DIMENSIONAL OPTIMIZATION OF AIRCRAFT CANOPY ACTUATION MECHANISM

BERK KEMAL PANK

FEBRUARY 2021

DIMENSIONAL OPTIMIZATION OF AIRCRAFT CANOPY ACTUATION MECHANISM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCE OF ÇANKAYA UNIVERSITY

BY

BERK KEMAL PANK

IN PARTIAL FULLFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

FEBRUARY 2021

ABSTRACT

DIMENSIONAL OPTIMIZATION OF AIRCRAFT
CANOPY ACTUATION MECHANISM

PANK, Berk Kemal

Master of Science, Department of Mechanical Engineering

Supervisor: Prof. Dr. Sıtkı Kemal İDER

Co-Supervisor: Assist. Prof. Dr. Özgün SELVİ

February 2021, 96 pages

This thesis presents a method for generating suitable mechanism for a defined problem step by step. The problem dealt with is finding an optimum solution for opening an aircraft canopy providing also locking in the same solution. For the objective, two types of mechanisms are used. Firstly, a Stephenson III type six-bar linkage is synthesized, then it is modified to geared seven-bar to get more movement capability. This work includes both synthesis with motion generation and optimization with genetic algorithm. 10 poses are defined to be approximated by the motion, and an original code of genetic algorithm is created to be used for dimensional optimization of the mechanism. Many trials are applied for both six-bar and geared seven-bar linkages, and a satisfying solution is obtained by the geared seven-bar mechanism finally. The mechanism described in this thesis is designed for an aircraft canopy actuation, but the methods can be used and applied for different purposes and fields.

Keywords: Dimensional optimization, Six-Bar linkage, Geared Seven-Bar linkage, Genetic algorithm, Motion generation, Aircraft canopy actuation mechanism

ÖZ

UÇAK KANOPİSİ HAREKETLENDİRME MEKANİZMASININ
BOYUTSAL OPTİMİZASYONU

PANK, Berk Kemal

Yüksek Lisans, Makine Mühendisliği Anabilim Dalı

Tez Yöneticisi: Prof. Dr. Sıtkı Kemal İDER

Ortak Tez Yöneticisi: Dr. Öğr. Üyesi Özgün SELVİ

Şubat 2021, 96 sayfa

Bu tez, tanımlanmış bir probleme yönelik uygun bir mekanizmayı adım adım üretmek için bir yöntem sunar. Ele alınan sorun, aynı çözümde kilitlemeyi de sağlayan, bir uçak kanopisinin açılması için optimum bir çözüm bulmaktır. Amaç için iki tür mekanizma kullanılmaktadır. İlk olarak, Stephenson III tipi altı çubuklu bir mekanizma sentezlenir, ardından daha fazla hareket kabiliyeti elde etmek için bu mekanizma üzerinde değişiklikler yapılarak, dişli seti eklenmiş bir yedi çubuk mekanizması elde edilir. Bu çalışma hem hareket üretme ile mekanizma sentezi, hem de sentezlenen mekanizmanın genetik algoritma ile optimizasyonunu içermektedir. Hareketi tanımlamak için 10 konum belirlenir ve mekanizmanın boyutsal optimizasyonu için kullanılmak üzere orijinal bir genetik algoritma kodu oluşturulur. Hem altı çubuklu hem de dişli yedi çubuklu mekanizmalar için birçok deneme uygulanır ve en sonunda dişli seti eklenmiş yedi çubuklu mekanizma ile istenen bir sonuç elde edilir. Bu tezde çalışılan mekanizma bir uçak kanopisini çalıştırması için tasarlanmıştır, ancak yöntemler farklı amaçlar ve alanlar için kullanılabilir ve uygulanabilir.

Anahtar Kelimeler: Boyutsal optimizasyon, 6-Kollu mekanizma, Dişli 7-Kolu mekanizma, Genetik algoritma, Hareket sentezi, Uçak kanopisi çalıştırma mekanizması

To My Mother…

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Dr. Sıtkı Kemal İDER for his supports and guidance in the period of preparing this thesis.

I am grateful to my co-supervisor Asst. Prof. Özgün SELVİ for his great helps, and allocating plenty of time during hard work periods even if it is night.

I feel gratitude to my mother for her support in whole of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Mechanisms are composed of some machine elements and used to transfer a force or motion in order to provide requested output. In the area of use, there are some limitations which influence the design that are worthy of notice. To create a mechanism design respecting the limitations requires some procedures. Mechanism synthesis which is the main one of them, is used for creating mechanisms for the determined boundaries or movements. After determining the mechanism type, the dimensional synthesis is implemented. To obtain a desired motion, path tracking or function generation, a variety of mechanism types can be synthesized. However, some of them cannot be really efficient, some cannot be rational to be used or some cannot be applicable. In other words, some design criteria limits could prevent using of most of them. Here, optimization method comes up. Briefly, an optimization provides selecting best one from the different alternatives. By this way, the best fit solutions could be found for the defined problem. There are various optimization methods in the literature, each one has different advantages from the each other, and their utilized examples exist.

In this work, an aircraft canopy actuation mechanism's dimensional optimization problem is handled. The defined problem is providing that the canopy could be opened by using some determined poses to be followed as close as possible. Therefore, the optimization problem is specialized for this purpose.

For the aircraft canopies, there should be a locking method to hold the canopy stable after it has been closed. This locking is generally provided by using another mechanism or by external helps manually. This work has been focused on providing

also the locking by using canopy opening mechanism at the same time. For this purpose, the movement is defined to achieve this locking with the same mechanism that runs by one actuator simultaneously.

The aim of the study is to point out a method for finding the optimum solution of a complicated motion generation problem which has prescribed 10 points and respective angles of the output link. When the locking process is integrated into the canopy movement, the path of the motion generated with these points becomes refractive, so has different side curves which makes the problem more complex and requires a compound mechanism.

Different types of mechanisms are considered for this purpose, then one of them is selected to be synthesized and optimized firstly. It is understood that a four-bar cannot derive the refractive movement which has different side curves. Therefore, in the beginning, a Stephenson III type six-bar is used, then by modifying this mechanism, a geared seven-bar linkage is generated to reach the desired objective in this work.

In the optimization side of this work, Genetic Algorithm method is decided to be implemented. This method is utilized to optimize link lengths of the mechanism, its ground connection points' (pivots) coordinates, some specific angles and values. In this process, all of these values are used for the necessary calculations to obtain the defined 10 poses which consist of objective values. By using them, the calculated objective values are acquired and compared with the prescribed points. The algorithm tries to minimize the difference between the two desired and calculated objective values. For this specific problem, the algorithm is modified as required, so that more efficient algorithm is achieved.

To mention about work done in this thesis, first of all, an aircraft body and its special canopy models are created in Creo. The locking pins and hinges are added to the models to be used for definition of the motion. After that, the canopy movement is defined by changing its position in the assembly model, so that the 10 prescribed poses are defined. After that, mechanism creation work is followed.

The first mechanism configuration is created manually to see visual shape of it firstly in Mathematica. To construct the mechanism, design parameters are found. Then, the mechanism's tracking point's path which is formed regarding input angle is created in this configuration. After that, the prescribed poses are also added visually, and the path is tried to be approximated to prescribed poses by changing the variables. When an approximate path is obtained in Mathematica, the values that provide this path are transferred to optimization algorithm which is created in Matlab. In Matlab, many trials are applied, but a sufficient result that satisfy the prescribed motion completely cannot be acquired. As a result of this, the mechanism is modified to geared seven-bar, so that more freedom of movement is procured for the movement. After implementing same procedures with the six-bar, an effectual result can be achieved.

The results obtained by the algorithm are transferred to Creo drawings to see how the mechanism created by these values moves in the working area. If there is no problem occurred, a force analysis is applied for the mechanism lastly. Eventually, an effective and usable mechanism is obtained by using these methods.

The methods explained in this thesis can be implemented for any other mechanisms' designs and optimization problems. Initial creation of the mechanism in Mathematica decreases the time needed for optimization, and helps to find design parameters. For different mechanisms, these can be applied according to its necessities. In the optimization side, more restrictions can be added on the presented algorithm with respect to the required mechanisms' limits. Consequently, the methods and algorithm developed in this thesis can be modified and developed in accordance with requirements of different problems.

CHAPTER 2

LITERATURE SURVEY

2.1. Mechanism Synthesis

In the area of mechanism synthesis which includes path, function and motion generation, there are many published works. Shrinivas S. Balli and Satish Chand suggested an analytical method for synthesis of five-bar motion and path generators [3]. Gim Song Soh and J. Michael McCarthy demonstrated synthesis process for the design of a six-bar steering linkage with using motion generation [8]. They used Watt I and Stephenson I, II and III six-bar structures for five pose synthesis. Briefly, they consider a 3R chain with adding two RR constraints to have the six-bar structures. They indicated that one degree of freedom systems for six links and seven joints consist of two basic topologies which Watt and Stephenson chains base on this. Figure 1 shows these chains and their inversions.

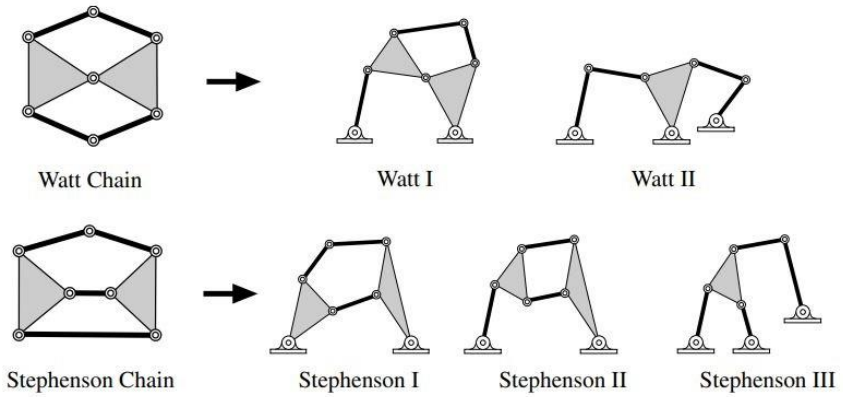

Figure 1 *The Watt and Stephenson six-bar chains, and their inversions.*

Except for the Watt II structure, the chains have a 3R chain inside. With their six-bar synthesis methodology, they stated that there are 63 possible linkage designs by evolving the Watt and Stephenson mechanisms. In conclusion, their design resulted in a single degree of freedom 14-bar linkage as a combination of Watt I linkages for each of the two front wheels with a crank and two couplers. Kevin Russell and Raj S. Sodhi presented motion generation for a five-bar mechanism which is one degree of freedom and geared [11]. They considered a two-phase moving pivot adjustment on rigid body of the five-bar which is used for defining poses for the motion generation. The mechanism can be seen in Figure 2.

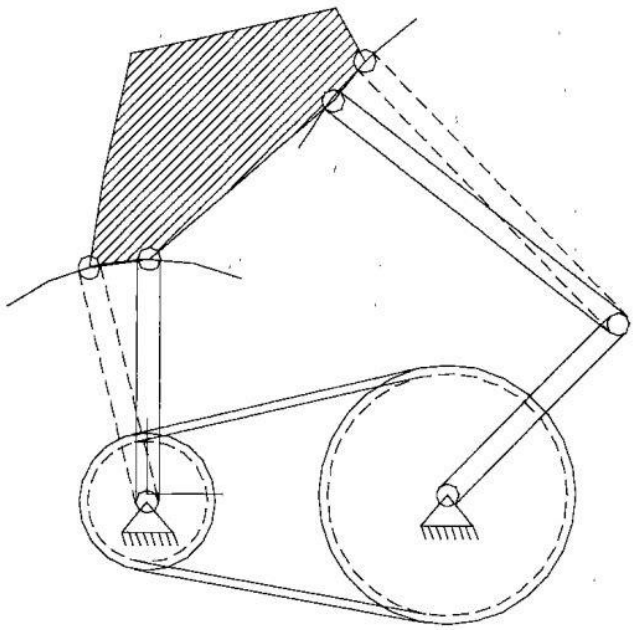

Figure 2 *Moving pivoted geared five-bar mechanism.*

Unlike the traditional method, they chose to describe the rigid body as three points instead of choosing a single point and displacement angle for their computation way. The advantage of their design is ability of utilizing the same mechanism for multiple phases of prescribed rigid body poses. Mark M. Plecnik and J. Michael McCarthy worked on a design procedure of Stephenson I, II and III type six-bar mechanisms with 11 configurations approach along the trajectory, which were intended to use as a leg

mechanism of a walking machine [16]. Each candidate designs arisen from the Stepheson mechanisms was analyzed. Four types of the Stephenson mechanisms can be seen in Figure 3.

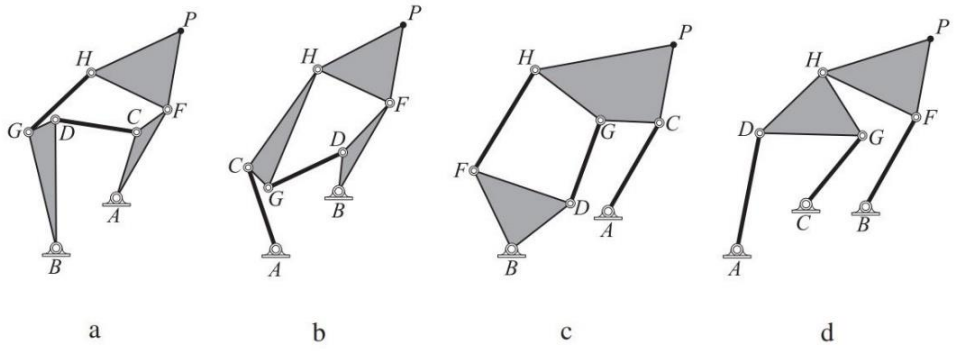

Figure 3 *Four types of Stephenson six-bar mechanisms.*

They took advantage of Mathematica with its functions like "NSolve" to find solutions of the kinematic equations. They also remarked that, a path generator should reach all specified points on a single trajectory without passing through a singularity for a successful design. Ying XI, Ying XIA, Xuanxuan LI and Yingbiao SUN worked on parametric design of a mathematical model of six-bar linkage which is for auto-feeding soup machine as feeder mechanism of die castings [17]. They modelled, simulated and optimized the parameters by using the software ADAMS. Khalid Nafees and Aas Mohammad presented dimensional synthesis of a Stephenson II type six-bar linkage [18]. Their objective was path generation with 15 prescribed points. They used Matlab to calculate loop closure equations and SAM software to verify the determined link lengths with their configurations.

2.2. Optimization Methods

While Genetic Algorithm method has been implemented, there are also different types of optimization techniques have been used in optimization problems in the existing literature. To mention some of them for the different methods with GA, Hitesh R. Patel and M.J. Mungla have took advantage of genetic algorithm [1]. They used the GA optimization technique because of high number of parameters to find optimal solution. They also mentioned GA works with probability search technique that needs less running time compared to other techniques. J.A. Cabrera, A.Simon and M.Prado utilized genetic algorithm [2]. They used randomly selected individuals. They create disturbing vectors and they used them for reproduction with crossover. They also mentioned two types of crossover and these can be seen in Figure 4.

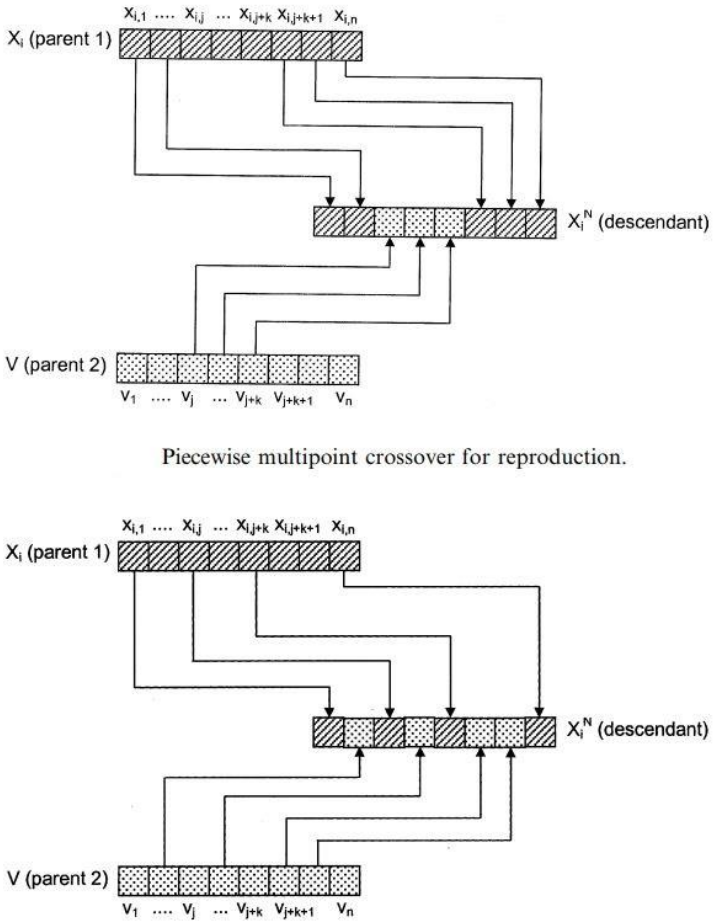

Figure 4 *Discrete multipoint crossover for reproduction.*

Moreover, they imposed some conditions for the populations. Then they used them by checking in the objective function. If the conditions fail, objective function which search for minimum value goes to much higher value. Observation of them is that more iterations required for genetic algorithms compared to gradient based methods. However, the gradient based methods' evaluation is more complex, which means total computation time of GA becomes lower generally. Yahia M. Al-Smadi, Qiong Shen, Kevin Russell & Raj S. Sodhi have worked on motion generation with a new design constraint, which is driving link static torque, for their synthesis of four-bar braking mechanism [4]. A representative of their mechanism can be seen in Figure 5.

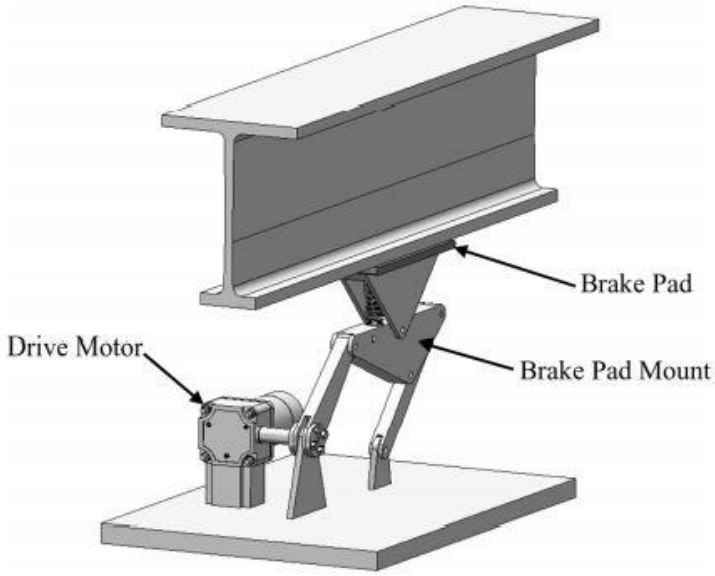

Figure 5 *Four-bar traveler braking mechanism.*

Kevin Russell and John Shen demonstrated a synthesis optimization model for both path and motion generation of four-bar linkage applied for a landing gear [6]. They considered the linkages should be free of order and branch defects with remarking these are inherent in path and motion generation, that means poses is in an order and true configuration (all four-bar linkages have two configurations) of the four-bar mechanism is applied. They used Sequential Quadratic Programming algorithm implemented in Matlab. The mechanism that is worked on can be seen in Figure 6.

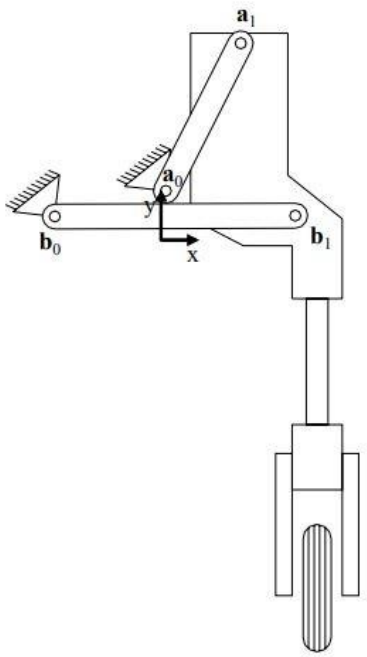

Figure 6 *Landing gear with synthesized four-bar.*

Galal A. Hassaan, Mohammed A. Al-Gamil and Maha M. Lashin worked on optimization of four-bar crank-rocker mechanism synthesis with Powell technique [5]. P.S. Shiakolas, D. Koladiya and J. Kebrle worked on a Stephenson 3 type six-bar mechanism synthesis [7]. They presented a methodology which combines Differential Evolution (DE) optimization technique and Geometric Centroid of Precision Positions (GCPP) technique. They used GCPP for the initial bounds of the design variables. They indicated that Differential Evolution contains same processes with Genetic Algorithm like initial population, crossover, mutation and selection. However, they indicated also DE uses the vector differentiation method to generate a new vector unlike the GA. They also mentioned that, solutions don't likely exist in the initial range of design variables in the classical optimization techniques which requires initial guesses for each design variable, so the quality of the final results is dependent to initial guesses or design variable bounds. Therefore, they used GCPP approach which automatically defines the initial bounds of the design variables. They proposed 3

different approaches. One is two stage synthesis that consists of synthesis of four-bar firstly, then synthesis of remaining dyad of six-bar. The other ones consist of direct synthesis of six-bar. They observed that the six-bar mechanism can be synthesized by using any of the proposed approaches, but the synthesizing firstly four-bar mechanism requires a smaller number of function evaluations since it reduces the number of design variables. Radovan R. Bulatovic, Stevan R. Dordevic and Vladimir S. Dordevic worked on synthesis of a Stephenson III type six-bar mechanism, which can be an alternative to cams or to meet certain special requirements that might not be satisfied by a four-bar mechanism [9]. They used a new metaheuristic algorithm, Cuckoo Search (CS). Metaheuristic algorithms have two major components: intensification and diversification, and they are more advanced and efficient than heuristic algorithms. Diversification means to generate diverse solutions on the global scale, while intensification means to focus on the search in a local region. The good combination of these mostly provide the global optimality is achievable with preventing the solutions being trapped al local optima, and increasing the diversity of the solutions. They exploited Cuckoo Search (CS) algorithm for the synthesis optimization. They also mentioned some of the other developed metaheuristic algorithms which are mostly nature-inspired: Simulated Annealing (SA), Genetic Algorithms (GA), Ant Colony Optimization (ACO), Bee Algorithms (BA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Harmony Search (HS), Firefly Algorithm (FA), Cuckoo Search (CS), and Bat-Inspired Algorithm (BA). Hamid Mehdigholi and Saeed Akbarnejad worked on a one degree of freedom six-bar mechanism that generates straight and parallel motion to be used for their robot's legs [10]. The reason choosing the six-bar is that, it simulates walking of animals very well. They also took advantage of GA to find optimal lengths of links. D. Mundo, G. Gatti and D.B. Dooner presented a dimensional synthesis approach and a genetic algorithm optimization for a five-bar linkage with non-circular gears that, its coupler point moves along a prescribed trajectory which consist of two parallel straight lines [12]. These non-circular gears reduce mobility of the five-bar to one by hooking one of the two angular inputs on other. To visualize, the mechanism they created can be seen in Figure 7.

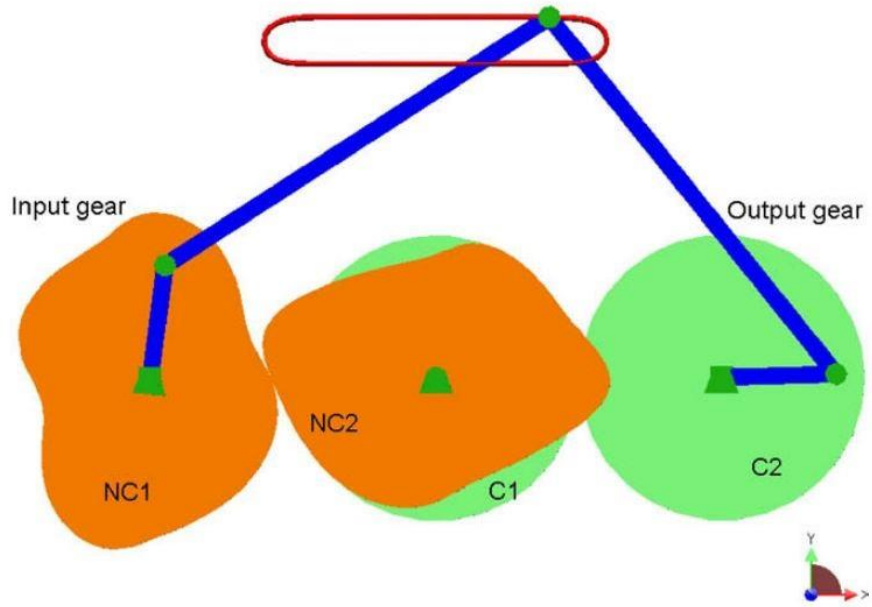

Figure 7 *CAD assembly model of Geared Five-Bar Mechanism (GFBM) with non-circular gears.*

In this work, a genetic algorithm is used to find a global optimum solution. Their methodology has two phases: Inverse kinematic analysis of the linkage (path generation) and synthesis of the non-circular gears. In the GA, they employed a set of design rules and penalty values to eliminate unfeasible solutions, and weighting factors are used for the objectives. To do this, they utilized trial and error approach. They chose randomly generated population of 100 individuals and 50 iterations to get a desired solution. Khalid Nafees and Aas Mohammad presented dimensional synthesis of Stephenson I type six-bar mechanism for path generation with 12 points, and optimized it by using GA [13]. They also used the crank angle's 12 points' values as input values. They stated that, error of a desired path can be decreased by increasing the number of links, but it can cause increasing errors at each joint of the mechanism and significant deviation at the same time, so balancing the two of them is important. They also remarked that, many researchers have worked on optimal synthesis for four-bar mechanisms. Some of them have worked on synthesis of five-bar and seven-bar whose degree of freedoms are more than one, so their synthesis work is based on two finitely separated poses. However, they asserted that there is no extensive research on optimal synthesis of mechanisms with high number of links. In addition, they

expressed that in GA, the initial random population size is considered as 2 to 4 times of variable count generally. Junli Shen, Guoqiang Wang, Qiushi Bi and Junna Qu studied for a mathematical model of loaders' Z-bar mechanism [14]. It includes syntheses of both a four-bar and also a six-bar Watt linkage. They tried a new method, comprehensive genetic algorithm which uses sequential quadratic programming (SQP) method too, to optimize non-linear equation with multi-constraints. The flowchart of their algorithm can be seen in Figure 8.

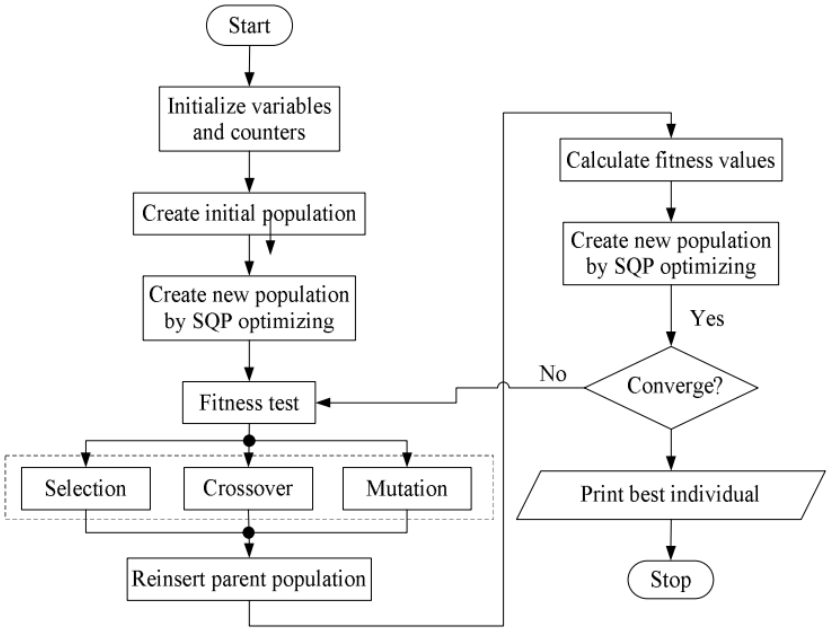

Figure 8 *Flowchart of GA and SQP optimization.*

They used SQP to optimize individuals of each generation in local optimum region, and these local optimums compose new populations. Then, the GA continues to normal work. This provides to prevent falling into local optimal region of the population. In GA, they made benefit of normalization and weighting methods to decrease more objective functions to a single. J. Afshari, F. Nazari, S. Baghalian and S.S. Malihi investigated a six-bar mechanism used in a prosthetic knee with computer simulation and optimization method of genetic algorithm [15]. They considered to use 4 different type of Watt and Stephenson mechanisms, and chose one of them for this work. In the

optimization, they took advantage of weighting factors for the objective function too. In addition, this six-bar is compared to a four-bar mechanism, and they indicated that the expected trajectory is better achieved by the six-bar. Wen-Yi Lin studied on optimization synthesis of a geared five-bar mechanism whose coupler point needed to trace some defined target trajectories [19]. These trajectories consist of up to 41 points. The motion of this five-bar mechanism's link with number of 5 has been derived by motion of link 2. He stated that more complex solutions can be provided by a geared five-bar rather than a four-bar. The gear set can contain belt and pulleys. A representative picture of his mechanism can be seen in Figure 9.

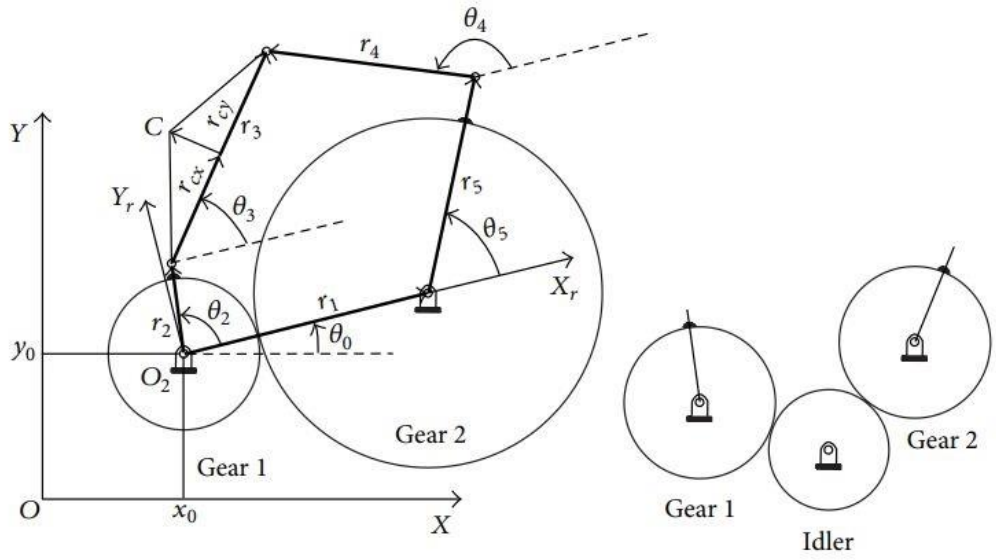

Figure 9 *Geared five-bar mechanism Free Body Diagram.*

He took advantage of GA and DE evolutionary algorithm to optimize the synthesis. Input variables are given with input angle values of the link 2 for each point. For the objective function, error function is used with square deviation method which is needed to be minimized. He used Roulette Wheel method for selection of the individuals, and Elitism method is utilized also. The results have been validated by SolidWorks program with animation.

CHAPTER 3

MATERIALS AND METHODS

3.1. Selecting Prescribed Values

Firstly, an aircraft canopy draft and its counter body (also aircraft fuselage) are created on PTC Creo (A 2D drawing from side view) and defined as a base line.

Then, the canopy locking method which is provided by actuation movement is chosen.

The maximum opening angle of the canopy is defined as 30°.

Movement path of the canopy opening is defined step by step. 10 poses of the canopy are selected for mechanism synthesis. Where required more, poses are chosen closer to each other.

To provide the desired movement, a six-bar mechanism is predicted to be more appropriate compared to a four-bar mechanism.

3.2. Canopy FBD

The canopy's free body diagram with the frame and point P is shown in Figure 10. The point P which is on left bottom corner of the canopy, is considered as the point to be followed together with the angle of the canopy with respect to the prescribed 10 points and 10 angles. The location of mechanism connecting points to the canopy is planned to be defined after the best fit mechanism is found.

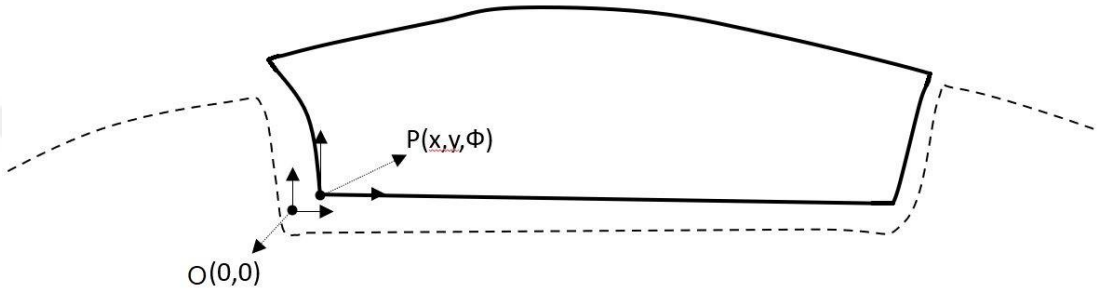

Figure 10 *Free Body Diagram (FBD) of the Canopy with Aircraft Frame.*

3.3. Creo Views of the Canopy for Different Poses

In Creo, models are created to show aircraft fuselage, canopy and seat perspective from side view in the prescribed orientation. As the canopy movement is planar, this side view (2-D) is used to exhibit the locations of the 10 points.

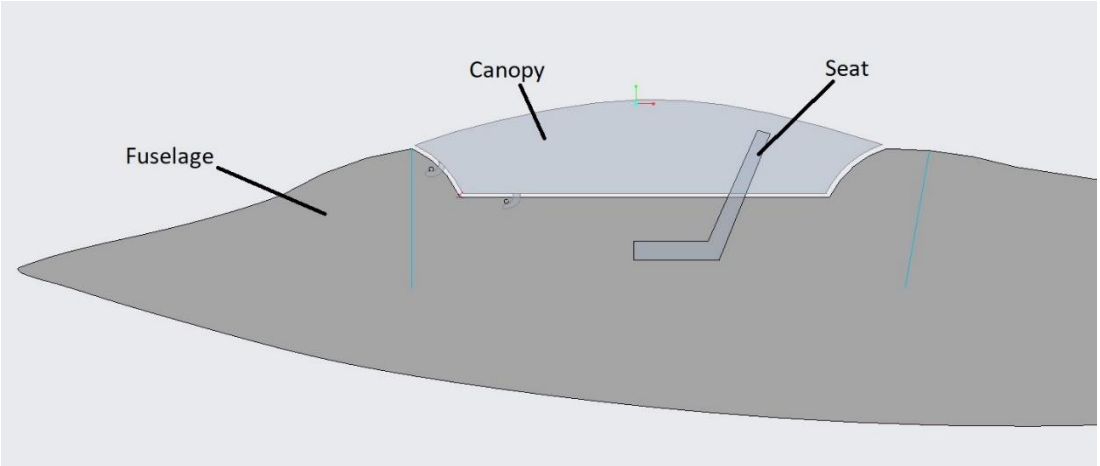

Figure 11 *Side view of Aircraft.*

As they have been shown in free body diagram, the base coordinate frame O and point P on the Creo drawing are shown in Figure 12 and Figure 13. Base coordinate frame O is 12 mm above from canopy sill (fuselage border). Here, the hinges for locking the canopy are shown too. The hardest point for the canopy opening mechanism movement is to disentangle these hinges from the hinge pins. This is because, the range of motion is limited here and the movement of point P must be fine until the hinges disentangle completely.

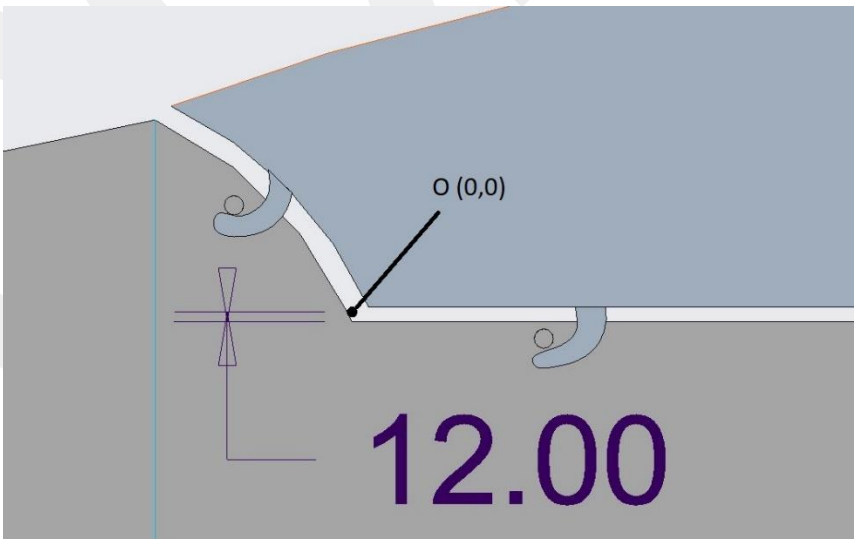

Figure 12 *Location of point O.*

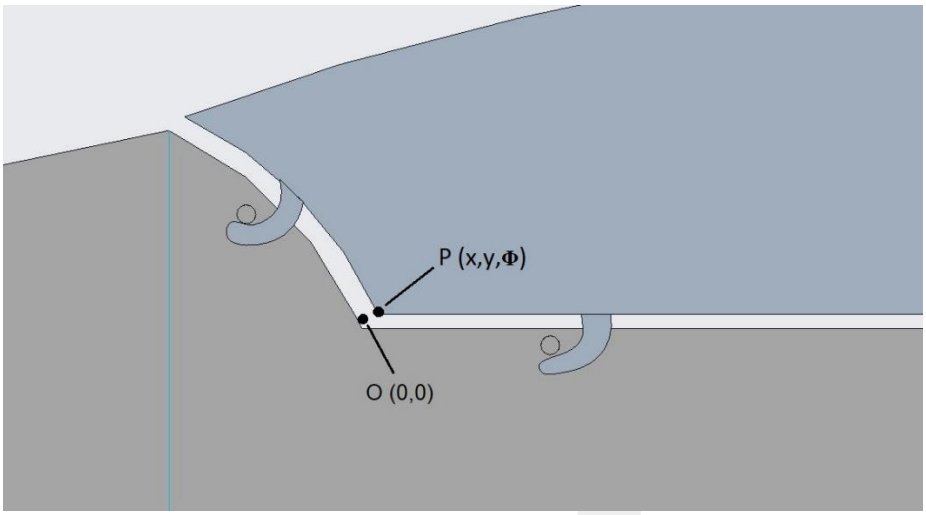

Figure 13 *Points O and P together.*

3.4. 10 Points Definition

The 10 points' poses are defined step by step by considering the hinge-lock poses. First five points represent disentangling from the pins, that also creates a curve when they are joined together. The other five points represent the canopy opening. However, when the other five points are joined, this creates a reverse side curve. The curves created by the first 5 points and the last 5 points can be seen in Figure 14 and Figure 15.

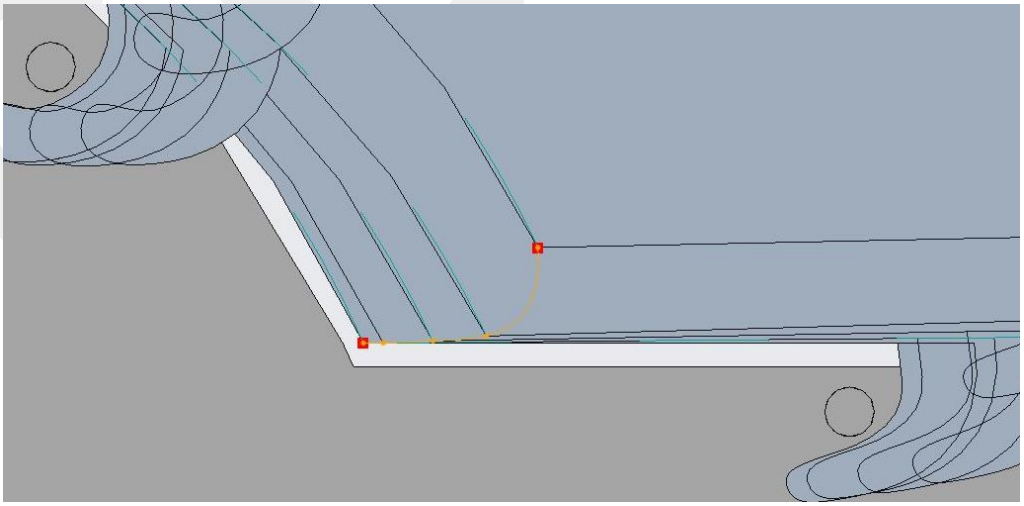

Figure 14 *The curve created by first 5 points.*

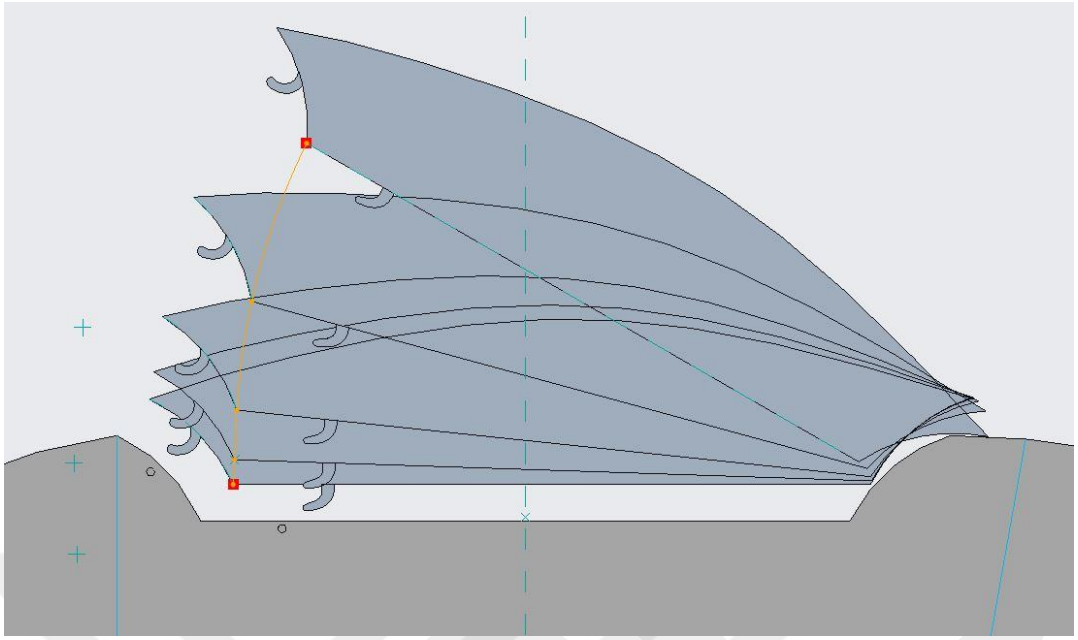

Figure 15 *The curve created by last 5 points.*

To define the 10 locations of the point P, the selected poses P(x,y,Φ) are used. In figures 16-25 below, the 10 poses' side view representatives declare the definitive variables x, y, and angle Θ. All the prescribed values of these 10 points poses with x, y and Θ angle is shown in Table 1 below.

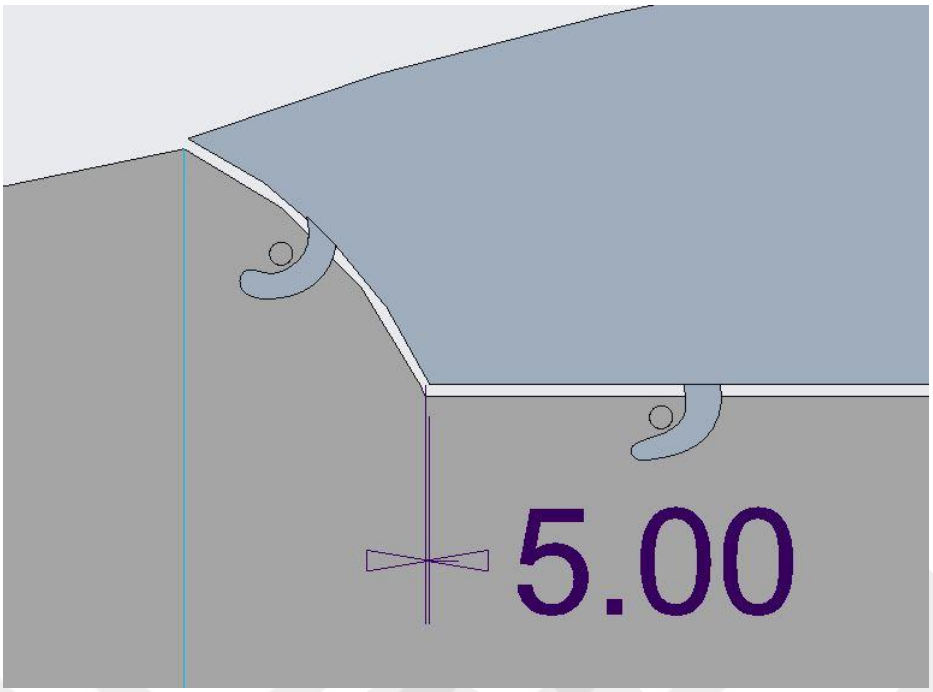

Figure 16 *Prescribed point 1.*

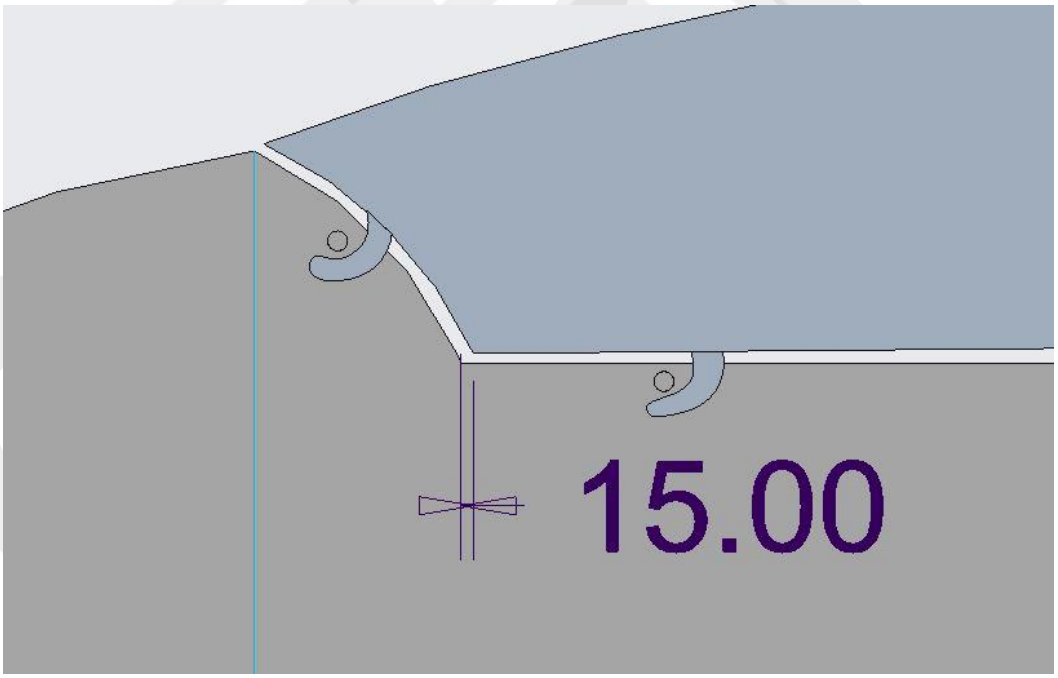

Figure 17 *Prescribed point 2.*

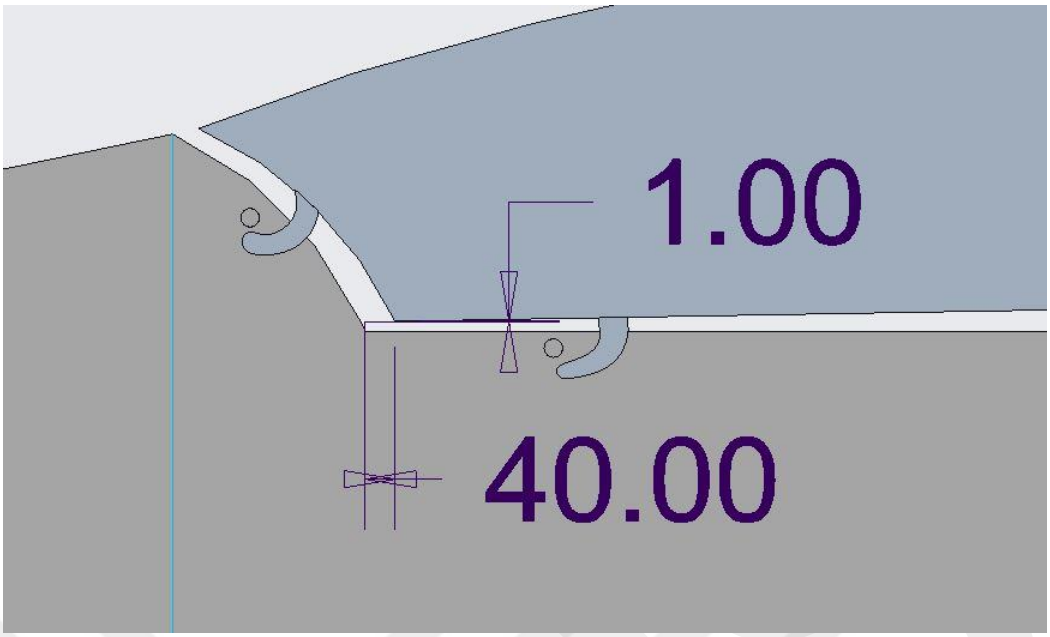

Figure 18 *Prescribed point 3.*

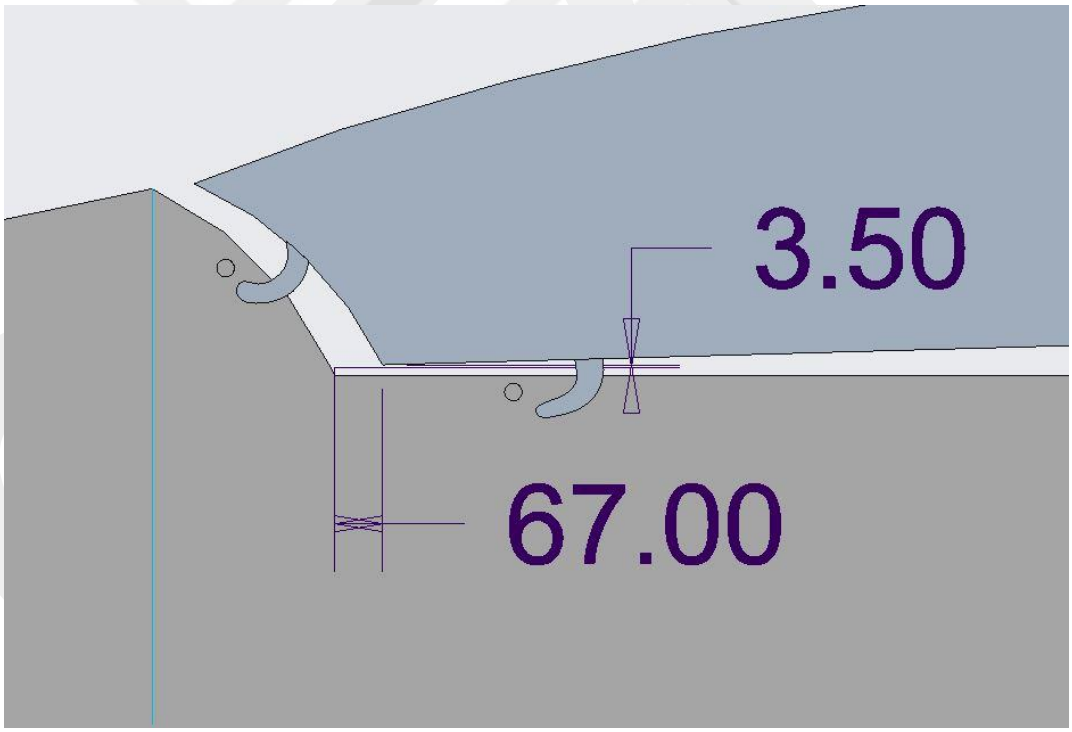

Figure 19 *Prescribed point 4.*

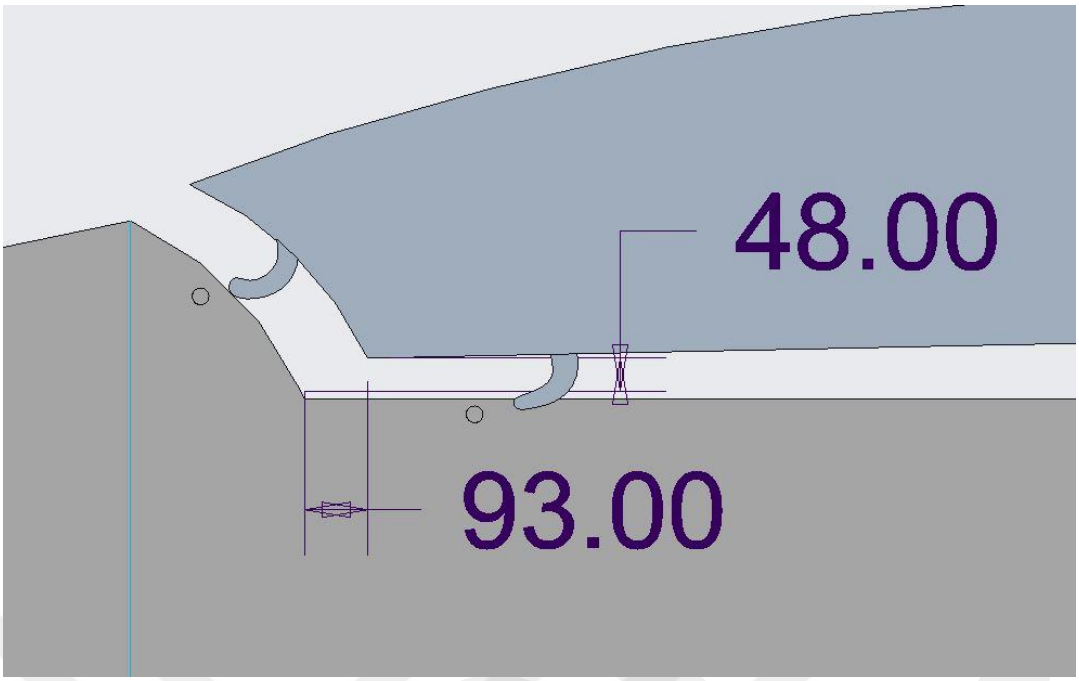

Figure 20 *Prescribed point 5.*

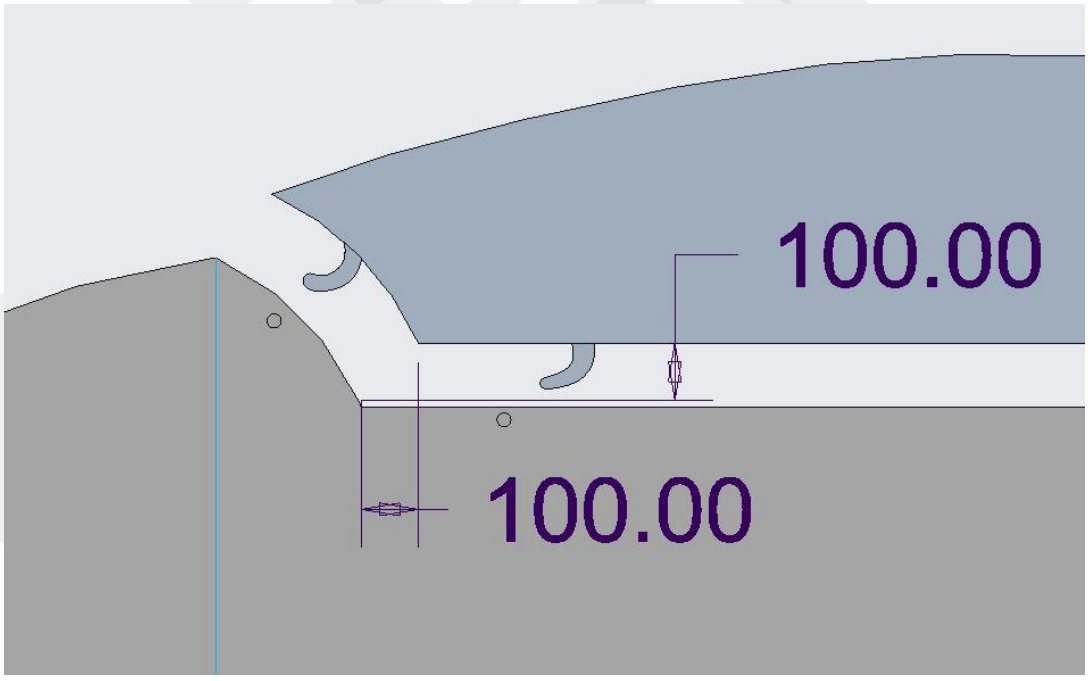

Figure 21 *Prescribed point 6.*

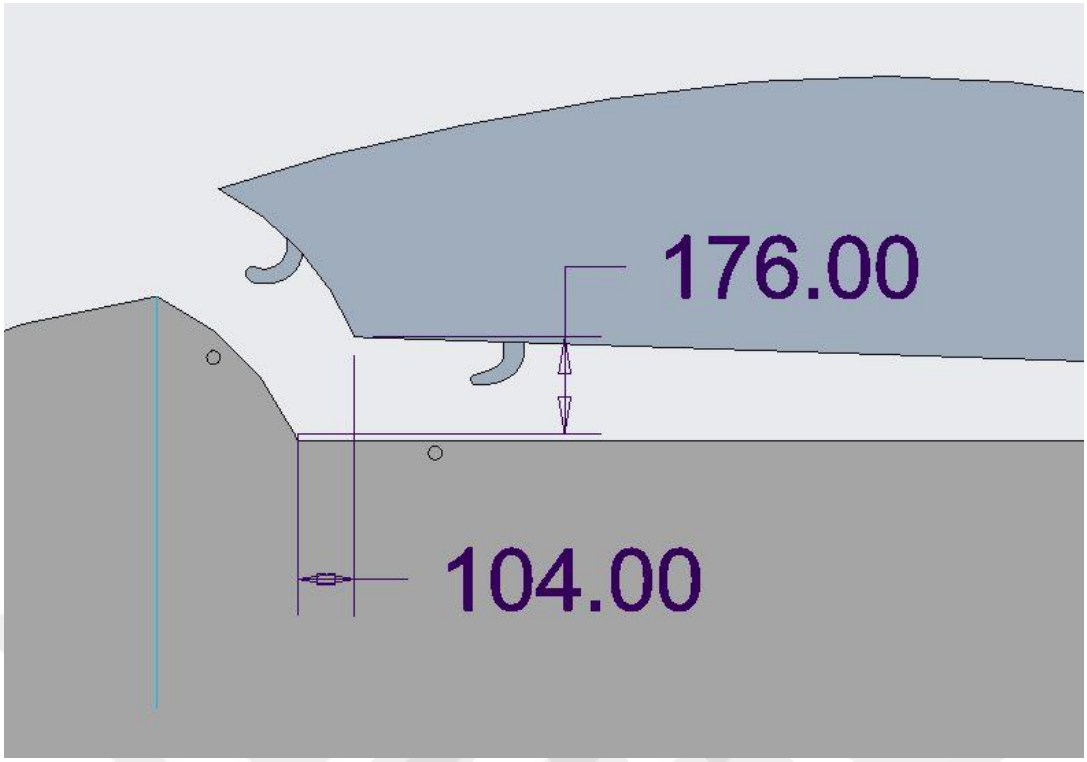

Figure 22 *Prescribed point 7.*

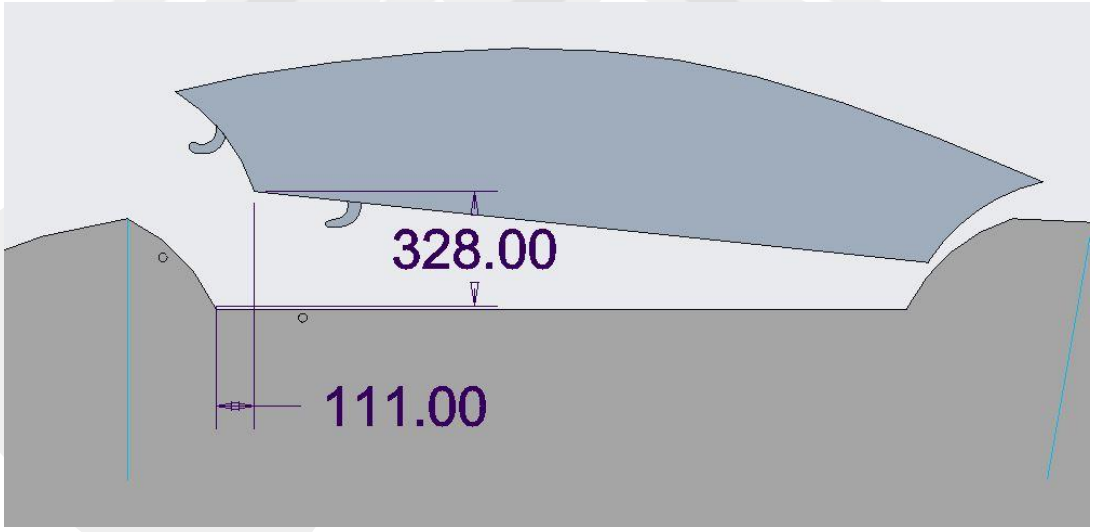

Figure 23 *Prescribed point 8.*

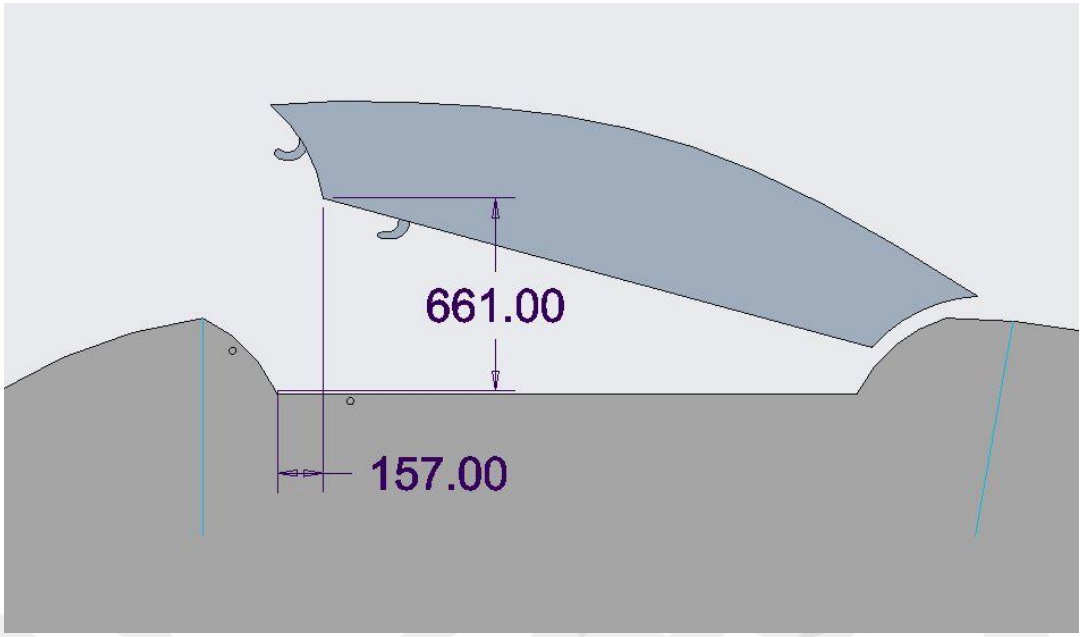

Figure 24 *Prescribed point 9.*

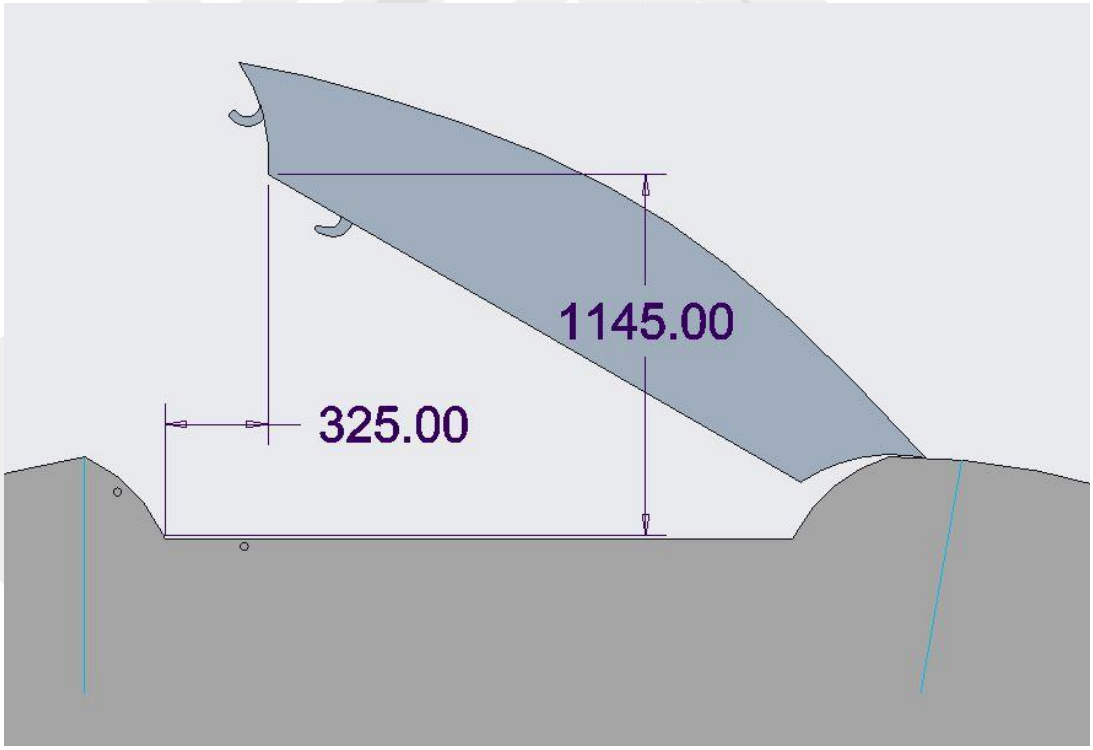

Figure 25 *Prescribed point 10.*

Table 1 Prescribed 10 poses' values.

Poses	P (x) (mm)	P (y) (mm)	P (Φ) (deg)
1	5	0	0
2	15	0	-0.5
3	40	1	-1
4	67	3.5	-1.6
5	93	48	-1.19
6	100	100	0
7	104	176	1.9
8	111	328	6
9	157	661	15
10	325	1145	30

The dimensional synthesis will be performed to the mechanisms by referencing these defined points' coordinates and angles. For a successful synthesis, the mechanisms should be sufficiently close to all the 10 prescribed points and 10 prescribed angles on a single trajectory. Figure 26 shows the 10 poses of the canopy at the same time.

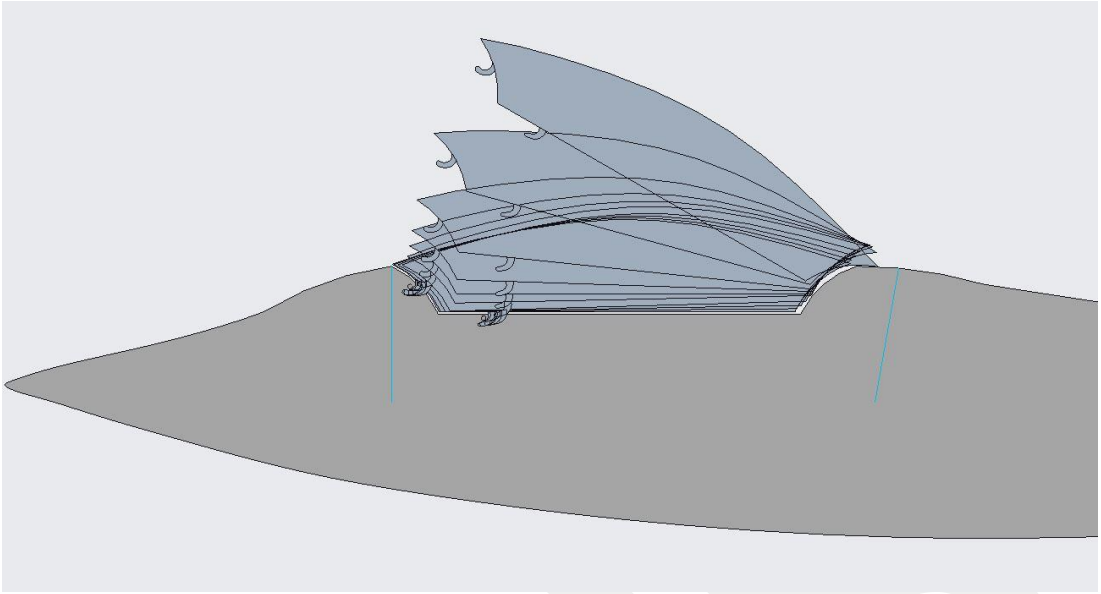

Figure 26 Representation for the 10 prescribed poses of the canopy in same figure.

3.5. First Mechanism Approach – Stephenson III Type Six-Bar

It is obvious that a four-bar cannot maintain that kind of complicated motion that has different side curves. The chosen mechanism should have had more mobility. Therefore, a six-bar linkage which is Stephenson III type is chosen and set to obtain the desired motion for the beginning. The mechanism's free body diagram with link, joint, angle names and frame are shown in Figure 27.

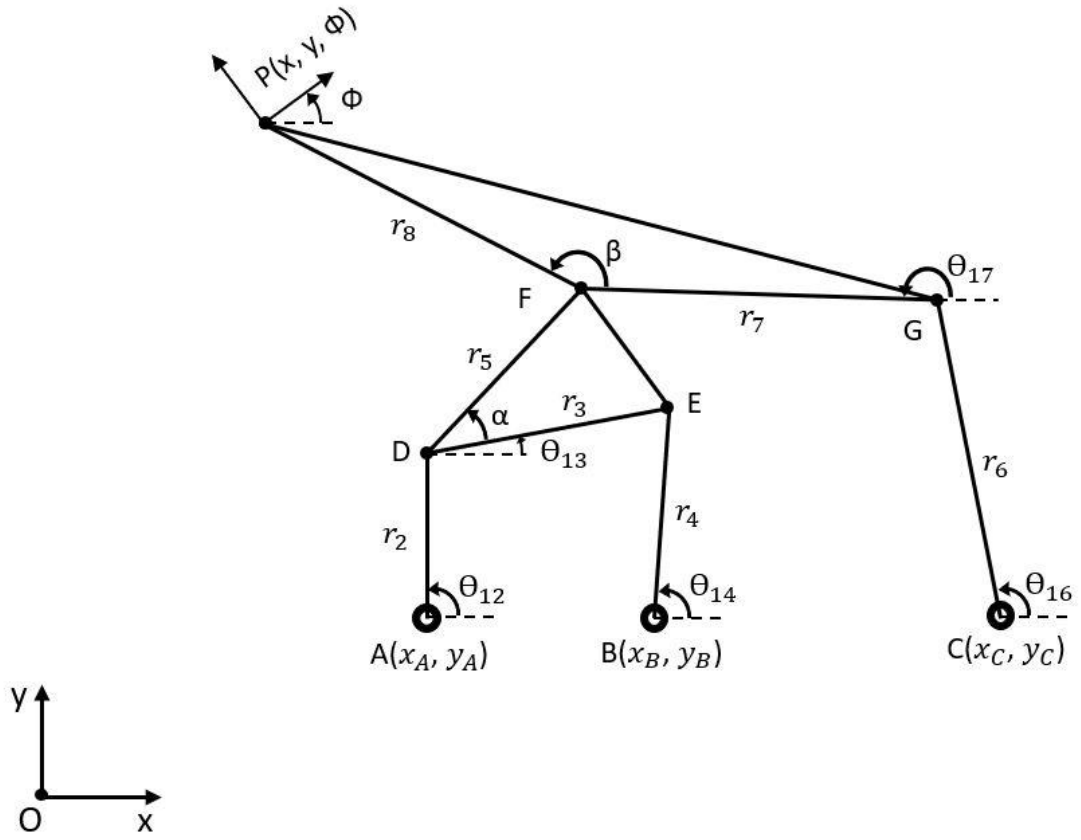

Figure 27 *Free Body Diagram (FBD) of six-bar mechanism.*

There are two ternary links that represent rigid bodies, FGP and DEF. The ternary link consisting of points F, G and P can be a part of canopy or fixedly connected to it via some connection points. Here, there are 15 constant (input) values xa, ya, xb, yb, xc, yc, r2, r3, r4, r5, r6, r7, r8, α and β. In addition to these, Θ12 is defined as variable input value of the mechanism. The 4 calculable variables are defined as Θ13, Θ14, Θ16, Θ17. Moreover, px, py and Φ which are coordinates and angle with respect to the frame, are regarded as output values.

Except the pivot points A, B and C, all the points' coordinates change as the mechanism moves. The coordinates of these points are calculated by Loop Closure Equations (LCEs). The input movement of the mechanism is imparted by rotation of the link 2 (crank), in other words, when the angle Θ12 changes, the mechanism moves.

To obtain motion equations, the LCEs are described. There are two main equations for the six-bar mechanism, and one more equation is written to reach the point P. These equations with their components' expansions can be seen below:

LCE 1: $\overrightarrow{AD} + \overrightarrow{DF} = \overrightarrow{AC} + \overrightarrow{CG} + \overrightarrow{GF}$

LCE 2: $\overrightarrow{AD} + \overrightarrow{DE} = \overrightarrow{AB} + \overrightarrow{BE}$

LCE 3: $\overrightarrow{OP} = \overrightarrow{OA} + \overrightarrow{AD} + \overrightarrow{DF} + \overrightarrow{FP} = \overrightarrow{OC} + \overrightarrow{CG} + \overrightarrow{GF} + \overrightarrow{FP}$

$$\overrightarrow{AD} = r_2.e^{i\theta_{12}} = r_2.cos\theta_{12} + r_2.isin\theta_{12}$$

$$\overrightarrow{DF} = r_5.e^{i(\theta_{13}+\alpha)} = r_5.cos(\theta_{13}+\alpha) + r_5.isin(\theta_{13}+\alpha)$$

$$\overrightarrow{AC} = (x_C - x_A) + \text{i}.(y_C - y_A)$$

$$\overrightarrow{CG} = r_6.e^{i\theta_{16}} = r_6.cos\theta_{16} + r_6.isin\theta_{16}$$

$$\overrightarrow{GF} = r_7.e^{i\theta_{17}} = r_7.cos\theta_{17} + r_7.isin\theta_{17}$$

$$\overrightarrow{DE} = r_3.e^{i\theta_{13}} = r_3.cos\theta_{13} + r_3.isin\theta_{13}$$

$$\overrightarrow{AB} = (x_B - x_A) + \text{i}.(y_B - y_A)$$

$$\overrightarrow{BE} = r_4.e^{i\theta_{14}} = r_4.cos\theta_{14} + r_4.isin\theta_{14}$$

$$\overrightarrow{FP} = r_8.e^{i(\theta_{17}+\beta-\pi)} = r_8.cos(\theta_{17}+\beta-\pi) + r_8.isin(\theta_{17}+\beta-\pi)$$

The LCE 2 is a typical four-bar mechanism equation. With the LCE 1 and LCE 2 together, all unknowns of the six-bar can be obtained in terms of the mechanism parameters, and the mechanism's general course of motion can be found. To obtain point P's coordinates, the LCE 3 is needed which helps to find offset difference between mechanism and output point's values.

Initially, hand calculations are tried to apply to equate the variables regarding these LCEs. The progress on it can be seen below:

For LCE 1:

Re: $r_2.cos\theta_{12} + r_5.cos(\theta_{13} + \alpha) = (x_C - x_A) + r_6.cos\theta_{16} + r_7.cos\theta_{17}$

Im: $r_2.sin\theta_{12} + r_5.sin(\theta_{13} + \alpha) = (y_C - y_A) + r_6.sin\theta_{16} + r_7.sin\theta_{17}$

For LCE 2:

Re: $r_3.cos\theta_{13} = (x_B - x_A) + r_4.cos\theta_{14} - r_2.cos\theta_{12}$

Im: $r_3.sin\theta_{13} = (y_B - y_A) + r_4.sin\theta_{14} - r_2.sin\theta_{12}$

Because to calculate these and find the variables' equations by hand is getting hard, these equations are transferred to program called Mathematica. Thanks to it, results can be obtained easier and faster. In addition, Mathematica provide crosschecking the hand calculations and progressing on it. Kinematic equations of six-bar are solved via NSolve function of Mathematica, and these equations can be seen below:

$$A1 = -2.r_4.(x_A - x_B + r_2.cos\theta_{12})$$

$$B1 = -2.r_4.(y_A - y_B + r_2.sin\theta_{12})$$

$$C1 = -\left[{r_2}^2 - {r_3}^2 + {r_4}^2 + (x_A - x_B)^2 + (y_A - y_B)^2 + 2.r_2.\left((x_A - x_B).cos\theta_{12} + (y_A - y_B).sin\theta_{12}\right)\right]$$

Firstly, there are two equations, one is for x and the other one is for y direction, derived from LCE 2. By using them, Θ13 is eliminated, and Θ14 is found as an equation. After that, Θ13 could be found using Θ14 equation.

$$\theta_{14} = Arctan\left(\frac{B1}{A1}\right) + Arccos\left(\frac{C1}{\sqrt{A1^2 + B1^2}}\right)$$

$$\theta_{13} = Arctan\left(\frac{\frac{(y_B - y_A) + r_4.sin\theta_{14} - r_2.sin\theta_{12}}{r_3}}{\frac{(x_B - x_A) + r_4.cos\theta_{14} - r_2.cos\theta_{12}}{r_3}}\right)$$

Similar to LCE 2's derived calculations, Θ17 is eliminated from the LCE 1's derived equations, then Θ16 and Θ17 are found respectively.

$$A2 = 2.r_6.(x_A - x_B) - r_2.r_6.cos\theta_{12} - 2.r_5.r_6.cos(\theta_{13} + \alpha)$$

$$B2 = -2.r_6.\left(y_A - y_C + r_2.sin\theta_{12} + r_5.sin(\theta_{13} + \alpha)\right)$$

$$\begin{aligned} C2 = {} & r_2{}^2 + r_5{}^2 + r_6{}^2 - 2.r_7{}^2 + (x_A - x_C)^2 + (y_A - y_C)^2 + 2.r_2.x_A.cos\theta_{12} \\ & - 2.r_2.x_C.cos\theta_{12} + 2.r_5.x_A.cos(\theta_{13} + \alpha) \\ & - 2.r_5.x_C.cos(\theta_{13} + \alpha) + 2.r_2.r_5.cos(\theta_{13} + \alpha - \theta_{12}) \\ & + 2.(y_A - y_C).\left(r_2.sin\theta_{12} + r_5.sin(\theta_{13} + \alpha)\right) \end{aligned}$$

$$\theta_{16} = Arctan\left(\frac{\frac{-B2.C2 - A2.\sqrt{B2^2.(A2^2 + B2^2 - C2^2)}}{B2^2.(A2^2 + B2^2)}}{\frac{-A2.C2 + \sqrt{B2^2.(A2^2 + B2^2 - C2^2)}}{A2^2 + B2^2}}\right)$$

$$\theta_{17} = Arctan\left(\frac{\frac{r_2.sin\theta_{12} + r_5.sin(\theta_{13} + \alpha) - (y_C - y_A) - r_6.sin\theta_{16}}{r_7}}{\frac{r_2.cos\theta_{12} + r_5.cos(\theta_{13} + \alpha) - (x_C - x_A) - r_6.cos\theta_{16}}{r_7}}\right)$$

To find the target point P's coordinates, the following equations are used.

$$x_P = x_A + r_2.cos\theta_{12} + r_5.cos(\theta_{13} + \alpha) + r_8.cos(\theta_{17} + \beta - \pi)$$

$$y_P = y_A + r_2.sin\theta_{12} + r_5.sin(\theta_{13} + \alpha) + r_8.sin(\theta_{17} + \beta - \pi)$$

After all the required equations found, proving trials are applied to see how the mechanism looks like on Mathematica with “Animate” function. The output of the function can be seen in Figure 28.

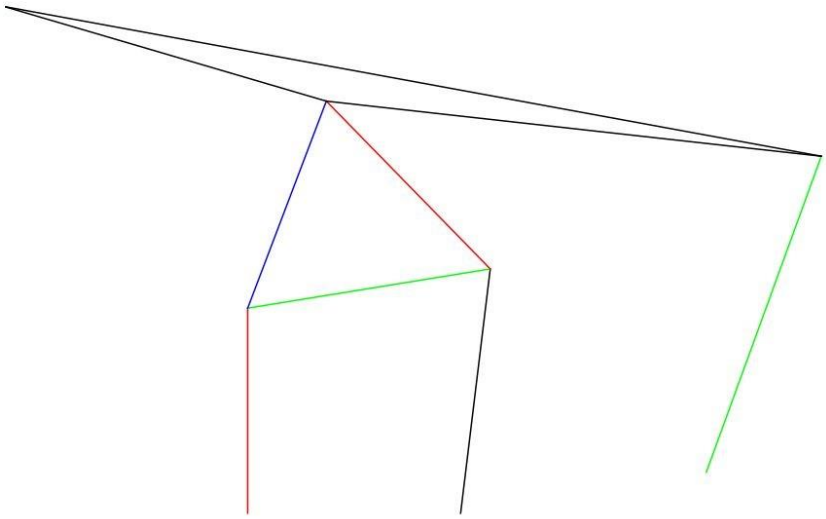

Figure 28 *The six-bar mechanism representation created by "Animate" function.*

The input variables are given approximately by measuring dimension values of the preliminary drawn mechanism. Firstly, wrong assembly configurations are observed with this work, so some of the mechanism links are placed inversely. After that, other configuration of the equations is taken, so that it corrects the view and configuration, and the expected mechanism assembly configuration is achieved.

On this created mechanism animation, tracking path of point P is added. This helps to see how the point moves when mechanism crank starts to rotate. This formation can be seen in Figure 29.

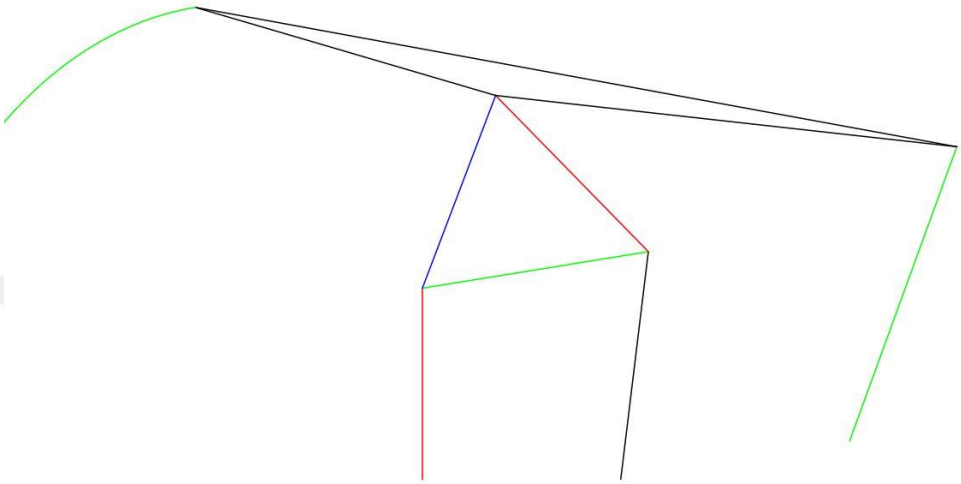

Figure 29 *The six-bar mechanism representation with path of point P.*

Finding the correct assembly configuration makes the next step easier, which is trying to find best path visually on Mathematica. By using prescribed values, 10 different 2 mm size links are created starting from the 10 points' coordinates through their related angles for each. Figure 30 shows the prescribed 10 poses created in Mathematica.

Figure 30 *The prescribed 10 poses' representation without mechanism.*

This shows prescribed locations of the canopy with required angles at these locations. To level size of the mechanism with the prescribed location coordinate sizes, a coefficient is used for the mechanism dimensions. This coefficient is determined as 15 after some trials. From now on, the coefficient is always used for equalize mechanism dimensions with the measured coordinates in both Mathematica works and algorithm which is applied later. Then, “Manipulate” command is used to graphically create prescribed canopy locations and the complete mechanism visual at the same time. In addition, a line is added also here to see movement path of mechanism’s P point. The command helps to create bars of the input values, so all of the values can change and the mechanism can work actively with this command. The mechanism with added tracking path and bars can be seen in Figure 31.

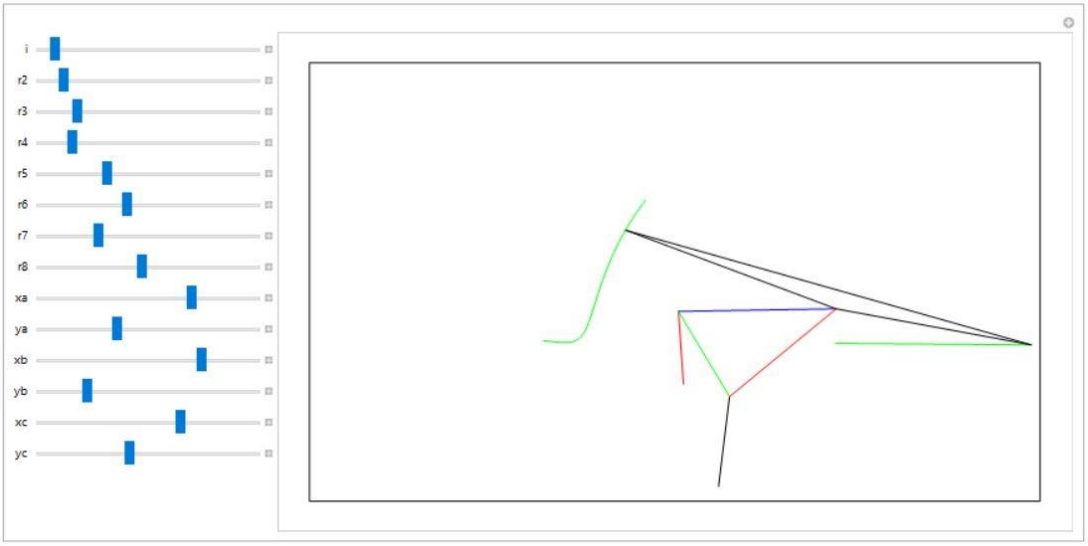

Figure 31 *The six-bar created by "Manipulate" function with active bars.*

By this way, trials can be applied to find best paths that to be reached to the prescribed locations. Using the bars actively change the orientation of mechanism and generates new paths at that moment. In Figure 32 and Figure 33, the applied method can be understood easily. They show, how the same mechanism changes, and how they result the path when change.

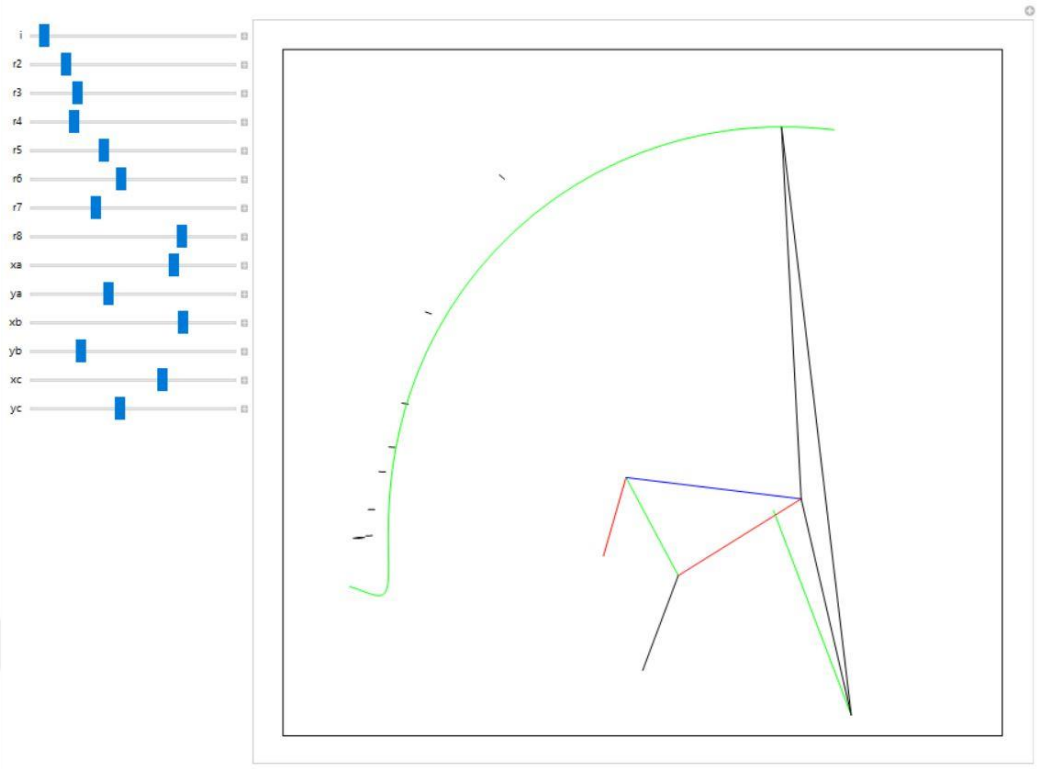

Figure 32 *A roughly created mechanism and its tracking path.*

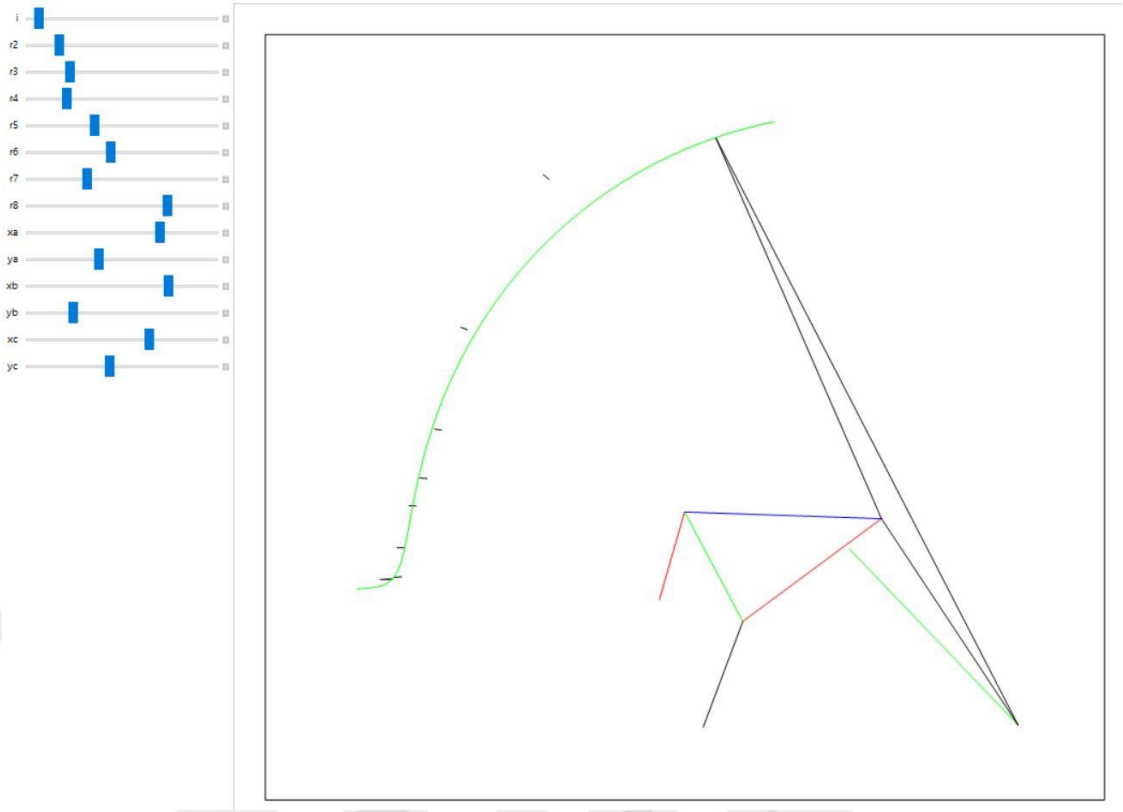

Figure 33 *The six-bar created by "Manipulate" command. By using the bars left, the mechanism's variables are changed and the green path can be approximated to the prescribed poses.*

After applying some trials to find close path to the prescribed 10 points by using the bars manually, the best defined input variables are chosen to be used for optimization starting variables. Table 2 demonstrates the firstly selected values for the algorithm.

Table 2 *The firstly selected initial values for the six-bar optimization. The values are in mm.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	x_A	y_A	x_B	y_B	x_C	y_C
25	34	31	54	67	48	114	78	-5	90	-40	130	9

3.6. Optimization Method

For optimizing the synthesis of the problem, Genetic Algorithm (GA) method is selected. As many researchers stated in their studies that, the discovery of genetic

algorithm dates back some time to 1960s. As Hitesh R. Patel and M.J. Mungla have mentioned [1], this method is suitable for high number of parameters as the 10 points and 10 angles approach has. Moreover, as J.A. Cabrera, A. Simon and M. Prado have stated [2], although it seems like the GA wastes more time than the other methods to generate the algorithm and to find an optimal solution, the computational time of the GA is lower generally. This is because, while GA requires high number of iterations, evaluation of the function is much more complex for deterministic methods. Genetic algorithm can also be used in many diverse areas too rather than mechanism optimizations by means of that kind of advantages.

The GA logic is based on probability distribution of randomly designated values. Therefore, when high number of trials are applied, the higher probability results will remain and it will get closer to the best results as long as the trials continue. Like all living creatures' genetic changes, this optimization technique is based on natural evolution theory.

This algorithm creates a "population" at first, which contains sets of the input variables. The population has "chromosomes" that, each chromosome represents a set of all the input variables to be used for each trial. Each variable in a chromosome is called as "gene". It works with eliminating chromosomes which have low survivability regarding the defined rules in the environment, so that the better chromosomes that give better results continue to survive. As its name signifies that, the genetic algorithm provides to find best solutions by keeping alive individuals that reflect closer results to the desired values in the environment defined. To achieve the desired results, it applies some processes to the population like natural selection, crossover and mutation. In this way, new populations are derived, and the evaluations continue on these.

In this work, a genetic algorithm code is created originally on MATLAB without using a tool for it. As a beginning, a simple algorithm is generated to see progression and to keep the possibility of making a mistake low. The variables and their equations (equations of design parameters) are taken from the Mathematica codes as they have been used in it. The input values to be calculated in the first written code consist of

only components of the connection points to the ground and the link lengths (xa, ya, xb, yb, xc, yc, r2, r3, r4, r5, r6, r7 and r8). Moreover, these values that are chosen in the Mathematica by the manual trials build the populations' initial values. The other values α, β and 10 different values for the input angle Θ12 which have also been determined by Mathematica are given as constants which are fixed, in other words they are not calculated within the first created algorithm. After that, objective values which consist of the prescribed values of the point P's coordinates and angles are added to the code as reference values to be approximated to.

To explain the algorithm's working principle elementarily, it applies the necessary calculations by using the input variables to find the point P's objective values. After acquiring these calculated objective values, it finds out the difference between the calculated values and the corresponding prescribed values. Then, it takes squares of them in order and sums. The algorithm tries to minimize this summed value by doing these processes with natural selection, crossover and mutation for many times. It eliminates the input values that result the farther objective values from the minimum as long as the process continue. Thus, it reveals the input values which gives the more desired objective values.

The algorithm consists of 5 different functions in general. One of them which contains the necessary calculations found by Mathematica is the main (fitness included) function. These calculations are applied by using equations of the design parameters. Additionally, this main function calls the other functions that, one is for finding the objective values and the other three of them are for natural selection, crossover and mutation.

In the main function, some necessary auxiliary values are defined firstly. These include population size and dimension, crossover and mutation probabilities and a coefficient for mutation value. These also include lower and upper limits for creating random numbers to be used for both derivation of new populations and the mutation function. In the function, the prescribed points' poses of x, y and Φ are also defined as a main objective in a matrix form of one row and 30 columns for 10 poses. After that, constant values of α, β and 10 different values of the input angle Θ12 are added. Then, the

algorithm generates random numbers to create the first population. These random values are added to the population's initial values that are defined in Mathematica, so that the input variables' initial values get formed. For the first trials of the algorithm, it is allowed to be changed of population by crossover and mutation directly without requirement of addition to the initial values after the initial population is created. In this way, the populations change again and again via processes of the natural selection, crossover and mutation.

The created population is in the form of a matrix whose rows indicate population size, and columns indicate population dimension. Each row which indicates a chromosome consists of a set of input variables. Each chromosome is limited to count of the input variables, in other words count of genes, which can also be called as population dimension. The algorithm should create many populations to be used for the calculations, and to be compared to find betters.

After input variables are calculated, the design parameters are found by using the defined equations for each input set, and also for all the 10 poses of each input set. These are calculated in two loops, one is for each input set, the other is for 10 different poses for an input set. Here, error functions are firstly added as "if" loops. They check that if the calculated design parameters cause an error in the mechanism or not. If an error occurs, the commands provide assigning new input values instead of the error causing input variables by breaking the loop. Then, it resumes to the loop from the failed order with the new input set. Then, the algorithm starts the iteration loop. In the first algorithm trial, iteration value is selected as 50.

First of all, the objective finding function is called to find objective values in the iteration loop. This function utilizes the calculated design parameters, and find x_p and y_p for the 10 different poses. It also calculates the angle values Φ for the 10 poses by taking difference of the 10 values of θ_{17} from its 1st value one by one. Since the canopy is thought to be assembled to the mechanism from the ternary link PFG rigidly, the change of Φ values is same as change of θ_{17} values. Nonetheless, the angle between the canopy angle and the ternary link won't be same. This value is found after the best optimization values are found. That's why θ_{17} are used to find Φ values. The first

pose's angle values of both desired and calculated values are chosen as 0°. Therefore, when first value of θ_{17} is subtracted from itself, the result is 0. When the other values of θ_{17} are subtracted from its first value, changes of the angles can be found. After the values of x_p, y_p and Φ are found, the function results these in a matrix form with one row and 30 columns for these found values in order, so that the objective matrix is created with these calculated values.

In the iteration loop, an objective difference matrix is created by subtracting the calculated objective values from the main objective values. The values of this matrix represent each output values' difference from the target values in same order. By using the objective difference matrix, an expectancy matrix is created. This matrix takes squares of objective difference matrix's each value, and sums values of the rows separately to create rows of itself. With the summed rows, the expectancy matrix is formed as population sized rows and one column. Thus, the total errors of each chromosome are calculated in the expectancy matrix.

A control activity with if loop is added then. It checks if minimum value of expectancy matrix is lower than the best value found up to this point. If the condition is met, the best value is changed to the minimum value of expectancy matrix. Previously, the best value has been defined as 1,000,000,000. Therefore, for the first iteration, the best value changes most probably. The control activity also keeps the best results that gives the minimum value if the condition is met. This is done by selecting the same index (row) of the calculated objective matrix, because in the algorithm, all the indexes are adjusted with their relations. This means that, an index of the population matrix creates same index of input variables matrix, and same index of the input variables gives the same index of objective matrix and the respective index of expectancy matrix. If the condition is met, the control activity lastly keeps the best input set that provides the best results, by also selecting the same index of the population matrix.

After the control activity, an incremental objective iteration matrix is generated to record the best value of the relative iteration. As the iteration increases, magnitude of this matrix increases too, so that decreases regarding the iteration number can be seen graphically by plotting after the calculation is finished completely.

The natural selection function is called then. For the natural selection, “roulette wheel selection” method is used in the algorithm. This method finds the selection probabilities of all the individuals, and provides selection of them for the next population regarding magnitudes of their probabilities. The name of the method comes from similarity to the roulette wheels in casinos. The probabilities are put in some portion of the wheel like a pie chart regarding their magnitudes. When the wheel is revolved and stopped, the selector is on one of the portions. This portion is selected, so higher probabilities have more chance to be selected. This function firstly takes reciprocals of values of the expectancy matrix, i.e. it takes inverse of the expectancy matrix. The sum of the inversed expectancy matrix’s values is found. A probability matrix is generated by dividing all of the inversed expectancy matrix’s values to the summed value. Thus, probabilities are between 0 and 1. After that, a cumulative probability matrix is created in a loop. This includes cumulative values of the probabilities from first probability to the last incrementally. For instance, first cumulative probability value is equal to the first probability value, second cumulative probability value is equals to sum of first and second probability values. The highest cumulative possibility value becomes 1 then. Next, a random number matrix with population size rows and one column is generated. This matrix's values are between 0 and 1. In a loop, these random numbers are checked from first row to the last, to find where it is firstly lower than values of the cumulative probability matrix. This is because, the values of cumulative probability matrix increase from first row to the last, and once a random number falls behind one of the cumulative probability matrix’s value, it also falls behind the other values of the cumulative probability matrix as row index increases. These found values represent indexes (row numbers) of selected probability matrix, and its source population matrix's indexes as well. When an index number is found in the loop, this is used to equalize new population’s loop relative row to the source population’s this index numbered row. From first row of the random number matrix to the last, this process is repeated. By this way, the new population is generated. In this function, some of the individuals are killed, and some of them are used more than one times in the new population. It is obvious that individuals that give the best results cannot be always transferred to the new generation. Nevertheless, as

iteration number increases, best individuals with best results comes to light just as in the nature.

The next function is crossover that is called by the main function. This function firstly creates a row matrix containing a random permutation of the integers from 1 to population size magnitude without repeating elements. Then, the function works in a loop created that runs half of the population size times. Two parent index values are generated by using the created row matrix. One of them takes even values of the row matrix while the other one odd values. It is provided in twos in one iteration of the loop in order. To exemplify, in the first iteration of the loop, first parent index value equals to first value of the row matrix, the second parent index value equals to second value of the row matrix. For the second iteration, the first parent index value is equals to third value, the second parent index value is equals to fourth value and it goes on like this up to last iteration. After that, two parents are created by taking population's rows (chromosomes) in a similar way. The parent index values determine the population's row indexes to be chosen for the two parents. Then, a random number is generated to be used for an if loop that checks whether the number is lower than the crossover probability value defined in the main function or not. If the condition is met, a crossover point number is defined as integer randomly in the range of population dimension. This number determines column number of the genes that are intercrossed after. In this way, new chromosomes of the population are generated by changing their column values (genes), and new population is generated too. Due to existence of this process, the population size should always be even.

The last function is mutation. In the function, a random numbered matrix with population size rows and population dimension column is created between values of 0 and 1. After that, all values of the matrix are checked in order by an if loop whether the values is lower than the mutation probability value or not. If the condition is met, a random number is generated between -1 and 1. With this number, range value between the lower limit and upper limit and mutation coefficient which are defined in the main function are multiplied. Finally, this multiplied value and the selected gene of the population are summed. The selected genes are chosen by using the random

numbered matrix’s row and column indexes that provide the if condition. Thus, new population is generated as well as new genes are created.

After new populations are generated by the functions of crossover and mutation, some checking procedures must be implemented to detect whether one of the individuals causes singularity that prevent the mechanism’s working or not. The implementation of these procedures is provided by loops that start to check from design parameter results of first row of the population matrix to the last's results. In other words, design parameters that are created by all the individuals with genes that represent different set of inputs are checked to find out errors if exist. Firstly, the design parameters are calculated at this stage too like done at the creating of first population. Secondly, some parts of design parameter equations whose results must be in their limits are checked by using if loops. These processes start from the first input set with its 10 different poses, and end at the last input set in the same way. If the result cannot be obtained as required even for one pose of the one input set, the population is sent to crossover and mutation functions respectively to generate a new population. Then, the checking loop is resumed on. A flowchart is created to represent procedures up to here, and can be seen in Figure 34.

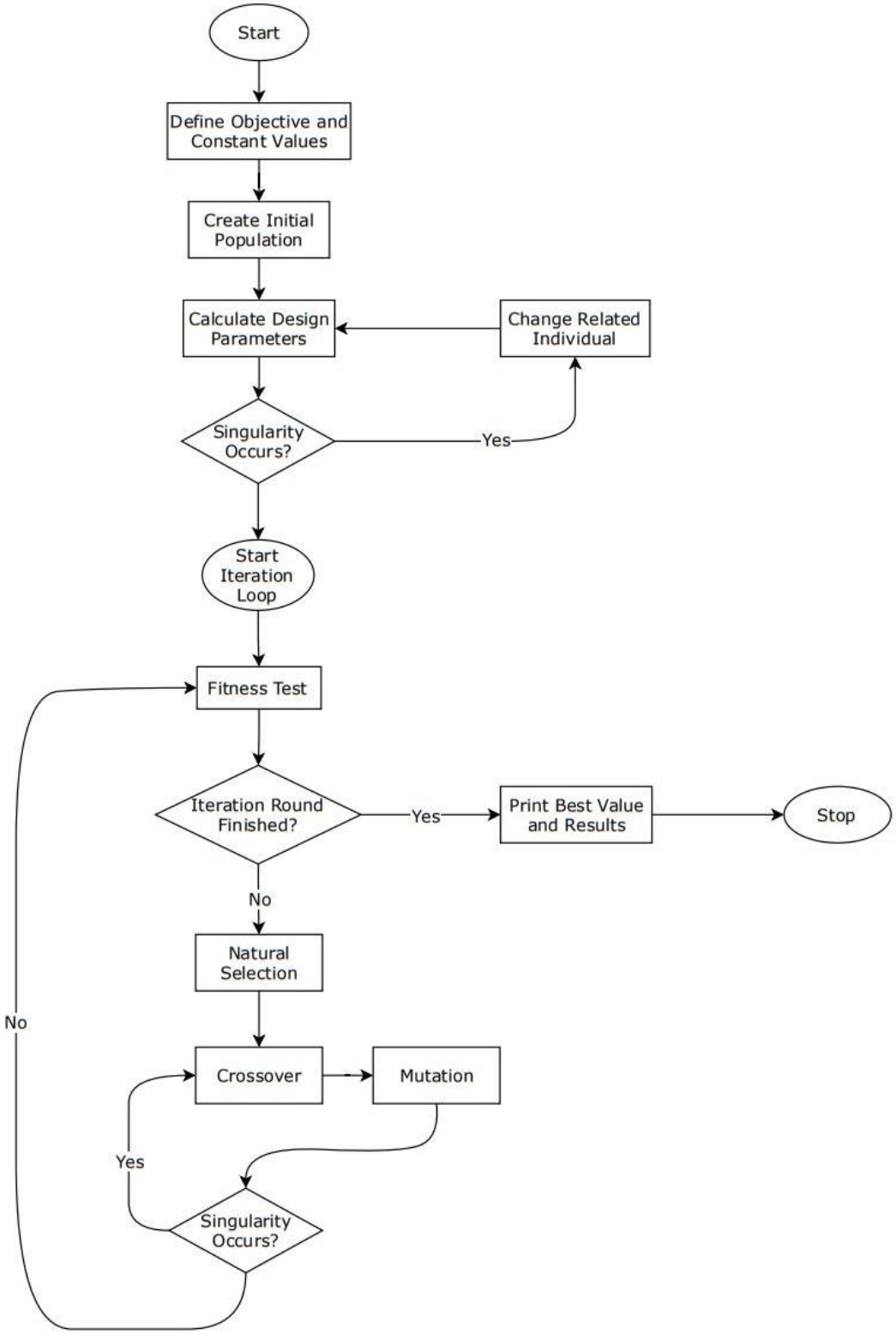

Figure 34 *Initial form of the algorithm, flowchart 1.*

For the next stage, the errorless values found up to the errored one observed are kept to be used for objective finding function in the next iteration of the main function. Thus, the errorless resulted input design parameters are not eliminated unnecessarily. This helps to decrease the computational time of the algorithm. The checking process continues up to last row's design parameters that are calculated without an error. Finally, the loop is ended, and the main function passes to the next iteration to start the calculations with the objective finding function by using the newly produced population's design parameters.

To mention the objective function, it consists of 30 values. First 10 of them are for x coordinates, second 10 of them y coordinates, and the last 10 of them are the respective angle values. The objective function is created in the algorithm as below:

$$obj = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10},$$
$$\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, \Phi_6, \Phi_7, \Phi_8, \Phi_9, \Phi_{10}]$$

Basically, the firstly created algorithm works as described, and flowchart of it can be seen in Figure 35.

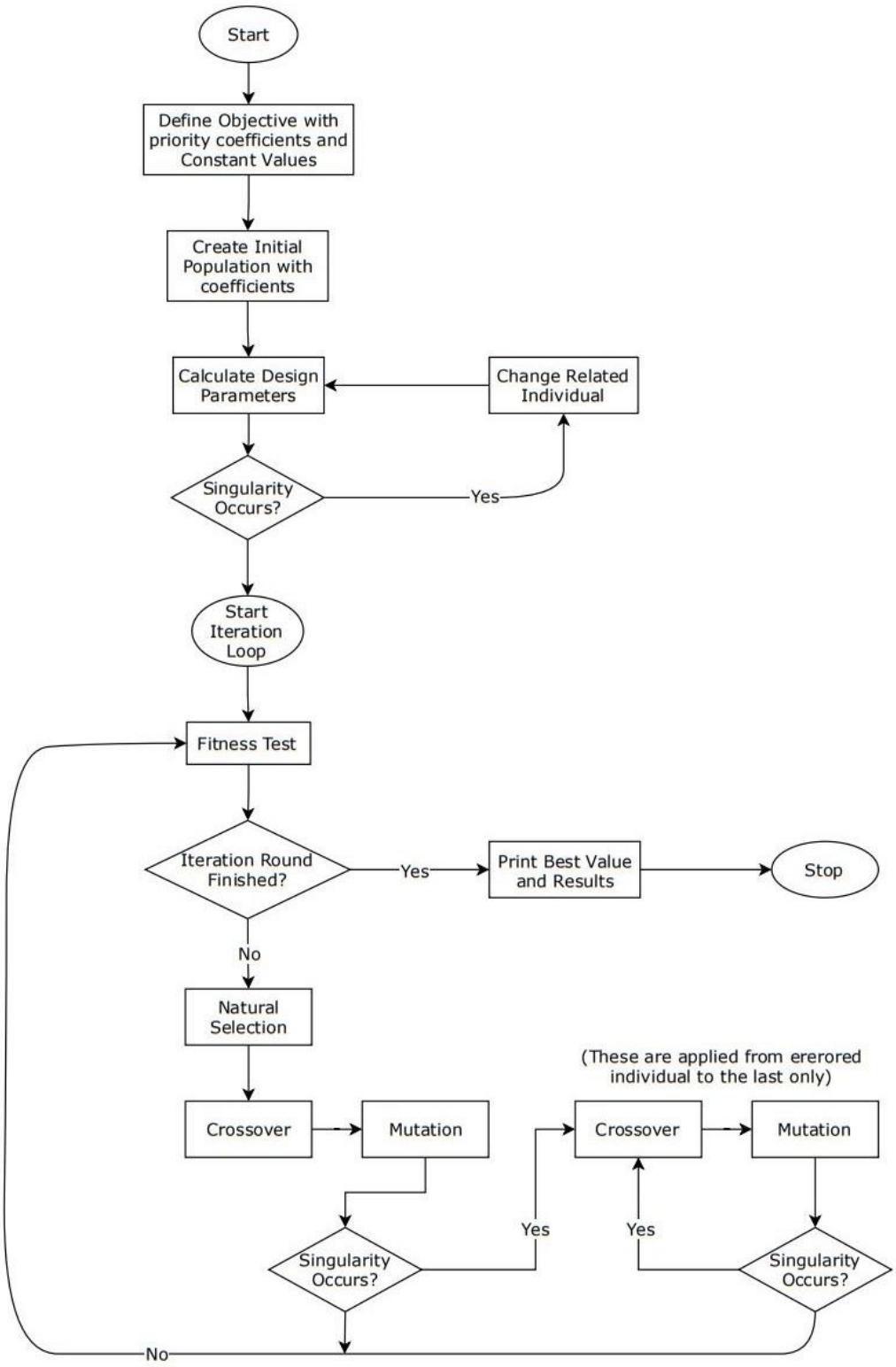

Figure 35 *The flowchart of the created algorithm, Flowchart 2.*

3.7. Trials on the 6-bar

By using the created algorithm, some optimum finding trials are applied for the six-bar. However, even approximate to proper results are not got. It is realized that, when the populations' values which are generated by crossover and mutation are directly used as new population values without adding them to the initial values chosen in the Mathematica at the beginning, the results can go away from the initially approximated mechanism orientation. As a consequence of this, the results of the algorithm can go infinite rather than getting closer results. Of course, this situation is not valid for all situations, and this is inconsistent with nature of the genetic algorithm. However, finding the global optimum via the algorithm directly, which uses the random generated values in the global, requires so much more iterations to be evaluated. This requires an ultrahigh-speed computer and so much time like maybe weeks and months. Therefore, for the starting, some approximative values, which are obtained by hand easily but are not optimum, are used as helper. These values orientate the algorithm to find the optimum near the created mechanism's values. Therefore, the initial values coming from Mathematica are used for the other generated populations as additives.

One more trial in addition to Figure 33 and Table 2 to find link lengths and connection coordinates can be seen in Figure 36 and the found values for algorithm's initial values can be seen in Table 3 as an example.

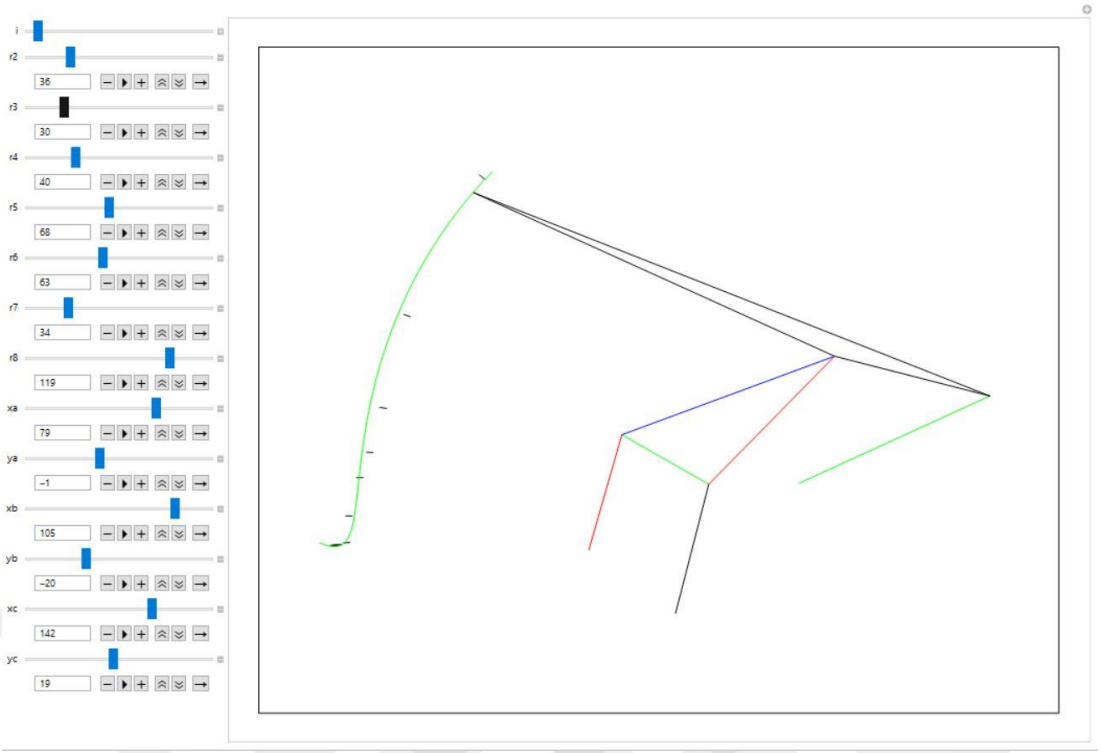

Figure 36 *A Mathematica trial for six-bar with Manipulate to obtain approximate path.*

Table 3 *Found values by a Mathematica trial.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	x_A	y_A	x_B	y_B	x_C	y_C
36	30	40	68	63	34	119	79	-1	105	-20	142	19

After this stage, while generating population in the algorithm, created random values are added to the initial values of population regardless of whether it is first generation or mutation.

The algorithm started with the necessary auxiliary values as in Table 4 for the six-bar linkage.

Table 4 *First optimization trial's auxiliary value set.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-2	2	4	13	0,095	0,05	0,05	50

After some trials, it is experienced how the calculation can be modified so that better results are obtained. One of the initial experiences is that, so much chromosomes have to be produced to approximate better results, and lower and upper limits should be minimized so that the results does not diverge. Hence, a new set of auxiliary value inputs are chosen as seen in Table 5.

Table 5 *A new set of auxiliary value inputs.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-1	1	1024	13	0,095	0,05	0,05	50

There is no close solution obtained with these exercises. After many trials, it is decided to use the angles α and β, which are defined as constants previously, as input variables. For these values, population dimension is increased two more. Figure 37 and Table 6 demonstrates one trial applied in Mathematica after addition of the two variables as input variables, and all the input variables obtained with this orientation.

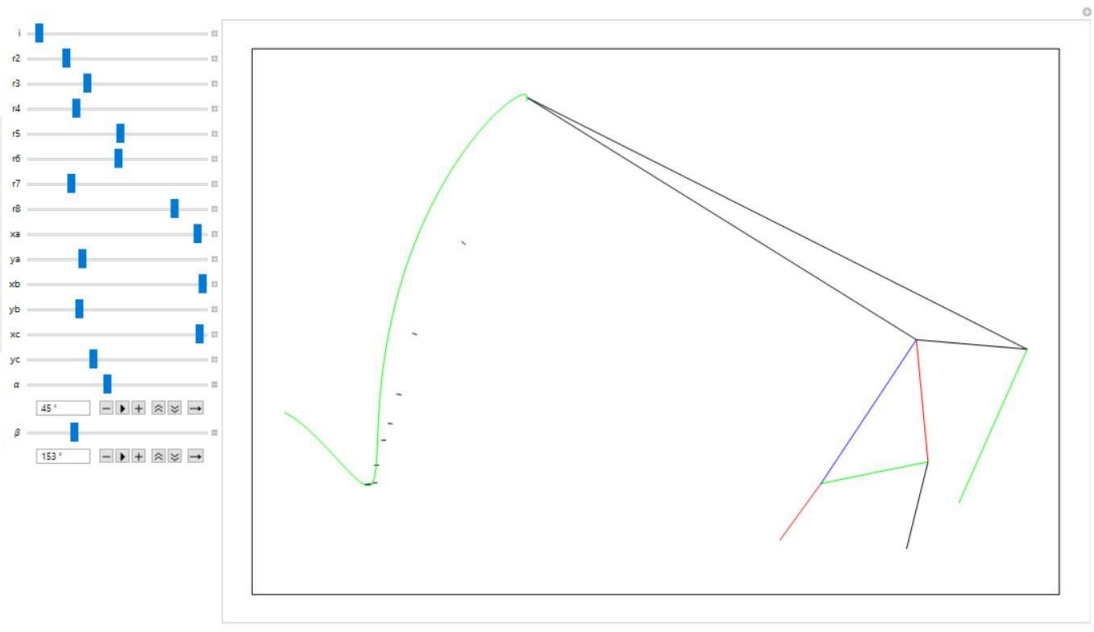

Figure 37 *Mathematica trial after addition of alpha and beta angles.*

Table 6 *After addition alpha and beta, found input values in Mathematica.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	x_A	y_A	x_B	y_B	x_C	y_C	α	β
32	50	41	79	77	36	210	192	-25	250	-29	274	-8	45	53

Their firstly defined values are used to be added to the populations' generated new two values as it is for the link lengths and pose components. Moreover, the input angle Θ12's 10 different values for the 10 poses, which are defined as constants previously too, are defined as input variables, so the population dimension is increased ten more by the input angle variables. Unlike the other input variables, a different matrix is created for the input angle values with 10 values. The first value of this matrix expresses the initial value of the input angle for the first pose which is defined before on the Mathematica. The other 9 values of this matrix express the angle differences of them from the previous angle value of each other for the other 9 poses. Like the other input variables' requirement of being calculated after each population is generated, a loop is created to calculate these input angles separately to be used for the design parameter calculations. When a new population is generated, the input angle loop takes the relative 10 values and sum the first pose value with the matrix's first value. Then, the other values are calculated by adding both populations' generated values and the matrix's other values in order. Table 7 shows an example of selected values of the input angle matrix for one trial.

Table 7 *For one trial, selected 10 different values of input angle.*

θ_{12}^{1}	θ_{12}^{2}	θ_{12}^{3}	θ_{12}^{4}	θ_{12}^{5}	θ_{12}^{6}	θ_{12}^{7}	θ_{12}^{8}	θ_{12}^{8}	θ_{12}^{10}
37	3	3	6	16	10	5	7	13	28

In addition, a coefficient value for the initial input angles matrix, and a coefficient matrix for the other components of the input variables are created. With these coefficients, ranges of change regarding lower and upper limits for the input angles and the other input variables can be controlled. To define the importance difference of the 10 poses' values, an importance matrix is also created. This matrix is multiplied

with the objective difference matrix that is used for the defining of expectancy matrix's values, so that it can be provided that the critical poses can have priority. All of these changes are applied to give the algorithm more flexibility to find optimum. The final state of the auxiliary input values is changed also to get better results, and can be seen in Table 8.

Table 8 *Auxiliary input value set with higher population size.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-1	1	1024	25	0,5	0,05	0,05	500

Up to this stage, a lot of trials with different auxiliary input sets, with positional input angle matrix instead of incremental one, and also with taking absolutes of population values generated for input angle values have been applied to get better results. A couple of poses have been provided, but it is understood that an approximate solution for the desired complete motion cannot be obtained with this mechanism. After deciding the 6-bar mechanism with this orientation cannot satisfy the desired motion, a new mechanism conformation is thought.

3.8. A Particular Second Mechanism Approach – Geared Seven-Bar

As the previous study fails to produce the desired result, it is necessitated that the mechanism should have more movement capability for the point P. Therefore, it is decided to develop the previous mechanism to have the capability of producing effectual motion.

To develop the mechanism in such a way that having more flexible movement ability by controlling input parameters, one more link is added between the Link 6 and ground point C. This new link is named as “Link 9” and its angle with respect to the ground is Θ19. Thus, the connection between points G and C is provided by two different links instead of one. This gives the point G freer movement capability rather that pivoting around one revolute joint as it previously does. Course of this movement changes

based on regarding orientations of the links 6 and 9, which also affect the movement of point P. This change helps to control the movement with one more aspect, in other words it makes dependent on one parameter more that changes the movement distinctively.

In this way, the changed mechanism has two degrees of freedom. However, for the design problem, it should be one to have ability of being controlled by one actuator only. Therefore, it is decided to make the movement of added link with number of 9 dependent to movement of Link 2. To do this, a couple of solutions are considered, and one has predominated over the others.

The selected solution to decrease the degree of freedom is integrating a gear set between links 2 and 9. This gear set makes the Link 9 dependent to Link 2. At least two gears are required for the gear set, because the links 2 and 9 need to be connected to the two gears separately. The gears' pivot points are located at points A and C, and when the links are connected to the gears, they rotate together around these points. In this form, the geared 7-bar mechanism can be seen in Figure 38.

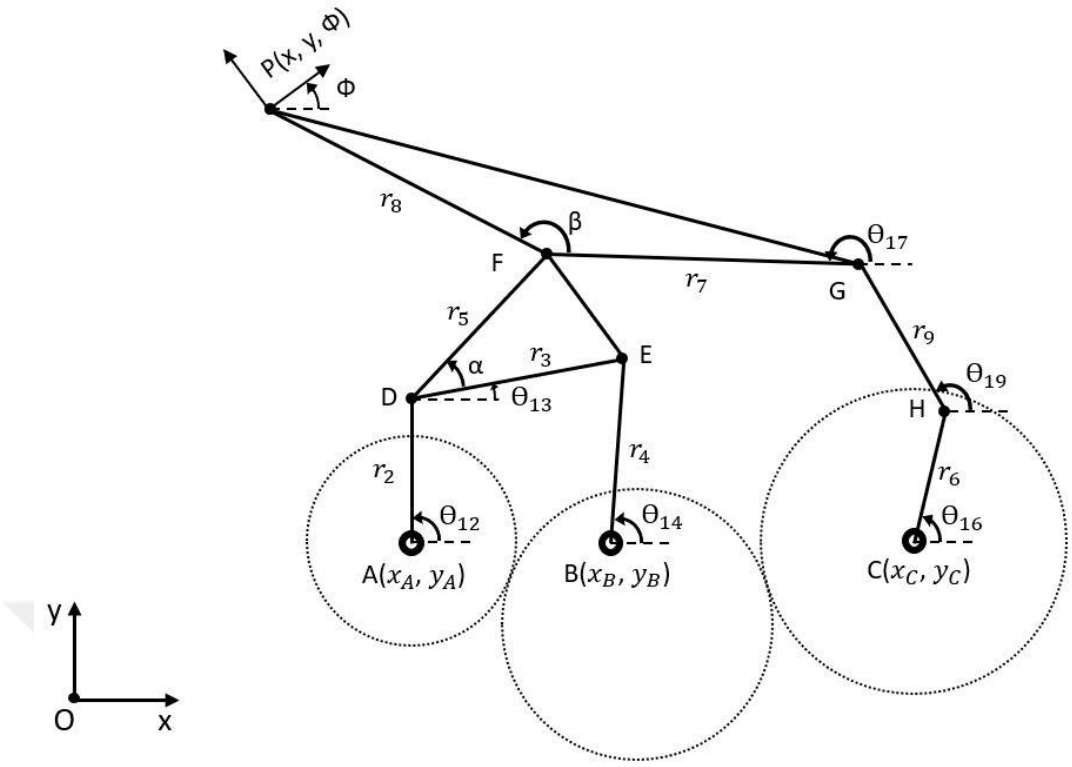

Figure 38 *Free Body Diagram (FBD) of geared seven-bar mechanism.*

The gear set can consist of only two gears or can have more, which does not matter. A pulley-belt set can be implemented also to have the same function. The important thing is ratio between input and output of it. For this purpose, the gear set is mentioned as "gear" only as gear ratio is important. With this procedure, two more input variables are needed to be added to the algorithm. One of them is length of Link 9, the other one is the gear ratio. Mathematica program is used again to find the new mechanism's design parameters which are to be added to Matlab afterwards. The new mechanism's loop closure equations can be seen below:

LCE 1: $\overrightarrow{AD} + \overrightarrow{DF} = \overrightarrow{AC} + \overrightarrow{CH} + \overrightarrow{HG} + \overrightarrow{GF}$

$\overrightarrow{CH} = r_6.e^{i\theta_{16}} = r_6.cos\theta_{16} + r_6.isin\theta_{16}$

$\overrightarrow{HG} = r_9.e^{i\theta_{19}} = r_9.cos\theta_{19} + r_9.isin\theta_{19}$

The LCE 2 and LCE 3 are same, because addition of the gear set and new link don't affect them. The gear ratio is defined as Z, and its equation regarding the affecting parameters is created as below:

$$\theta_{16} = \theta_{12} . Z$$

The new mechanism's kinematic equations found via Mathematica can be seen below:

$$A2 = -2.r_9.\left(x_A - x_C + r_2.cos\theta_{12} - r_6.cos\theta_{16} + r_5.cos(\theta_{13} + \alpha)\right)$$

$$B2 = -2.r_9.\left(y_A - y_C + r_2.sin\theta_{12} - r_6.sin\theta_{16} + r_5.sin(\theta_{13} + \alpha)\right)$$

$$\begin{aligned} C2 = -[&{r_2}^2 + {r_5}^2 + {r_6}^2 - {r_7}^2 + {r_9}^2 + {x_A}^2 - 2.x_A.x_C + {x_C}^2 + {y_A}^2 \\ &- 2.y_A.y_C + {y_C}^2 + 2.r_2.(x_A - x_C).cos\theta_{12} \\ &- 2.r_6.(x_A - x_C).cos\theta_{16} - 2.r_2.r_6.cos(\theta_{12} - \theta_{16}) \\ &+ 2.r_5.x_A.cos(\theta_{13} + \alpha) - 2.r_5.x_C.cos(\theta_{13} + \alpha) \\ &+ 2.r_2.r_5.cos(\alpha - \theta_{12} + \theta_{13}) - 2.r_5.r_6.cos(\alpha - \theta_{16} + \theta_{13}) \\ &+ 2.r_2.y_A.sin\theta_{12} - 2.r_2.y_C.sin\theta_{12} - 2.r_6.y_A.sin\theta_{16} \\ &+ 2.r_6.y_C.sin\theta_{16} + 2.r_5.y_A.sin(\theta_{13} + \alpha) \\ &- 2.r_5.y_C.sin(\theta_{13} + \alpha)] \end{aligned}$$

Similar to six-bar's derived calculations, Θ19 and Θ17 are found respectively:

$$\theta_{19} = Arctan\left(\frac{B2}{A2}\right) + Arccos\left(\frac{C2}{\sqrt{A2^2 + B2^2}}\right)$$

$$\theta_{17} = Arctan\left(\frac{\frac{r_2.sin\theta_{12} + r_5.sin(\theta_{13}+\alpha) - (y_C - y_A) - r_6.sin\theta_{16} - r_9.sin\theta_{19}}{r_7}}{\frac{r_2.cos\theta_{12} + r_5.cos(\theta_{13}+\alpha) - (x_C - x_A) - r_6.cos\theta_{16} - r_9.cos\theta_{19}}{r_7}}\right)$$

In Mathematica, applying some trials manually to find close path to the prescribed 10 points is needed again. Addition of more input variables to the algorithm requires more time for calculation and makes finding optimum harder. Therefore, it is tried to obtain quite close result during the manual trials in Mathematica firstly. Finally, an approximate solution is found in Mathematica manually after some trials, which can be seen in Figure 39, and the defined input variables which represent initial values for the Matlab code can be seen in the Table 9.

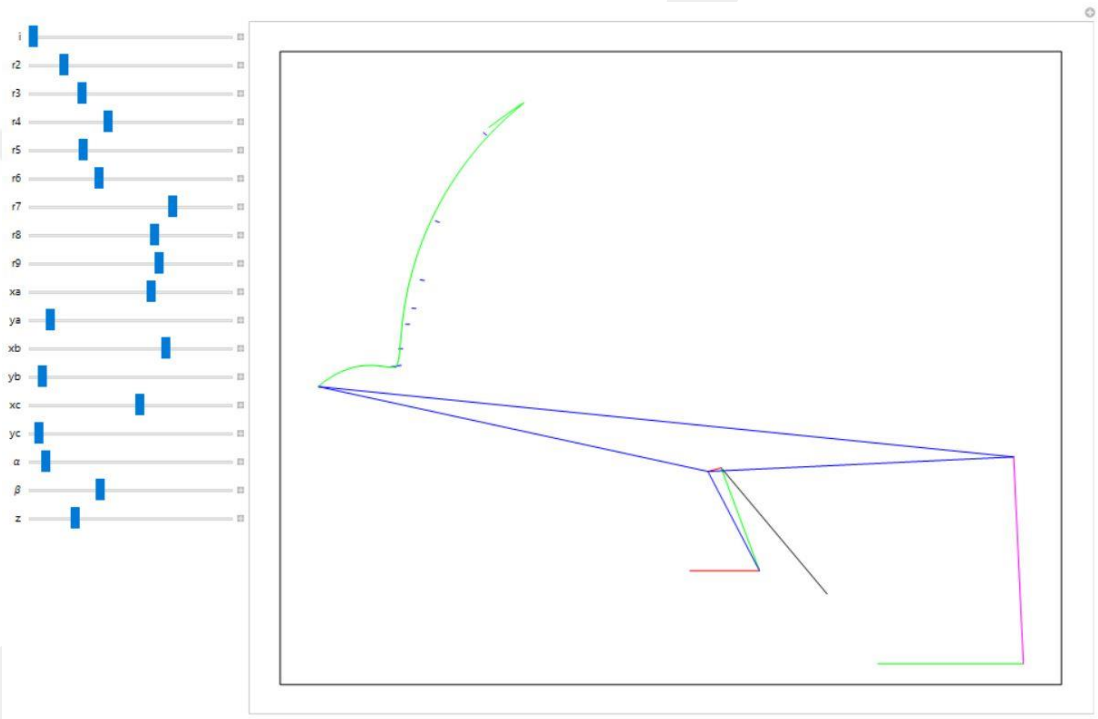

Figure 39 *First found 9-bar mechanism with approximated path after some trials.*

Table 9 *Input values of the first found 9-bar mechanism.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	x_A	y_A	x_B	y_B	x_C	y_C	α	β	Z
33	52	78	53	69	145	189	98	144	-96	209	-107	233	-140	7	165	1.1

3.9. Optimization Trials

The optimization process is started by using the same algorithm, which has been created for the six-bar linkage before, with some basic changes. To explain these changes, the new input parameters are added, and design parameters are changed to be appropriate for the geared seven-bar. The new input parameters consist of length of Link 9, which is r9, and the gear ratio Z.

After many trials run in the algorithm, some further changes have been implemented too. Firstly, probability of mutation value is increased to 0.1 from 0.05, because it is observed that more changes are required as iteration continues. Secondly, the error functions with loops are changed with respect to the new mechanism. Next, the mutation coefficient is decreased from 0.05 to 0.01 to limit the difference from its ancestor when a gene is exposed to mutation. Then, the objective importance matrix is updated so that the coefficients are closer to each other, because when some poses are made extremely important, the algorithm focuses only those points. With these changes, the auxiliary value set becomes like seen in Table 10.

Table 10 *Auxiliary input value set after some trials for 9-bar.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-1	1	64	27	0,5	0,1	0,01	50,000

There are also applied some simple changes like about input angles' coefficients, other input variables' coefficients, lower and upper limits etc. as the trials continue. Finally, an important thing is discovered about population size – iteration size equilibrium.

In the beginning, the population size has been tried to be chosen as much as higher, because it is thought that if higher number of populations, which means high number of input variable sets, created to be calculated and used for the trials, the algorithm would find closer results to the objective, and this would decrease the total time needed for finding the optimum. However, the fact of the matter is that, obtaining different solutions by using the functions of natural selection, crossover and mutation is more helpful rather than creating many populations in the beginning. There are two main

reasons for that. Firstly, when the algorithm applies natural selection, the closer results are saved, so better individuals survive, and algorithm continues to change genes of the betters progressively while with higher population size, many unrelated individuals are created. This process continues as much as the iteration number. The second reason is necessity of eliminating effective individuals too when one individual causes an error in the population. When the population size is higher, these processes of elimination of errored population and creating a new population from the beginning wastes more time while this process is so short for low population sizes. The advantage of decreasing population size and increasing iteration loop is found out when it is started to see better solutions by increasing iteration number quite a lot. After this important benefit is discovered, the trials are focused on decreasing population size and increasing the iterations.

With first 4 poses of the 10 prescribed poses the algorithm is tried firstly to see benefit of the discovered run condition. The population size is selected as 64, and iteration number is increased to 50,000. The result which needs to be minimized decreases to below 1. The auxiliary input set of this trial can be seen in Table 11.

Table 11 *Trial with high number of iterations for 4 poses.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-20	20	128	21	0,5	0,1	0,02	50,000

By this way, the calculation of the algorithm is finished faster, and obtained result is satisfactory when the found values are transferred to code of Mathematica. The path obtained with these values can be seen in Figure 40, which shows the first 4 poses are approximated. Besides, success of this trial's points out accuracy of the algorithm.

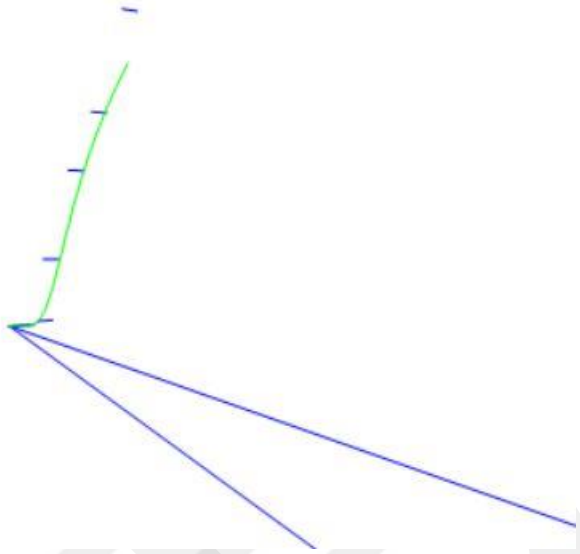

Figure 40 *Approximation to 4 poses.*

For this trial, the input variable count is decreased to 21 due to decreasing input angle Θ12's 6 values are eliminated. The main objective function and calculated objective function sizes are also be decreased to 12 when the points are decreased to 4. After that, it is started to try this for the 10 prescribed points.

When the algorithm is run in this way for the 10 points, it is observed that the iteration number is not sufficient to find optimum. Therefore, it is increased much more. Firstly, it is tried with 500,000 iterations. Then the trials are applied with 1,000,000 and 2,000,000 iterations too.

During the trials, two problems more have come to light. The first problem is that, sometimes the algorithm keeps calculating everlastingly, which means the program doesn't finish, continues finding new populations and cannot be released from an iteration. When the algorithm starts to change the population via crossover and mutation in the iteration loop after the first population creation, sometimes it falls into undesirable region. The reason of this is that, some individuals of the population generated in the penultimate iteration form mechanisms close to singularity. As a result of this, the evolved individuals can fall into error easily if their parent individuals are

close to tend the error region. In other words, the genes in the individuals can cause errors when they are changed by crossover or mutation, because they verge on error region. These are determined in the error checking loops. If an error occurs in this loop, the algorithm changes the population and continues checking from the errored individual normally as previously mentioned. Sometimes some individuals in an iteration are stuck due to failing of generating a valid individual. It tries so many times that, but the algorithm runs to the infinite.

To solve this problem, an error correction function is created by using an if loop and a counter. The solution found is correction of the individual that causes errors. If the algorithm tries the generating a new individual a defined number of times via crossover and mutation, and if it cannot achieve to get a valid one, the algorithm changes this individual with the best individual found up to that time. Firstly, the counter increases in the error checking loops for each error. The error correction loop is added beginning of the error checking loop. If the counter exceeds the threshold value, the individual equalized to the best individual found for current stage, and the counter is zeroized. Thus, the algorithm can continue to progress.

This function works like Elitism technique which retains the best individual so far. Hence, the algorithm has works like combination of Elitism and Roulette Wheel methods with this modification. After application of this development, the flowchart of the algorithm modified too, and that can be seen in Figure 41.

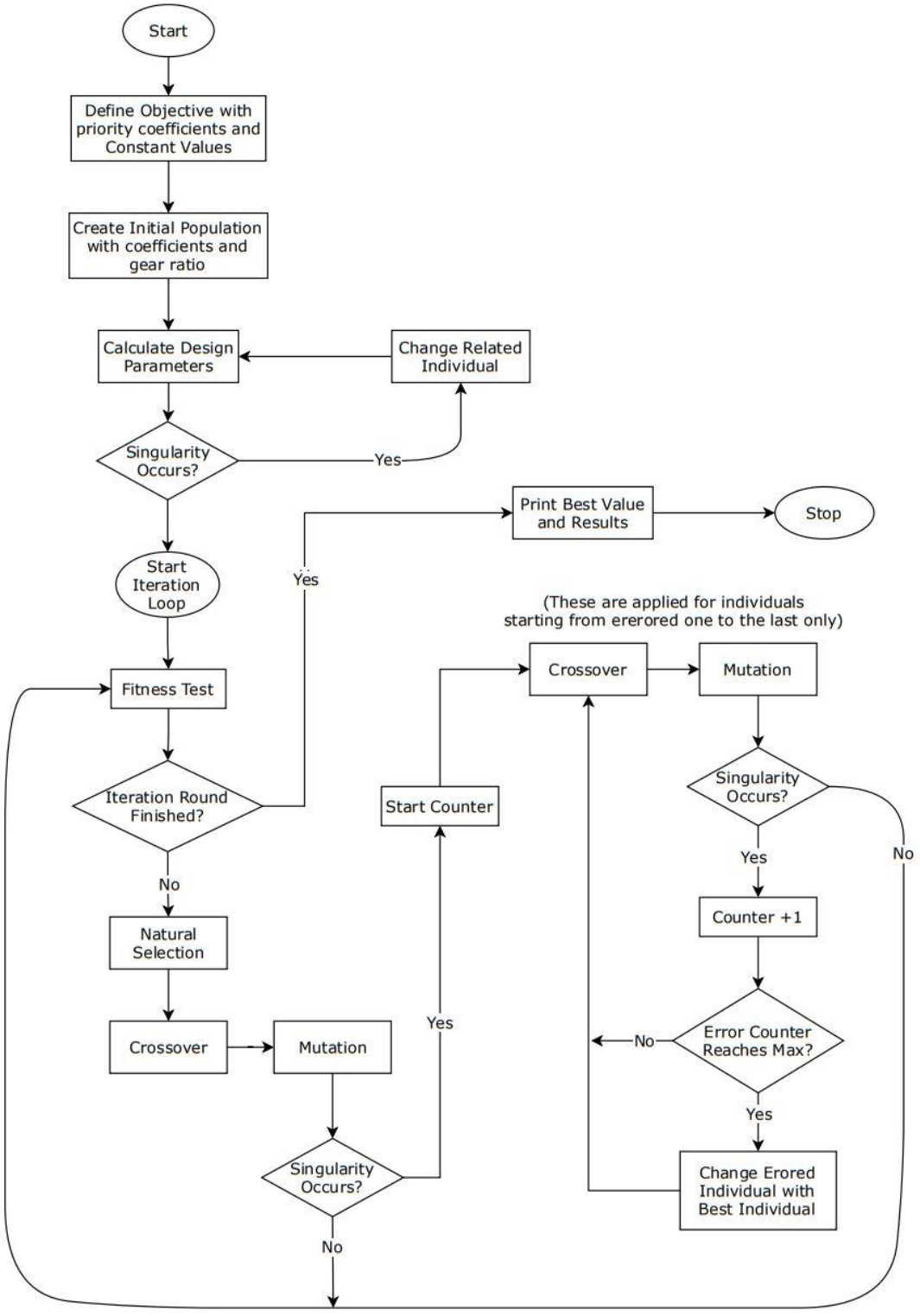

Figure 41 *Algorithm flowchart after addition of elitism technique, Flowchart 3.*

The second problem is losing some individuals which provider of used design parameters if an error occurs after first row in the error checking loop. To illustrate, in an error condition, the population is regenerated regardless of it is coming from mutation or changing with best individual. When the population is regenerated, the algorithm continues to check from error causing individual, in other words it continues from the related row. On the other hand, the design parameters found for individuals before this individual, in other words previous rows' calculated variables stay same to be calculated for the objective when the algorithm continues from the errored one. As mentioned before, if all the individuals are changed in case an error, the proper individuals are erased, and also unnecessary burden comes up for the algorithm to check these rows again.

To solve this second problem, a backup population is generated. This backup population's each row is equalized to the population's related rows in the error checking loop after checking of each iteration is finished properly. By this way, when the population is changed via crossover and mutation as a result of an error, the previously found design parameters' provider individuals are saved in the backup population. At the end, the population is equalized to the backup population before finding the expectancy matrix, the best input set and so on. This is because that kind of functions in the algorithm recognize population rather than backup population. Thus and so, the algorithm reaches to its ultimate form.

Further, it can be notified that, when the mechanism is changed, all the required changes are applied at the main function. The objective finding, natural selection, crossover and mutation functions do not need to be changed. This is because, the modifications of the mechanism and algorithm does not affect these general functions.

As an extra, one more step further is tried with the algorithm. The all calculations are tried to repeated for a defined iteration also. The repetitive iterations' initial values are chosen from the found values of the previous iterations. The flowchart shows this modification in Figure 42.

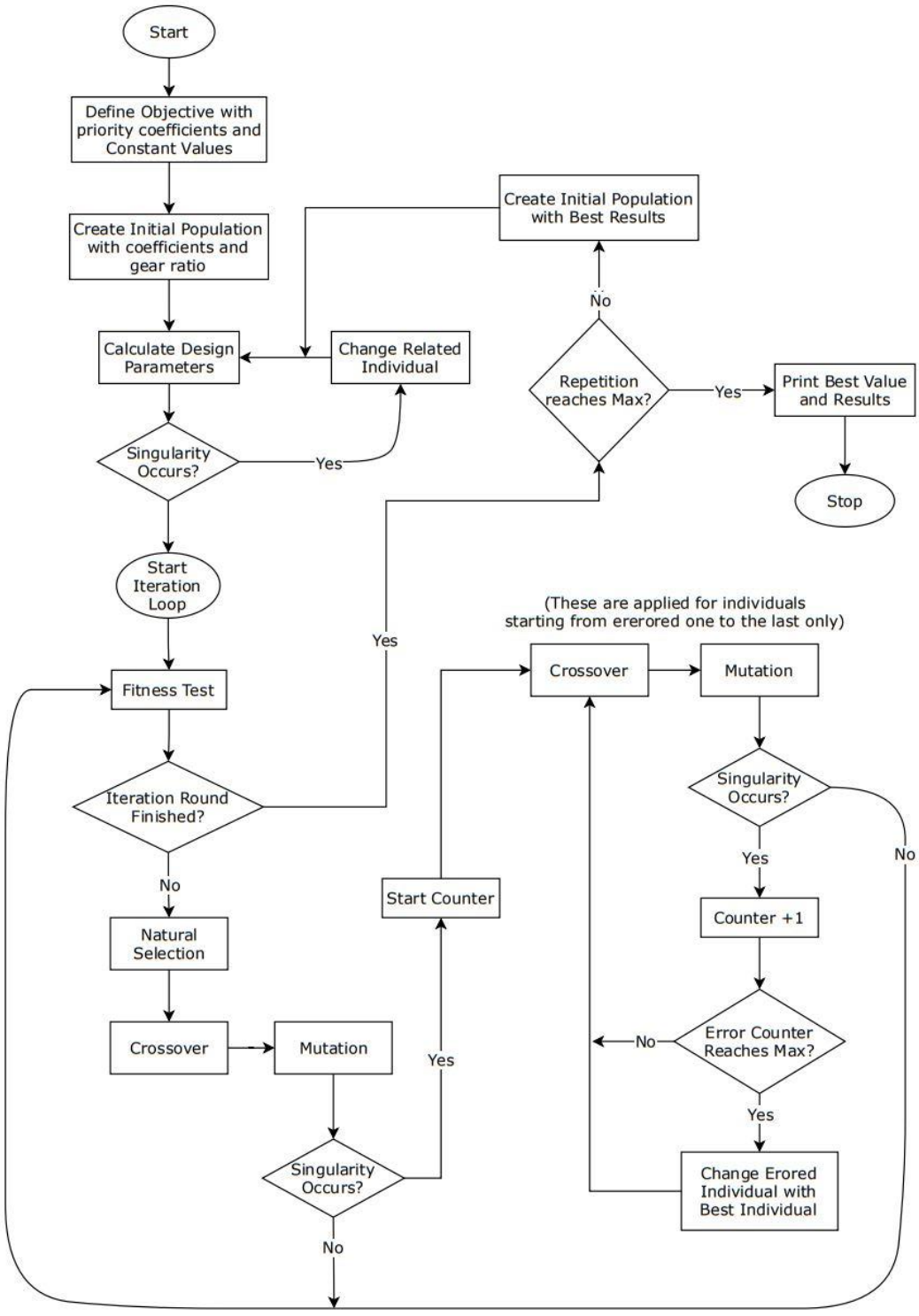

Figure 42 *Algorithm flowchart of repetitive calculation, Flowchart 4.*

Although it is thought that this development of the algorithm can provide better results, this method moves the results from desired values. Therefore, this method is given up, and the optimization process is continued as presented in 3th flowchart.

3.10. Trials on the Seven-Bar

With the final created algorithm, some close results are obtained. The trials are applied with changing the auxiliary values of the algorithm. To show examples of these trials, the population size is decreased up to 32 to decrease calculation time and these variables can be seen in Table 12. However, an acceptable result cannot be obtained via this auxiliary input set.

Table 12 *Auxiliary value set for trial applied for geared seven-bar.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-1	1	32	27	0,5	0,1	0,01	50,000

As Khalid Nafees and Aas Mohammad mentioned in their article [13], the population size is considered 2 to 4 times of the variable count generally, the population size with 32 cannot satisfy the desired results. Therefore, the trials are implemented with high number of population size afterwards. Another trial's auxiliary input seen can also be seen in Table 13.

Table 13 *Another auxiliary value set for trial applied for geared seven-bar.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-10	10	256	27	0,5	0,1	0,02	2,000,000

Here, some applied trials' results are added to the Mathematica to see how the found mechanisms look like. The some resulted mechanisms obtained by these trials can be seen in Figure 43 and Figure 44.

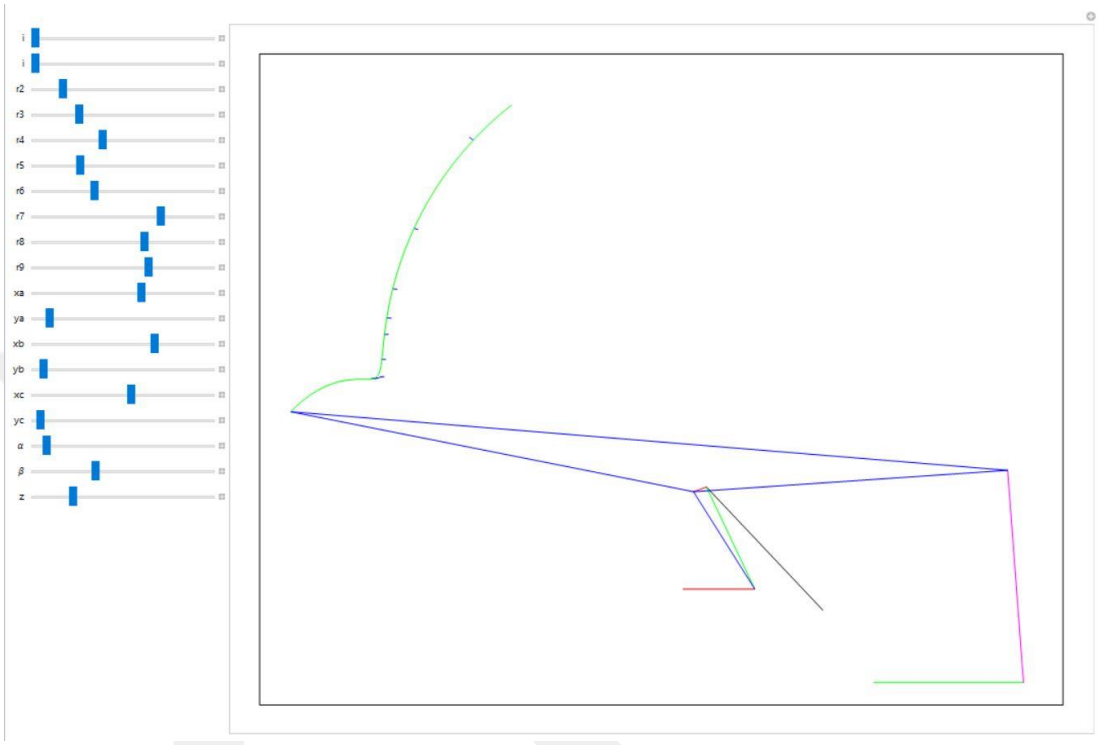

Figure 43 *Optimization resulted geared seven-bar mechanism of a trial.*

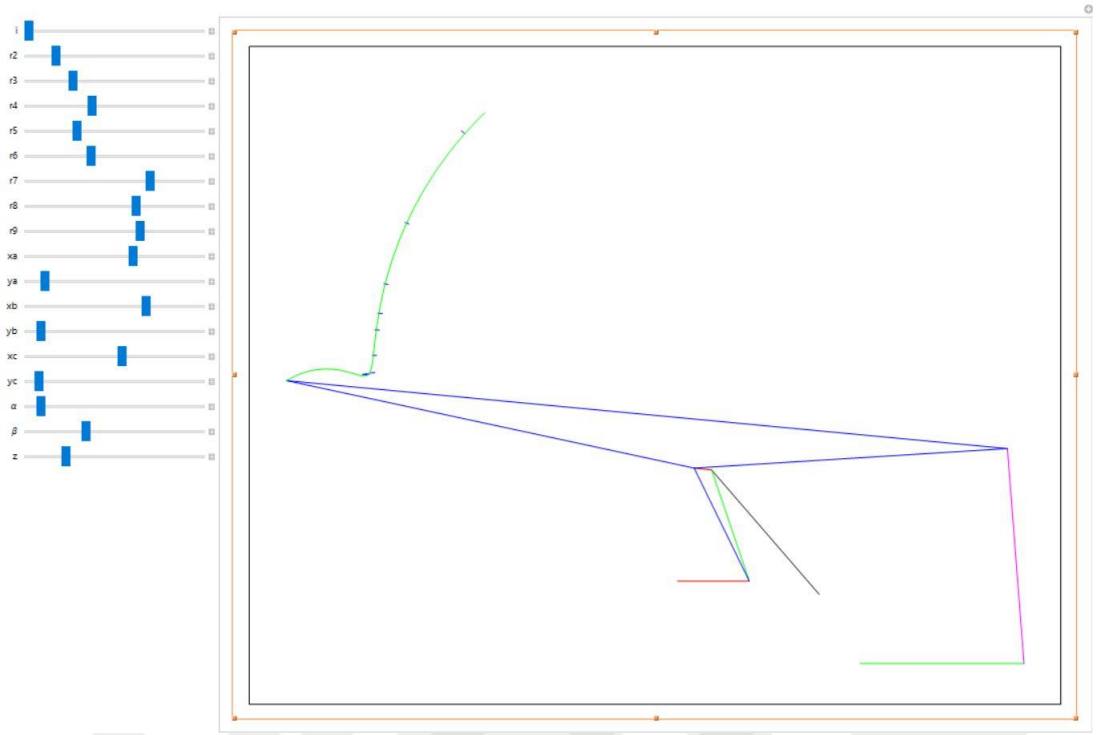

Figure 44 *Optimization resulted geared seven-bar mechanism of another trial.*

After a couple of trials applied, and after the mechanism orientation is changed a little bit, a close solution to the desired solution could be obtained. This obtained mechanism representation can be seen in Figure 45.

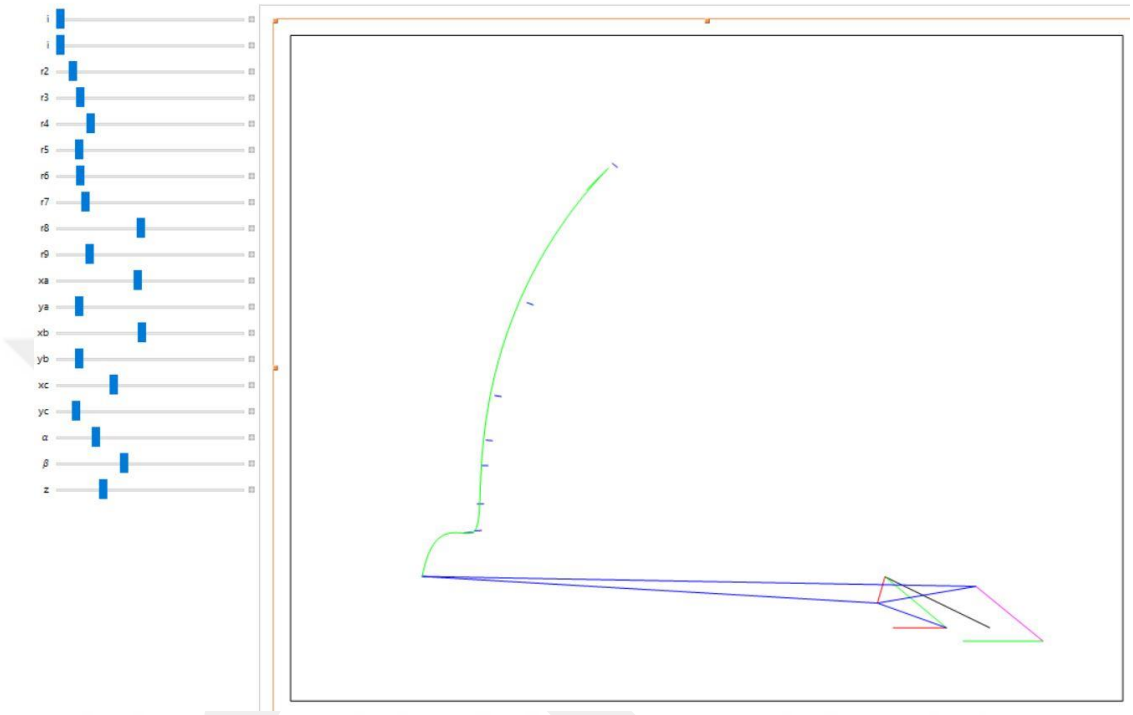

Figure 45 *A close result obtained by geared seven-bar.*

CHAPTER 4

RESULTS AND DISCUSSION

In this work, a lot of trials have been applied to find a good solution for the described problem. These trials include tests of six-bar which is Stephenson III type firstly, and a geared seven-bar that has been modified from the six-bar later. Despite the fact that the six-bar mechanism could not be successful to get close findings for the desired solution, it has been achieved to get close results by the geared seven-bar. The algorithm and the code used in this work are created originally without using the tool of Matlab, so that a modified genetic algorithm has been developed. During the trials applied, many errors have occurred during both works done using Mathematica and the algorithm works in Matlab. These errors have been corrected one by one when they emerged. Hence, a successful set of Mathematica codes and optimization algorithm could be obtained. By applying trials for only the first four poses, it has been proven that the algorithm works correctly. Then, the algorithm has been run for the 10 poses to get better solutions than the one obtained by hand initially. Even though the results can move away from the predefined objective values when more independence is given to the algorithm to create input variables (population), this algorithm succeeds to give desired results when this fact is taken into consideration. The reason of this is that the algorithm searches at the global area to find optimum and it can diverge from the desired shape of motion when the limits are given widely. Normally, this is a desirable property for finding the optimum in the global space. However, for the problems with high number of variables like this work, it takes so much time to find global optimum with giving independence to the algorithm. Until it finds the global optimum, it gives results far away from the objective. In order to avoid this, some extra limits can be defined for each variable in the algorithm. Nevertheless, an initial approximation method is selected to be used instead. By using this approximation, the limits in the

algorithm are selected in small space, and the algorithm becomes guided for the approximated orientation. In this way, desired solutions could be obtained.

To give an additional information, the dimensions of the geared seven-bar mechanism at the first trial were too large so that it could not fit into the aircraft. Then as the next trial the mechanism is made smaller and the results of this modified mechanism rendered it compatible with the aircraft, and other design limits are not required to be implemented. In other words, limits for the mechanism connection points can also be defined for this algorithm in case it is needed due to lack of enough space at connection areas. However, for the present work, it is not considered since the mechanism orientation and the connection locations has not been found unacceptable, and there is no limit defined.

The last approximated initial values that are found by Mathematica can be seen in Table 14 and Table 15 and mechanism representation with these values can be seen in Figure 46.

Table 14 *Last initial values found in Mathematica, Link Lengths.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
16	24	35	22	24	30	137	26

Table 15 *Last initial values found in Mathematica, Pivot Coordinates.*

x_A	y_A	x_B	y_B	x_C	y_C	α	β	Z
131	-28	160	-28	152	-32	20	167	1.2

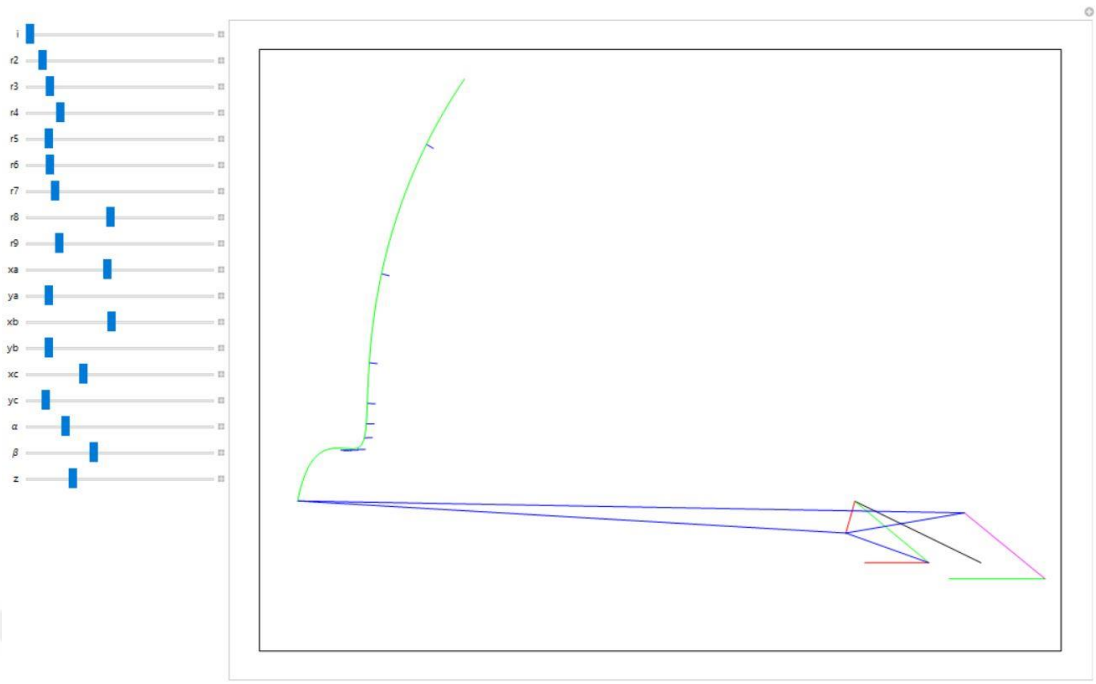

Figure 46 *Lastly found mechanism with initial values, representation in Mathematica.*

Consequently, it is provided that satisfactory results are obtained by this work. One mechanism orientation with the desired values finally could be reached. This mechanism that provides the desired result can be seen in Figure 47 as 2D which is created in Mathematica.

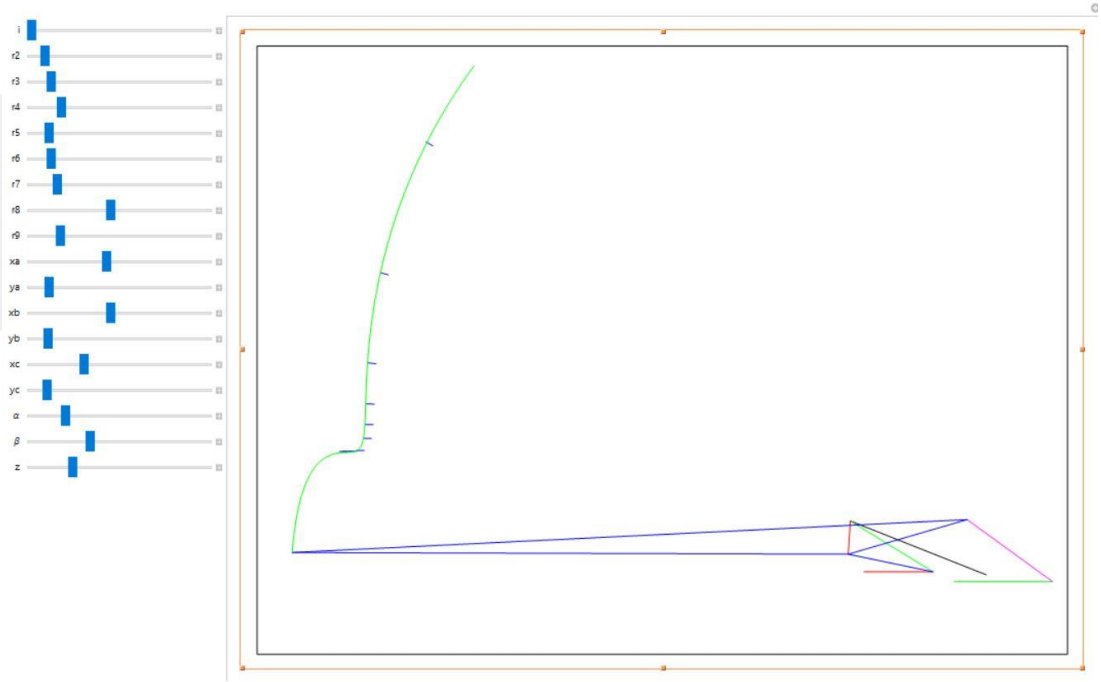

Figure 47 *The movement path of final mechanism that is found by optimization.*

The auxiliary input values of the resulted mechanism's optimization can be seen in Table 16.

Table 16 *Final solution's auxiliary input value set.*

Lower Limit	Upper Limit	Population Size	Population Dimension	Crossover Probability	Mutation Probability	Mutation Coefficient	Iteration
-2	2	128	27	0,5	0,1	0,02	50,000

The output values found by the optimization program to reach that result can be seen in Table 17, Table 18 and Table 19.

Table 17 *The generated best mechanism's link lengths.*

r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
17.0972	23.9822	36.1022	21.3883	24.2469	30.5595	137.2273	25.9761

Table 18 *The generated best mechanism's connection points' coordinates, angle values (in Degree) and gear ratio.*

x_A	y_A	x_B	y_B	x_C	y_C	α	β	Z
129.7295	-29.6681	159.9515	-30.4189	152.0045	-32.0417	19.8010	163.6095	1.2024

Table 19 *The generated best mechanism's input angle values (in Degree) for 10 poses.*

θ_{12}^{1}	θ_{12}^{2}	θ_{12}^{3}	θ_{12}^{4}	θ_{12}^{5}	θ_{12}^{6}	θ_{12}^{7}	θ_{12}^{8}	θ_{12}^{9}	θ_{12}^{10}
28.0559	29.6295	33.7390	39.5728	48.9963	54.3548	59.4309	66.1581	77.2057	91.4027

The pose values found can be seen in Table 20. In addition to this, by using the optimized values of x and y coordinates found in Matlab, a path is also obtained by Plot function in Matlab. This path represents the tracking path too and can be seen in Figure 48.

Table 20 *The found pose values by the final optimization.*

Points	P (x) (mm)	P (y) (mm)	P (Φ) (deg)
1	5.2224	-0.3	0
2	15.6111	0.2	-0.2881
3	40.0473	1.4	-0.9255
4	67.4060	3.4	-1.6010
5	92.9411	48.1	-1.1517
6	99.0054	103.8	0.0932
7	101.8105	179.5	1.9828
8	108.1701	328.2	5.9366
9	152.8687	659.8	15.2142
10	323.7655	1145.4	30.1225

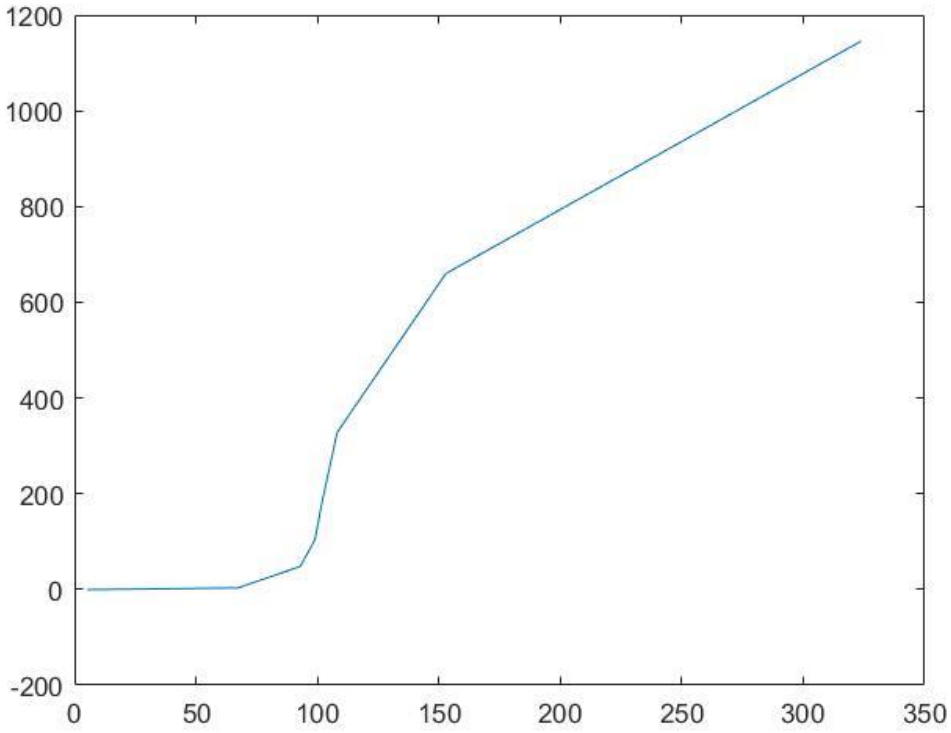

Figure 48 *The obtained path as 10 steps by using x and y coordinates in Matlab.*

It is required to change these obtained values to real values in mm. For this purpose, they are multiplied by 15 which has been selected as the coefficient value for the whole work. This has been selected during the six-bar mechanism's trials. The multiplied values can be seen in Table 21 and Table 22.

Table 21 *The generated best mechanism's link lengths after multiplying by 15.*

r_2 (mm)	r_3 (mm)	r_4 (mm)	r_5 (mm)	r_6 (mm)	r_7 (mm)	r_8 (mm)	r_9 (mm)
256.458	359.733	541.533	320.8245	363.7035	458.3925	2058.4095	389.7915

Table 22 *The generated best mechanism's connection points' coordinates after multiplying by 15.*

x_A (mm)	y_A (mm)	x_B (mm)	y_B (mm)	x_C (mm)	y_C (mm)
1945.9425	-445.0215	2399.2725	-456.2835	2280.0675	-480.6255

The Figure 49 shows the graph of objective error values with respect to iteration number. The errors shown in the figure decreases rapidly until the 10,000th iteration. Therefore, the remaining iterations may be considered unnecessary. However, they have decreased the objective function by some more amount though.

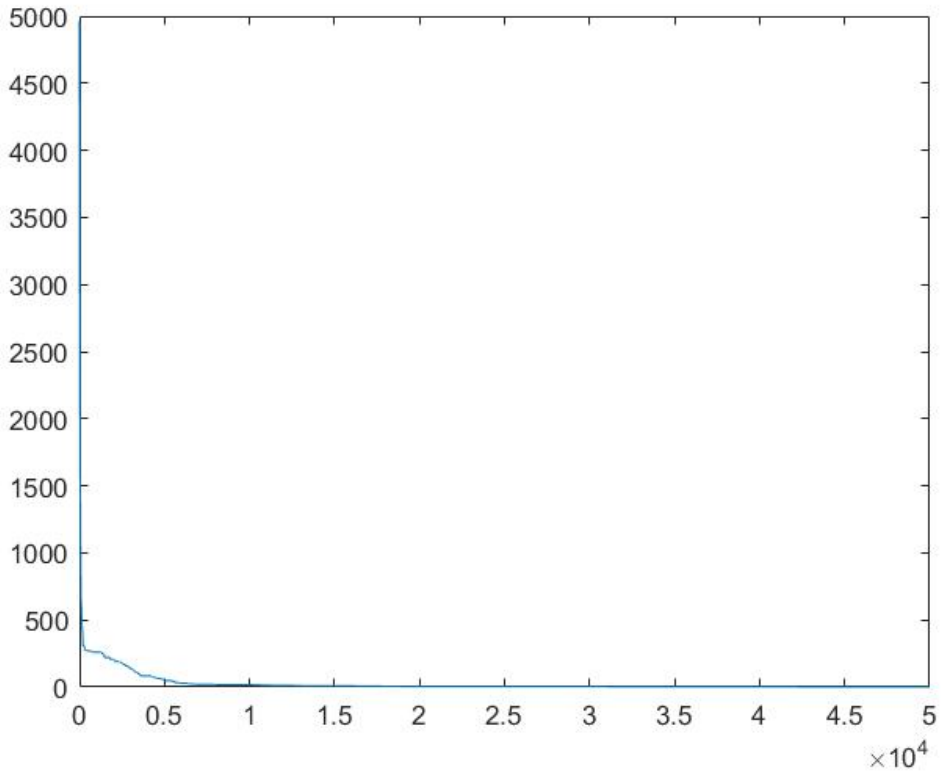

Figure 49 *The error values of objective differences with respect to iteration number.*

Moreover, the resulting mechanism's more visual shapes, and some of its poses can be seen in Figures 50-60 respectively. They are created in Creo to visualize results and give more perspicacity to the readers.

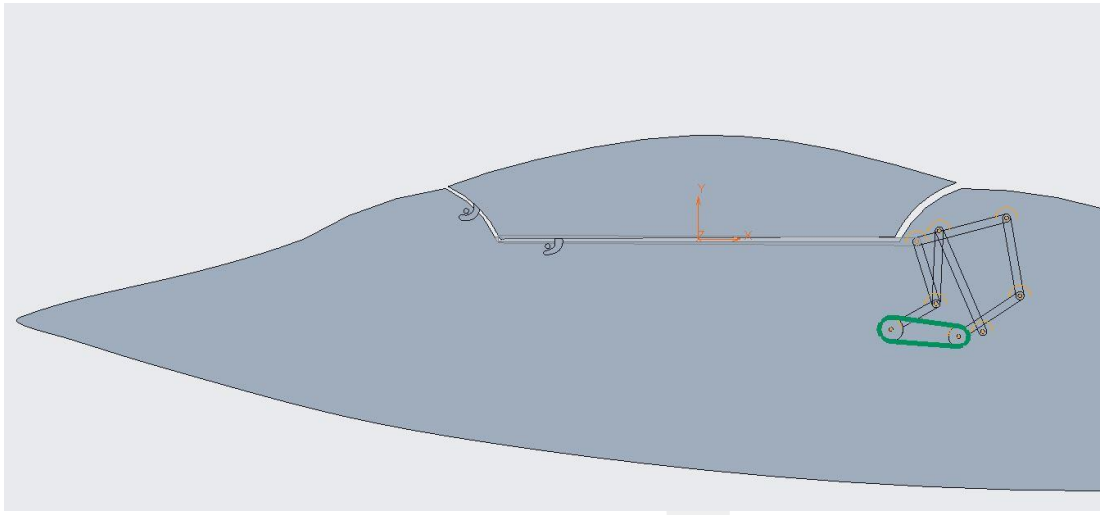

Figure 50 *The general view, when the canopy is closed.*

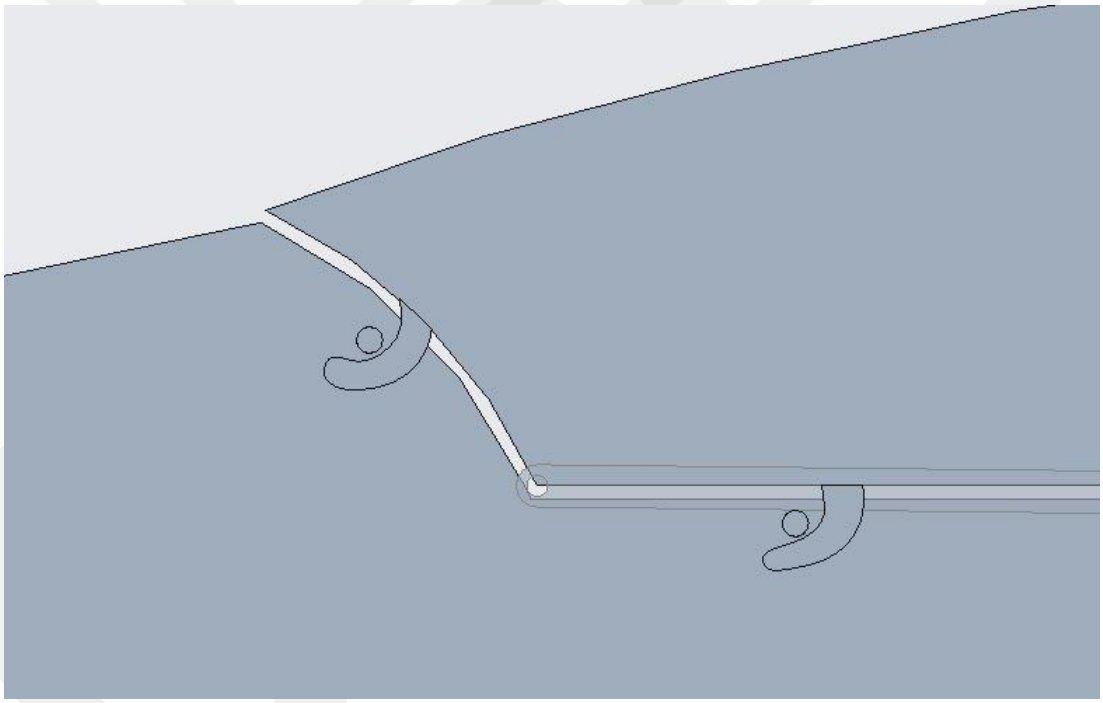

Figure 51 *A close up view to the locks, when the canopy is closed.*

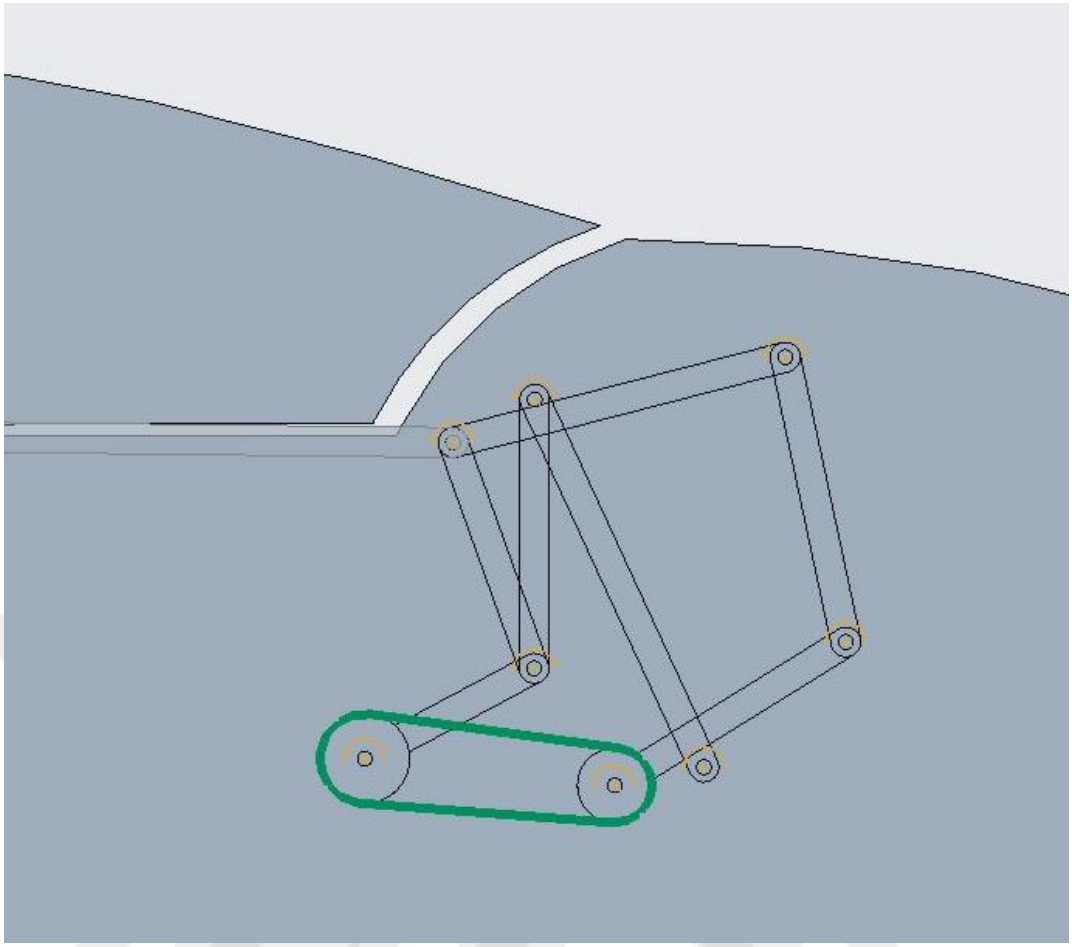

Figure 52 *A close up view to the mechanism, when the canopy is closed.*

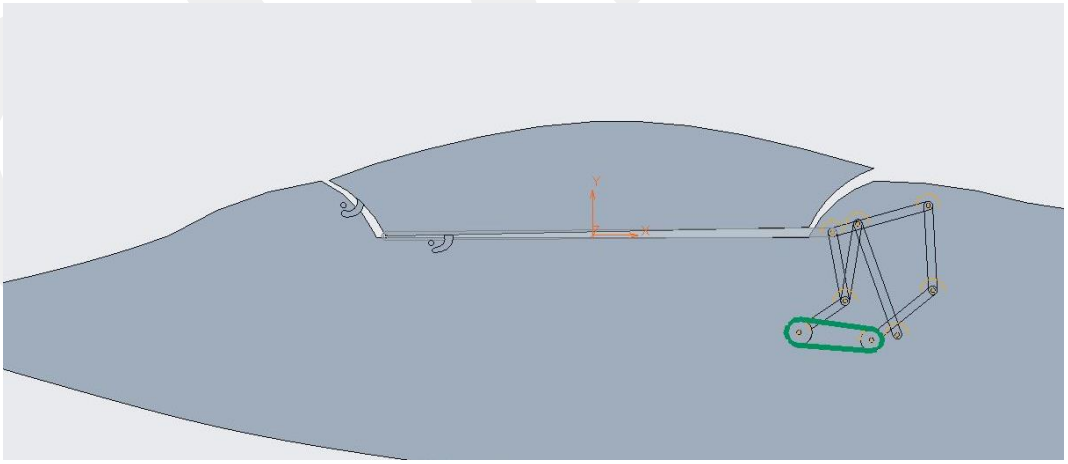

Figure 53 *The general view, when the mechanism starts to actuate.*

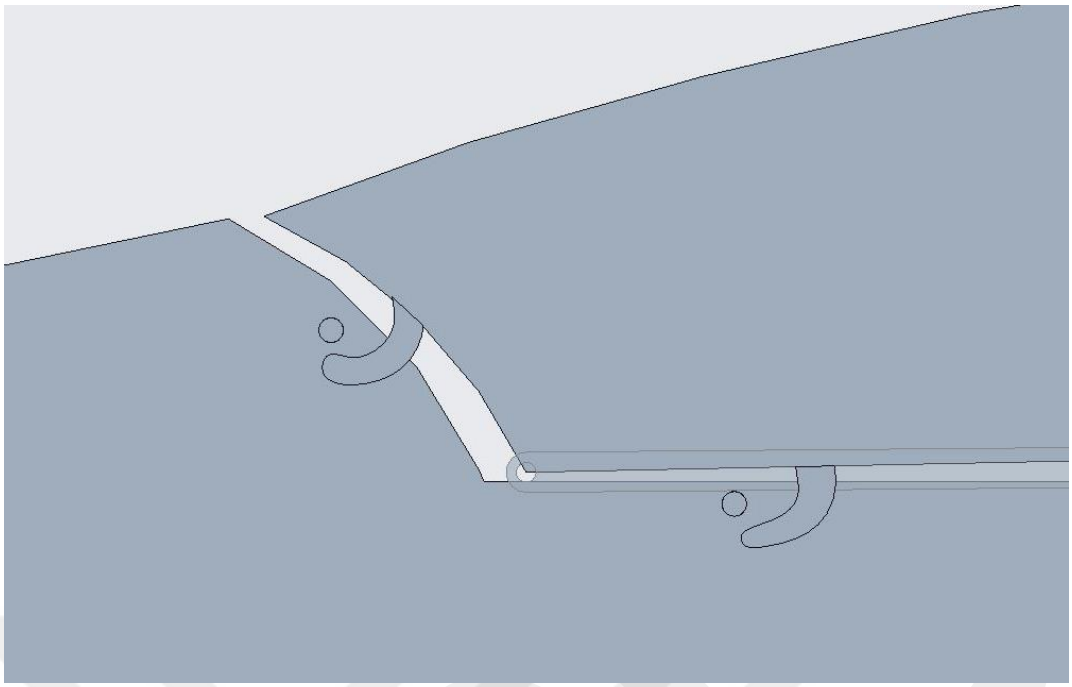

Figure 54 *A close up view to the locks, when the mechanism starts to actuate.*

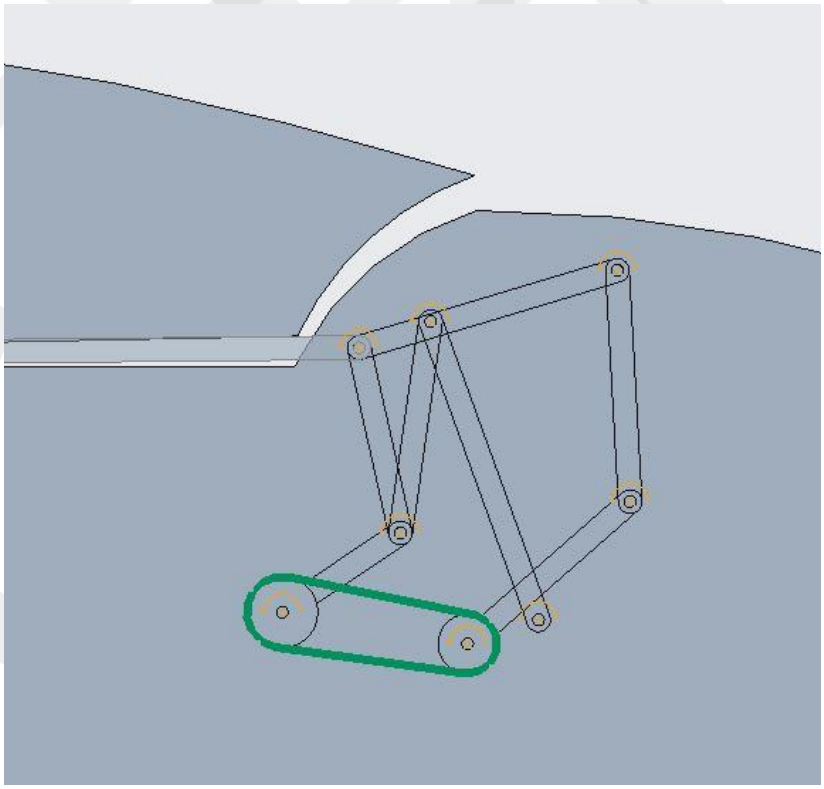

Figure 55 *A close up view to the mechanism, when the mechanism starts to actuate.*

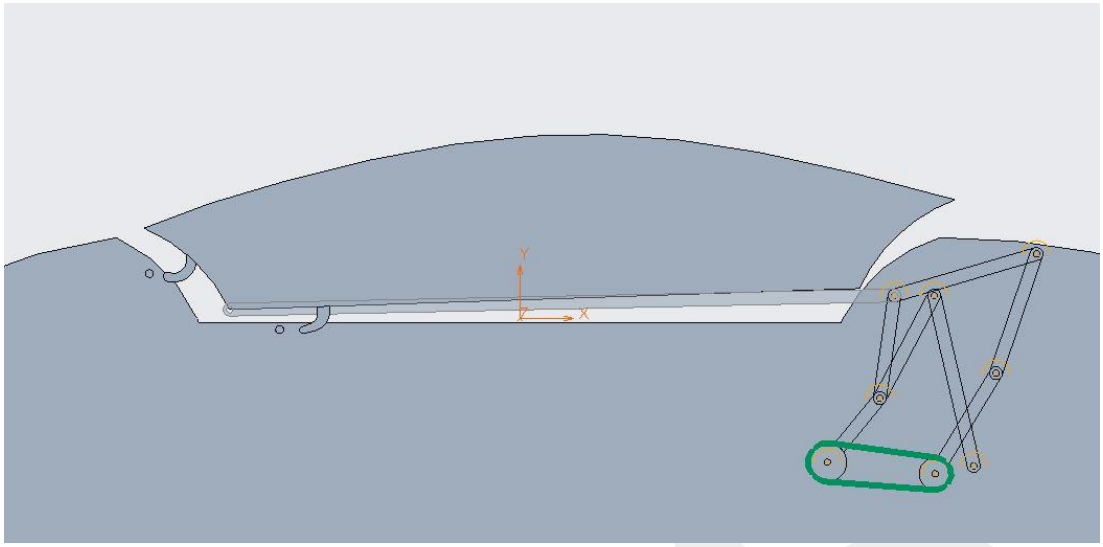

Figure 56 *The general view, when the canopy disentangled from the locks.*

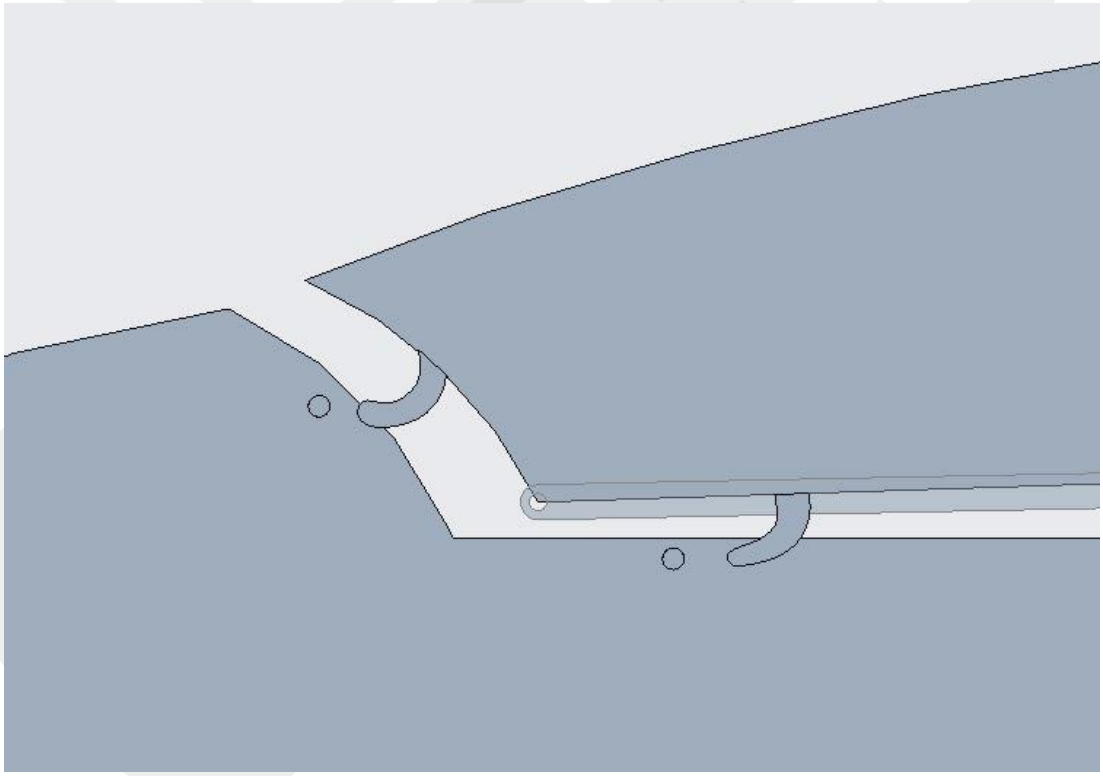

Figure 57 *A close up view to the locks, when the canopy disentangled from the locks.*

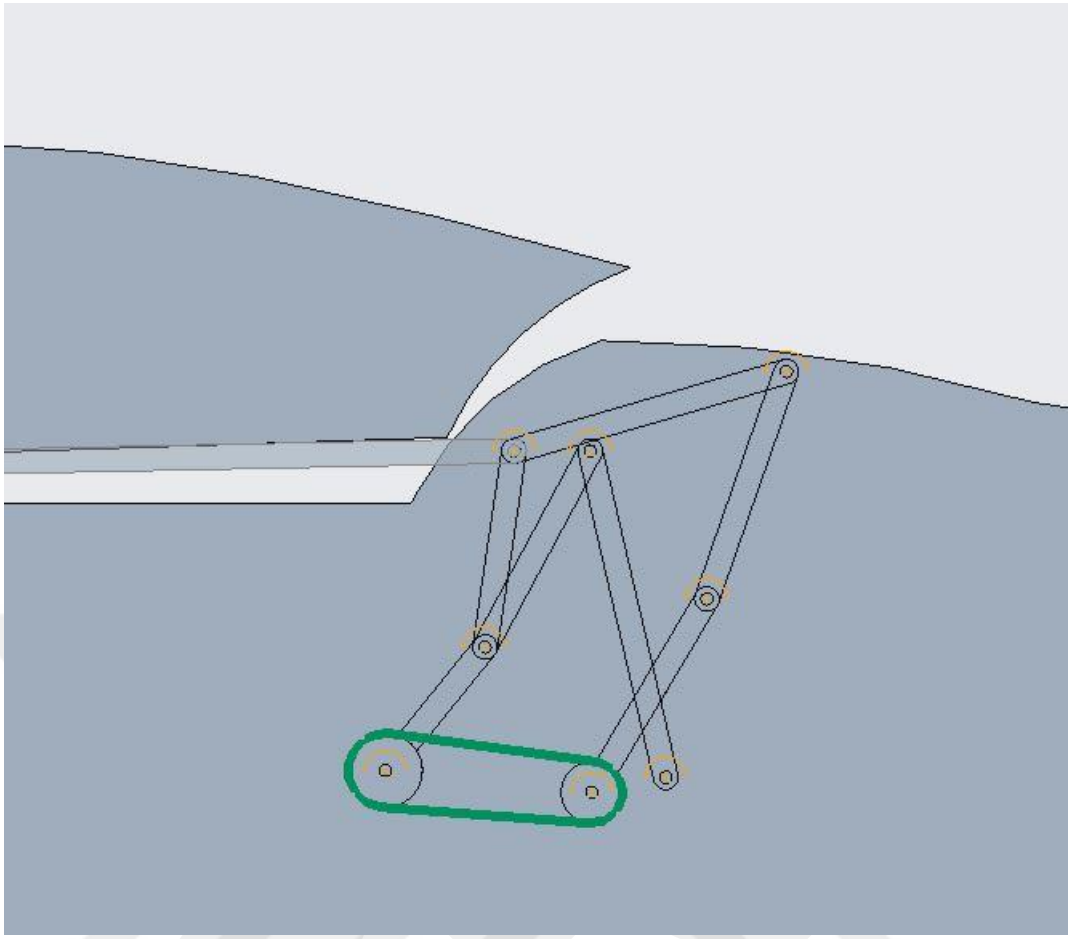

Figure 58 *A close up view to the mechanism, when the canopy disentangled from the locks.*

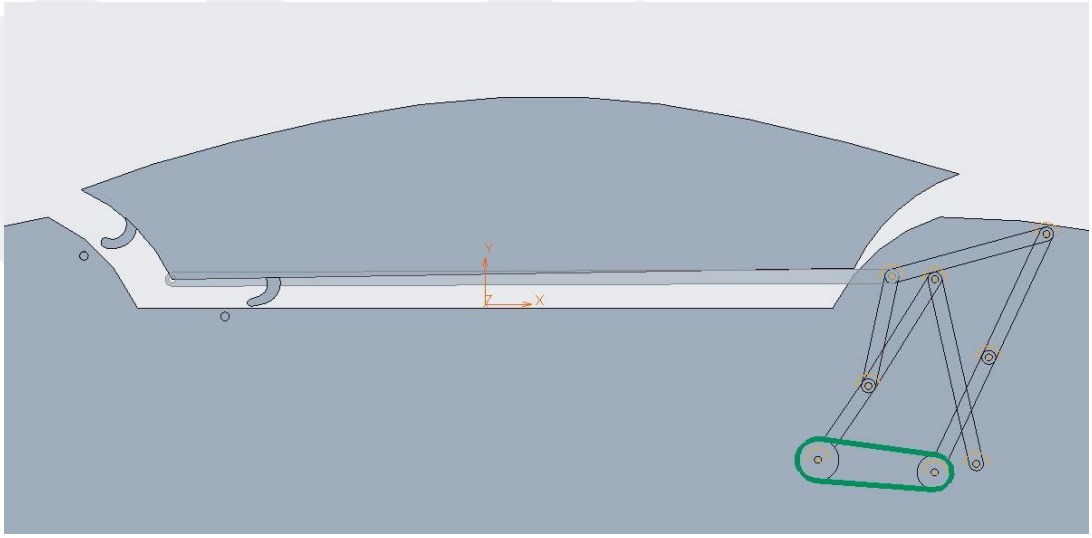

Figure 59 *A general view, when the mechanism passes to the second curve. (After the first curve finished).*

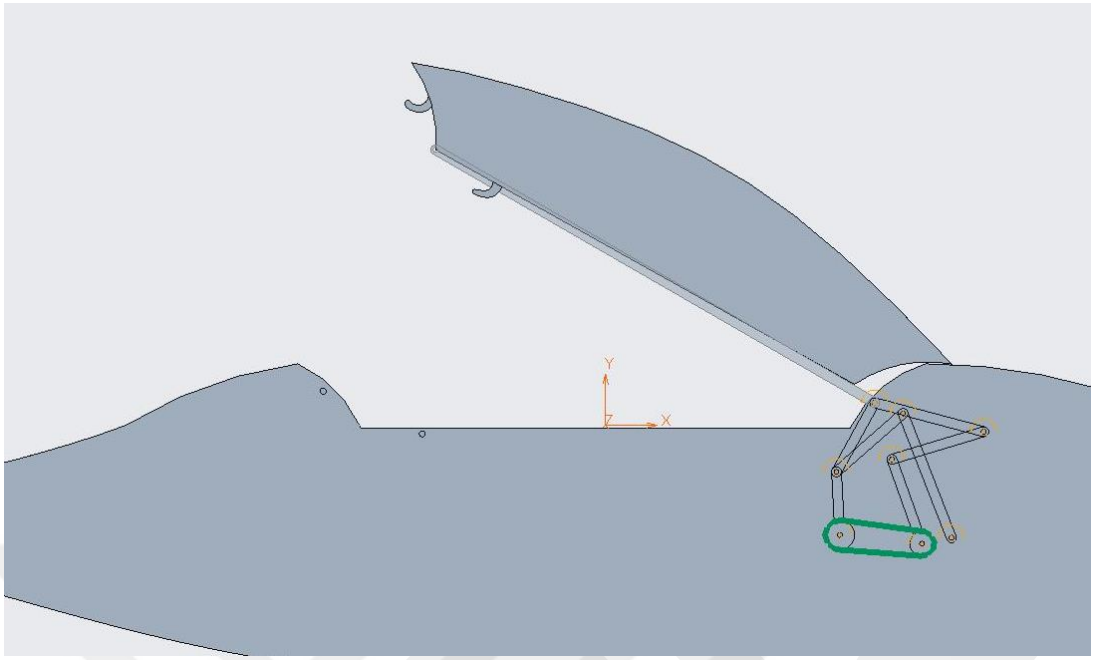

Figure 60 *The general view, when the canopy is fully opened.*

It has to be indicated that, when the values are put into the algorithm to find the optimum solution, the best found value has been decreased up to 1.6168 for 50,000 iterations, and with the auxiliary input value set in the Table 16. It is needed to keep in mind that the best found value consists of sum of 30 different values' squares. As pose count is higher, the value cannot be decreased so much to fit all of the poses equally.

On the mechanism finally obtained, a force analysis is performed to see required force or torque values to actuate the mechanism. This procedure is applied on Excel by transferring the design parameter equations and necessary calculations. Then, when the input values are entered in the excel, the requested results can be seen. For the last formation of the mechanism, the generated mechanism on Excel can be seen in Figure 61.

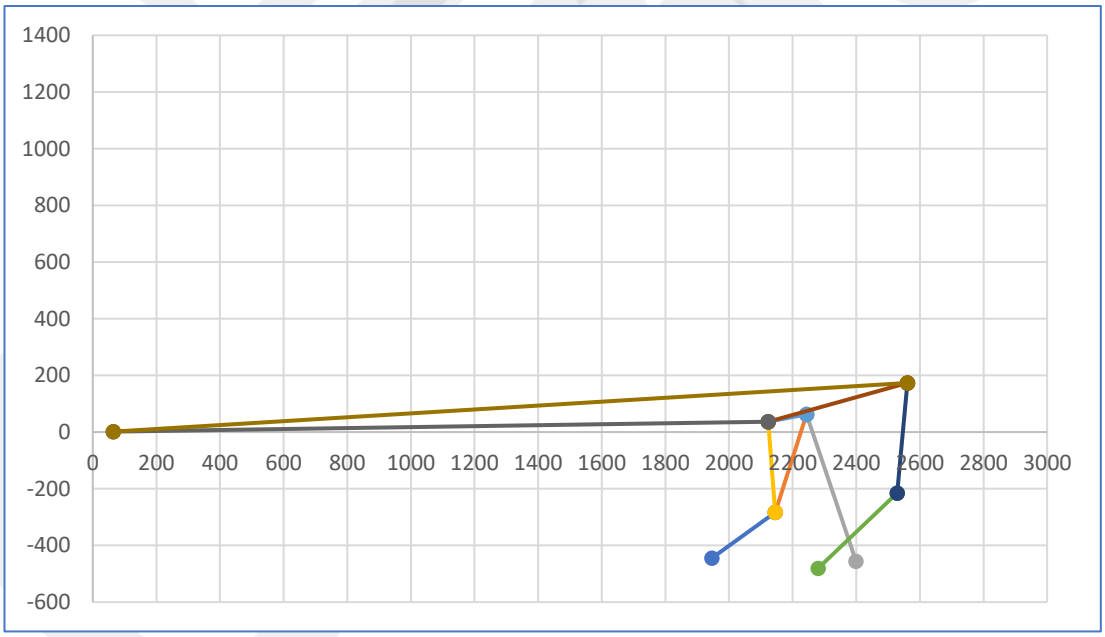

Figure 61 *Mechanism generated on Excel for force analysis.*

In the excel work, the canopy mass is taken into account as 100 Kg, and placed 1000 mm further on x-axis from the point P. The other values like link masses, frictions etc. are neglected since the effects of them are much smaller than this. The required torque values for a rotary actuator which is placed at pivot of link 2 are calculated as the input angle is varied. The maximum required torque (actuator torque) is observed when the input angle is 44°. This value is found as approximately 215.9 Nm.

To calculate the maximum stress applied on a link, the bending moments, shear forces and axial forces are considered at various positions. Links 2 and 6 are more critical where the bending stresses are dominant. Therefore, the maximum stress is calculated on the link which has maximum bending moment only. The maximum bending moment is observed on link 2 at the position of the mechanism where input angle is 39°.

By using the maximum bending moment and axial force applied on link 2 at that position, the axial stress and bending stress are calculated. By summing these two values, the resultant normal stress is found (the other stresses are neglected since they are so smaller). The found total stress on link 2 with respect to the link's width (b) and height (h) values can be seen in Table 23.

Table 23 *Stress values of link 2.*

b (mm)	h (mm)	Axial Stress (MPa)	Bending Stress (MPa)	Resultant Stress (MPa)
10	40	8.9519363	298.41572	307.36766

It can be understood that, the links can be produced with these values. If desired, the b and h values can be increased, so that the stresses can be further decreased.

For future works, these methods can be utilized to obtain results for similar mechanisms. Moreover, for different works, these methods can be modified or developed, and can also be used for them. To say the final word, this study is thought to be a guide for forthcoming mechanism studies.

CHAPTER 5

CONCLUSION

In this thesis, a problem of creating an appropriate mechanism for an aircraft canopy that should provide the desired movement, and its optimization has been tackled. Firstly, representative aircraft body and a suitable canopy are created as CAD models. The CAD model also comprises locking pins in the aircraft body and hinges at the canopy. By respecting the locking, the desired movement has been determined by picking 10 points and their angle values for the approximate opening movement of the canopy by using the CAD model, so that the motion generation problem has been created. After that, it has been considered which types of mechanism are suitable for this purpose, and decided to start with a Stephenson III type six-bar. Before the optimization, the mechanism has been created in Mathematica with its design parameter equations. The Mathematica has also helped to generate an approximate path visually. The results in the Mathematica have been used for the algorithm generated in Matlab. With the design parameter equations, the general construction of function has been created, with input values obtained in Mathematica have been used as starter values of the algorithm.

For the optimization process, Genetic Algorithm has been utilized. The reason is that, this algorithm bases on evolutionary calculation which has been inspired from Darwin's evolution theory, and generally helps to find a global optimum. To mention other advantages, this algorithm is suitable for problems with a high number of variables. It can contain high probabilities and set of rules. It provides better solutions from random solutions by using error functions and uncertainties. This algorithm has some functions like natural selection, crossover and mutation. By crossover and mutation functions, it changes the solutions and applies natural selection to them. In

this way, the better solutions are selected as algorithm continues. In this thesis, roulette wheel selection has been used generally for the natural selection method. In addition, elitism method is also implemented for some error cases. Some special fault cases' solutions and priority equations are integrated too. Thus, a modified genetic algorithm has been generated to get better results and speed up the algorithm in line with this purpose.

The six-bar mechanism orientations generated by the algorithm have not provided expected results for the defined motion. Although the desired path could be obtained, the respective angle orientations could not be achieved. Therefore, it has been decided to modify the mechanism. To give the mechanism tracking point more movement ability, one more link has been added between the ternary link and 6th link, so that previous one link is replaced by two links there. Upon addition of this link, the degree of freedom has gone up to two. To decrease the DOF to one, it has been decided to add a gear set or a bel-pulley set between 2nd link and 6th link, so that movement of 6th link has become dependent to input movement Θ12. With this modification, the mechanism has become a geared seven-link mechanism. For this mechanism, the processes like finding design parameter equations, generating an approximate path and taking its provider input values to be used for the algorithm have been applied in Mathematica again. Then, the values found have been transferred to Matlab for optimization process. In the algorithm, some auxiliary values like population size, iteration number, mutation probability are changed for different trials. These values are selected in such a way to be appropriate for the desired solution, preventing the mechanism to go into an implausible result. By deciding a convenient set of the auxiliary input values with a good prepared input variable set of the mechanism, it has been proven that the algorithm could find optimum solutions in a short time. If these values are roughly selected, the time needed to find optimum can increase tremendously. This can even make it impossible to find a suitable solution. Hence, a considerable time has been spent to determine these values firstly. Some more trials have been implemented to get a proper solution set. Finally, a result that satisfies the required motion parameters of the mechanism has been obtained by variables found by the algorithm. By using these parameters, the mechanism with canopy has been

created in Creo to see whether the mechanism can move without any inaccuracy or not. It has been understood that the created mechanism can work as requested on the aircraft body, without protruding from aircraft skin, or without taking up so much space in the aircraft body. The created mechanism can be used for an aircraft with the dimensions found. If special limits are defined, the desired different mechanisms could also be generated by the expressed methods.

After all, a force analysis has been applied on Excel by using the found mechanism formation. By this analysis, the needed torque to actuate the mechanism is found. Moreover, the maximum stress values acting on the most critical link are found. Thus, it has been verified that the mechanism links can be produced by materials which are commonly used in the industry.

By all these works, this thesis has been finalized. It has shown a methodology for similar purposes, and it is thought that it can be helpful for forthcoming works.

REFERENCES

[1] **PATEL, H. R., & MUNGLA, M. J.** (2019). Optimal synthesis for function generation of double-loop planer four-bar mechanism using genetic algorithm. *SN Applied Sciences*, *1*(5), 423.

[2] **CABRERA, J. A., SIMON, A., & PRADO, M.** (2002). Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and machine theory*, *37*(10), 1165-1177.

[3] **BALLI, S. S., & CHAND, S.** (2002). Five-bar motion and path generators with variable topology for motion between extreme positions. *Mechanism and machine theory*, *37*(11), 1435-1445.

[4] **AL-SMADI, Y. M., SHEN, Q., RUSSELL, K., & SODHI, R. S.** (2009). Planar Four-Bar Motion Generation with Prescribed Static Torque and Rigid-Body Reaction Force#. *Mechanics based design of structures and machines*, *37*(1), 73-85.

[5] **HASSAAN, G. A., AL-GAMIL, M., & LASHIN, M.** (2013). Optimal kinematic synthesis of a 4-bar planar crank-rocker mechanism for a specific stroke and time ratio. *International Journal of Mechanical and Production Engineering Research and Development*, *3*(2), 87-98.

[6] **RUSSELL, K., & SHEN, J.** (2011). Planar four-bar motion and path generation with order and branching conditions. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, *5*(4), 264-273.

[7] **SHIAKOLAS, P. S., KOLADIYA, D., & KEBRLE, J.** (2005). On the optimum synthesis of six-bar linkages using differential evolution and the geometric centroid of precision positions technique. *Mechanism and Machine Theory*, *40*(3), 319-335.

[8] **SOH, G. S., & MCCARTHY, J. M.** (2008). The synthesis of six-bar linkages as constrained planar 3R chains. *Mechanism and Machine Theory*, *43*(2), 160-170.

[9] **BULATOVIĆ, R. R., ĐORĐEVIĆ, S. R., & ĐORĐEVIĆ, V. S.** (2013). Cuckoo search algorithm: a metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. *Mechanism and Machine Theory*, *61*, 1-13.

[10] **MEHDIGHOLI, H., & AKBARNEJAD, S.** (2012). Optimization of watt's six-bar linkage to generate straight and parallel leg motion. *International Journal of Advanced Robotic Systems*, *9*(1), 22.

[11] **RUSSELL, K., & SODHI, R. S.** (2004). Kinematic synthesis of planar five-bar mechanisms for multi-phase motion generation. *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, *47*(1), 345-349.

[12] **MUNDO, D., GATTI, G., & DOONER, D. B.** (2009). Optimized five-bar linkages with non-circular gears for exact path generation. *Mechanism and Machine Theory*, *44*(4), 751-760.

[13] **NAFEES, K., & MOHAMMAD, A.** (2016). Optimal Dimensional Synthesis of Six-Bar Stephenson I Mechanism for path generation. *International Journal of Mechanical Engineering and Technology*, *7*(6), 535-546.

[14] **SHEN, J., WAND, G., BI, Q., & QU, J.** (2013). A comprehensive genetic algorithm for design optimization of Z-bar loader working mechanism. *Journal of Mechanical Science and Technology*, *27*(11), 3381-3394.

[15] **AFSHARI, J., NAZARI, F., BAGHALIAN, S., & MALIHI, S. S.** (2012). Applying genetic algorithm for optimization of six-bar mechanism of prosthetic knee joint. *International Journal of Engineering and Applied Sciences*, *4*(1), 9-16.

[16] **PLECNIK, M. M., & MCCARTHY, J. M.** (2016). Design of Stephenson linkages that guide a point along a specified trajectory. *Mechanism and Machine Theory*, *96*, 38-51.

[17] **XI, Y., XIA, Y., LI, X., SUN, Y., & WANG, H.** (2010). Kinematic analysis and parameter optimization of six-bar linkage based on ADAMS. In *2010 International Conference on Mechanic Automation and Control Engineering* (pp. 10-13).

[18] **NAFEES, K., & MOHAMMAD, A.** (2016). Dimensional synthesis of six-bar Stephenson II linkage for fifteen precision points path generation. *Perspectives in Science*, *8*, 485-487.

[19] **LIN, W. Y.** (2013). Optimum path synthesis of a geared five-bar mechanism. *Advances in Mechanical Engineering*, *5*, Article ID: 757935.

APPENDICES

Matlab Codes

```
function [eniyigirdi, eniyicozum, eniyideger, objit] =
sonuc_14_02_2021_10()

as=-2; us=2; d = 27; psize = 128; pcross = 0.5; pmutation= 0.1;
delta = 0.02; eniyideger = 1000000000000; eniyicozum = zeros (1,30);
hata = 0; points = 10;
    ka = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]; ka = 1 * ka;
    aci = [28 1.5 4.0 5.8 9.2 4.9 4.8 6.7 10.5 14];
division = 15;

    pk = [1 1 1 1 0.01 0.01 1 0.01 0.5 0.5 0.5 0.5 0.5 0.01 0.01
0.01 0.0001];
    pk = pk * 15;

for iterasyon = 1:1

lamda = unifrnd(as, us, [psize, d]); i = 0;

while i < psize
    i = i + 1;

        population(i, :) = [(lamda(i,1)) (lamda(i,2)) (lamda(i,3))
(lamda(i,4)) (lamda(i,5)) (lamda(i,6)) (lamda(i,7)) (lamda(i,8))
(lamda(i,9)) (lamda(i,10)) (lamda(i,11)) (lamda(i,12)) (lamda(i,13))
(lamda(i,14)) (lamda(i,15)) (lamda(i,16)) (lamda(i,17))
(lamda(i,18)) (lamda(i,19)) (lamda(i,20)) (lamda(i,21))
(lamda(i,22)) (lamda(i,23)) (lamda(i,24)) (lamda(i,25))
(lamda(i,26)) (lamda(i,27))];

        xa(i) = division * 131 + pk(1) * population (i,1);
        ya(i) = division * -28 + pk(2) * population (i,2);
        xb(i) = division * 160 + pk(3) * population (i,3);
        yb(i) = division * -28 + pk(4) * population (i,4);
        xc(i) = division * 152 + pk(5) * population (i,5);
        yc(i) = division * -32 + pk(6) * population (i,6);

        r2(i) = division * 16 + pk(7) * population (i,7);
        r3(i) = division * 24 + pk(8) * population (i,8);
        r4(i) = division * 35 + pk(9) * population (i,9);
        r5(i) = division * 22 + pk(10) * population (i,10);
        r6(i) = division * 24 + pk(11) * population (i,11);
```

```
        r7(i) = division * 30 + pk(12) * population (i,12);
        r8(i) = division * 137 + pk(13) * population (i,13);
        r9(i) = division * 26 + pk(14) * population (i,14);

        alpha(i) = 20 + pk(15) * population(i,25);
        Alpha(i) = (pi/180) * alpha(i);

        beta(i) = 167 + pk(16) * population (i,26);
        Beta(i) = (pi/180) * beta(i);

        Z(i) = 1.2 + pk(17) * population (i,27);

        for z = 1:points
            if z == 1
        theta12(i,z) = ka(z) * population (i,z+14) + aci(z);
            end
            if z > 1
        theta12(i,z) = theta12(i,z-1) + ka(z) * abs(population
(i,z+14)) + aci(z);
            end
        end
        Theta12(i, :) = (pi/180) * theta12(i,:);

        for j = 1:points

        A1(i,j) = -2 * r4(i) * (xa(i) - xb(i) + r2(i) *
cos(Theta12(i,j)));
        B1(i,j) = -2 * r4(i) * (ya(i) - yb(i) + r2(i) *
sin(Theta12(i,j)));
        C1(i,j) = -(r2(i)^2 - r3(i)^2 + r4(i)^2 + (xa(i) - xb(i))^2
+ (ya(i) - yb(i))^2 + 2 * r2(i) * ((xa(i) - xb(i)) *
cos(Theta12(i,j)) + (ya(i) - yb(i)) * sin(Theta12(i,j))));

            if (A1(i,j).^2 + B1(i,j).^2) < 0 || (C1(i,j) ./
(sqrt(A1(i,j).^2 + B1(i,j).^2))) > 1 || C1(i,j) ./ (sqrt(A1(i,j).^2
+ B1(i,j).^2)) < -1
                fprintf('error '); lamda(i, :) = unifrnd(as, us, [1,
d]); i = i - 1; break
            else
                Theta14(i,j) = atan2(B1(i,j), A1(i,j)) +
acos(C1(i,j) ./ (sqrt(A1(i,j).^2 + B1(i,j).^2)));
                Theta13(i,j) = atan2((((yb(i)-ya(i)) + r4(i) .*
sin(Theta14(i,j)) - r2(i) .* sin(Theta12(i,j))).*(r3(i))),(((xb(i)-
xa(i)) + r4(i) .* cos(Theta14(i,j)) - r2(i) .*
cos(Theta12(i,j))).*(r3(i))));
            end

        Theta16(i,j) = Theta12(i,j) * Z(i);

        A2(i,j) = -2 * r9(i) * (xa(i) - xc(i) + r2(i) *
cos(Theta12(i,j)) - r6(i) * cos(Theta16(i,j)) + r5(i) *
cos(Alpha(i)+Theta13(i,j)));
        B2(i,j) = -2 * r9(i) * (ya(i) - yc(i) + r2(i) *
sin(Theta12(i,j)) - r6(i) * sin(Theta16(i,j)) + r5(i) *
sin(Alpha(i)+Theta13(i,j)));
        C2(i,j) = -(r2(i)^2 + r5(i)^2 + r6(i)^2 - r7(i)^2 + r9(i)^2
+ xa(i)^2 - 2 * xa(i) * xc(i) + xc(i)^2 + ya(i)^2 - 2 * ya(i) *
yc(i) + yc(i)^2 + 2 * r2(i) * (xa(i) - xc(i)) * cos(Theta12(i,j)) -
```

```
2 * r6(i) * (xa(i) - xc(i)) * cos(Theta16(i,j)) - 2 * r2(i) * r6(i)

* cos(Theta12(i,j)-Theta16(i,j)) + 2 * r5(i) * xa(i) *
cos(Alpha(i)+Theta13(i,j)) - 2 * r5(i) * xc(i) *
cos(Alpha(i)+Theta13(i,j)) + 2 * r2(i) * r5(i) * cos(Alpha(i)-
Theta12(i,j)+Theta13(i,j)) - 2 * r5(i) * r6(i) * cos(Alpha(i)-
Theta16(i,j)+Theta13(i,j)) + 2 * r2(i) * ya(i) * sin(Theta12(i,j)) -
2 * r2(i) * yc(i) * sin(Theta12(i,j)) - 2 * r6(i) * ya(i) *
sin(Theta16(i,j)) + 2 * r6(i) * yc(i) * sin(Theta16(i,j)) + 2 *
r5(i) * ya(i) * sin(Alpha(i)+Theta13(i,j)) - 2 * r5(i) * yc(i) *
sin(Alpha(i)+Theta13(i,j)));

            if (A2(i,j).^2 + B2(i,j).^2) < 0 || (C2(i,j) ./
(sqrt(A2(i,j).^2 + B2(i,j).^2))) > 1 || (C2(i,j) ./ (sqrt(A2(i,j).^2
+ B2(i,j).^2))) < -1
                fprintf('error2 '); lamda(i, :) = unifrnd(as, us,
[1, d]); i = i - 1; break
            else
                Theta19(i,j) = atan2(B2(i,j), A2(i,j)) +
acos(C2(i,j) ./ (sqrt(A2(i,j).^2 + B2(i,j).^2)));
                Theta17(i,j) = atan2((((r2(i) .* sin(Theta12(i,j)))
+ r5(i) .* sin(Theta13(i,j) + Alpha(i)) - (yc(i) - ya(i)) - r6(i) .*
sin(Theta16(i,j)) - r9(i) .* sin(Theta19(i,j))./(r7(i))),((r2(i) .*
cos(Theta12(i,j)) + r5(i) .* cos(Theta13(i,j) + Alpha(i)) - (xc(i) -
xa(i)) - r6(i) .* cos(Theta16(i,j)) - r9(i) .*
cos(Theta19(i,j))./(r7(i))));
            end
        end
end

iteration = 1;


eniyigirdi = zeros (1,d);


asilobj = [5 15 40 67 93 100 104 111 157 325 0 0 1 3.5 48 100 176
328 661 1145 0 0.5 1 1.6 1.19 0 -1.9 -6 -15 330];
onemk = [1 1 1 1 1 0.1 0.1 0.1 0.1 0.1 1 1 1 1 1 0.1 0.1 0.1 0.1 0.1
1 1 1 1 0.5 0.5 0.5 0.5 0.5 1];

while (iteration < 50000)

    beklenti = zeros (psize, 1);
    objfark = zeros (psize, (points*3));

    [obj] = girdiler_14_02_2021_10(psize, Theta12, xa, ya, r2, r5,
Theta13, Alpha, r8, Theta17, Beta, points);

    for i = 1:psize
       objfark(i, :) = asilobj - obj(i, :);
    end

    objfark = objfark .* onemk;

    for i = 1:psize
        beklenti(i) = sum(objfark(i, :).^2);
    end
```

```
    if (min(beklenti) < eniyideger)
       eniyideger = min(beklenti);
       idx = find(beklenti == eniyideger);
       eniyicozum = obj(idx, :);
       eniyigirdi = population(idx, :);
    end

    objit(iteration) = eniyideger;

[arapop] = dogalsecilim_14_02_2021_10(population, beklenti, psize);

[arapop] = crossover_14_02_2021_10(arapop, psize, pcross, d);
population = mutation_14_02_2021_10(arapop, pmutation, psize, d,
delta, us, as);

i = 0;
while i < psize
    i = i + 1;
    r = i; % to be used for yedekpop

        if hata == 2
            population (i,:) = eniyigirdi;
            hata = 0; fprintf('//');
        end

        xa(i) = division * 131 + pk(1) * population (i,1);
        ya(i) = division * -28 + pk(2) * population (i,2);
        xb(i) = division * 160 + pk(3) * population (i,3);
        yb(i) = division * -28 + pk(4) * population (i,4);
        xc(i) = division * 152 + pk(5) * population (i,5);
        yc(i) = division * -32 + pk(6) * population (i,6);

        r2(i) = division * 16 + pk(7) * population (i,7);
        r3(i) = division * 24 + pk(8) * population (i,8);
        r4(i) = division * 35 + pk(9) * population (i,9);
        r5(i) = division * 22 + pk(10) * population (i,10);
        r6(i) = division * 24 + pk(11) * population (i,11);
        r7(i) = division * 30 + pk(12) * population (i,12);
        r8(i) = division * 137 + pk(13) * population (i,13);
        r9(i) = division * 26 + pk(14) * population (i,14);

        alpha(i) = 20 + pk(15) * population(i,25);
        Alpha(i) = (pi/180) * alpha(i);

        beta(i) = 167 + pk(16) * population (i,26);
        Beta(i) = (pi/180) * beta(i);

        Z(i) = 1.2 + pk(17) * population (i,27);

        for z = 1:points
            if z == 1
        theta12(i,z) = ka(z) * population (i,z+14) + aci(z);
            end
            if z > 1
        theta12(i,z) = theta12(i,z-1) + ka(z) * abs(population
(i,z+14)) + aci(z);
            end
        end
```

```
        Theta12(i, :) = (pi/180) * theta12(i,:);

   for j = 1:points

        A1(i,j) = -2 * r4(i) * (xa(i) - xb(i) + r2(i) *
cos(Theta12(i,j)));
        B1(i,j) = -2 * r4(i) * (ya(i) - yb(i) + r2(i) *
sin(Theta12(i,j)));
        C1(i,j) = -(r2(i)^2 - r3(i)^2 + r4(i)^2 + (xa(i) - xb(i))^2
+ (ya(i) - yb(i))^2 + 2 * r2(i) * ((xa(i) - xb(i)) *
cos(Theta12(i,j)) + (ya(i) - yb(i)) * sin(Theta12(i,j))));

            if (A1(i,j).^2 + B1(i,j).^2) < 0 || (C1(i,j) ./
(sqrt(A1(i,j).^2 + B1(i,j).^2))) > 1 || C1(i,j) ./ (sqrt(A1(i,j).^2
+ B1(i,j).^2)) < -1
                fprintf('!');
                arapop = population;
                [arapop] = crossover_14_02_2021_10(arapop, psize,
pcross, d);
              population = mutation_14_02_2021_10(arapop,
pmutation, psize, d, delta, us, as);
                i = i - 1; hata = hata + 1; break
            else
                Theta14(i,j) = atan2(B1(i,j), A1(i,j)) +
acos(C1(i,j) ./ (sqrt(A1(i,j).^2 + B1(i,j).^2)));
                Theta13(i,j) = atan2((((yb(i)-ya(i)) + r4(i) .*
sin(Theta14(i,j)) - r2(i) .* sin(Theta12(i,j))./(r3(i))),(((xb(i)-
xa(i)) + r4(i) .* cos(Theta14(i,j)) - r2(i) .*
cos(Theta12(i,j)))/(r3(i))));
            end

        A2(i,j) = -2 * r9(i) * (xa(i) - xc(i) + r2(i) *
cos(Theta12(i,j)) - r6(i) * cos(Theta16(i,j)) + r5(i) *
cos(Alpha(i)+Theta13(i,j)));
        B2(i,j) = -2 * r9(i) * (ya(i) - yc(i) + r2(i) *
sin(Theta12(i,j)) - r6(i) * sin(Theta16(i,j)) + r5(i) *
sin(Alpha(i)+Theta13(i,j)));
        C2(i,j) = -(r2(i)^2 + r5(i)^2 + r6(i)^2 - r7(i)^2 + r9(i)^2
+ xa(i)^2 - 2 * xa(i) * xc(i) + xc(i)^2 + ya(i)^2 - 2 * ya(i) *
yc(i) + yc(i)^2 + 2 * r2(i) * (xa(i) - xc(i)) * cos(Theta12(i,j)) -
2 * r6(i) * (xa(i) - xc(i)) * cos(Theta16(i,j)) - 2 * r2(i) * r6(i)
* cos(Theta12(i,j)-Theta16(i,j)) + 2 * r5(i) * xa(i) *
cos(Alpha(i)+Theta13(i,j)) - 2 * r5(i) * xc(i) *
cos(Alpha(i)+Theta13(i,j)) + 2 * r2(i) * r5(i) * cos(Alpha(i)-
Theta12(i,j)+Theta13(i,j)) - 2 * r5(i) * r6(i) * cos(Alpha(i)-
Theta16(i,j)+Theta13(i,j)) + 2 * r2(i) * ya(i) * sin(Theta12(i,j)) -
2 * r2(i) * yc(i) * sin(Theta12(i,j)) - 2 * r6(i) * ya(i) *
sin(Theta16(i,j)) + 2 * r6(i) * yc(i) * sin(Theta16(i,j)) + 2 *
r5(i) * ya(i) * sin(Alpha(i)+Theta13(i,j)) - 2 * r5(i) * yc(i) *
sin(Alpha(i)+Theta13(i,j)));

            if (A2(i,j).^2 + B2(i,j).^2) < 0 || (C2(i,j) ./
(sqrt(A2(i,j).^2 + B2(i,j).^2))) > 1 || (C2(i,j) ./ (sqrt(A2(i,j).^2
+ B2(i,j).^2))) < -1
                fprintf('?');
                arapop = population;
                [arapop] = crossover_14_02_2021_10(arapop, psize,
pcross, d);
```

```
                population = mutation_14_02_2021_10(arapop, pmutation,
psize, d, delta, us, as);
                    i = i - 1; hata = hata + 1; break
                else
                    Theta19(i,j) = atan2(B2(i,j), A2(i,j)) +
acos(C2(i,j) ./ (sqrt(A2(i,j).^2 + B2(i,j).^2)));
                    Theta17(i,j) = atan2((((r2(i) .* sin(Theta12(i,j)))
+ r5(i) .* sin(Theta13(i,j) + Alpha(i)) - (yc(i) - ya(i)) - r6(i) .*
sin(Theta16(i,j)) - r9(i) .* sin(Theta19(i,j))).*(r7(i))),((r2(i) .*
cos(Theta12(i,j)) + r5(i) .* cos(Theta13(i,j) + Alpha(i)) - (xc(i) -
xa(i)) - r6(i) .* cos(Theta16(i,j)) - r9(i) .*
cos(Theta19(i,j))).*(r7(i))));
                end
                if j == points
                yedekpop(r, :) = population (r, :);
                end
    end
end
population = yedekpop;
 %   iterasyon
iteration
    iteration = iteration + 1;
    eniyideger
end
end
xp = eniyicozum(1:10);
yp = eniyicozum(11:20);
phi = eniyicozum(21:30);

Xa = 131 + (pk(1) / 15) * eniyigirdi (1);
Ya = -28 + (pk(2) / 15) * eniyigirdi (2);
Xb = 160 + (pk(3) / 15) * eniyigirdi (3);
Yb = -28 + (pk(4) / 15) * eniyigirdi (4);
Xc = 152 + (pk(5) / 15) * eniyigirdi (5);
Yc= -32 + (pk(6) / 15) * eniyigirdi (6);
R2 = 16 + (pk(7) / 15) * eniyigirdi (7);
R3 = 24 + (pk(8) / 15) * eniyigirdi (8);
R4 = 35 + (pk(9) / 15) * eniyigirdi (9);
R5 = 22 + (pk(10) / 15) * eniyigirdi (10);
R6 = 24 + (pk(11) / 15) * eniyigirdi (11);
R7 = 30 + pk(12) / 15 * eniyigirdi (12);
R8 = 137 + pk(13) / 15 * eniyigirdi (13);
R9 = 26 + pk(14) / 15 * eniyigirdi (14);
aalpha = 20 + pk(15) * eniyigirdi (25);
bbeta = 167 + pk(16) * eniyigirdi (26);
ZZ = 1.2 + pk(17) * eniyigirdi (27);

for z = 1:points
    if z == 1
        T12(z) = ka(z) * eniyigirdi (z+14) + aci(z);
    end
    if z > 1
        T12(z) = T12(z-1) + ka(z) * abs(eniyigirdi (z+14)) + aci(z);
    end
end
cozum = [R2 R3 R4 R5 R6 R7 R8 R9 Xa Ya Xb Yb Xc Yc aalpha bbeta ZZ];

end
```

```
function [obj] = girdiler_14_02_2021_10(psize, Theta12, xa, ya, r2,
r5, Theta13, Alpha, r8, Theta17, Beta, points)

for i = 1:psize
    for j = 1:points
xp(i,j) = xa(i) + r2(i) * cos(Theta12(i,j)) + r5(i) *
cos(Theta13(i,j) + Alpha(i)) + r8(i) * cos(Theta17(i,j) + Beta(i) -
pi);
yp(i,j) = ya(i) + r2(i) * sin(Theta12(i,j)) + r5(i) *
sin(Theta13(i,j) + Alpha(i)) + r8(i) * sin(Theta17(i,j) + Beta(i) -
pi);

% xd = xa + r2 * cos(Theta12(j));
% yd = ya + r2 * sin(Theta12(j));

% xe = xb + r4 * cos(Theta14(i,j));
% ye = yb + r4 * sin(Theta14(i,j));

% xf = xd + r5 * cos(Theta13(i,j) + Alpha);
% yf = yd + r5 * sin(Theta13(i,j) + Alpha);

% xg = xc + r6 * cos(Theta16(i,j));
% yg = yc + r6 * sin(Theta16(i,j));
% Eğer kullanılacaksa, bunların güncellenmesi lazım mathemacica'ya
göre

theta17(i,j) = Theta17(i,j) / (pi/180);
Phi(i,j) = theta17(i,j) - theta17(i,1);

    end

end

obj = [xp yp Phi];

end
```

```
function [arapop] = dogalsecilim_14_02_2021_10(population, beklenti,
psize)

beklenti = 1 ./ beklenti;
sumbeklenti = sum(beklenti);
probs = beklenti / sumbeklenti;
cprobs = probs;

for i = 2:psize

    cprobs(i) = cprobs(i-1) + probs(i);

end

rs = unifrnd(0, 1, [psize, 1]);
arapop = population;

for i = 1:psize

    idx = find(rs(i) < cprobs, 1);
    arapop(i, :) = population(idx, :);

end

end
```

```
function [arapop] = crossover_14_02_2021_10(arapop, psize, pcross,
d)

pairs = randperm(psize);

for i = 1:psize/2

    parent1idx = pairs(2 * i - 1);
    parent2idx = pairs(2 * i);
    parent1 = arapop(parent1idx, :);
    parent2 = arapop(parent2idx, :);
    rs = unifrnd(0, 1);
    if (rs < pcross)
      % fprintf('çaprazladı ');
        cpoint = unidrnd(d-1);                      %crosspoint
        dummy = parent1(cpoint + 1 : end);          %birinin
verisini saklamak için oluşturulan şey
        parent1 (cpoint + 1 : end) = parent2 (cpoint + 1 : end);
        parent2 (cpoint + 1 : end) = dummy;
        arapop (parent1idx, :) = parent1;
        arapop (parent2idx, :) = parent2;

    end

end

end
```

```
function [arapop] = mutation_14_02_2021_10(arapop, pmutation, psize,
d, delta, us, as)

rs = unifrnd(0, 1, [psize, d]);

for i = 1:psize
    for j = 1:d

        if (rs(i,j) < pmutation)
         %   fprintf('mutasyona uğradı ');
            rs2 = unifrnd(-1, 1);
            arapop (i,j) = arapop (i,j) + rs2 * delta * (us - as);

        end

    end
end

end
```