UNIVERSITY OF CALIFORNIA,
IRVINE

Computer Aided Design of Eight-bar Linkages

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Kaustubh Hiralal Sonawale

2015

DEDICATION

To my
family, friends and lab

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor J. Michael McCarthy, for his support and guidance throughout my Masters and Ph.D. It's been an honor to work with him. He has been a constant inspiration for me for both his excellence in mechanical engineering and critical thinking.

I would also like to extend my thanks to Professor J.E.Bobrow, Prof. K. Mease, Prof. D. Reinkensmeyer and Prof. F. Jabbari for their wonderful classes. I also appreciate the interest of the dissertation committee, Prof. K. Mease and Prof. L. Valdevit, in my research.

I thank my colleagues Brandon Tsuge, Alex Arredondo, Mark Plecnik, Brian Parrish, Shramana Ghosh, Yang Liu and Jeff Glabe in the Robotics and Automation Laboratory for their constant support throughout my stay in Irvine. I enjoyed long conversations with Mark Plecnik regarding the future of the kinematics field, and also collaborating with Alex Arredondo and Jeff Glabe on the MechGen software. I specifically learnt a lot from Brian Parrish and enjoyed working with him. Also learnt a few life lessons from Brandon Tsuge.

Being at UCI has been a very rewarding experience for me. The beautiful and peaceful campus really helped me calm down during stressful times. An afternoon walk around the Aldrich park is truly a serene experience. I gratefully acknowledge the support of the department of Mechanical and Aerospace Engineering at UC Irivne and the National Science Foundation.

ABSTRACT OF THE DISSERTATION

Computer Aided Design of Eight-bar Linkages

By

Kaustubh Hiralal Sonawale

Doctor of Philosophy in Mechanical and Aerospace Engineering

University of California, Irvine, 2015

Professor J. Michael McCarthy, Chair

This dissertation describes research into an automated methodology for designing eight-bar linkage systems for specialized tasks. The user specifies the desired motion as five discrete task positions, and a backbone chain robot that can achieve those task positions. The design algorithm then uses the graph theory approach to find all the different ways of constraining the robot using mechanical constraints, which are rigid binary links (RR constraints), to generate single degree of freedom linkages.

Linkage solutions synthesized using this method are only guaranteed to reach the five positions. They can exhibit branch defects which results in discontinuities during their motion in between the task positions. An analysis algorithm is used to discard the defective solutions and ensure that the filtered solutions will move the end-effector smoothly through all the five desired task positions. Once the original task positions are explored for finding linkage solutions, the design system runs the algorithm iteratively over randomized task positions, within acceptable variations provided by the user. This results in the generation of large number of defect-free linkage solutions. These solutions are further checked for a minimum and maximum allowed link length criteria to further filter out linkages. The result is a list of successful linkage solutions that are presented to the user.

The RR constraining strategy for the user specified robot has provided some useful insights in to linkage design. It has been observed that as we increase the complexity of the starting serial or closed chain robot, the number of linkage solutions that could be synthesized increases rapidly. This increases the probability of finding useful solutions and hence more design options can be presented to the designer for selection. The design system has been implemented in a computer aided linkage software system called MechGen for four-bar and six-bar linkages, and has demonstrated its effectiveness to design better devices for several applications.

Chapter 1

Introduction

The process of designing a mechanical device to perform a desired task is often referred to as synthesis of linkages or mechanisms. Linkage design is often considered as a creative process and requires a combination of knowledge, experience and intuition. As the mechanism synthesis science has developed over the last 100 years, this process is becoming more rational.

Linkages are generally preferred over robots for operations involving repetitive motion, owing to their inherent structural strength and cost effectiveness due to less actuators. For a robot, in order to achieve the desired motion, the various motors driving the links are syncronized under software control. On the other hand, in a linkage, mechanical constraints in the form of rigid links are used instead of software, to control relative motion between the various links. This enables us to use less number of actuators to get the desired motion, often turning out to be more cost effective for specialized tasks.

The motivation for the proposed research is to remove intuition out of the design process and use the computer to rapidly search through the entire design space for all the feasible solutions and present them to the user in the CAD package itself. The dissertation is an attempt to provide an automated methodology for designing eight-bar linkages so that it lends itself naturally to be used on a computer. The user can specify either a 3 degree-of-freedom (DOF) 6R loop robot or a 4 DOF serial robot, that can reach all the five task positions. For the 6R loop case, two RR constraints are required to constrain it to single DOF eight-bar linkage, whereas in case of 4R chain, three RR constraints are required.

Following the synthesis, each eight-bar candidate linkage is analyzed to find the forward kinematic solutions. For a particular input link angle, each forward kinematics solution depicts a linkage configuration, which represents a specific way to assemble the various links of the eight-bar linkage. As the input link in rotated in the range, all the different configurations are tracked and then sorted into branches. A defect free linkage has all the five linkage configurations at the five task positions, lying on the same branch.

A linkage qualifies as a successful linkage if it satisfies two requirements: i) it is defect-free and ii) the shortest and the longest link have to be within certain limits, specified by the user. This strategy of finding successful eight-bar linkages ensures smooth movement of the end-effector through the five task positions.

Providing the user freedom to select the 6R loop or 4R chain reduces the scale of the problem, in terms of the computation requirement, as less number of unknowns are required to be solved. It is also very natural to start with a parallel or serial chain robot to achieve certain positions inside its workspace. Such a specification also results in to greater control over the synthesized linkage. A similar remark can be made about specification of only five task positions. It is well known that an RR dyad can be synthesized exactly for up to five prescribed poses, and extensive research has been reported on this Burmester problem using different approaches.

This method of constraining the open and closed chain robot can be applied to more complex cases. Since, the application of each RR constraint reduces one degree of freedom (DOF) of the system; this method can be applied to partially constrain robots to achieve dexterous work-space manipulation. The vision behind this research work is to provide an expert rule based system that can help designer to create machines that can achieve complex motion characteristics. The primary feature for such a system would be to offer the engineer lot of linkage design choices and also an easy to use user interface to make changes to the design requirements. Such an system will greatly enhance project outcomes. The software MechGen was created with this intention for synthesizing planar and spherical four-bar and six-bar linkages and has been successfully used by students in undergraduate and graduate classes at UC Irvine for their design projects.

1.1 Overview of Linkage Design

Linkages can be broadly classifies based on the type of motion they perform, namely, motion, path and function generation. In motion generation, the requirement is to guide a body through certain positions, such that both the position and orientation of the body is specified. In path generation, a specific point on the coupler link is required to trace a certain path. In case of function generation, the output link rotates at an angle which follows a certain function of the input crank angle. This dissertation focuses on motion generating six-bar and eight-bar linkages that can achieve five task positions.

Eight-bar linkages, in general, have the advantage of achieving more complex motion characteristics compared to four-bars or six-bars. Wunderlich [1] derived an expression for the highest degree possible for a coupler curve of a linkage connected by revolute joints. This expression gives degree 6, 18 and 54 for the four-bar, six-bar and eight-bar linkage coupler curves respectively. Freudenstein et al. [2] showed the complexity of coupler curves for six-bar linkages is upto degree 16 and for a specific eight-bar topology, it is degree 30.

1.2 Literature Review

Given the task specification motion/path/function generation or their combinations, the next step in design is type synthesis and dimensional synthesis [3]. In type synthesis, the topology of the linkage is found whereas in dimensional synthesis the physical dimensions of the linkage are calculated, that can perform the desired task. Mueller (1953) [4] proposed a graphical approach for eight-bar linkage synthesis, to coordinate motion of input, output and coupler links. Hain (1967) [5] demonstrated a graphical procedure to design an eight-bar linkage, to simultaneously produce two rectilinear displacements in arbitrary orientations, for four prescribed positions. Soni (1973) [6] proposed a technique to synthesize the double flier eight-bar linkage, having five links in each of the its three loops, for a varieties of motion programs namely, motion, path, function and their combinations.

Freudenstein, Primrose and Roth (1966) [2] studied the coupler curves for the eight-bar linkages. Soni (1971) [7] studied the existence of coupler cognate mechanisms for a class of eight-bars, while Pennock (2005) [8] studied the coupler curves of the double flier eight-bar linkage.

Angeles and Chen [9,10] developed a method to synthesize a specific eight-bar topology, that can reach 11 task positions exactly. Their method couples two four-bar solutions to guide a coupler through the 11 task positions. Soh and McCarthy [11–13] introduced a synthesis procedure for the design of eight-bar linkages as mechanically constrained 6R parallel robots with revolute and prismatic joints. Sonawale and McCarthy [15–18] developed this procedure further to systematically find all possible eight-bars by applying two RR constraints to the user defined 6R loop, and three RR constraints to the 4R serial chain. Soh [19] also worked on synthesis of mechanically constrained 4R chains. Central to this process is the calculation of an RR dyad that connects two relatively moving bodies introduced by Burmester(1888) [20], also Sandor(1984) [21]. For a discussion of Burmester's work see [22].

By allowing the task specifications to be randomized within acceptable variations, and running the design algorithm iteratively on these randomized tasks, more linkage solutions can be generated. Sonawale and McCarthy [14, 15] have demonstrated the effectiveness of this method for the design of planar and spherical linkages.

Analysis of an linkage is finding the forward kinematics solutions for a linkage. Neilson and Roth [23], and Wampler [24] applied the Dixon determinant elimination method [25, 26] for analyzing planar mechanisms. Dhingra [27] used several elimination techniques to reduce the forward kinematics problem for eight-bars to an univariate polynomial. Parrish and McCarthy [28–30] developed an algorithm for automated generation of loop equations, which is the first step towards analysis using the Dixon determinant. It uses the adjacency matrix of a linkage graph to characterize the design, [31]. This dissertation employs their algorithm for the forward kinematics analysis.

F. Freudenstein and G. N. Sandor [32] initiated the computer aided design of linkages by assembling the loop equations of a four-bar linkage in a set of task positions, and then using the newly developed digital computer to solve these equations to determine its dimensions. R. E. Kaufman's KINSYN [33, 34] was a combination of computer aided solution to polynomial equations and a graphical user interface display that united Freudenstein's approach with the geometrical methods of Burmester [20] to form *KINSYN*. This was followed by A. G. Erdman's *LINCAGES* system [35, 36] and K. J. Waldron's *RECSYN* system [37]. Computer graphics based linkage synthesis has been extended to spherical four-bar linkages by Ruth, Larochelle and McCarthy [38,39] and to spatial chains by Collins and Perez [40,41]. Recently, Kinzel et al (2006, 2007) [42,43] showed that the kinematic synthesis of a four-bar linkage can be integrated into a solid modeling package using the constraint solver that is part of the sketch mode.

Automation of linkage synthesis and analysis is crucial to the development of a standalone computer aided design software. Soni (1988) [44] developed a rule-based expert system for mechanism type selection and dimensional synthesis for six and eight-bar linkages, for a variety of motion programs. It is based on finding unique paths that connect a coupler link and the ground link to themselves or other coupler links, thus providing constraints for the design equations. Hansen (1993) [45] also worked on a computer program for automated synthesis of mechanisms by generating solutions to mutually independent vector loop equations. For finding solutions, both authors used numerical continuation methods, to solve the highly non linear synthesis equations.

Many researchers explored optimization and genetic algorithm for simultaneous type and dimensional synthesis. Liu and McPhee (2007) [46] gives a good overview of this.

1.3 Contribution

This dissertation provides the first systematic design algorithm for eight-bar linkages using RR constraints for the backbone chain, namely (a)constraining a 6R loop using two RR constraints, and (b)constraining a 4R serial chain using three RR constraints. The algorithm is topology independent and hence performs both type and dimensional synthesis. Out of the possible 16, 15 topologies could be synthesized using this design system. It also ensures that the synthesized linkages are verified for performance and are branch defect-free.

The design system allows user to specify a backbone chain robot that can achieve the desired task. This is specification of 12 design parameters (x, y) for six pivots in case of 6R loop and eight parameters (x, y) for four pivots in case of 4R serial chain. It then constraints these chains to generate multi-bar linkages. This approach allows the user greater control over the location of ground pivots, location of moving pivots, overall shape of the linkage, and its movement.

In addition the design system also successfully employs randomization of the task positions within user defined tolerance zones to generate more linkage solutions.

Since the design system performs search for linkage solutions with no built in assumption of topology, it lends itself to be run as an automated system. This design algorithm was successfully implemented in a computer aided linkage design software MechGen for four-bar and six-bar linkages, as an add-in for SolidWorks. MechGen allows the user to specify the desired task in the SolidWorks sketch environment and computes linkage solutions as assemblies that the designer can use in SolidWorks. This software provides a rapid product development environment and has generated quite an interest with JPL and NASA Langley Research Center, with LaRC buying a copy of our software.

Chapter 2

Mathematical Background

In this chapter the fundamental mathematical concepts will be laid out. The first step in any linkage design process is a way to capture the motion requirement in a mathematical formulation. In linkage design the motion requirement is often broken down as finite task specifications. For example, for designing a four-bar linkage, the required motion of the end effector is broken down into five snap shots of the end-effector covering the entire motion. This first abstraction of the motion lends itself to a set of polynomial equations, which are solved to generate linkage solutions. These solutions represent the physical dimensions of the linkage, that when built into a physical model will ensure that the desired movement is achieved. A broad classification for linkages based on the motion they can achieve, is given as - motion, path and function generation. In this dissertation, the research focuses on the motion generation problem only. Specifically the design of six-bar and eight-bar linkages is explored to perform specialized motion.

2.1 Linkage Synthesis Theory

The primary design building block for the design of linkages dealt in this dissertation, is the synthesis of an RR constraint. An RR constraint is a binary link with two revolute joints. Every RR constraint added between the various links of the user specified backbone chain, will take away a degree of freedom of the system. This could be referred to as mechanical programming to constrain the relative movement of various links in a linkage.

In this dissertation, the formulation of Burmester's synthesis equations is used, which assumes that the RR constraint connects a moving body M to a fixed body F. For the purposes of this design system, a generalization is made by assuming five positions of the first moving body represented by frames M_j and five positions of the second moving body represented by frames F_j, $j = 1, \ldots 5$, which are known. The synthesis equations for the planar and spherical RR constraint differ slightly and hence are dealt separately in the following sub-sections.

2.1.1 Planar RR Constraint

Let the 3×3 homogeneous transformations $[R_j]$ and $[S_j]$ define the position and orientation of M_j and F_j, $j = 1, \ldots 5$, respectively, in the ground frame, given by,

$$[R_j] = \begin{bmatrix} \cos\gamma_j & -\sin\gamma_j & a_j \\ \sin\gamma_j & \cos\gamma_j & b_j \\ 0 & 0 & 1 \end{bmatrix}, \; [S_j] = \begin{bmatrix} \cos\sigma_j & -\sin\sigma_j & c_j \\ \sin\sigma_j & \cos\sigma_j & d_j \\ 0 & 0 & 1 \end{bmatrix}, \quad j = 1, \ldots, 5, \tag{2.1}$$

where γ_j and σ_j represents the orientations of the frames $[R_j]$ and $[S_j]$ respectively, and (a_j, b_j) and (c_j, d_j) represent their locations. Let $\mathbf{w} = (w_x, w_y, 1)$ be the homogeneous coordinates of point fixed in the frame M and, similarly, let $\mathbf{g} = (g_x, g_y, 1)$ be fixed in

F, so that,

$$\mathbf{W}^j = [R_j]\mathbf{w}, \quad \mathbf{G}^j = [S_j]\mathbf{g}, \quad j = 1, \dots, 5. \tag{2.2}$$

The constraint equations for an RR crank that connect the frames M_j and F_j, $j = 1, \dots 5$, are given by,

$$(\mathbf{W}^j - \mathbf{G}^j) \cdot (\mathbf{W}^j - \mathbf{G}^j) = R^2, \quad j = 1, \dots, 5, \tag{2.3}$$

where the dot denotes the usual vector dot product, and R is a constant that defines the length of the RR crank. These five equations can be solved for the coordinates of $\mathbf{w}$ and $\mathbf{g}$ and the length R.

It is convenient to reformat the equations in (2.2) so that the coordinates of the RR crank pivots are defined in the ground frame G as $\mathbf{W}^1 = (x, y, 1)$ and $\mathbf{G}^1 = (u, v, 1)$. This is done by introducing the relative transformations,

$$[R_{1j}] = [R_j][R_1]^{-1} \quad [S_{1j}] = [S_j][S_1]^{-1}, \quad j = 1, \dots, 5, \tag{2.4}$$

so that

$$\mathbf{W}^j = [R_{1j}]\mathbf{W}^1 \quad \mathbf{G}^j = [S_{1j}]\mathbf{G}^1, \; j = 1, \dots, 5. \tag{2.5}$$

The constraint equations for the RR crank now take the form,

$$([R_{1j}]\mathbf{W}^1 - [S_{1j}]\mathbf{G}^1) \cdot ([R_{1j}]\mathbf{W}^1 - [S_{1j}]\mathbf{G}^1) = R^2, \; j = 1, \dots, 5. \tag{2.6}$$

Subtract the first of the equations (2.6) from the remaining to eliminate R^2 and obtain the four bilinear synthesis equations in four unknowns, $\mathbf{r} = (u, v, x, y)$ as,

$$([R_{1j}]\mathbf{W}^1 - [S_{1j}]\mathbf{G}^1) \cdot ([R_{1j}]\mathbf{W}^1 - [S_{1j}]\mathbf{G}^1)$$
$$-(\mathbf{W}^1 - \mathbf{G}^1) \cdot (\mathbf{W}^1 - \mathbf{G}^1) = 0, \; j = 2, \ldots, 5. \tag{2.7}$$

The solution of these synthesis equations yields as many as four sets of design parameters, $\mathbf{r} = (u_i, v_i, x_i, y_i)$, $i = 1, 2, 3, 4$, defining the RR cranks $\mathbf{W}^1\mathbf{G}^1$. See [12], [20] and [21].

2.1.2 Spherical RR Constraint

For the spherical case, the orientation of each link can be captured using three angles $\{\alpha, \beta, \gamma\}$, which are rotations about local axes Y, X and Z respectively. Let angles $\{\alpha_{Mj}, \beta_{Mj}, \gamma_{Mj}\}$ and angles $\{\alpha_{Fj}, \beta_{Fj}, \gamma_{Fj}\}$ define the orientations of frames M_j and F_j, $j = 1, \ldots 5$, respectively. Now 3×3 rotation matrices $[T_j]$ and $[K_j]$ can be used to define the orientation of M_j and F_j as,

$$[T_j] = [Y(\alpha_{Mj})][X(\beta_{Mj})][X(\gamma_{Mj})],$$
$$[K_j] = [Y(\alpha_{Fj})][X(\beta_{Fj})][X(\gamma_{Fj})], \;\; j = 1, \ldots, 5, \tag{2.8}$$

where,

$$[Y(\alpha)] = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \;\; [X(\beta)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix},$$
$$[Z(\gamma)] = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.9}$$

Let $\mathbf{w} = (w_x, w_y, 1)$ be the homogeneous coordinates of the joint axis direction in frame M and, similarly, let $\mathbf{g} = (g_x, g_y, 1)$ be the joint axis direction in frame F, so that

$$\mathbf{W}^j = [T_j]\mathbf{w}, \quad \mathbf{G}^j = [K_j]\mathbf{g}, \quad j = 1, \dots, 5. \tag{2.10}$$

The constraint equations for an spherical RR crank that connects the joints $\mathbf{W^j}$ and $\mathbf{G^j}$, $j = 1, \dots 5$, are given by,

$$\mathbf{W}^j \cdot \mathbf{G}^j = |\mathbf{W}^j||\mathbf{G}^j| \cos\rho, \quad j = 1, \dots, 5, \tag{2.11}$$

where the dot denotes the usual vector dot product, and ρ is a constant angle that defines the angular length of the spherical RR crank.

Again, It is convenient to reformat the equations in (2.10) so that the coordinates of the RR crank pivot directions are defined in the ground frame G as $\mathbf{W}^1 = (x, y, 1)$ and $\mathbf{G}^1 = (u, v, 1)$. This is done by introducing the relative transformations,

$$[T_{1j}] = [T_j][T_1]^{-1} \quad [K_{1j}] = [K_j][K_1]^{-1}, \quad j = 1, \dots, 5, \tag{2.12}$$

so that

$$\mathbf{W}^j = [T_{1j}]\mathbf{W}^1 \quad \mathbf{G}^j = [K_{1j}]\mathbf{G}^1, \quad j = 1, \dots, 5. \tag{2.13}$$

The constraint equations for the RR crank now take the form,

$$[T_{1j}]\mathbf{W}^1 \cdot [K_{1j}]\mathbf{G}^1 = |[T_{1j}]\mathbf{W}^1||[K_{1j}]\mathbf{G}^1| \cos\rho = |\mathbf{w}||\mathbf{g}| \cos\rho, \qquad j = 1, \dots, 5. \tag{2.14}$$

Subtract the first of the equations (2.14) from the remaining to eliminate $|\mathbf{w}||\mathbf{g}|\cos\rho$ and obtain the four bilinear synthesis equations in four unknowns, $\mathbf{r} = (u, v, x, y)$ as,

$$[T_{1j}]\mathbf{W}^1 \cdot [K_{1j}]\mathbf{G}^1) - \mathbf{W}^1 \cdot \mathbf{G}^1 = 0, \qquad j = 2, \ldots, 5. \tag{2.15}$$

The solution of these synthesis equations yields as many as six sets of design parameters, $\mathbf{r} = (u_i, v_i, x_i, y_i)$, $i = 1, \ldots, 6$, defining the spherical RR cranks $\mathbf{W}^1\mathbf{G}^1$.

2.2 Linkage Analysis

Analysis of single degree-of-freedom linkage is also referred to as it's direct or forward kinematics. It involves determining values for all the joint parameters for a given position of the input link. Since the governing kinematic loop closure equations are non-linear in nature, multiple solutions are generally obtained for a single position of the input link. These solutions can be thought of as different configurations possible for the linkage, for the same position of the input link as discussed in Myszka et al. (2012) [54].

The solution methods for these nonlinear kinematic equations can be broadly divided into two classes: numerical (iterative) methods and closed form (analytical) methods. The commonly used iterative methods are variants of either the Newton-Raphson or conjugate gradient methods. These methods require an initial guess at the solution. If the initial guess is not close enough to a solution, the iterative process may converge slowly, converge to an unacceptable solution, or may diverge altogether. As iterative methods yield only one solution closest to the starting guess a repeated application of it is required. But even with multiple starting guesses, it is not guaranteed to yield all possible solutions. A bootstrap method was developed by Roth and Freudenstein (1963) [55] to find all possible solutions. A refined

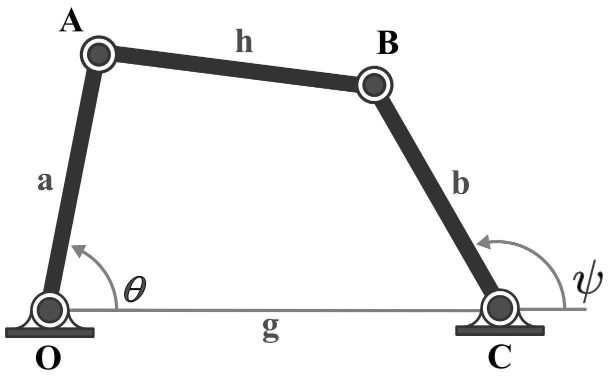


Figure 2.1: Planar four-bar linkage

form of this method known as the homotopy or numerical continuation methods developed by Wampler et al. [50], is capable of finding all possible solutions and eliminate the need for a good initial estimate to the solution. Closed form solutions to kinematic equations can be obtained using elimination theories based on resultants or Grobner bases. In general, there are two types of resultants: the first based on elimination of one variable; the second attempts to eliminate all but one variable simultaneously, see Almadi et al.(1999) [56] .

For the four-bar linkages, the loop closure constraint equation yields all the configuration angles required for its assembly as described in McCarthy (2010) [12]. Even the six-bar Watt I linkage, owing to its special topology of a four-bar on top of another four-bar, results in a straight forward method for the analysis as discussed later in this section. The analysis is very important because the synthesis routine can only guarantee that the candidate linkage will reach the five positions, but it might happen that the linkage fails during its motion in between these task positions. The linkage is said to be a successful linkage if the input crank when turned within the range, results in the end effector moving smoothly through all the five task positions. This information is used to animate the linkage.

2.2.1 Planar Four-bar Analysis

The process of analyzing a planar four-bar is explained in McCarthy (2010) [12] and is summarized as follows. Figure 2.1 shows a planar four-bar linkage **OABC** constructed by connecting the end links of two RR chains viz. **OA** and **CB**. Let a, b, h, g be the link lengths of the links **OA**, **CB**, **AB** and **OC** respectively.

The relationship between the input crank angle and output crank angle is given by the mechanical constraint that the points **A** and **B** remain at a fixed distance apart h, throughout the motion of the linkage. This constraint equation is given by

$$(\mathbf{B}-\mathbf{A}).(\mathbf{B}-\mathbf{A}) = h^2, \tag{2.16}$$

which is used to determine the output angle ψ as a function of the input angle θ. The coordinates of **A** and **B** are given by

$$\mathbf{A} = \begin{Bmatrix} acos\theta \\ asin\theta \end{Bmatrix}, \quad \mathbf{B} = \begin{Bmatrix} g + bcos\psi \\ bsin\psi \end{Bmatrix}, \tag{2.17}$$

Substituting this in the Eqn. (2.16) the following equation is obtained,

$$b^2 + g^2 + 2gbcos\psi + a^2c - 2(acos\theta(g + bcos\psi) + absin\theta sin\psi) - h^2 = 0 \tag{2.18}$$

Collecting the coefficients of $cos\psi$ and $sin\psi$ in this equation, the constraint equation takes the form

$$A(\theta)cos\psi + B(\theta)sin\psi \;=\; C(\theta) \tag{2.19}$$

where

$$\begin{aligned} &A(\theta) = 2abcos\theta - 2gb, \\ &B(\theta) = 2absin\theta, \\ &C(\theta) = g^2 + b^2 + a^2 - h^2 - 2agcos\theta. \end{aligned} \tag{2.20}$$

This equation has the solution given by

$$\psi(\theta) = arctan\left(\frac{B}{A}\right) \pm arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right) \tag{2.21}$$

It is important to note here that there are two output angles ψ associated with each input angle θ. The two angles result from the fact that the triangle Δ **ABC** can be assembles with **B** on either side of the diagonal **AB**. The angle $\delta = arctan(B/A)$ locates the diagonal **AC** from the ground link and $k = arccos(C/\sqrt{A^2+B^2})$ is the angle above and below this diagonal that locates the driven crank.

The argument of the arc-cosine function must be in the range -1 to $+1$. This can be used along with the input angle θ to impose a condition on **A**, **B** and **C** as given below

$$A(\theta)^2 + B(\theta)^2 - C(\theta)^2 \geqslant 0 \tag{2.22}$$

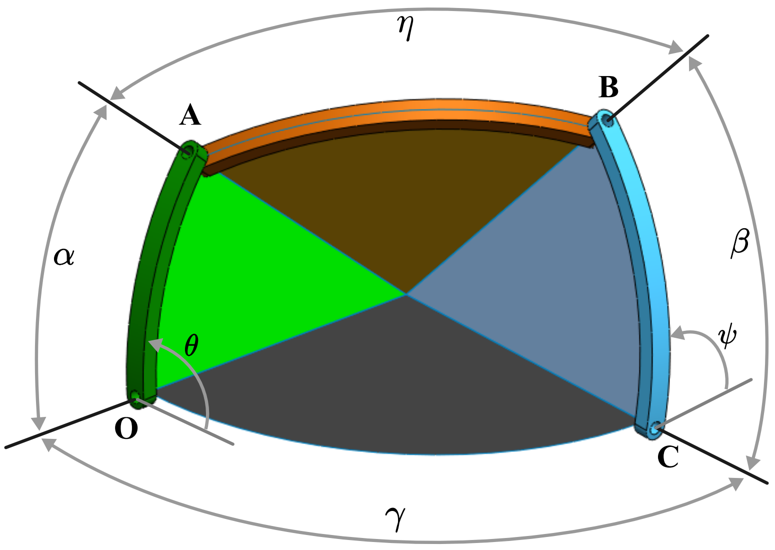


Figure 2.2: Spherical four-bar linkage

If this constraint is not satisfied then the linkage cannot be assembled for the specified value of θ. These two values for ψ give us all the information needed for the analysis. This is a general analysis procedure for the planar four-bar linkage.

2.2.2 Spherical Four-bar Analysis

The process of analyzing a spherical four-bar is similar to the planar four-bar as explained in McCarthy (2010) [12] and is summarized as follows. Figure 2.2 shows a spherical four-bar linkage **OABC** constructed by connecting the end links of two spherical RR chains viz. **OA** and **CB**. Let α, β, η, γ be the angular lengths of the spherical links **OA**, **CB**, **AB** and **OC** respectively. The relationship between the input crank angle and output crank angle is given by the mechanical constraint that the angular dimension η between the moving axes **A** and **B** is constant throughout the movement of the linkage. This constraint equation is given by

$$\mathbf{A}.\mathbf{B} = cos\eta, \tag{2.23}$$

which is used to determine the output angle ψ as a function of the input angle θ. The coordinates of $\mathbf{A}$ and $\mathbf{B}$ are given by

$$\mathbf{A} = \left\{ \begin{array}{c} cos\theta sin\alpha \\ sin\theta sin\alpha \\ cos\alpha \end{array} \right\}, \quad \mathbf{B} = \left\{ \begin{array}{c} cos\gamma cos\psi sin\beta + sin\gamma cos\beta \\ sin\psi sin\beta \\ -sin\gamma cos\psi sin\beta + cos\gamma cos\beta \end{array} \right\}, \tag{2.24}$$

Substituting this in the Eqn. (2.23) the following equation is obtained,

$$c\theta s\alpha(c\gamma c\psi s\beta + s\gamma c\beta) + s\theta s\alpha s\psi s\beta + c\alpha(-s\gamma c\psi s\beta + c\gamma c\beta) \; = \; c\eta \tag{2.25}$$

where s and c denote the *sine* and the *cosine* functions

Collecting the coefficients of $cos\psi$ and $sin\psi$ in this equation, the constraint equation takes the form

$$A(\theta)cos\psi + B(\theta)sin\psi \; = \; C(\theta) \tag{2.26}$$

where

$$\begin{aligned}
A(\theta) &= cos\theta\, sin\alpha\, cos\gamma\, sin\beta - cos\alpha\, sin\gamma\, sin\beta,\\
B(\theta) &= sin\theta\, sin\alpha\, sin\beta,\\
C(\theta) &= cos\eta - cos\theta\, sin\alpha\, sin\gamma\, cos\beta - cos\alpha\, cos\gamma\, cos\beta.
\end{aligned} \tag{2.27}$$

This equation has the solution given by

$$\psi(\theta) = arctan\left(\frac{B}{A}\right) \pm arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right) \tag{2.28}$$

It is important to note here that there are two output angles ψ associated with each input angle θ. The two angles result from the fact that the spherical triangle Δ **ABC** can be assembles with **B** on either side of the diagonal **AB**. The angle $\delta = arctan(B/A)$ locates the diagonal **AC** from the ground link and $k = arccos(C/\sqrt{A^2+B^2})$ is the angle above and below this diagonal that locates the driven crank.

The argument of the arc-cosine function must be in the range -1 to $+1$. This can be used along with the input angle θ to impose a condition on **A**, **B** and **C** as given below

$$A(\theta)^2 + B(\theta)^2 - C(\theta)^2 \geqslant 0 \tag{2.29}$$

If this constraint is not satisfied then the linkage cannot be assembled for the specified value of θ. These two values for ψ give us all the information needed for the analysis. Again this is a general analysis procedure for the spherical four-bar linkage. For analyzing a linkage obtained by five task positions synthesis, the linkage parameters like link lengths and elbow position are known.

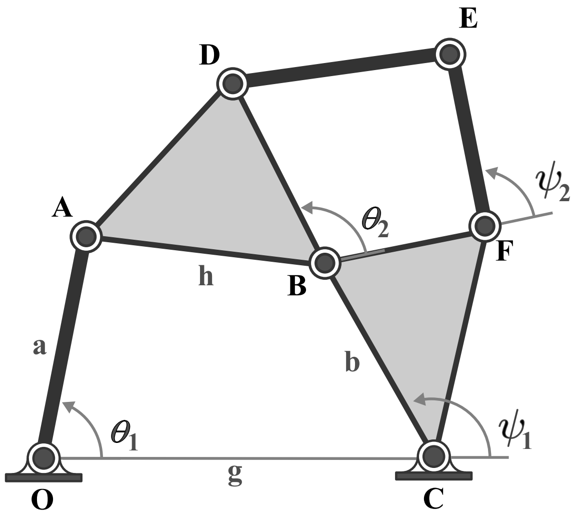


Figure 2.3: Planar six-bar linkage

2.2.3 Planar Watt I Six-bar Analysis

The special topology of the Watt I six-bar linkage converts its analysis problem to an easier problem of analyzing two four-bars. The procedure described earlier for analyzing four-bar linkage, gives the two output $\psi1$ angles associated with the two configurations of the link **CB** for which the four-bar linkage **OABC** can be assembled. Next the second four-bar **BDEF** is considered. As the input link **OA** of the first four-bar **OABC** is rotated, the angle between the two ternary links **ABD** and **CBF** changes and is completely defined by the first four-bar. This angle $\theta2$, measured between the two link vectors **BF** and **BD**, serves as an input angle for the second four-bar with ternary link **CBF** as an instantaneous ground link, as shown in Fig 2.3. Now the two output angles $\psi2$ can be calculated using the method described earlier. Thus the entire planar Watt I six-bar linkage is analyzed.

2.2.4 Spherical Watt I Six-bar Analysis

The methodology used for the planar Watt I six-bar linkage applies to the spherical variant and the analysis for the two spherical four-bars Fig.2.4 will follow the method described

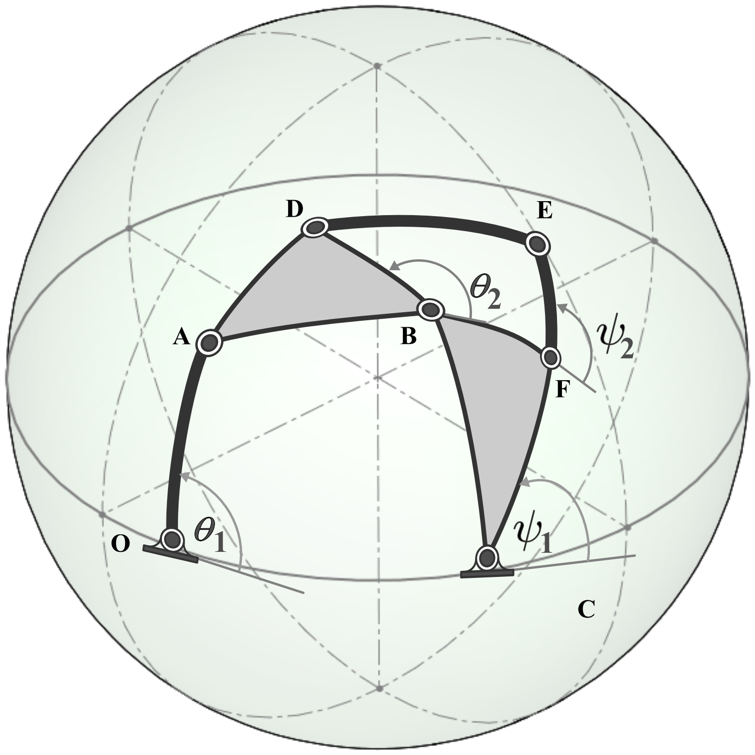


Figure 2.4: Spherical six-bar linkage

earlier.

2.2.5 Planar Eight-bar Analysis

For eight-bar linkages, the design system uses an analysis algorithm described in Parrish and McCarthy [28] for forward kinematics. The algorithm uses an automated loop generation technique based on Graph theory. It uses the Dixon Determinant approach to find all possible solutions for all the unknown joint angles, for a given input angle. This determines every possible assembly configuration for the linkage. This section describes the information exchange by the design system with the analysis algorithm, and post processing the forward kinematic solutions to find useful eight-bar linkages.

Input data to the Analysis Algorithm

The main program, shown in Fig.2.5, uses the analysis routine seamlessly in it. Since the analysis routine is designed for any generic single dof linkage, the linkage information sent to this routine has to be formatted in a specific way. This sub-section describes the formatting of the input data for the analysis routine.

An adjacency matrix is convenient way to represent the connections (revolute joints) between the various links. Consider an example linkage shown in Fig.2.6, the adjacency matrix is given as

$$P_o = \begin{pmatrix} 0&1&0&0&0&1&0&1 \\ 1&0&1&0&0&0&0&0 \\ 0&1&0&1&0&1&0&0 \\ 0&0&1&0&1&0&1&0 \\ 0&0&0&1&0&0&0&1 \\ 1&0&1&0&0&0&1&0 \\ 0&0&0&1&0&1&0&0 \\ 1&0&0&0&1&0&0&0 \end{pmatrix}. \tag{2.30}$$

For the input to the analysis routine, each synthesized candidate linkage is expressed as a modified adjacency matrix $[P]$, with the *1s* in $[P_0]$ replaced by the joint coordinates (x, y) for each of the joints $\{C_1, \ldots, C_{10}\}$. For example, $[P_{1,2}]$ and $[P_{2,1}]$ representing the connection between links a_1 and a_2, will display the coordinates C_2. The input data $\mathbf{V}$, for the analysis algorithm is a list, consisting of the modified adjacency matrix $[P]$, ground link a_8 and the input link number a_1, expressed as:

$$\mathbf{V} = \Big\{[P], \quad a_8, \quad a_1\Big\}. \tag{2.31}$$

For the example linkage the input is,

$$\mathbf{V} = \left\{ \begin{bmatrix} 0 & C_2 & 0 & 0 & 0 & 0 & C_9 & C_1 \\ C_2 & 0 & C_3 & 0 & 0 & C_7 & 0 & 0 \\ 0 & C_3 & 0 & C_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_4 & 0 & C_5 & C_8 & 0 & 0 \\ 0 & 0 & 0 & C_5 & 0 & 0 & 0 & C_6 \\ 0 & C_7 & 0 & C_8 & 0 & 0 & C_{10} & 0 \\ C_9 & 0 & 0 & 0 & 0 & C_{10} & 0 & 0 \\ C_1 & 0 & 0 & 0 & C_6 & 0 & 0 & 0 \end{bmatrix}, 8, 1 \right\} \tag{2.32}$$

Output data from the Analysis Algorithm

The output $\mathbf{W}$ from the analysis routine is,

$$\mathbf{W} = \Big\{ [M], [N], \mathbf{K}, r, \mathbf{s}, \mathbf{t}, [Q], FTLA \Big\}. \tag{2.33}$$

Matrices $\{[M], [N]\}$ form a matrix pair, which will be used to find the unknown joint angles by solving the generalized eigenvalue problem. Vector $\mathbf{K}$ for the example linkage is

$$\begin{aligned} \mathbf{K} = \{ & T_3T_4,\ T_3T_5,\ T_4T_5,\ T_3T_6,\ T_4T_6,\ T_5T_6,\ T_3T_7, \\ & T_4T_7,\ T_5T_7,\ T_3T_4T_6,\ T_3T_5T_6,\ T_4T_5T_6,\ T_3T_4T_7, \\ & T_3T_5T_7,\ T_4T_5T_7,\ T_3T_6T_7,\ T_4T_6T_7,\ T_5T_6T_7\}. \end{aligned} \tag{2.34}$$

Variable r represents which joint angle will correspond to the eigenvalues. Vector $\mathbf{s}$ represent a list of other five joint angles, which will be obtained from the eigenvectors. Vector $\mathbf{t}$ is the ratio index information. Modified adjacency matrix $[Q]$ enables us to make a one to one correspondence between the joint naming conventions used by the analysis routine and this program, by comparing with matrix $[P]$. For the example linkage,

$$[Q] = \begin{bmatrix} 0 & j1t2 & 0 & 0 & 0 & 0 & j1t7 & j8t1 \\ j1t2 & 0 & j2t3 & 0 & 0 & j2t6 & 0 & 0 \\ 0 & j2t3 & 0 & j3t4 & 0 & 0 & 0 & 0 \\ 0 & 0 & j3t4 & 0 & j4t5 & j6t4 & 0 & 0 \\ 0 & 0 & 0 & j4t5 & 0 & 0 & 0 & j5t8 \\ 0 & j2t6 & 0 & j6t4 & 0 & 0 & j7t6 & 0 \\ j1t7 & 0 & 0 & 0 & 0 & j7t6 & 0 & 0 \\ j8t1 & 0 & 0 & 0 & j5t8 & 0 & 0 & 0 \end{bmatrix}. \qquad (2.35)$$

Comparing with the matrix $[P]$, it is found that $j1t2 = C_2, j2t3 = C_3, j3t4 = C_4, j4t5 = C_5, j5t8 = C_6, j2t6 = C_7, j6t4 = C_8, j1t7 = C_9, j7t6 = C_{10}$. $FTLA$ is a compact representation for the three loop equations and will be used not only for the forward kinematics but also for animating the linkage.

Forward Kinematics

This section describes how the solution to the generalized eigenvalue problem is converted to forward kinematic solutions for the joint angles. From the inverse kinematics of the first 3R chain $\{C_1, C_2, C_3\}$ of the 6R loop, the five input angles $\theta_{1,j}$, $j = 1, \ldots, 5$ for a_1 corresponding to the five task positions can be obtained, ref Fig.**??**. These five angles form four angle ranges between them $1-2$, $2-3$, $3-4$, $4-5$. These ranges are discretized to form an array of input angles, $\theta_{1,j}$, $j = 1, \ldots, n$ of length n.

The matrices $[M]$ and $[N]$ have T_1 and T_{c1} as the only unknowns. So a loop of size n is setup, and for each iteration $j = 1, \ldots, n$, the values for $T_{1,j}$ and $T_{c1,j}$ are calculated as $T_{1,j} = e^{i\theta_{1,j}}$ and $T_{c1,j} = e^{-i\theta_{1,j}}$ respectively. These values are substituted in the two matrices $[M]$ and

$[N]$ and the generalized eigenvalue problem is solved as,

$$[N_j]\mathbf{v} = \lambda[M_j]\mathbf{v}, \tag{2.36}$$

where the generalized eigenvalues λ correspond to all the possible joint angles, in isotropic form, for the link specified by the variable r. For the linkage discussed in the example, $r = \theta_2$, which means that the eigenvalues correspond to all the possible angles for link 2, for the given input link angle θ_{1j}. The generalized eigenvectors $\mathbf{v}$ correspond to the other joint angles in isotropic form. It should be noted that $\mathbf{v} = v_i,\ i = 1, \ldots, 18$ is defined up to a constant multiple, say μ. Therefore the other joint angles in isotropic form are found by computing the ratios,

$$\begin{aligned} &T_{3,j} = \frac{v_{10}}{v_5}, \quad T_{4,j} = \frac{v_{10}}{v_4}, \quad T_{5,j} = \frac{v_{11}}{v_4}, \\ &T_{6,j} = \frac{v_{10}}{v_1}, \quad T_{7,j} = \frac{v_{13}}{v_1}. \end{aligned} \tag{2.37}$$

The ratio information is obtained from the vector $\mathbf{t}$ and can be confirmed by checking the corresponding ratios on the vector $\mathbf{K} = K_i,\ i = 1, \ldots, 18$, as shown below:

$$\begin{aligned} &T_{3,j} = \frac{K_{10}}{K_5}, \quad T_{4,j} = \frac{K_{10}}{K_4}, \quad T_{5,j} = \frac{K_{11}}{K_4}, \\ &T_{6,j} = \frac{K_{10}}{K_1}, \quad T_{7,j} = \frac{K_{13}}{K_1}. \end{aligned} \tag{2.38}$$

An isotropic angle T is real if its norm is equal to 1. The angle in radians is extracted from the isotropic angle by finding its argument. For an input angle $\theta_{1,j}$ a maximum of 18 solutions for $\{\theta_{2,j}, \theta_{3,j}, \theta_{4,j}, \theta_{5,j}, \theta_{6,j}, \theta_{7,j}\}$ can be obtained. This means that for a given input angle $\theta_{1,j}$, a maximum of 18 different configurations are possible for the linkage. In actuality this number is a lot smaller. For $j = 1, \ldots, n$, these solutions are compiled as a list, given by $\{\theta_{1,j}, \theta_{2,j}, \theta_{3,j}, \theta_{4,j}, \theta_{5,j}, \theta_{6,j}, \theta_{7,j}, \theta_{8,j}\}$. All solution lists are stored in a matrix $[S]$, which is of size $(8x18xn)$, such that

$$[S_j] = \begin{bmatrix} \theta_{1,j} & \theta_{1,j} & \cdots & \theta_{1,j} & \theta_{1,j} \\ \theta_{2,1,j} & \theta_{2,2,j} & \cdots & \theta_{2,17,j} & \theta_{2,18,j} \\ \theta_{3,1,j} & \theta_{3,3,j} & \cdots & \theta_{3,17,j} & \theta_{3,18,j} \\ \theta_{4,1,j} & \theta_{4,2,j} & \cdots & \theta_{4,17,j} & \theta_{4,18,j} \\ \theta_{5,1,j} & \theta_{5,2,j} & \cdots & \theta_{5,17,j} & \theta_{5,18,j} \\ \theta_{6,1,j} & \theta_{6,2,j} & \cdots & \theta_{6,17,j} & \theta_{6,18,j} \\ \theta_{7,1,j} & \theta_{7,2,j} & \cdots & \theta_{7,17,j} & \theta_{7,18,j} \\ \theta_8 & \theta_8 & \cdots & \theta_8 & \theta_8 \end{bmatrix}_{(8x18)} , j = 1, \ldots, n \qquad (2.39)$$

Note that the angle of the ground link a_0 is constant θ_8. For every $[S_j]$, each column represents one solution or rather one assembly configuration of the eight-bar linkage.

Sorting Branches

In this dissertation, a linkage configuration is denoted by a set of angles $\{\theta_2, \ldots, \theta_7\}$, made by the links $\{a_2, \ldots, a_7\}$. A linkage configuration is a compact representation of how the various links are assembled. For a given input angle θ_1, there can be many configurations possible as discussed before. The forward kinematic solutions $[S_j], j = 1, \ldots, n$ (columns) give us all possible linkage configurations, when the input angle θ_1 is incremented from the starting angle to the ending angle, that is from $\theta_{1,1}$ to $\theta_{1,n}$.

The goal of the sorting algorithm is to track the different configurations and then sort them into branches. Figure 2.7 displays five linkage configurations sorted into branches in each of the plots for the angles $\{\theta_2, \ldots, \theta_7\}$ drawn against the input angle θ_1. The procedure for sorting the branches is described in Plecnik and McCarthy [67] and is explained briefly below.

An eight-bar linkage has three loops. Each loop will have two loop closure equations respectively for the x and y coordinates of the pivots involved in the loop. The loop equations are obtained from the $FTLA$ format, provided by the analysis algorithm and are represented by the vector,

$$\mathbf{f} = \left\{ \begin{array}{c} f_{1x}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \\ f_{1y}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \\ f_{2x}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \\ f_{2y}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \\ f_{3x}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \\ f_{3y}(\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7) \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right\}. \qquad (2.40)$$

Next the Jacobian for the loop equations vector is obtained, $\mathbf{f}$, by taking partial derivatives with respect to the configuration angles $\{\theta_2, \dots, \theta_7\}$ as,

$$[J_f] = \begin{bmatrix} \frac{\partial f_{1x}}{\partial \theta_2} & \cdots & \frac{\partial f_{1x}}{\partial \theta_7} \\ \frac{\partial f_{1y}}{\partial \theta_2} & \cdots & \frac{\partial f_{1y}}{\partial \theta_7} \\ \frac{\partial f_{2x}}{\partial \theta_2} & \cdots & \frac{\partial f_{2x}}{\partial \theta_7} \\ \frac{\partial f_{2y}}{\partial \theta_2} & \cdots & \frac{\partial f_{2y}}{\partial \theta_7} \\ \frac{\partial f_{3x}}{\partial \theta_2} & \cdots & \frac{\partial f_{3x}}{\partial \theta_7} \\ \frac{\partial f_{3y}}{\partial \theta_2} & \cdots & \frac{\partial f_{3y}}{\partial \theta_7} \end{bmatrix}. \tag{2.41}$$

For the sake of being succinct, the linkage configurations of $[S_j]$ (columns) is referred to as $\mathbf{Z}_{j,k}, k = 1, \dots, 18$. Using $\mathbf{Z}_{1,k}$ from $[S_1]$ as the initial condition, the Newton's Raphson method is employed to find the evolution of each $\mathbf{Z}_{1,k}$, when the input angle θ_1 is incremented from $\theta_{1,1}$ to $\theta_{1,2}$. It is referred to it as $\mathbf{Z}_{(2,k)p}$, where p the stands for predicted value. This is given by:

$$\mathbf{Z}_{(2,k)p} = \mathbf{Z}_{1,k} + [J_{\mathbf{f}}(\mathbf{Z}_{1,k})]^{-1}\mathbf{f}(\theta_{1,2}, \mathbf{Z}_{1,k}), \quad k = 1, \dots, 18 \tag{2.42}$$

Next the configurations of $\mathbf{Z}_{(2,k)p}$ are matched with each of the corresponding configurations in $[S_2]$, that is $\mathbf{Z}_{2,k}$ and save the sorted solutions in the list $\mathbf{W}$ as branches. Now the input angle is incremented to $\theta_{1,3}$ and with the sorted $\mathbf{Z}_{(2,k)}$ as initial conditions, Newton-Raphson method is used again to find $\mathbf{Z}_{(3,k)p}$. This process is continued while $j \leq n$ and after each iteration the branches are updated and saved to the list $\mathbf{W}$. Figure 2.7 displays the five sorted branches in each of the 6 plots individually.

Using this information, it can be determined if a particular eight-bar linkage is defect-free. First a search is performed for the branches in the list $\mathbf{W}$ that are of length n. Each of these branches represent a linkage configuration, that exist throughout the motion of the input link θ_1, j for $j = 1, \ldots, n$. This means that, when the input link is rotated, the linkage moves smoothly in that configuration. If a branch is of length less than n, then it means that singularity has occurred at some input angle θ_1, j. This locks the linkage and hence have to be discarded.

If none of the branches in the list $\mathbf{W}$ are of length n, the candidate linkage is discarded. If there are a few branches that are of length n, only these branches are retained in the list $\mathbf{W}$. Now for each complete branch, it is checked, if the end-effector reaches all the five task positions when the linkage moves in this configuration. This is done by verifying whether the five task position configurations lie on a single branch. If yes, then that particular branch is the correct configuration for the linkage and the linkage is deemed defect-free and the configuration saved.

Figure 2.7 shows a defective linkage. For this linkage, three task position configurations 2, 3 and 4 lie on branch3, while task position configuration 1 and 5 lie on branches 5 and 2 respectively. So if the linkage moves in the configuration of branch3, only task positions 2, 3 and 4 will be hit. This verifies that this candidate linkage suffers from branch defect and hence it is discarded. Figure 5.10 shows a defect-free linkage, where all the five task position configurations lie on a single branch (branch1). If the defect-free linkage also satisfies the permissible values for minimum and maximum link length values, specified by the user, it is deemed as useful.

2.3 Linkage Defects

A linkage is said to fail in its operation if it does not satisfy the desired movement requirements of the designer. Thus a motion generating linkage may be said to have failed if it does not guide the moving body through all the task positions.

Defects in mechanisms has always been a important topic in the field of synthesis. Balli and Chand [57] did a comprehensive literature review to capture the major trends and contributions in this field. Filemon [58] was the first researcher to address many of the linkage defects. Waldron et al. [59], Barker et al. [60] and Gupta et al. [61] are the major contributors to this field. Chase and Mirth [62, 63] investigated these linkage defects and proposed improved definitions for the same to eliminate the ambiguity inherent in previous usages of the terms.

Circuit: A circuit of a linkage is defined as all possible orientations of the links which can be realized without disconnecting any of the joints. A mechanism that must be disassembled to move between two positions must be described as "changing circuits."

Branch: A circuit may contain positions of the linkage where the derivative of the angle of the output link with respect to the angle of the driving link becomes infinite, defined as the stationary configurations also referred to as singular positions. If such stationary configurations exist on a circuit, a branch is defined as a continuous series of positions of the mechanism on the circuit between two stationary configurations. Thus, the stationary configurations divide a single circuit into a series of branches. A mechanism that must pass through a stationary configuration, but need not be disassembled, to move between two positions is described as "changing branch within a circuit."

The knowledge of the circuits and branches of a planar mechanism is essential for successfully synthesizing a mechanism and is also useful for analyzing the mechanism. A change in circuit is a stronger defect than a change in branch within a circuit. A mechanism that must change circuits to move between two desired positions is useless, since the mechanism must be disassembled and re-assembled. A mechanism that changes branch may suffer from drivability problems. It is important to note that interchanging the functions of the input and output links will not affect the circuit attributes of the mechanism. In fact, any of the moving links could be selected as driver without affecting the circuit attributes. However the branch attributes change substantially if the functions of the input and output links are interchanged.

Subtle circuit changes are very common in multi-loop mechanisms. Multi-loop mechanisms may have many more circuits than four-bars. For example, the standard six-bar chains may have up to six possible circuits (Primrose et al., 1967) [64], while a fourbar can have a maximum of two. Therefore, multi-loop mechanisms are more likely to suffer a change in circuits.

2.3.1 Filtering defect-free linkages in MechGen

Since MechGen currently only deals with four-bar and Watt-I six-bar, which is composed of two four-bars, a common check routine is employed to filter out the defective linkages. Currently the software only allows for single-dof actuation, which is given to the input crank. So the range is calculated for the input link. It is made sure that the task positions lie on the same branch of the same circuit as different circuit would require disassembly and reassembly and different branch would require actuation at the output link which is currently unavailable in the software. The Jacobian information alone is demonstrated to be insufficient to fully quantify the circuit and branch properties of a four-bar linkage as mentioned by Chase et al. [62].

In order to filter the candidate linkages generated by the synthesis process for defects, Mech-

GEN uses the constraint equations Eqn. (2.18) and Eqn. (2.25) to analyze both the planar and spherical four-bar linkages using the theory presented in McCarthy (2010) [12]. Solving this equation yields two values for the output crank angle for a given input crank angle as shown in Eqn. (3.23). The two angles result from the fact the coupler link and the output crank can be assembled in two configurations namely elbow up and elbow down for the same input crank angle, which corresponds to the plus and minus values in the equation,

$$\psi(\theta) = \arctan\left(\frac{B}{A}\right) \pm \arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right). \tag{2.43}$$

.

This equation not only gives us the two possible output crank angles, but also gives us a formula for the range of the input crank angle through the argument of the arccos term which is subjected to bounds with the range -1 to $+1$. If this is not satisfied then the linkage cannot be assembled for the specified input crank angle. The maximum and minimum values for the input crank angle θ are obtained by setting the equality condition in Eqn. (2.22) for planar four-bar and Eqn. (2.29) for spherical four-bar which yields a quadratic equation in $cos\theta$. The solutions to these equations are:

$$\begin{aligned} \theta_{min} &= \pm \arccos\left(\frac{g^2+a^2-(h-b)^2}{2ag}\right), \\ \theta_{max} &= \pm \arccos\left(\frac{g^2+a^2-(h+b)^2}{2ag}\right), \end{aligned} \tag{2.44}$$

for planar and

$$\begin{aligned} \theta_{min} &= \pm \arccos\left(\frac{cos(\eta-\beta) - cos\alpha\, cos\gamma}{sin\alpha\, sin\gamma}\right), \\ \theta_{max} &= \pm \arccos\left(\frac{cos(\eta+\beta) - cos\alpha\, cos\gamma}{sin\alpha\, sin\gamma}\right), \end{aligned} \tag{2.45}$$

for spherical four-bar linkages, where a, b, c, d and $\eta, \beta, \alpha, \gamma$ are the link lengths for the planar and spherical four-bars respectively as shown in Fig.2.1 and Fig.2.2.

The θ_{min} and θ_{max} values define the range of movement of the input crank. The *cosine* function cannot distinguish between $\pm\theta$ so there are actually two limits for each case $\pm\theta_{min}$ and $\pm\theta_{max}$ above and below the link **OC**

MechGen uses Eqn. (3.23) and the θ_{min} and θ_{max} values from the Eqn (2.44) and Eqn (2.45) to filter both the first four-bars and second four-bars. The two checks used to filter defective linkages are:

1. ***The orientation of the output crank with respect to the coupler that is the elbow up or elbow down configuration***

   The elbow of the output crank with respect to the coupler should be either up or down for all the five task positions. This avoids circuit change when the input crank is a crank (range 0 to 2π), branch change when it is a 0-rocker or π-rocker or just rocker, see McCarthy (2010) [12].

2. ***The limits on the input crank***

   The 5 crank angles for the five task positions should lie in the range specified by θ_{min} and θ_{max} or $-\theta_{min}$ and $-\theta_{max}$. This defect occurs in the case, when the input crank is a rocker. The positions might all have elbow up or down, that is they may all seem to lie on the same branch but they might be on a different circuit. So this check prevents circuit change when input crank is rocker.

These two checks are sufficient to take care of circuit or branch defects and the resulting four-bar or Watt I six-bar linkages are usable linkages. The check is independent on the resolution used for the input angle for solving the equations. So a usable, defect-free linkage is guaranteed with this method.

Table 2.1: Four-bar linkage types based on input angle

Sr. no.	θ_{min}	θ_{max}	Linkage Type	Circuits	Branches
1	img	img	Crank Rocker/ Double Crank	2	0
1	img	real	00 / 0π Double Rocker	1	2
1	real	img	$\pi\pi$ / $\pi 0$ Double Rocker	1	2
1	real	real	Rocker crank/ Double Rocker	2	4

It is important to note that MechGen does not find all the possible circuits and branches for the Watt I six-bar linkage. The algorithm only makes sure that the motion of the two four-bar lie in a single branch when the end effector moves through all the five task positions. The order defect mentioned in Balli and Chand [57] is not accounted for in MechGen. Also MechGen does not take in to consideration the folding or near folding linkages. These linkages with real world tolerances and material stiffness can cause problems, especially causing a subtle change in circuits. For a detailed classification of both the planar and spherical linkages, refer McCarthy (2010) [12].

Based on the values of θ_{min} and θ_{max}, imaginary or real, the four-bar linkages can be classified as shown in Tab. 2.1. It is really amazing how much information about the linkage is packed in the two values θ_{min} and θ_{max}. The following section explains the same.

Allowable ranges of motion for four-bar linkages

$\theta_{\mathbf{min}}$: imaginary and $\theta_{\mathbf{max}}$: imaginary

If θ_{min} and θ_{max} are both imaginary, then this means that both the lower limit and the upper limit for the input does not exist. So two circuits exist for this linkage for elbow up and elbow down position of the output crank. Since there are no limits on the input crank, this means that a singular position or stationary configuration when the coupler link and output crank line up, never occurs. Hence there are no branches for any of the two circuits. And because the singular positions never show up, the output crank if in elbow up position will continue in that position for the entire range of the input crank. In order to have it in the elbow down position, it necessitates the disassembly and re-assembly of the output crank in the elbow down position. This means change of circuit occurs. This confirms that two circuits exist for this case. The configuration possible is crank rocker or double crank (both Grashof)as described in McCarthy (2010) [12]. The synthesis procedure can give us a linkage that has elbow up configuration for the first four task positions and elbow down configuration for the fifth task position. If this linkage have these θ_{min} and θ_{max} values, then it can be said that this linkage changes circuits at the fifth task position and hence the linkage has be filtered out.

$\theta_{\mathbf{min}}$: imaginary and $\theta_{\mathbf{max}}$: real

In this case, there is no lower limit for the input crank but the existence of θ_{max} (which gives two input crank angles, positive and negative due to the *arccosine* nature), imposes bounds on the upper limits of the input crank. So in this case there exists a single circuit with two branches. Since there are upper limits on the input crank, this means that a singular position or stationary configuration occurs when the coupler link and output crank line up at each of the two extreme angles. Hence there are two branches, because at the singular position further rotating the crank in the opposite direction can result in either elbow up or elbow down configuration. This causes the linkage to have two branches. It can be observed that, in order to have the output crank in the elbow up or down position, there is no need to disassemble and re-assemble the linkage. This means that there is no change in the circuit and hence this proves that only one circuit exists for this case. The configuration possible is called 00 double rocker and 0π double rocker (both non-Grashof) refer McCarthy (2010) [12]. Now if there is a situation similar to the earlier case and the linkage has these θ_{min} and θ_{max} values, then it can be said that this linkage changes branches at the fifth task position. Again this is problematic situation for this single degree of freedom linkage and hence should be filtered out.

$\theta_{\mathbf{min}}$: real and $\theta_{\mathbf{max}}$: imaginary

In this case, there is no upper limit for the input crank but the existence of θ_{min} (which gives two output crank angles, positive and negative, due to *arccosine* nature), imposes bounds on the lower limits of the input crank. So in this case a single circuit exist with two branches. Since there are lower limits on the input crank, this means that a singular position or stationary configuration occurs when the coupler link and output crank line up at each of the two extreme angles. Hence there are two branches, because at the singular position further rotating the crank in the opposite direction can result in either elbow up or elbow down configuration. This causes the linkage to have two branches. Now it can be observed that, in order to have the output crank in the elbow up or down position, there is no need to disassemble and re-assemble the linkage. This means that there is no change in the circuit and hence proves that only one circuit exists for this case. The configuration possible is called $\pi 0$ double rocker and $\pi\pi$ double rocker (both non-Grashof) refer McCarthy (2010) [12]. Now again if there is a situation similar to the first case and the linkage has these θ_{min} and θ_{max} values then it can be said that this linkage changes branches at the fifth task position. Again this is problematic situation for this single degree of freedom linkage and hence should be filtered out.

$\theta_{\mathbf{min}}$: real and $\theta_{\mathbf{max}}$: real

This is an interesting case. Both the lower and upper limits are defined. Now due to the *arccosine* nature two ranges exist; one from θ_{min} and θ_{max} and other from $-\theta_{max}$ to $-\theta_{min}$. these two ranges give rise to two circuits. Now since the lower limit and upper limit are defined for both the ranges, this means that a singular position or stationary configuration occurs when the coupler link and output crank line up at each of the two extreme angles. Hence there are two branches for each range or circuit, because at the singular position further rotating the crank in the opposite direction can result in either elbow up or elbow down configuration. Thus this case has two circuits and four branches in all. The configuration possible is called Rocker crank (Grashof) and Double Rocker (Grashof), refer McCarthy (2010) [12]. Now again if there is a situation similar to the first case and the linkage has these θ_{min} and θ_{max} values then, it can be said that this linkage changes branches or changes both branches and circuits at the fifth task position. In fact for this tricky case, it might happen that the linkage might change circuits but not branches. Again this is problematic situation for this single degree of freedom linkage and hence should be filtered out.

An important thing to note is that the θ_{min} and θ_{max} values are so valuable that they can give an indication whether the linkage is close to folding itself. For the values of θ_{min} very close to 0 and θ_{max} very close to π, the linkage is very close to folding itself and if possible such linkage solutions should be avoided.

2.3.2 Filtering defect-free eight-bar linkages

For eight-bar linkages as seen in the Linkage Analysis section, an additional linkage defect check is not required as the sorted branches ensure the smooth movement of the end-effector through the five task positions. It is important to note that defect check for eight-bar linkages using the dixon determinant approach can only be as accurate as the resolution of the input link angle. For example, selecting a larger angular step can cause a branch defect to not show up in the analysis. Hence the step size for incrementing the input link angle in its range of motion has to be carefully selected and often selection of a very small angle must be done.

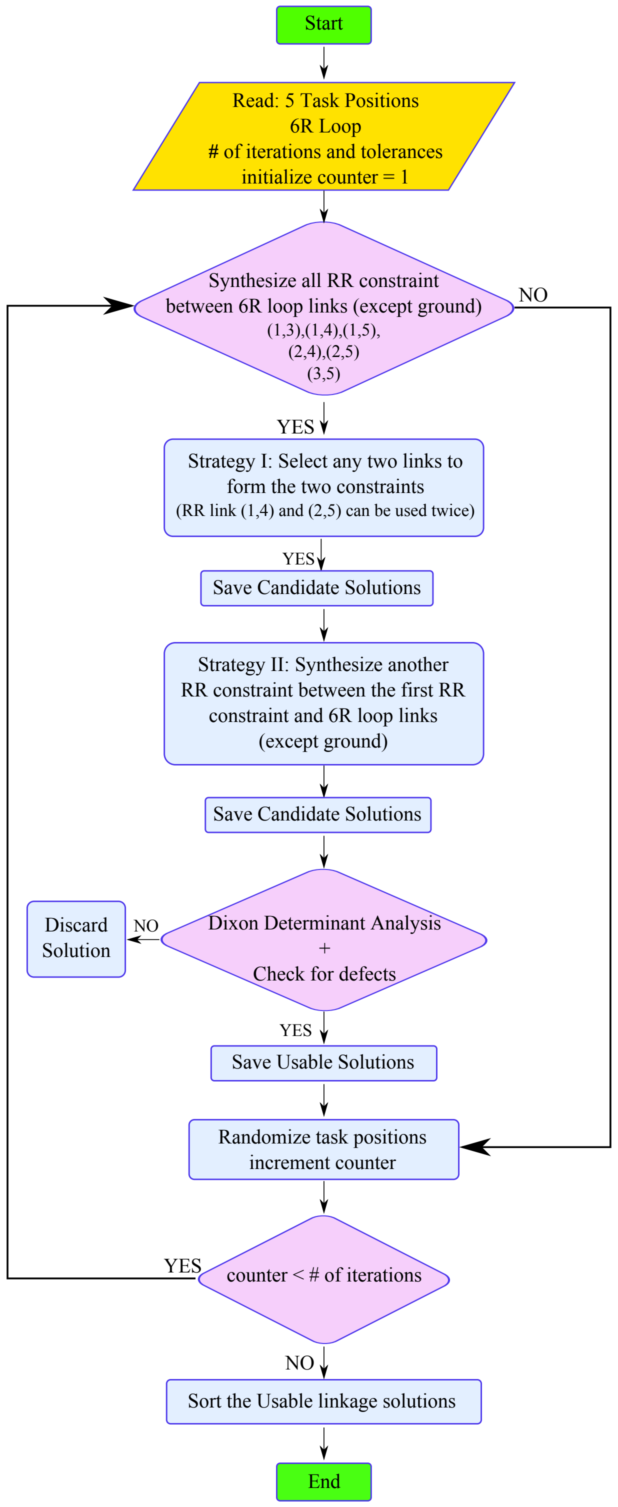
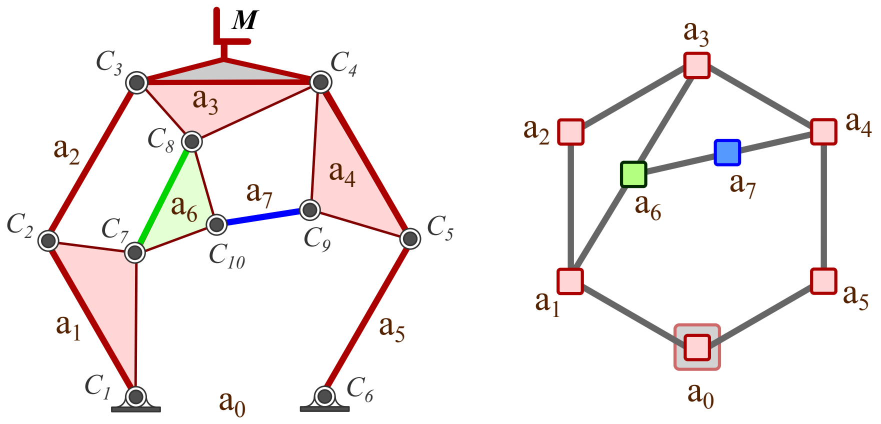

Start
Read: 5 Task Positions
6R Loop
# of iterations and tolerances
initialize counter = 1
Synthesize all RR constraint
between 6R loop links (except ground)
(1,3),(1,4),(1,5),
(2,4),(2,5)
(3,5)
NO
YES
Strategy I: Select any two links to
form the two constraints
(RR link (1,4) and (2,5) can be used twice)
YES
Save Candidate Solutions
Strategy II: Synthesize another
RR constraint between the first RR
constraint and 6R loop links
(except ground)
Save Candidate Solutions
Discard
Solution
NO
Dixon Determinant Analysis
+
Check for defects
YES
Save Usable Solutions
Randomize task positions
increment counter
YES
counter < # of iterations
NO
Sort the Usable linkage solutions
End

Figure 2.5: Design procedure flow chart



🟩 **First RR Crank**
🟦 **Second RR Crank**

Figure 2.6: Example eight-bar linkage topology.

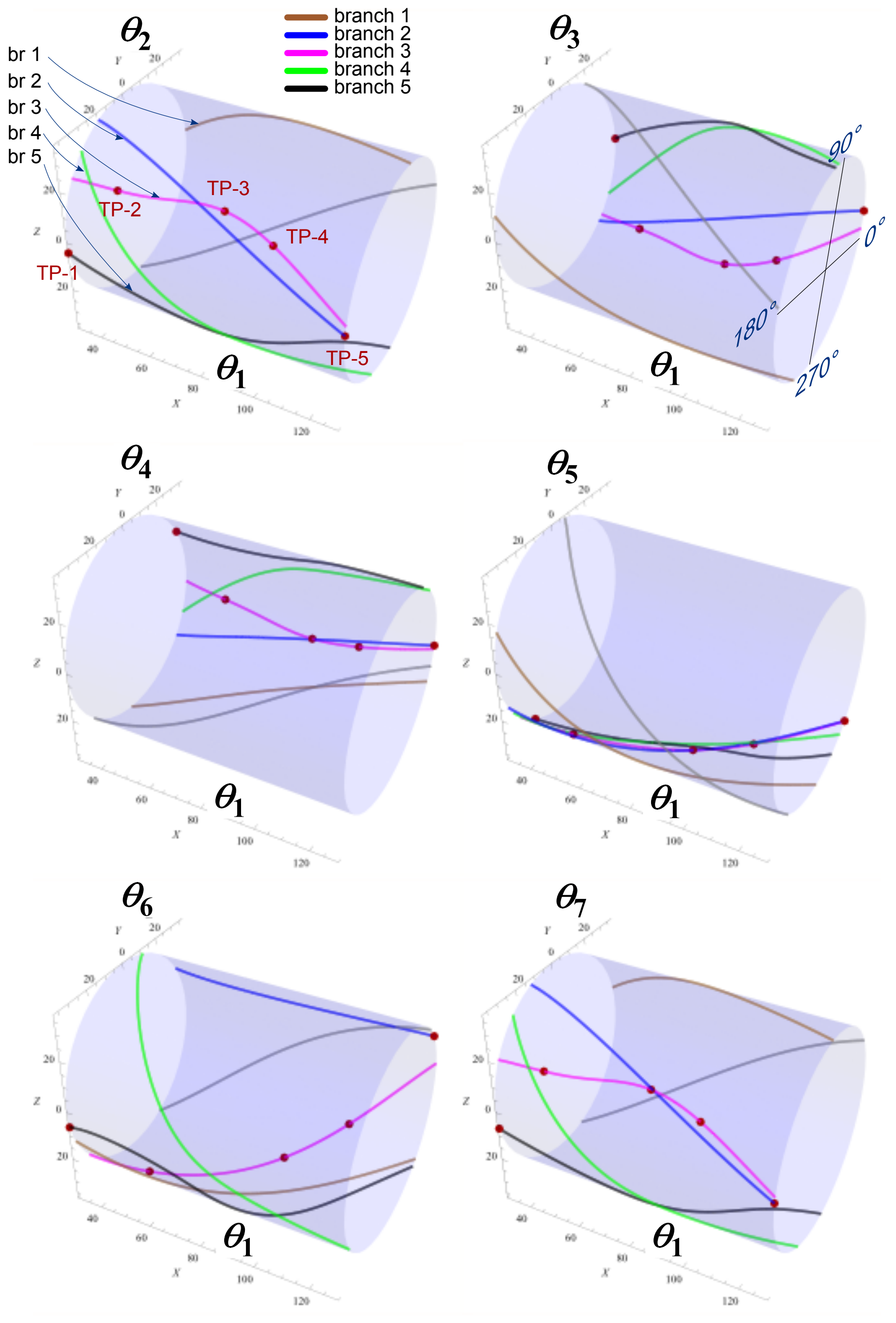


Figure 2.7: Linkage configurations (branches) for an eight-bar linkage suffering from branch defect. Task position configurations 1 and 5 lie on branches 5 and 2 respectively, while task position configurations 2, 3 and 4 lie on the same branch 3.

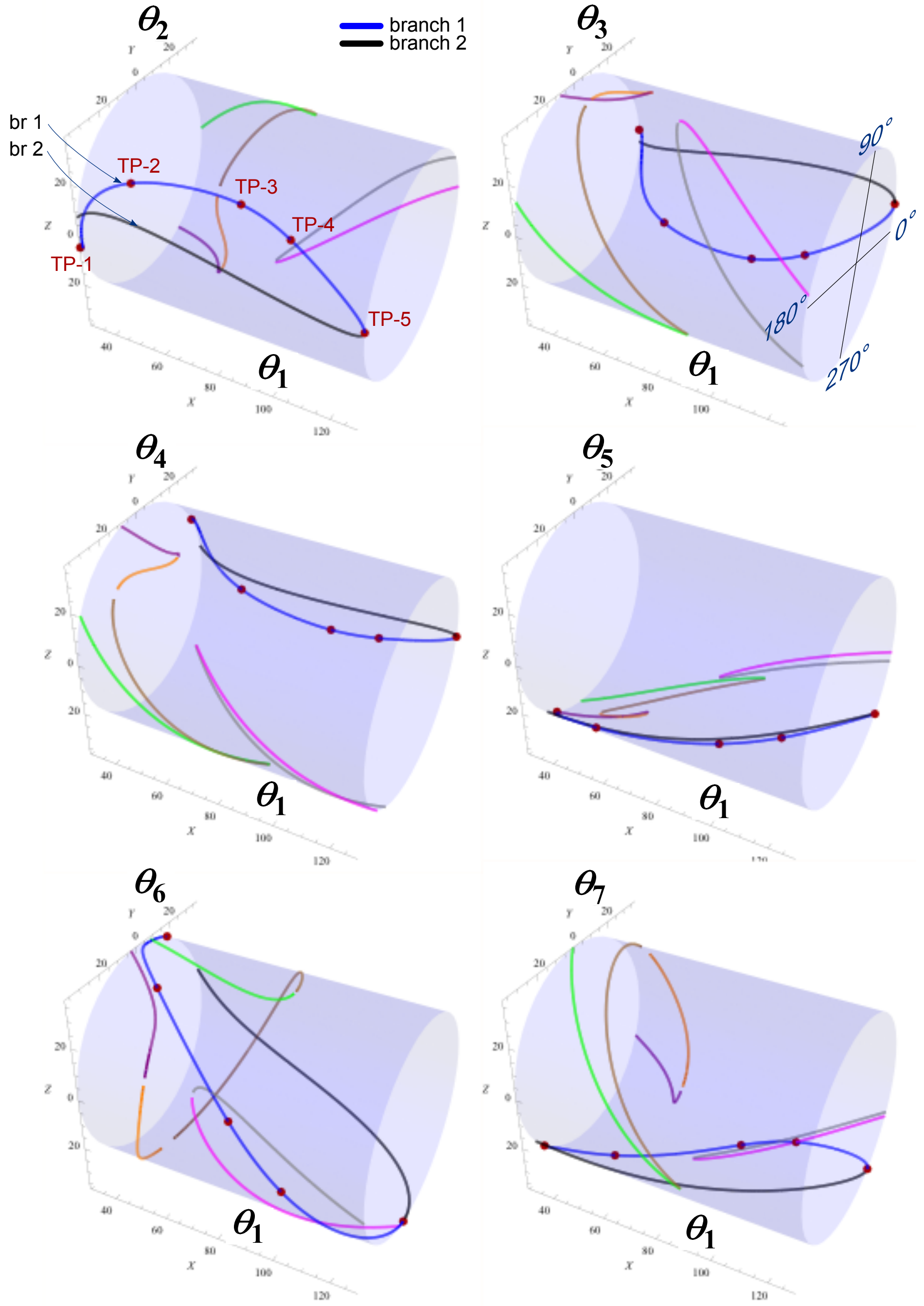


Figure 2.8: Linkage configurations (branches) for a useful eight-bar linkage. All the five task position configurations lie on branch 1.

Chapter 3

Six-bar Linkage Design

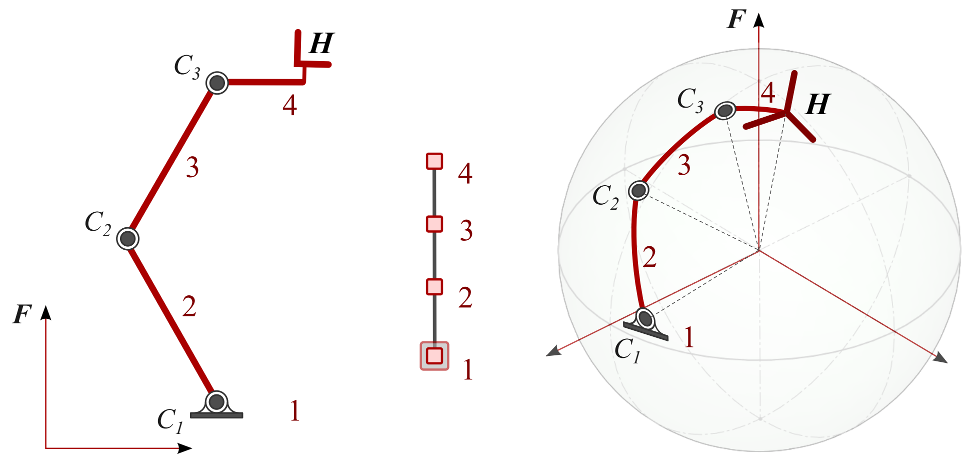


Figure 3.1: A planar and spherical 3R serial chain are shown with joints $C_1, \ldots, C_3$. Body 1 is the ground link and body 4 is the end-effector link. Both chains have the same linkage graph.

In this chapter, the design system for planar and spherical six-bar linkages is presented. The designer specifies five task positions and the backbone 3R chain that can reach these positions. Using the graph theory approach, it is found that there are in all 6 ways to add the two RR constraints to the 3R chain to yield a maximum of 63 planar six-bar linkages and 165 spherical six-bar linkages. These linkages are then verified for branch defects and successful linkages are presented to the designer.

McCarthy and Choe [47] showed that kinematic synthesis equations regularly fail to yield designs that meet the performance requirements, because the task positions fall on separate branches. In order to overcome this challenge, the design system runs the algorithm iteratively over task positions with small random variations added to them, which are within user specified tolerance zones. This increases the probability of finding successful solutions. This system has been incorporated in a computer aided linkage design package MechGen, see Sonawale et al. (2013) [14], and has been thoroughly tested in classes at both undergraduate and graduate level.

A six-bar spherical example of a car door linkage is used to explain the design process for MechGen. The most important feature of this software is its integration with SolidWorks. MechGen allows the user to specify the task requirements along with the backbone chain in the sketch environment of SolidWorks. It reads the input from the sketch and generates solutions, which displayed as animations in its user interface. For the selected linkage, it generates a SolidWorks assembly, which the user can incorporate in his/her parent assembly. This enables for rapid prototyping of ideas. MechGen combines recent research by Kinzel et al. (2006, 2007) [42, 43] in the synthesis of four-bar linkages using the constraint solver in sketch mode of solid modelers, with the six-bar linkage synthesis techniques of Soh, et al. (2006) [48]. The result is a convenient computer-aided design system for planar and spherical six-bar linkages. An interesting example of folding structures for emergency shelters is used to demonstrate the effectiveness of the overall design strategy.

3.1 Attachment of two RR Dyads to a 3R Serial Chain

In this section the goal is to find all the different ways in which two RR dyads can be attached to the 3R serial chain. The convention introduced here is to represent spherical linkages on the $+Z$ hemisphere, so that planar linkage and spherical linkage graphs are the same, see

Tsai (2000) [31]. The two RR dyads can be either applied independently to the links of the 3R chain, or in sequence where the second RR dyad is connected between the first RR dyad and one of the links of the 3R chain. In order to systematically find the different ways of applying them, its linkage graph is introduced, see Tsai(2000) [31]. Let the graph, $G =< V, E >$, be defined by the list of links V and the list of joints E represented by the links they connect.

The convention used for numbering the links of a 3R chain is shown in Figure 3.1. This yields a linkage graph for the 3R chain defined by,

$$\begin{aligned} G &=< V, E > \\ &=< \{1, 2, 3, 4\}\, , \{\{1, 2\}, \{2, 3\}, \{3, 4\}\} > . \end{aligned} \tag{3.1}$$

The six-bar linkages obtained by adding two RR dyads to a 3R serial chain can be viewed as defined by the linkage graphs $L_{(ij)(kl)}$ obtained from G by adding two subgraphs,

$$\begin{aligned} A_{ij} &=< \{i, j, 5\}\, , \{\{i, 5\}, \{j, 5\}\} >, \\ B_{kl} &=< \{k, l, 6\}\, , \{\{k, 6\}, \{l, 6\}\} > . \end{aligned} \tag{3.2}$$

The vertex pairs (i, j) and (k, l) are to be enumerated to determine all possible six-bar linkages given by,

$$L_{(ij)(kl)} = G \cup \{\{5, 6\}\, , \{\{i, 5\}, \{j, 5\}, \{k, 6\}, \{l, 6\}\}\}. \tag{3.3}$$

In order to enumerate the vertex pairs (i, j) and (k, l), the vertex list $V_4 = \{1, 2, 3, 4\}$ of the 3R chain and the vertex list $V_5 = \{1, 2, 3, 4, 5\}$ of the linkage after A_{ij} is attached, is introduced. For applying the first RR constraint, any two links from vertex list V_4 are picked, and for applying the second RR constraint any two links from vertex list V_5 are picked. Enumeration of pairs of vertices in the lists V_4 and V_5 that are available for the attachment of the RR dyads yields the lists P_A and P_B,

$$\begin{aligned} P_A =& \{(i,j) : i,j \in V_4, i \neq j\},\ |P_A| = \binom{4}{2} = 6, \\ P_B =& \{(k,l) : k,l \in V_5, k \neq l\},\ |P_B| = \binom{5}{2} = 10. \end{aligned} \tag{3.4}$$

This shows that the maximum number of linkage graphs, $L_{(ij)(kl)}$ obtained by combination of these lists is given by,

$$|L_{(ij)(kl)}| = |P_A||P_B| = 60. \tag{3.5}$$

However, these 60 linkage graphs are reduced due to existence of duplicate graphs, invalid graphs in which links combine to form a structure, and constraints imposed by the user to have certain design features in the six-bar linkage.

3.1.1 Ordering

The possibility that two different linkage graphs describe the same six-bar linkage can be eliminated by introducing an ordering to the labeling of the attached RR dyads. Specifically, introduce the convention,

$$i < j, \quad k < l. \tag{3.6}$$

Now, if $i, j, k, l \neq 5$, order the linkage graphs $L_{(ij)(kl)}$, such that the vertex pairs $(ij)(kl)$ are listed in order of the first entries i and k and then in order of the second entries j and l. This means the two dyads connecting links $\{2, 4\}$ and $\{1, 3\}$ are listed as (13)(24) while the dyads connecting links $\{3, 4\}$ and $\{1, 5\}$, are listed as (34)(15).

This ordering eliminates 15 duplicate graphs reducing the number of linkage graphs to 45.

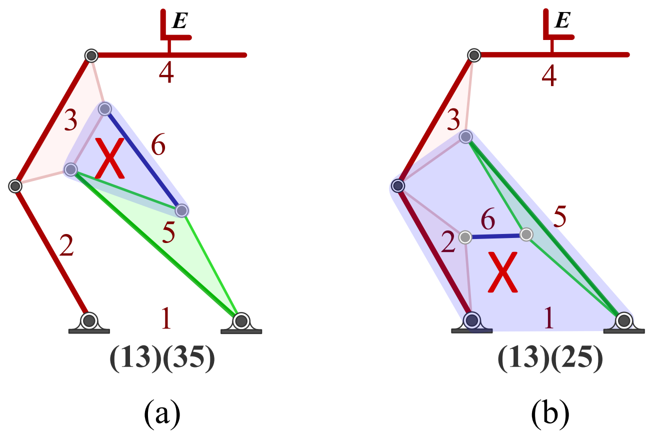


Figure 3.2: RR constraints that form substructures: (a) the linkage graph $L_{(13)(35)}$ includes triangular substructure $\{3, 5, 6\}$; (b) the linkage graph $L_{(13)(25)}$ has a constrained quadrilateral $\{1, 2, 3, 5, 6\}$ that forms a structure.

3.1.2 Structure Subgraphs

There are two cases in which the addition of two RR dyads creates a substructure, which collapses several links into one. The two cases are attachments that:

1. Form a triangle: If three links $\{r, s, t\}$ are connected by three joints $\{\{r, s\}, \{s, t\}, \{t, r\}\}$, the result is a pin-jointed triangle which forms a structure with no relative movement. An example is the sequence of edges $\{\{3, 5\}, \{5, 6\}, \{3, 6\}\}$, shown in Figure 3.2(a).

2. Connect opposite sides of a quadrilateral: If four vertices $\{r, s, t, w\}$ form a quadrilateral, then the addition of a dyad connecting either $\{r, t\}$ or $\{s, w\}$ will forms a structure with no relative movement. Figure 3.2(b) shows the case where links $\{1, 2, 3, 5\}$ forms a structure with the addition of the dyad connecting $\{2, 5\}$.

Eliminating the linkage graphs that have these substructures reduces the number of six-bar linkage graphs to 8.

3.1.3 Design Features

This last condition is artificial in the sense that it is imposed to achieve designs that have a particular feature. Through experience, it has been observed that an important concern of linkage designers is the location of attachment points to the ground frame. For this reason, the six-bar linkage graphs are restricted to include only the two connections to the ground frame. This removes 2 more linkage graphs that include a third base pivot. The result is 6 six-bar linkage graphs for both planar and spherical six-bar linkages.

3.2 The Design Requirements

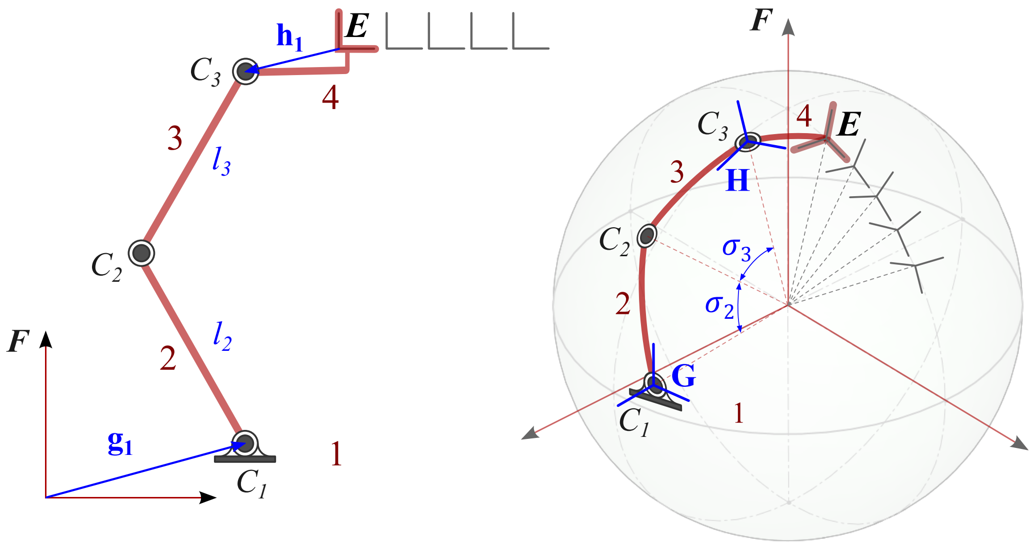


Figure 3.3: The input consists of five task positions and a 3R serial chain defined in the first task position for both planar and spherical six-bar linkage design systems.

The six-bar linkage design system requires specification of the five task positions for the end-effector and the dimensions of the 3R serial chain in the first position, see Figure 3.3. Specifically, the user defines the following requirements;

3.2.1 For Planar Six-bar Linkages

1. the five 3×3 homogeneous transformations, $[D_j], j = 1, \ldots, 5$, that define the movement of the end-effector frame,

$$[D_j] = \begin{bmatrix} \cos\phi_j & -\sin\phi_j & d_{j,x} \\ \sin\phi_j & \cos\phi_j & d_{j,y} \\ 0 & 0 & 1 \end{bmatrix}, \; j = 1, \ldots, 5; \tag{3.7}$$

2. the coordinates of the base pivot, $C_1 : \mathbf{g} = (g_x, g_y, 1)$;
3. the coordinates of the moving pivot attached to the end-effector, measured in the end-effector frame, $C_3 : \mathbf{h} = (h_x, h_y, 1)$;
4. the dimensions, $l_2 = |C_1C_2|$ and $l_3 = |C_2C_3|$; and,
5. whether the elbow pivot C_2 is in the positive or negative configuration.

3.2.2 For Spherical Six-bar Linkages

1. the five 3×3 rotation matrices, $[D_j], j = 1, \ldots, 5$, that define the movement of the end-effector frame,

$$[D_j] = [Y(\alpha_j)] \cdot [X(\beta_j)] \cdot [Z(\gamma_j)], \;\; j = 1, \ldots, 5, \tag{3.8}$$

   where $[Y(\alpha)]$, $[X(\beta)]$ and $Z(\gamma)$ are 3×3 rotation matrices, representing rotations about local axes Y, X and Z respectively.

2. the frame $[G]$ that maps the base, pivot C_1, of the 3R chain to the fixed frame F;
3. the frame $[H]$ that maps the tool frame in the last link of the 3R chain at pivot C_3;

4. the angular dimensions, σ_2 and σ_3; and,

5. whether the elbow pivot C_2 is in the positive or negative configuration.

This data is used to determine the position and orientation of each link of the 3R serial chain in each of the task positions. The relative positions of links in the 3R serial chain are then used to synthesize the RR dyads using equation (2.7).

3.3 Inverse Kinematics of the 3R serial chain

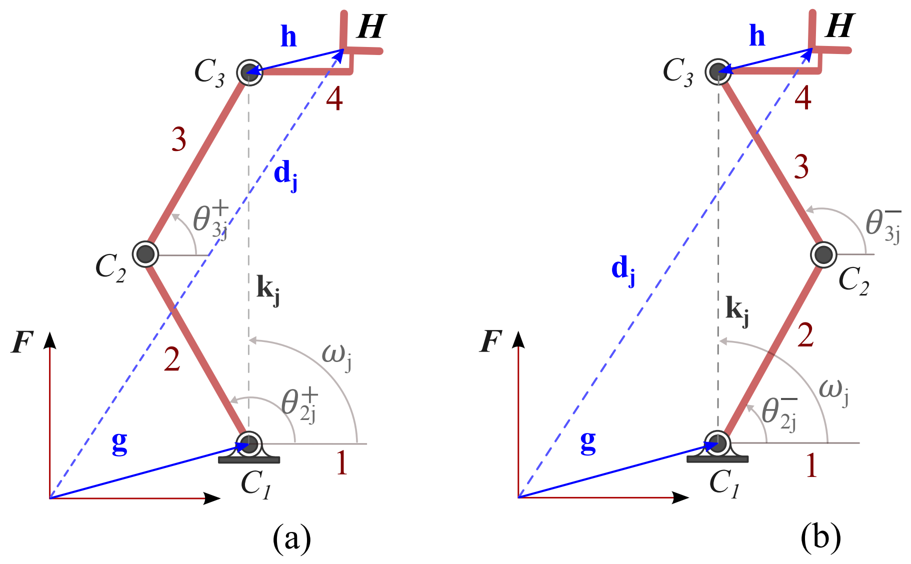


Figure 3.4: The elbow C_2 configuration for the chain is defined as (a) positive for $\theta_{3j} < \omega_j$, and (b) negative for $\omega_j < \theta_{3j}$.

The position and orientation of each of the links of the 3R serial chain $\{C_1, C_2, C_3\}$ can be determined by performing inverse kinematics on it, Fig. 3.3.

3.3.1 Planar Inverse Kinematics of the 3R chain

Using the Denavit Hartenberg convention, the kinematics equations for the 3R chain $\{C_1, C_2, C_3\}$ are given as,

$$\begin{aligned}[D_j] =& [T(\mathbf{g})][Z(\theta_{2,j})][X(l_2)][Z(\theta_{3,j} - \theta_{2,j})] \\ & \times [X(l_3)][Z(\phi_j - \theta_{3,j})][T(-\mathbf{h})], \quad j = 1, \ldots, 5;\end{aligned} \tag{3.9}$$

where the homogenous transformation matrices $[T(\mathbf{g})], [Z(\theta)]$ and $[X(l)]$ and are given as,

$$[T(\mathbf{g})] = \begin{bmatrix} 1 & 0 & g_x \\ 0 & 1 & g_y \\ 0 & 0 & 1 \end{bmatrix}, [Z(\theta)] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, [X(l)] = \begin{bmatrix} 1 & 0 & l \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.10}$$

In order to determine the angles $\theta_{2,j}$ and $\theta_{3,j}$, rearrange the kinematics equations into the form,

$$\begin{aligned}[K_j] = [Z(\theta_{2,j})][X(l_2)][Z(\theta_{3,j} - \theta_{2,j})][X(l_3)][Z(-\theta_{3,j})], \\ j = 1, \ldots, 5;\end{aligned} \tag{3.11}$$

where $[K_j]$ is the matrix of known constants,

$$[K_j] = [T(\mathbf{g})]^{-1}[D_j][T(-\mathbf{h})]^{-1}[Z(-\phi_j)]. \tag{3.12}$$

Expand equation (4.11) to obtain from the third column of $[K_j]$, the vector $\mathbf{k}_j = (k_{j,x}, k_{j,y})$, given by

$$\begin{aligned}k_{j,x} &= d_{j,x} + h_x \cos\phi_j - h_y \sin\phi_j \\ k_{j,y} &= d_{j,y} + h_y \cos\phi_j + h_x \sin\phi_j \end{aligned} \tag{3.13}$$

Next, simplify the right hand side of equation 4.10 to obtain,

$$\begin{aligned}k_{j,x} &= l_2 \cos\theta_{2,j} + l_3 \cos\theta_{3,j} \\ k_{j,y} &= l_2 \sin\theta_{2,j} + l_3 \sin\theta_{3,j}, \quad j = 1, \ldots, 5.\end{aligned} \tag{3.14}$$

Introduce the angle ω_j that defines the direction of the vector $\mathbf{k}_j$, which is given by

$$\omega_j = \arctan\left(\frac{k_{j,y}}{k_{j,j}}\right), \quad j = 1, \ldots, 5. \tag{3.15}$$

This combined with the cosine law of the triangle formed by $C_1C_2C_3$ yields,

$$\theta_{2,j} = \omega_j \pm \arccos\left(\frac{l_2^2 + |\mathbf{k}_j|^2 - l_3^2}{2l_2|\mathbf{k}_j|}\right), \quad j = 1, \ldots, 5. \tag{3.16}$$

The arccos function yields positive and negative values for $\theta_{2,j}$, which correspond to the positive and negative configurations of the elbow C_2 as shown in Figure 4.5(a) and (b) respectively.

The angle $\theta_{3,j}$ is now obtained from the equation (4.13), as

$$\theta_{3,j} = \arctan\left(\frac{k_{j,y} - l_2 \sin\theta_{2,j}}{k_{j,x} - l_2 \cos\theta_{2,j}}\right). \tag{3.17}$$

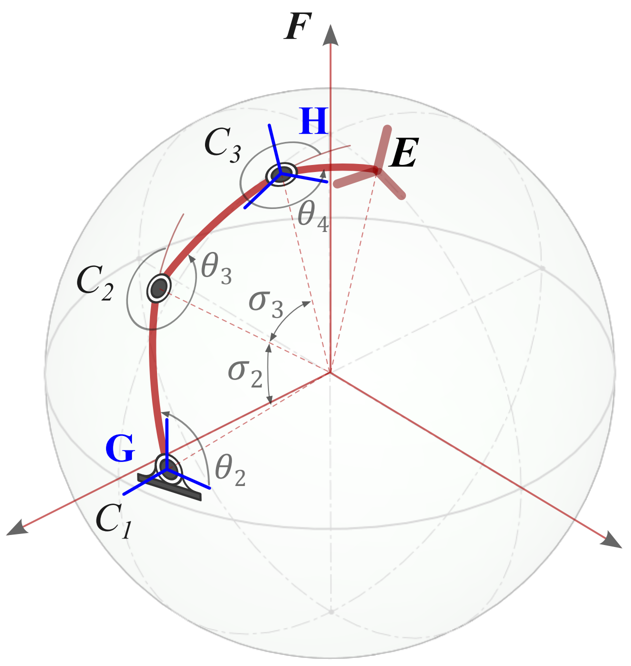


Figure 3.5: The joint angles for the spherical 3R serial chain are defined relative to the previous link.

3.3.2 Spherical Inverse Kinematics of the 3R chain

Using the Denavit Hartenberg convention, the kinematics equations for the spherical 3R chain $\{C_1, C_2, C_3\}$, with angles $\theta_2, \theta_3, \theta_4$ defined as shown in Fig.3.5, are given as,

$$[D_j] = [G][Z(\theta_{2,j})]Y(\sigma_2)][Z(\theta_{3,j})][Y(\sigma_3)][Z(\theta_{3,j})][H], \quad j = 1, \ldots, 5, \tag{3.18}$$

where transformations $[Y(\alpha)]$, $[X(\beta)]$ and $[Z(\gamma)]$ are rotation matrices, see Eq.2.9. Notice in spherical case, the angles $\theta_2, \theta_3, \theta_4$ are defined from previous links instead of all defined with respect to the ground as in planar case. Similar to the planar case, by solving the kinematic equations (3.18), two set of solutions for $\theta_2, \theta_3, \theta_4$ are obtained, based on whether the elbow C_2 is in the positive or negative position.

3.4 Synthesis Equations

The inverse kinematics data allows the determination of the position and orientation of each of the links, $V_\mu, \mu = 1, \ldots, 4$, of the 3R serial chain, in each of five task configurations, $[D_\nu], \nu = 1, \ldots, 5$. The result is the set of transformations,

$$[L_{\mu,\nu}] = \begin{bmatrix} \cos\theta_{\mu,\nu} & -\sin\theta_{\mu,\nu} & a_{\mu,\nu} \\ \sin\theta_{\mu,\nu} & \cos\theta_{\mu,\nu} & b_{\mu,\nu} \\ 0 & 0 & 1 \end{bmatrix}, \quad \mu = 1, \ldots, 4, \quad \nu = 1, \ldots, 5, \tag{3.19}$$

where θ represents the orientation and (a, b) represents the location, in case of the planar 3R chain, and

$$\begin{aligned} &[L_{\mu,\nu}] = [Y(\epsilon_{\mu,\nu})][X(\kappa_{\mu,\nu})][Z(\lambda_{\mu,\nu})], \\ &\qquad \mu = 1, \ldots, 4, \;\; \nu = 1, \ldots, 5, \end{aligned} \tag{3.20}$$

where $\epsilon, \kappa, \lambda$ are the three orientation angles, in case of spherical 3R chain.

The synthesis equations for an RR dyads between the links V_i and V_k are obtained by substituting,

$$[R_\nu] = [L_{i,\nu}], \; [S_\nu] = [L_{k,\nu}], \quad \nu = 1, \ldots, 5, \tag{3.21}$$

into equation (2.7) for planar case and

$$[T_\nu] = [L_{i,\nu}], \; [K_\nu] = [L_{k,\nu}], \quad \nu = 1, \ldots, 5, \tag{3.22}$$

equation (2.15) for spherical case. Notice that for convenience the symbol ν replaces the symbol j in the synthesis equations. Once the first RR constraint is applied, it is available for attachment for the second RR constraint. Hence, following the application of first RR constraint, which forms the link 5, the list of transformations, $[L_{\mu,\nu}]$ from Eqn. (4.17) and (3.20), are updated for this new link 5.

3.5 Calculating Candidate Designs

The design system solves the synthesis equations (2.7), to calculate the two RR constraints, for each of the of the 6 unique linkage graphs $L_{(ij)(kl)}$. The number of solutions must account for RR constraint solutions that satisfy the synthesis equations and are already part of the linkage.

This is done by distinguishing two cases, (i) sets of two RR constraints that include only the vertices, $\{1, 2, 3, 4\}$ of the original 3R serial chain, which are referred here as independent constraints, and (ii) sets of two RR constraints, that are applied in sequence including one connection to a vertex in the set $\{5\}$, which are referred as dependent constraints.

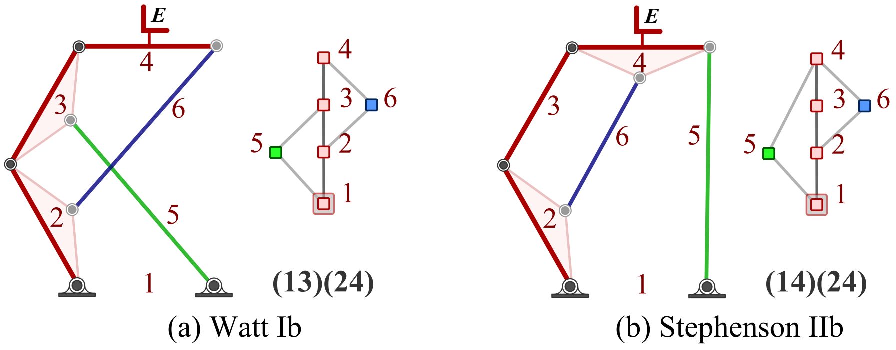


Figure 3.6: Two different six-bar linkage graphs $(ij)(kl)$ are obtained using two independent RR dyad constraints, such that $i, j, k, l \in \{1, 2, 3, 4\}$.

Independent constraints: A set of two RR constraints that connect to the vertices $i, j, k, l \in \{1, 2, 3, 4\}$ can be applied independent of each other. There are 2 unique ways of attaching the independent constraints as shown in Fig.4.12. They are topologies Watt Ib and Stephenson IIb respectively.

In order to count the number of design candidates obtained from the synthesis equations, consider the linkage graph example (13)(24) shown in Fig.4.12(a). Observe that for the attachment of the first RR constraint A_{13}, one solution C_1C_2 already exists, thus the synthesis equations yield at most three new RR constraints. This is true for the second RR constraint B_{24} as well. In this case, the various combinations of the RR solutions yields as many as $3 \times 3 = 9$ candidate designs in case of planar six-bar linkage, and $5 \times 5 = 25$ candidate designs in case of spherical six-bar linkage.

A systematic count for the two linkage graphs with independent constraints, yields as many as 21 six-bar candidate planar and 55 spherical six-bar linkage designs.

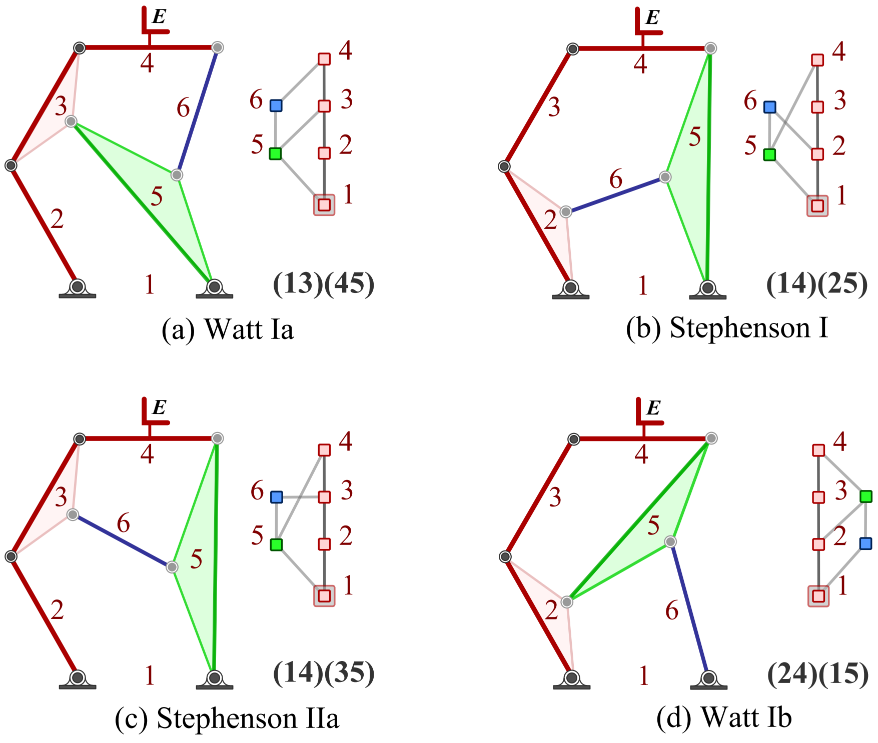


Figure 3.7: Four different six-bar linkage graphs $(ij)(kl)$ are obtained by connecting the second RR dyad, to the link of the first RR dyad, such that $i, j, k \in \{1, 2, 3, 4\}$ and $l \in \{5\}$.

Dependent constraints: A set of two RR constraints with $i, j, k \in \{1, 2, 3, 4\}$ and $l \in \{5\}$, will have the second RR constraint attached to the first RR constraint. There are 4 unique ways of attaching the dependent constraints as shown in Fig.4.13.

In order to count the number of design candidates obtained from the synthesis equations, consider the linkage graph (14)(25) shown in Figure 4.13(b). Observe that for the attachment of the first RR constraint A_{14}, all the four RR constraint solutions are available. For the application of second RR constraint B_{25}, one solution C_1C_4 already exists, thus the synthesis equations yield at most three new RR constraints. In this case, the various combinations of the RR solutions yields as many as $4 \times 3 = 12$ candidate designs in case of planar six-bar linkage, and $6 \times 5 = 30$ candidate designs in case of spherical six-bar linkage.

A systematic count for all the 4 linkage graphs with dependent constraints yields as many as 42 six-bar linkage candidate planar and 110 spherical six-bar linkage designs.

Finally, this yields a total of 63 planar six-bar and 165 spherical designs for the 6 linkage graphs.

3.6 Performance Verification

The performance of a six-bar linkage design is analyzed to ensure smooth movement movement of the end-effector through the five task positions. In order to be a successful design the end-effector must pass through the task positions in a single assembly.

3.6.1 Planar Six-bar Linkage

For the analysis of planar six-bar linkages, the automated system developed by Parrish et al. (2014) [28] is employed. This algorithm reads the location of the pivots of the six-bar linkage in the first task position and the adjacency matrix of the linkage graph to formulate the two loop equations for the six-bar linkage. These equations are solved analytically using the Dixon determinant approach in order to obtain all the assemblies of the linkage for a given value of the input angle.

A sorting algorithm collects the result of the analysis routine into various assemblies, that define the values of the 7 joint angles for k iterations of the input angle. Note that the angle made by the ground link remains constant. The results are the joint trajectories for each of the assemblies. For more details about sorting solutions into linkage assemblies to verify performance, see Plecnik and McCarthy (2013) [67]. In addition to this requirement the designer may also require the link lengths of the linkage to meet certain criteria to be of practical use.

3.6.2 Spherical Six-bar Linkage

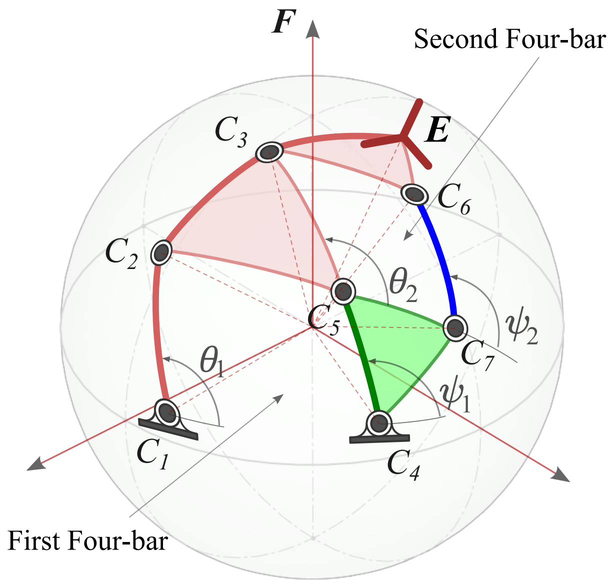


Figure 3.8: The Watt I six-bar linkage can be analyzed as an assembly of two four-bar linkages.

In case of spherical six-bar linkages, only the Watt I topology six-bar is analyzed for branch/-

circuit defects. The Watt I topology has two four-bar sub-loops as shown in Fig.3.8. In order to filter these linkages for defects, the design systems analyzes the two spherical four-bar sub-loops separately using the theory presented in McCarthy (2010 [12]. Solving the forward kinematics equation for four-bar linkage yields two values for the output crank angle ψ for a given input crank angle θ as shown in equation (3.23) below. The two angles result from the fact, that the coupler link and the output crank can be assembled in two configurations namely elbow up and elbow down for the same input crank angle, which corresponds to the plus and minus values in the equation,

$$\psi(\theta) = \arctan\left(\frac{B}{A}\right) \pm \arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right). \tag{3.23}$$

This equation also provides formula for finding the bounds for the range of motion of the input crank angle θ, through the argument of the arccos term, which is subjected to the bounds -1 and $+1$. If this is not satisfied then the linkage cannot be assembled for the specified input crank angle. Thus equation (3.23) can be used to filter both the first and second four-bars. The two checks used to filter defective linkages are:

1. **The elbow up or elbow down configuration**

   The elbow of the output crank with respect to the coupler should be either up or down for all the five task positions. This avoids circuit change when the input link is a crank, with range 0 to 2π. It also avoids a branch change, when the input link is a 0-rocker or π-rocker for non-Grashof linkages, or a Grashof rocker, see McCarthy (2010) [12].

2. **The limits on the input crank**

   The five crank angles associated with the five task positions must lie in the range defined by the values obtained from the arccos argument in (3.23). This defect occurs, when the input crank is a rocker. This check prevents circuit change when input crank is rocker, see McCarthy (2010) [12].

These two checks are sufficient to identify circuit and branch defects and ensure the system returns spherical Watt I six-bar linkages that are defect-free.

3.7 Tolerance Zones

Kinematic synthesis equations fail regularly to yield designs as they are dependent on the selection of the five task positions. [47] showed that this challenge could be overcome by introducing small variations to the task within designer specified tolerance zones. The result is a reliable synthesis procedure that has been implemented in the MechGen series of design systems, see Sonawale (2013) [14].

3.7.1 Planar Six-bar Linkage

The first run of the design algorithm $k = 1$ uses the user defined task positions, $(\phi_j, \mathbf{d}_j)$, $j = 1, \dots, 5$. For each subsequent run $k = 2, \dots, q$, where q are the number of iterations specified by the user, the task positions are modified by the addition a random variation from the tolerance zones $(\pm\Delta\phi_j, \pm\Delta\mathbf{d}_j)$, $j = 1, \dots, 5$, given by

$$\begin{aligned}\{(\phi_i, \mathbf{d}_j), j = 1, \dots, 5\}_k &= \{(\phi_i, \mathbf{d}_j),\ j = 1, \dots, 5\}\\ &\quad + \{\mathrm{Rand}(\pm\Delta\phi_j, \pm\Delta\mathbf{d}_j),\ j = 1, \dots, 5\},\quad k = 2, \dots, q.\end{aligned} \tag{3.24}$$

3.7.2 Spherical Six-bar Linkage

Similarly in case of spherical linkages, for the first run $k = 1$, the original task positions are used $(\alpha_j, \beta_j, \gamma_j)$, $j = 1, \dots, 5$. For the following runs $k = 2, \dots, q$, the task positions are randomized by the addition of small variations with the tolerance zones $(\pm\Delta\alpha_j, \pm\Delta\beta_j, \pm\Delta\gamma_j,)$,

$j = 1, \ldots, 5$, given by

$$\begin{aligned}\{(\alpha_j, \beta_j, \gamma_j),\ j = 1, \ldots, 5\}_k = {} & \{(\alpha_j, \beta_j, \gamma_j),\ j = 1, \ldots, 5\} \\ & +\{\text{Rand}(\pm\Delta\alpha_j, \pm\Delta\beta_j, \pm\Delta\gamma_j),\ j = 1, \ldots, 5\}, \\ & k = 2, \ldots, q. \end{aligned} \tag{3.25}$$

This strategy of performing search for defect free linkages iteratively by introducing small variations to the task positions, within designer specified tolerance zones, yields successful six-bar linkage designs.

3.8 Computer Aided Software Implementation: MechGen (Mechanism Generator)

In this section, a computer aided software implementation of the design system, MechGen, is discussed for spherical six-bar linkage is discussed. Currently, MechGen can design four-bar and six-bar, planar and spherical linkages, for five specified task positions. MechGen works as an Add-in for SolidWorks. The user specifies the desired motion as task positions in the sketch environment, along with the backbone chain robot. MechGen then captures the user sketch along with allowable tolerances on the task positions and number of iterations. It then generates linkage solutions and presents them to the user as animations. The user can browse through all the solutions and can generate a SolidWorks assembly of the desired linkage.

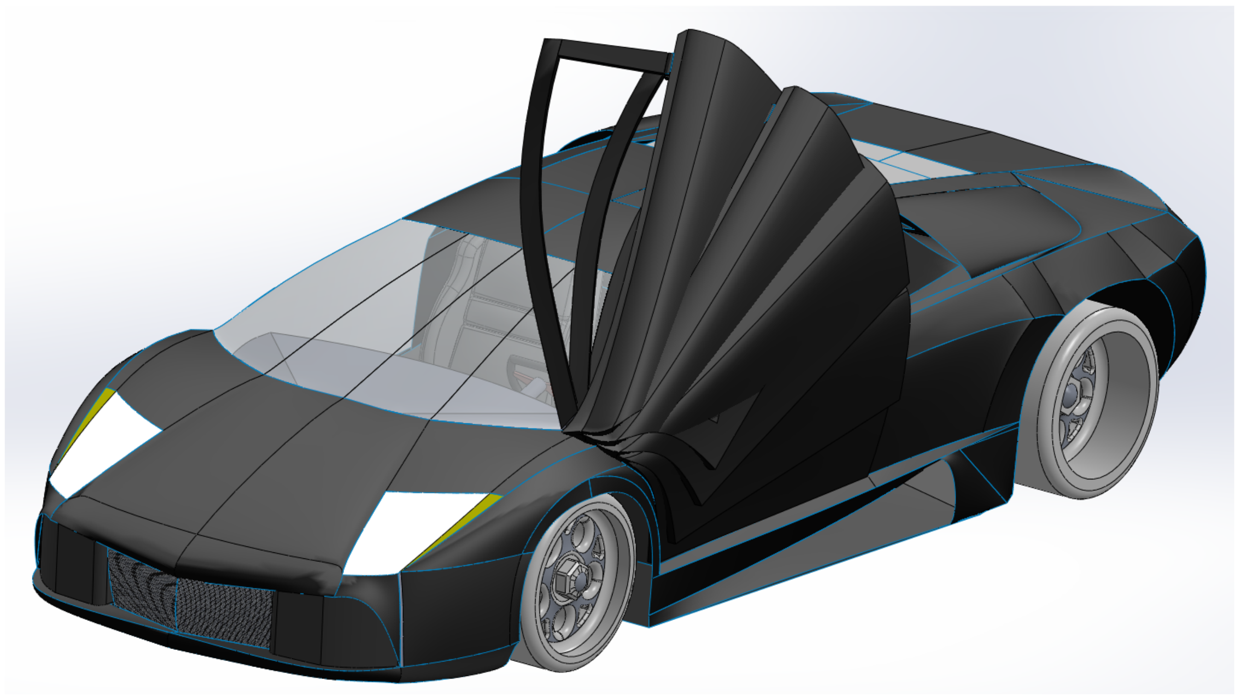


Figure 3.9: Task requirement of the designer

3.8.1 MechGen Design Process

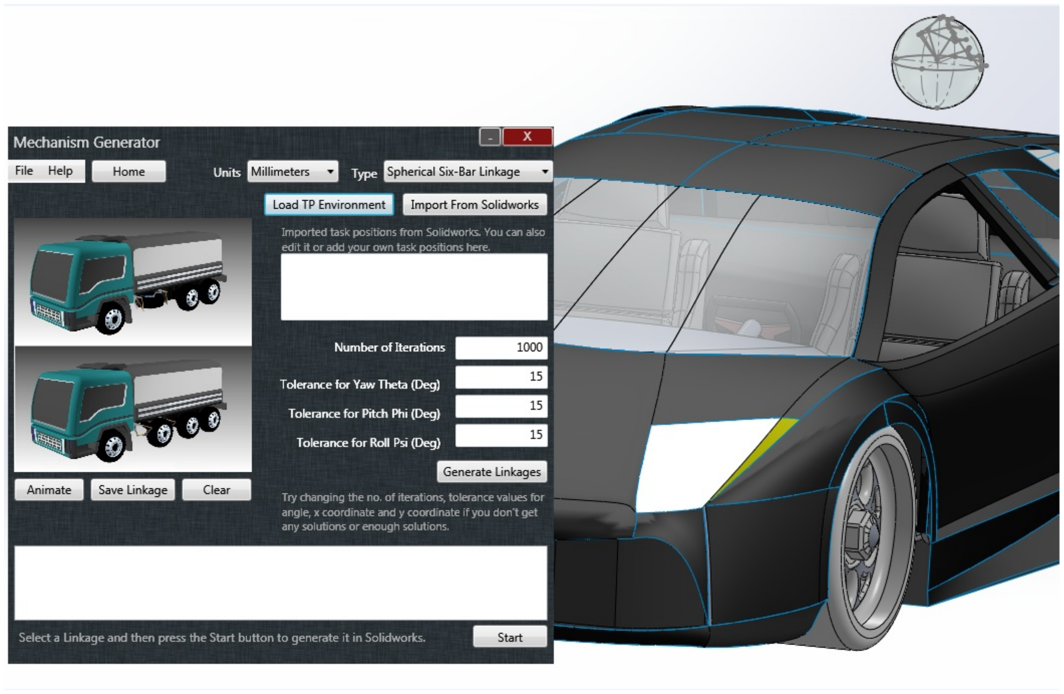
Here a *car door opening linkage* example is used to demonstrate the design process for MechGen.

Step 1: Motion Requirement

The user's requirement for the motion of the car door with respect to the car body, as five positions, is shown in Figure 3.9. Since MechGen can do only five position synthesis, the desired motion has to be discretized into five door positions.

Step 2: Specifying five Task Positions

Due to the inherent complexity in specifying the spherical task positions, MechGen has a part file embedded in it, which is referred to as the *Environment*. This part file has two sketches built in; one for the five task positions and one for the 3R serial chain. In this step MechGen software is opened and *Environment* part file is imported in the user assembly



Units: Millimeters

Type: Spherical Six-Bar Linkage

Imported task positions from Solidworks. You can also edit it or add your own task positions here.

Number of Iterations: 1000

Tolerance for Yaw Theta (Deg): 15

Tolerance for Pitch Phi (Deg): 15

Tolerance for Roll Psi (Deg): 15

Try changing the no. of iterations, tolerance values for angle, x coordinate and y coordinate if you don't get any solutions or enough solutions.

Select a Linkage and then press the Start button to generate it in Solidworks.

Figure 3.10: Import environment part file

as shown in Figure 3.10. The user edits the sketch for task positions to get a movement that is close to the requirement. Five doors are attached to the task positions to confirm this, as shown in Figure 3.11.

Step 3: Specifying the five 3R chains

In this step the user edits the 3R chain sketch in the *Environment* file to specify the backbone serial chain as shown in Figure 3.12.

Step 4: Import the task postion and 3R chain information in MechGen

In this step the user selects the 3R chain sketch in Environment and clicks on the *Import From SolidWorks* button. MechGen captures all the information and displays it to the user along with several verification checks to display errors if any as shown in Figure 3.13.

Step 5: Generate linkage solutions

In this step the user specifies the number of iterations and the tolerances and then clicks on

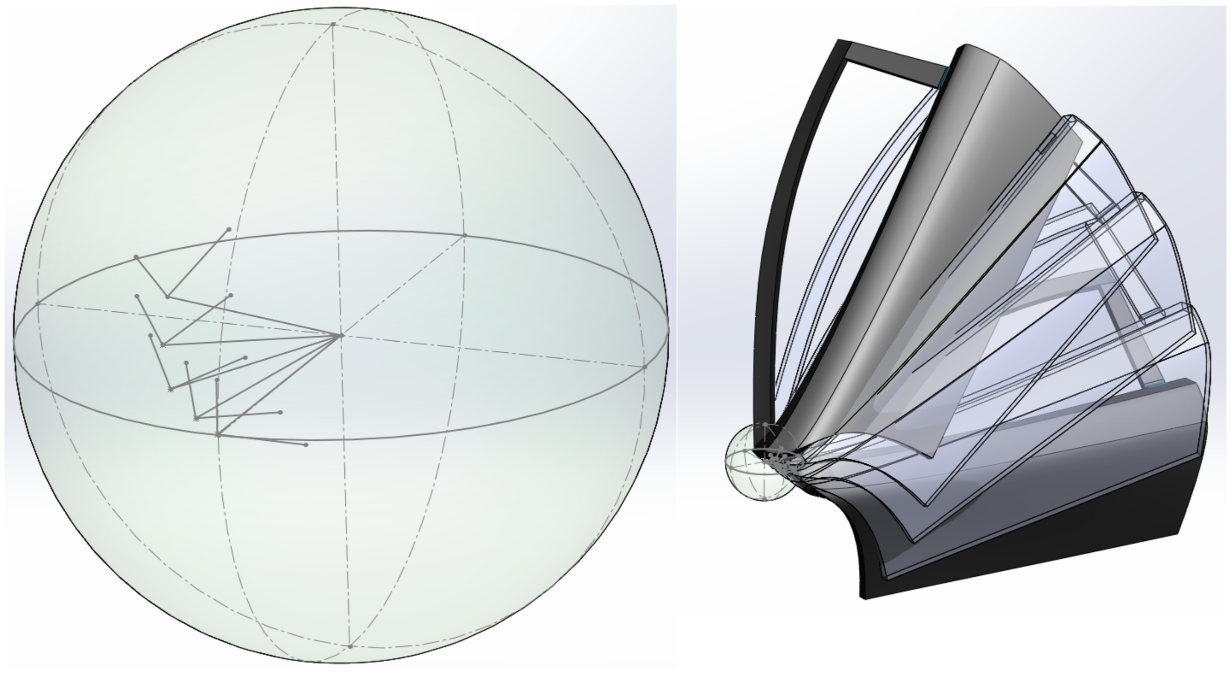


Figure 3.11: Edit task positions to suit the requirements

the *Generate Linkages* button to generate solutions. The generated solutions are displayed as shown in Figure 3.14. The user can select a linkage from the solution box and generate its Solidworks assembly oby clicking on the ”Start” button. Figure3.15 show four example linkages generated by MechGen in SolidWorks.

Step 6: Integrating desired linkage in user assembly

In this step the user imports the desired linkage solution in to the car assembly and integrates it. Figure3.16 show the final car door assembly with the door being guided by the spherical Watt I six-bar linkage through the five task positions. A video of this linkage in action can be found on `http://mechanicaldesign101.com`.

3.8.2 Generating Solid Geometries in SolidWorks

The development of this module in MechGen began with recording macros of simple shapes drawn in the SolidWorks user interface. When saved, these macros provided the commands,

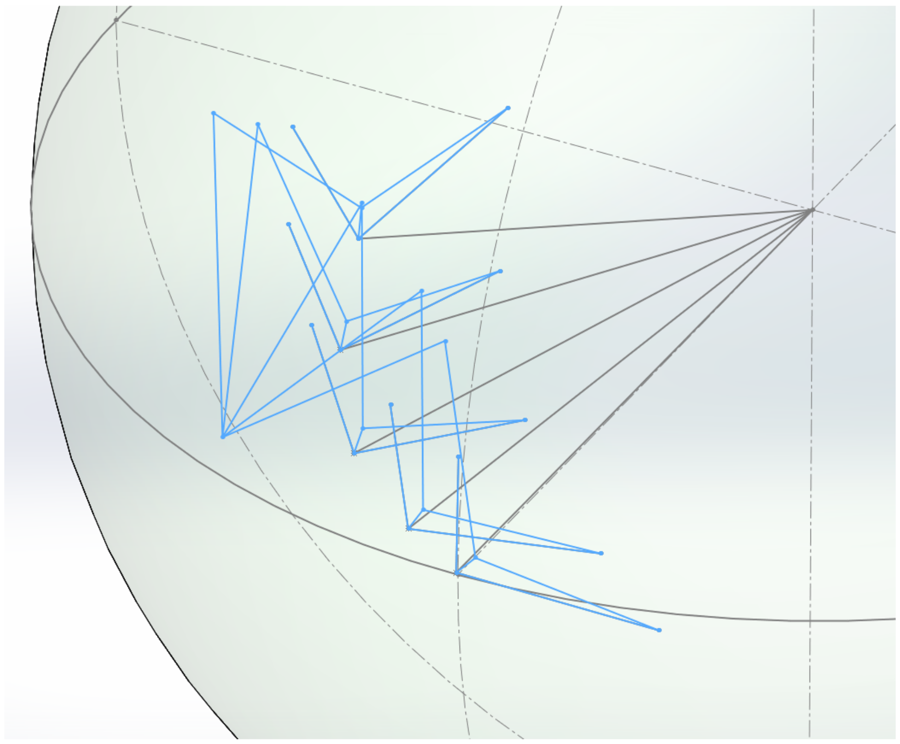
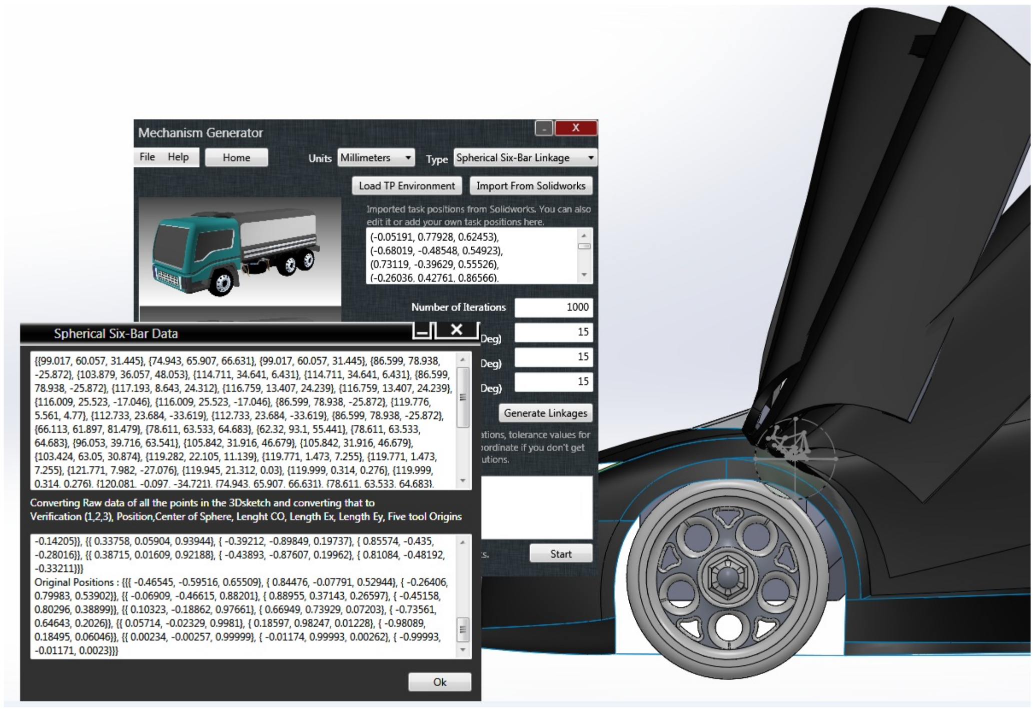


Figure 3.12: Edit the five 3R chains

parameters and code sequence required to produce simple shapes. The various macros were then altered and combined into a simple Visual Basic (VB6) program. As the program became robust at producing a SolidWorks model for the planar six-bar linkage using primitive shapes, the development of the software was shifted the to VB.NET to take advantage of the .NET Framework.

During the linkage generation process, MechGen creates a separate part file for each spherical link. A link is created using lines and arcs to form a closed loop and then extruded to give it a three dimensional structure. Once all the part documents are created, an assembly document is made and the parts are imported into the assembly model space. First, the software selects the origin of a link in a part document and mates it to the origin of the assembly document. The process is repeated until all the part document origins are coincident with the assembly document origin. Next, the software applies a collinear mate between the joint axis of the connecting links. Each link is connected in this manner until all joint axes



Imported task positions from Solidworks. You can also edit it or add your own task positions here.

```
(-0.05191, 0.77928, 0.62453),
(-0.68019, -0.48548, 0.54923),
(0.73119, -0.39629, 0.55526),
(-0.26036, 0.42761, 0.86566),
```

Number of Iterations 1000

Deg) 15

Deg) 15

Deg) 15

```
{{99.017, 60.057, 31.445}, {74.943, 65.907, 66.631}, {99.017, 60.057, 31.445}, {86.599, 78.938,
-25.872}, {103.879, 36.057, 48.053}, {114.711, 34.641, 6.431}, {114.711, 34.641, 6.431}, {86.599,
78.938, -25.872}, {117.193, 8.643, 24.312}, {116.759, 13.407, 24.239}, {116.759, 13.407, 24.239},
{116.009, 25.523, -17.046}, {116.009, 25.523, -17.046}, {86.599, 78.938, -25.872}, {119.776,
5.561, 4.77}, {112.733, 23.684, -33.619}, {112.733, 23.684, -33.619}, {86.599, 78.938, -25.872},
{66.113, 61.897, 81.479}, {78.611, 63.533, 64.683}, {62.32, 93.1, 55.441}, {78.611, 63.533,
64.683}, {96.053, 39.716, 63.541}, {105.842, 31.916, 46.679}, {105.842, 31.916, 46.679},
{103.424, 63.05, 30.874}, {119.282, 22.105, 11.139}, {119.771, 1.473, 7.255}, {119.771, 1.473,
7.255}, {121.771, 7.982, -27.076}, {119.945, 21.312, 0.03}, {119.999, 0.314, 0.276}, {119.999,
0.314, 0.276}, {120.081, -0.097, -34.721}, {74.943, 65.907, 66.631}, {78.611, 63.533, 64.683},
```

Converting Raw data of all the points in the 3Dsketch and converting that to Verification (1,2,3), Position,Center of Sphere, Lenght CO, Length Ex, Length Ey, Five tool Origins

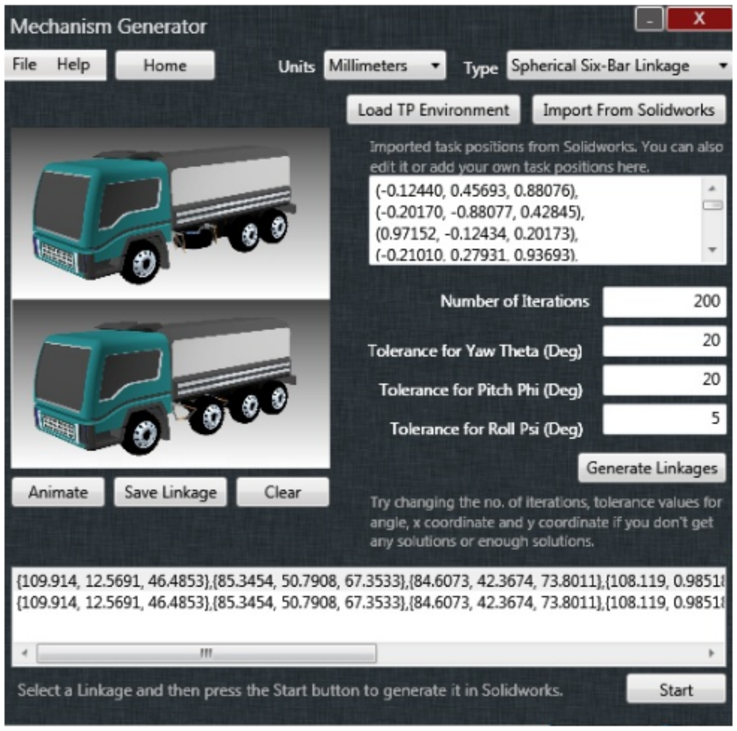
```
-0.14205}}, {{ 0.33758, 0.05904, 0.93944}, { -0.39212, -0.89849, 0.19737}, { 0.85574, -0.435,
-0.28016}}, {{ 0.38715, 0.01609, 0.92188}, { -0.43893, -0.87607, 0.19962}, { 0.81084, -0.48192,
-0.33211}}}
Original Positions : {{{ -0.46545, -0.59516, 0.65509}, { 0.84476, -0.07791, 0.52944}, { -0.26406,
0.79983, 0.53902}}, {{ -0.06909, -0.46615, 0.88201}, { 0.88955, 0.37143, 0.26597}, { -0.45158,
0.80296, 0.38899}}, {{ 0.10323, -0.18862, 0.97661}, { 0.66949, 0.73929, 0.07203}, { -0.73561,
0.64643, 0.2026}}, {{ 0.05714, -0.02329, 0.9981}, { 0.18597, 0.98247, 0.01228}, { -0.98089,
0.18495, 0.06046}}, {{ 0.00234, -0.00257, 0.99999}, { -0.01174, 0.99993, 0.00262}, { -0.99993,
-0.01171, 0.0023}}}
```

Figure 3.13: Import task position and 3R chain information in MechGen

are mated. This completes the assembly for the spherical Watt I sixbar linkage. Rotating the input link will cause the end effector to move through the five task positions.

3.9 Design System Application for Six-bar Linkages

In this section the implementation of the design system is demonstrated for the solid modeling software SolidWorks, using an example each for planar and spherical linkages. The task selected for the planar six-bar linkage is rectilinear motion, whereas for the spherical six-bar linkage it is a motion along the great circle.



Units: Millimeters
Type: Spherical Six-Bar Linkage

Imported task positions from Solidworks. You can also edit it or add your own task positions here.

```
(-0.12440, 0.45693, 0.88076),
(-0.20170, -0.88077, 0.42845),
(0.97152, -0.12434, 0.20173),
(-0.21010, 0.27931, 0.93693),
```

Number of Iterations: 200
Tolerance for Yaw Theta (Deg): 20
Tolerance for Pitch Phi (Deg): 20
Tolerance for Roll Psi (Deg): 5

Try changing the no. of iterations, tolerance values for angle, x coordinate and y coordinate if you don't get any solutions or enough solutions.

```
{109.914, 12.5691, 46.4853},{85.3454, 50.7908, 67.3533},{84.6073, 42.3674, 73.8011},{108.119, 0.9851
{109.914, 12.5691, 46.4853},{85.3454, 50.7908, 67.3533},{84.6073, 42.3674, 73.8011},{108.119, 0.9851
```

Select a Linkage and then press the Start button to generate it in Solidworks.

Figure 3.14: MechGen displaying linkage solutions

3.9.1 Planar Rectilinear Motion

Here the user first creates a sketch to define the five task positions and the five 3R chains as shown in Fig.3.17. The rectilinear motion of range 100mm is captured by the five task positions spaced apart by 25mm. The location of the ground pivot is $C_1 = (30, -100)$mm and the lengths of the links are $C_1C_2 = 90\text{mm}, C_2C_3 = 60\text{mm}$. The tolerances for the task positions are $(\pm\Delta 0.1^o, \pm\Delta 5\text{mm}, \pm\Delta 0\text{mm})$. Notice that the tolerance on the y coordinate of the task positions are all zero. The sample number of iterations specified are 500. MechGen imports this data from the sketch and generates six-bar linkages and presents them to the user as shown in Fig.3.18. For the example it was able to generate 191 successful linkages in 1 minute. The linkage solution selected is shown in Fig.3.19 and its joint coordinates $C_1, \dots, C_7$ are specified in Tab.3.1.

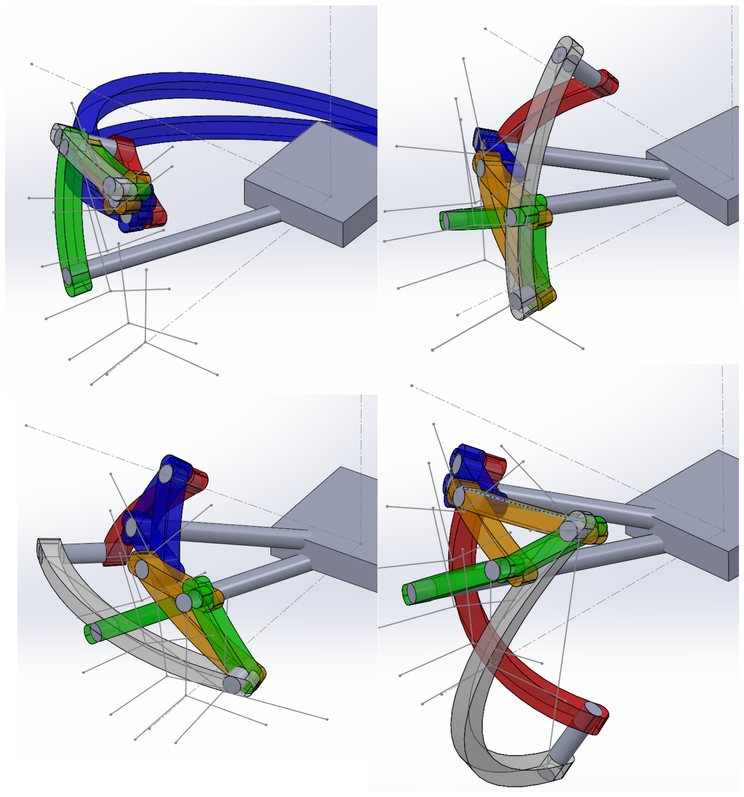


Figure 3.15: Example linkages generated by MechGen

3.9.2 Motion on a Great Circle

Here the user first creates a sketch to define the five spherical task orientations and the five 3R chains as shown in Fig.3.20. The great circular motion to be achieved for the angular range of 65^o is captured by the five task positions. The direction of the ground pivot axis is $C_1 = (165.06, 122.39, 190.21)$mm and the angular lengths of the spherical links are $C_1C_2 = 38^o, C_2C_3 = 36.3^o$. The tolerances on the orientations of the task positions are $(\pm\Delta 0.01^o, \pm\Delta 5^o, \pm\Delta 0.1^o)$. The sample number of iterations specified are 10. MechGen imports this data from the sketch and generates spherical six-bar linkages and presents them to the user. For this example, it was able to generate 6 successful linkages in approximately 5 minutes. The linkage solution selected is shown in Fig.3.21 and its joint coordinates $C_1, \ldots, C_7$ are specified in Tab.3.2.

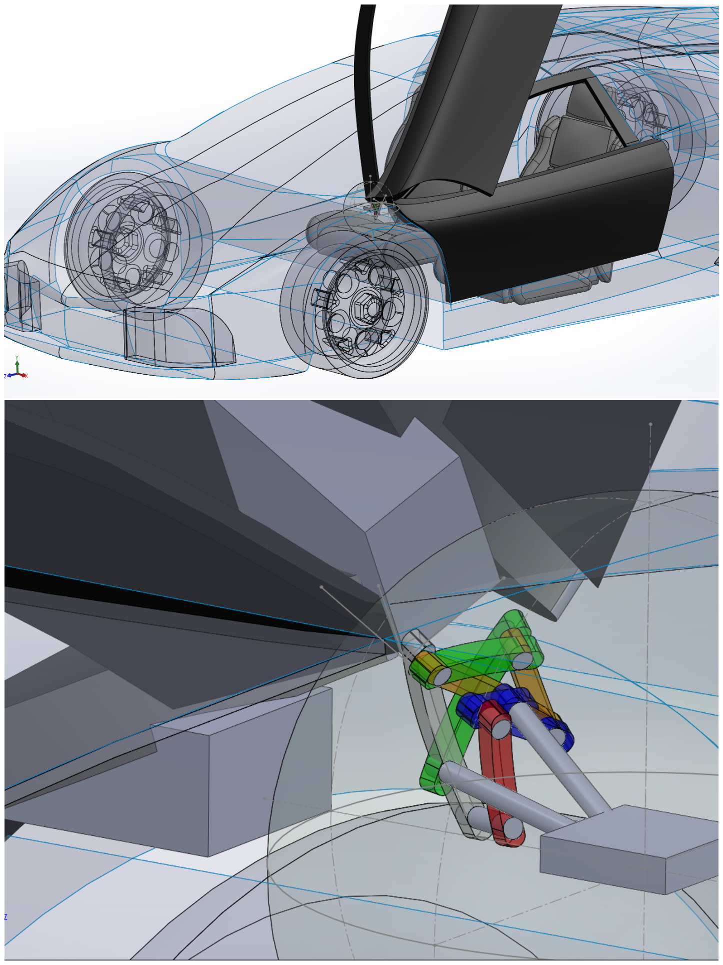


Figure 3.16: Intergration of spherical Watt I six-bar linkage in the user assembly

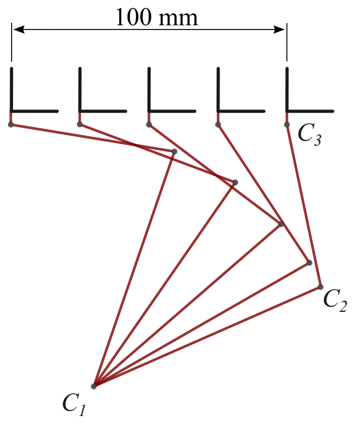
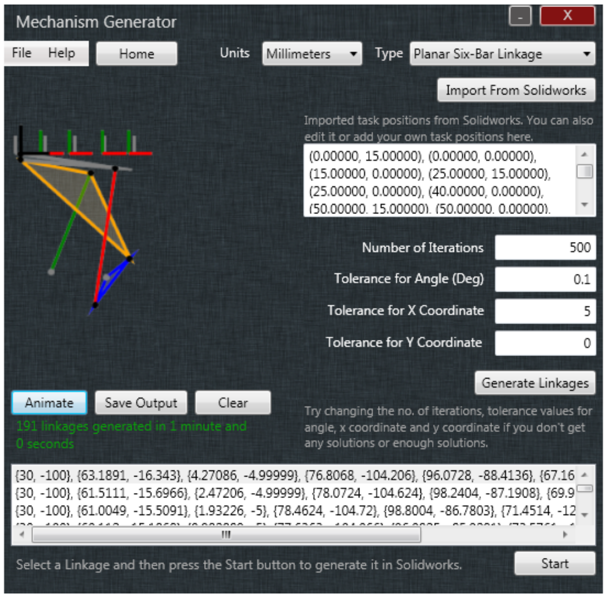


Figure 3.17: The rectilinear motion task positions and user-defined 3R planar serial chains are drawn in the sketch environment.



Units: Millimeters
Type: Planar Six-Bar Linkage

Imported task positions from Solidworks. You can also edit it or add your own task positions here.

```
(0.00000, 15.00000), (0.00000, 0.00000),
(15.00000, 0.00000), (25.00000, 15.00000),
(25.00000, 0.00000), (40.00000, 0.00000),
(50.00000, 15.00000), (50.00000, 0.00000),
```

Number of Iterations: 500
Tolerance for Angle (Deg): 0.1
Tolerance for X Coordinate: 5
Tolerance for Y Coordinate: 0

Try changing the no. of iterations, tolerance values for angle, x coordinate and y coordinate if you don't get any solutions or enough solutions.

191 linkages generated in 1 minute and 0 seconds

```
{30, -100}, {63.1891, -16.343}, {4.27086, -4.99999}, {76.8068, -104.206}, {96.0728, -88.4136}, {67.16
{30, -100}, {61.5111, -15.6966}, {2.47206, -4.99999}, {78.0724, -104.624}, {98.2404, -87.1908}, {69.9
{30, -100}, {61.0049, -15.5091}, {1.93226, -5}, {78.4624, -104.72}, {98.8004, -86.7803}, {71.4514, -12
```

Select a Linkage and then press the Start button to generate it in Solidworks.

Figure 3.18: The design system reads the sketch and computes candidate linkage designs for review by the designer.

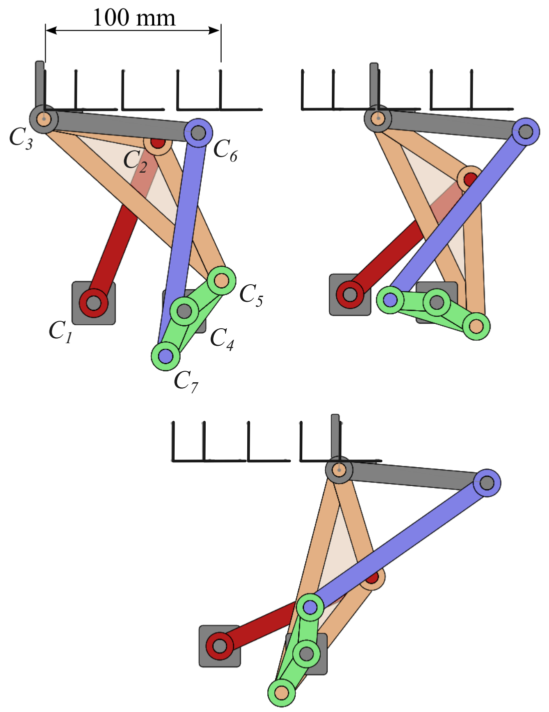


Figure 3.19: For a selected planar six-bar linkage design, the system writes parts and assembly files readable by the solid modeler, in this case, *SolidWorks*.

Table 3.1: Joint Coordinates for the Planar Six-bar Linkage Solution

Pivot	Location Data (x, y) in mm
C_1	$(30.0, -100.0, 0)$
C_2	$(63.19, -16.34)$
C_3	$(4.27, -4.99)$
C_4	$(76.81, -104.21)$
C_5	$(96.07, -88.41)$
C_6	$(67.16, -127.46)$
C_7	$(84.07, -12.11)$

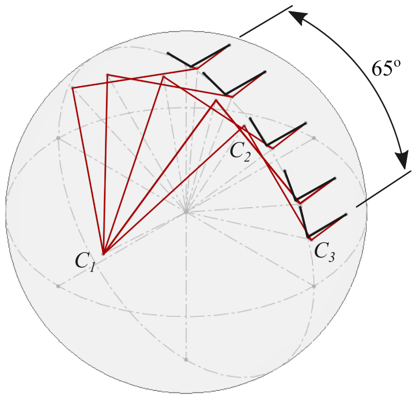


Figure 3.20: The task positions selected along a great circle and the user-defined 3R spherical chains are drawn in the sketch environment of the solid modeler, in this case,*SolidWorks*.

3.10 Summary

This chapter presented the design system for generating defect-free planar and spherical six-bar linkages. It is found that there are in all 6 ways to add the two RR constraints to the 3R chain to yield a maximum of 63 planar six-bar linkages and 165 spherical six-bar linkages. This design system has been successfully implemented in a computer aided linkage design package, MechGen. The design process for MechGen is explained using the example of a spherical car door linkage. Additional examples are provided that achieve rectilinear motion in the plane and motion along a great circle on the sphere. The integration of the design system with a solid modeler provides a practical environment for the design of complex

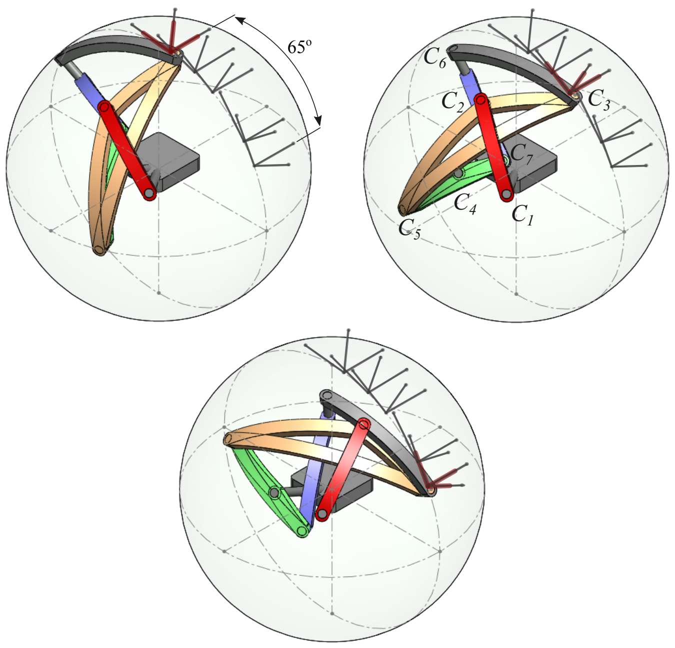


Figure 3.21: For a selected spherical six-bar linkage design, the system writes parts and assembly files readable by the solid modeler, in this case, *SolidWorks*.

mechanical systems.

Table 3.2: Joint Coordinate for the Spherical Six-bar Linkage Solution

Pivot Axis	Location Data (x, y, z) in mm
C_1	$(165.06, 122.39, 190.21)$
C_2	$(19.75, 228.36, 160.82)$
C_3	$(38.91, 277.23, -5.45)$
C_4	$(62.76, 139.40, 234.59)$
C_5	$(96.33, -16.18, 262.41)$
C_6	$(-193.21, 192.42, 63.5917)$
C_7	$(52.44, 215.29, 171.18)$

Chapter 4

Eight-bar Linkages Obtained from a 6R Loop

This chapter presents the design system for planar eight-bar linkages, obtained by adding two RR dyads to a user specified 6R loop, shown in Figure 4.1. The designer specifies five task positions and the 6R loop that can reach these positions. Using the graph theory approach, it is found that there are in all 32 ways to add the two RR constraints to the 6R loop to yield a maximum of 340 planar eight-bar linkages. These linkages are then verified for branch defects and successful linkages are saved. The number of design candidates is increased by varying the task requirements within user specified tolerance zones. This system is demonstrated by finding successful eight-bar linkage designs for a set of rectilinear motion task positions.

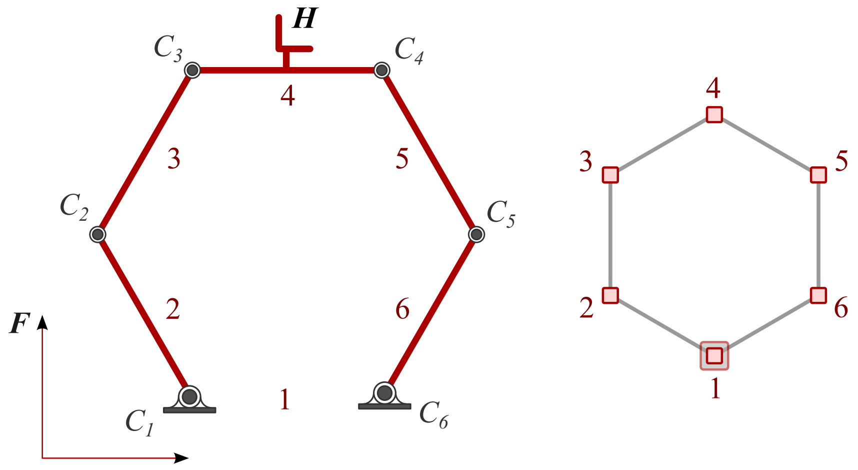


Figure 4.1: A 6R loop with hinged joints $C_1, \ldots, C_6$ and links $\{1, \ldots, 6\}$, together with its linkage graph. Notice that the ground is link 1 and the end-effector is link 4.

4.1 Finding 2 RR constraints using graph theory approach

In order to manage the attachment of two RR dyads to a 6R planar loop, its linkage graph is introduced, see [31]. Let the graph, $G =< V, E >$, be defined by the list of links V and the list of joints E, which represent the two links they connect. The convention used here for numbering the links of a 6R loop is shown in Figure 4.1.

This yields a linkage graph for the 6R loop defined by,

$$\begin{aligned} G &=< V, E > \\ &=< \{1,2,3,4,5,6\}\,,\ \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\},\{1,6\}\} > . \end{aligned} \tag{4.1}$$

The eight-bar linkages obtained by adding two RR dyads to a 6R loop can be viewed as defined by the linkage graphs $L_{(ij)(kl)}$ obtained from G by adding two subgraphs,

$$\begin{aligned} A_{ij} &= < \{i, j, 7\}\,,\, \{\{i, 7\}, \{j, 7\}\} >, \\ B_{kl} &= < \{k, l, 8\}\,,\, \{\{k, 8\}, \{l, 8\}\} > . \end{aligned} \tag{4.2}$$

The vertices (i, j) and (k, l) are to be enumerated to determine all possible eight-bar linkages given by,

$$L_{(ij)(kl)} = G \cup \{\{7, 8\}\,,\, \{\{i, 7\}, \{j, 7\}, \{k, 8\}, \{l, 8\}\}\}. \tag{4.3}$$

In order to enumerate the linkage graphs, $L_{(ij)(kl)}$, the vertex lists introduced are - V_6 for the 6R closed chain and the vertex list V_7 for the linkage after A_{ij} is attached. Enumeration of pairs of vertices in the lists V_6 and V_7 that are available for the attachment of the RR dyads yields the lists P_A and P_B,

$$\begin{aligned} P_A =& \{(i, j) : i, j \in V_6, i \neq j\},\ |P_A| = \binom{6}{2} = 15, \\ P_B =& \{(k, l) : k, l \in V_7, k \neq l\},\ |P_B| = \binom{7}{2} = 21. \end{aligned} \tag{4.4}$$

This shows that the maximum number of linkage graphs, $L_{(ij)(kl)}$ obtained by combination of these lists is given by,

$$|L_{(ij)(kl)}| = |P_A||P_B| = 315. \tag{4.5}$$

However, these 315 linkage graphs are reduced by constraints imposed to eliminate duplicate graphs, to remove links that combine to form a structure, and to add preferred design features to the eight-bar linkage.

4.1.1 Ordering

The possibility that two different linkage graphs describe the same eight-bar linkage can be eliminated by introducing an ordering to the labeling of the attached RR dyads. Specifically, introduce the convention,

$$i < j, \quad k < l. \tag{4.6}$$

Now, order the linkage graphs $L_{(ij)(kl)}$, such that the pairs $(ij)(kl)$ are listed in order of the first entries i and k and then in order of the second entries j and l. This means the two dyads connecting links $\{2, 6\}$ and $\{3, 5\}$ are listed as (26)(35) while the dyads connecting links $\{2, 6\}$ and $\{2, 3\}$, are listed as (23)(26).

This ordering eliminates 105 duplicate graphs reducing the number of linkage graphs to 210.

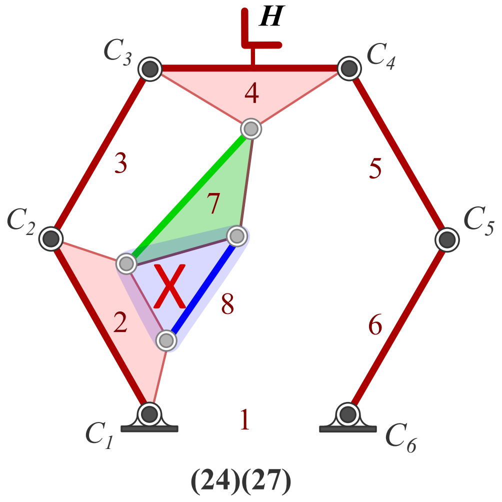
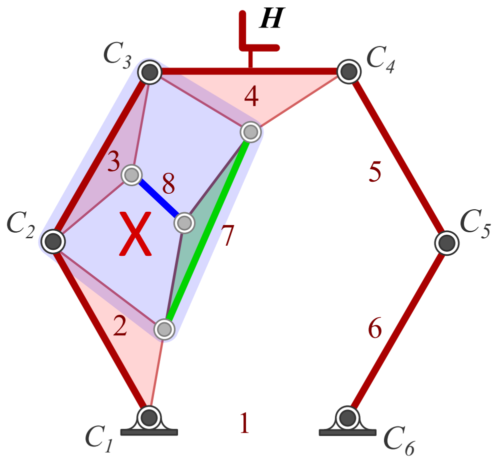


Figure 4.2: RR dyads connecting the links $\{2, 7, 8\}$ yields three joints $\{\{2, 7\}, \{2, 8\}, \{7, 8\}\}$ that form a structure.



(24)(37)

Figure 4.3: The sequence of links $\{2, 3, 4, 7\}$ form a quadrilateral. A dyad connecting vertices $\{2, 4\}$ or $\{3, 7\}$ forms a structure.

4.1.2 Structure Subgraphs

There are two cases in which the addition of two RR dyads creates a substructure, which collapses several links into one. The two cases are attachments that:

1. Form a triangle: If three links $\{r, s, t\}$ are connected by three joints $\{\{r, s\}, \{s, t\}, \{t, r\}\}$, the result is a pin-jointed triangle which forms a structure with no relative movement. An example is the sequence of edges $\{\{2, 7\}, \{2, 8\}, \{7, 8\}\}$ shown in Figure 5.2.

2. Connect opposite sides of a quadrilateral: If four vertices $\{r, s, t, w\}$ form a quadrilateral, then the addition of a dyad connecting either $\{r, t\}$ or $\{s, w\}$ will forms a structure with no relative movement. Figure 5.3 shows the case where links $\{2, 3, 4, 7\}$ forms a structure with the addition of the dyad connecting $\{3, 7\}$.

Eliminating the linkage graphs that have these substructures reduces the number of eight-bar linkage graphs to 69.

4.1.3 Design features

This last condition is artificial in the sense that it imposed to achieve designs that have a particular feature. Through experience it has been observed that an important concern of linkage designers is the location of attachment points to the ground frame. For this reason, the eight-bar linkage graphs are restricted to include only the two connections to the ground frame that were specified as part of the 6R loop. This removes 37 linkage graphs that include a third base pivot. The result is 32 eight-bar linkage graphs.

4.2 Specifying the Design Requirements

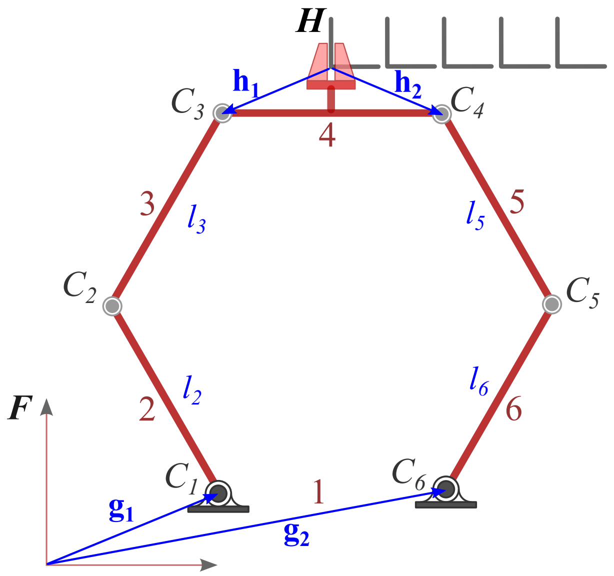


Figure 4.4: Five required task positions, and the configuration of the 6R loop in the first position.

The eight-bar linkage design system requires specification of the five task positions for the end-effector and the dimensions of the 6R loop in the first position, see Figure 4.4. Specifically, the user defines the following requirements,

1. the five transformations, $[D_j], j = 1, \ldots, 5$, that define the movement of the end-effector frame,

$$[D_j] = \begin{bmatrix} \cos\phi_j & -\sin\phi_j & d_{1x,j} \\ \sin\phi_j & \cos\phi_j & d_{1y,j} \\ 0 & 0 & 1 \end{bmatrix}, \; j = 1, \ldots, 5; \tag{4.7}$$

2. the coordinates of the two base pivots, $C_1 : \mathbf{g_1} = (g_{1x}, g_{1y}, 1)$, and $C_6 : \mathbf{g_2} = (g_{2x}, g_{2y}, 1)$;

3. the coordinates of the two moving pivots attached to the end-effector, measured in the end-effector frame, $C_3 : \mathbf{h_1} = (h_{1x}, h_{1y}, 1)$ and $C_4 : \mathbf{h_2} = (h_{2x}, h_{2y}, 1)$;

4. the dimensions, $l_2 = |C_1C_2|$, $l_3 = |C_2C_3|$, $l_5 = |C_4C_5|$ and $l_6 = |C_5C_6|$; and,

5. whether the elbow pivots C_2 and C_5 are configured in the positive or negative configurations.

This data is used to determine the position and orientation of each link of the 6R loop in each of the task positions. The relative positions of links in the 6R loop are then used to synthesis the RR dyads using equation (2.7).

4.3 Inverse Kinematics of the 6R loop

The position and orientation of the each of the links of the 6R loop can be determined by considering the inverse kinematics of the two 3R chains, $\{C_1, C_2, C_3\}$ and $\{C_6, C_5, C_4\}$, that form the 6R loop, see Figure 4.4.

Using the Denavit Hartenberg convention, the kinematics equations for the 3R chain $\{C_1, C_2, C_3\}$

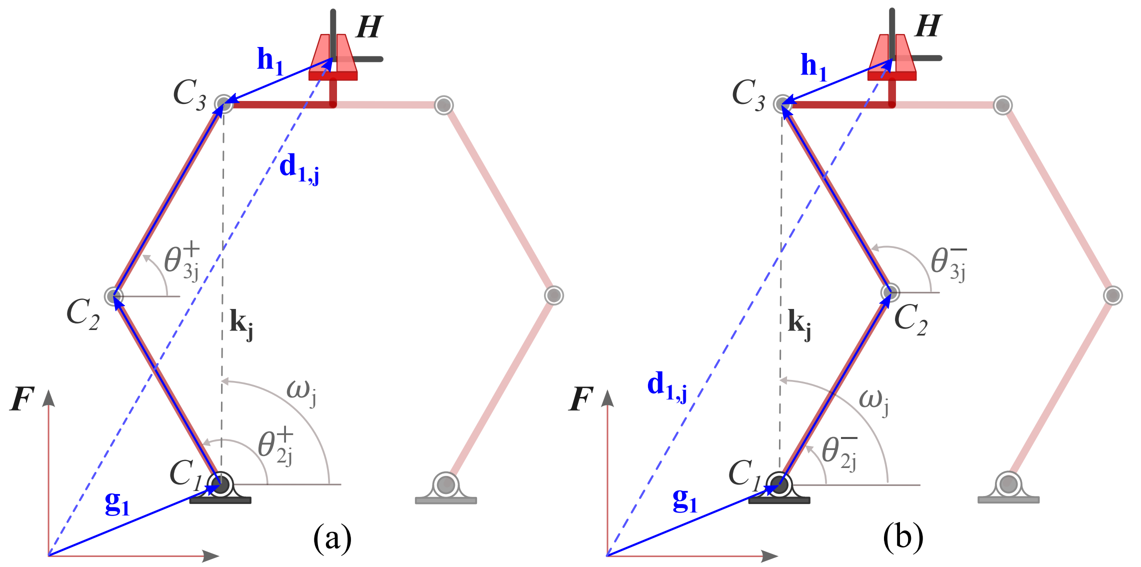


Figure 4.5: (a) is the positive configuration for elbow C_2, and (b) is the negative configuration for elbow C_2 for the left 3R chain $\{C_1, C_2, C_3\}$.

are given as,

$$[D_j] = [T(\mathbf{g}_1)][Z(\theta_{2,j})][X(l_2)][Z(\theta_{3,j} - \theta_{2,j})][X(l_3)][Z(\phi_j - \theta_{3,j})][T(-\mathbf{h}_1)],$$
$$j = 1, \ldots, 5; \tag{4.8}$$

where the homogenous transformation matrices $[T(\mathbf{g})]$, $[Z(\theta)]$ and $[X(l)]$ and are given as,

$$[T(\mathbf{g})] = \begin{bmatrix} 1 & 0 & g_x \\ 0 & 1 & g_y \\ 0 & 0 & 1 \end{bmatrix}, [Z(\theta)] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, [X(l)] = \begin{bmatrix} 1 & 0 & l \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.9}$$

In order to determine the angles $\theta_{2,j}$ and $\theta_{3,j}$, rearrange the kinematics equations into the form,

$$[K_j] = [Z(\theta_{2,j})][X(l_2)][Z(\theta_{3,j} - \theta_{2,j})][X(l_3)][Z(-\theta_{3,j})],$$
$$j = 1, \ldots, 5; \tag{4.10}$$

where $[K_j]$ is the matrix of known constants,

$$[K_j] = [T(\mathbf{g}_1)]^{-1}[D_j][T(-\mathbf{h}_1)]^{-1}[Z(-\phi_j)]. \tag{4.11}$$

Expand equation 4.11 to obtain from the third column of $[K_j]$, the vector $\mathbf{k}_j = (k_{x,j}, k_{y,j})$, given by

$$\begin{aligned} k_{x,j} &= d_{1x,j} + h_{1x}\cos\phi_j - h_{1y}\sin\phi_j \\ k_{y,j} &= d_{1y,j} + h_{1y}\cos\phi_j + h_{1x}\sin\phi_j \end{aligned} \tag{4.12}$$

Next, simplify the right hand side of equation 4.10 to obtain,

$$\begin{aligned} k_{x,j} &= l_2\cos\theta_{2,j} + l_3\cos\theta_{3,j} \\ k_{y,j} &= l_2\sin\theta_{2,j} + l_3\sin\theta_{3,j}, \quad j = 1, \ldots, 5. \end{aligned} \tag{4.13}$$

Introduce the angle ω_j that defines the direction of the vector $\mathbf{k}_j$, which is given by

$$\omega_j = \arctan\left(\frac{k_{y,j}}{k_{x,j}}\right), \quad j = 1, \ldots, 5. \tag{4.14}$$

This combined with the cosine law of the triangle formed by $C_1C_2C_3$ yields,

$$\theta_{2,j} = \omega_j \pm \arccos\left(\frac{l_2^2 + |\mathbf{k}_j|^2 - l_3^2}{2l_2|\mathbf{k}_j|}\right), \quad j = 1, \ldots, 5. \tag{4.15}$$

The arccos function yields positive and negative values for $\theta_{2,j}$, which correspond to the positive and negative configurations of the elbow C_2 as shown in Figure 4.5(a) and (b) respectively.

The angle $\theta_{3,j}$ is now obtained from the equation (4.13), as

$$\theta_{3,j} = \arctan\big(\frac{k_{y,j} - l_2 \sin\theta_{2,j}}{k_{x,j} - l_2 \cos\theta_{2,j}}\big). \tag{4.16}$$

A similar analysis yields the angles angles $\theta_{4,j}$ and $\theta_{5,j}$, the result is that the position and orientation of each link in the 6R loop is defined in each task position.

4.4 Synthesis equations

This data allows the determination of the position and orientation of each of the links, $V_\mu, \mu = 1, \ldots, 6$, of the 6R loop, in each of five task configurations, $D_\nu, \nu = 1, \ldots, 5$. The result is the set of transformations,

$$[K_{\mu,\nu}] = \begin{bmatrix} \cos\theta_{\mu,\nu} & -\sin\theta_{\mu,\nu} & a_{\mu,\nu} \\ \sin\theta_{\mu,\nu} & \cos\theta_{\mu,\nu} & b_{\mu,\nu} \\ 0 & 0 & 1 \end{bmatrix}, \quad \mu = 1, \ldots, 6, \quad \nu = 1, \ldots, 5. \tag{4.17}$$

The synthesis equations for an RR dyads between the links V_i and V_k are obtained by substituting,

$$[R_\nu] = [K_{i,\nu}],\ [S_\nu] = [K_{k,\nu}], \quad \nu = 1, \ldots, 5, \tag{4.18}$$

into equation (2.7). Notice that for convenience the symbol ν replaces the symbol j in the synthesis equations. Once the first RR constraint is applied, it is available for attachment for the second RR constraint. Hence, following the application of first RR constraint, which forms the link 7, the list of transformations, $[K_{\mu,\nu}]$ from equation 4.17, is updated for this new link 7.

4.5 Calculating Candidate Designs

The design system solves the synthesis equations (2.7), to calculate the two RR constraints, for each of the of the 32 unique linkage graphs $L_{(ij)(kl)}$. The number of solutions must account for RR constraint solutions that satisfy the synthesis equations and are already part of the linkage.

This is done by distinguishing two cases, (i) sets of two RR constraints that include only the vertices, $\{1, 2, 3, 4, 5, 6\}$ of the original 6R closed chain, which are referred to as independent constraints, and (ii) sets of two RR constraints, that include one connection to a vertex in the set $\{7\}$, which are referred to as dependent constraints.

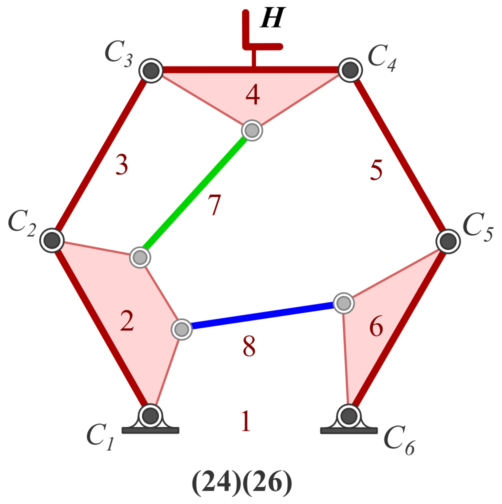


Figure 4.6: Eight-bar linkage graph (24)(26) can produce a maximum of 9 candidate linkage designs.

Independent constraints: A set of two RR constraints that connect to the vertices $i, j, k, l \in \{1, 2, 3, 4, 5, 6\}$ can be applied independent of each other. There are 17 unique ways of attaching the independent constraints as shown in Fig4.12.

In order to count the number of design candidates obtained from the synthesis equations, consider the linkage graph example (24)(26) shown in Figure 5.8. Observe that for the attachment of the first RR constraint A_{24}, one solution C_2C_3 already exists, thus the synthesis equations yield at most three new RR constraints. This is true for the second RR constraint B_{26} as well. In this case, the various combinations of the RR solutions yields as many as $3 \times 3 = 9$ candidate designs.

A systematic count for all the 17 linkage graphs with independent constraints, yields as many as 178 eight-bar candidate linkage designs.

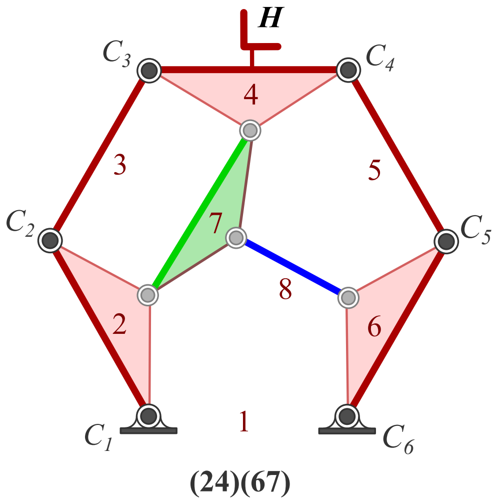


Figure 4.7: Eight-bar linkage graph (24)(67) can produce a maximum of 12 candidate linkage designs.

Dependent constraints: A set of two RR constraints with $i, j, k \in \{1, 2, 3, 4, 5, 6\}$ and $l \in \{7\}$, will have the second RR constraint attached to the first RR constraint. There are 15 unique ways of attaching the dependent constraints as shown in Fig4.13.

In order to count the number of design candidates obtained from the synthesis equations, consider the linkage graph (24)(67) shown in Figure 4.7. Observe that for the attachment of the first RR constraint A_{24}, one solution C_2C_3 already exists, thus the synthesis equations yield at most three new RR constraints. For the application of second RR constraint B_{67}, all the four RR constraint solutions are available. In this case, the various combinations of

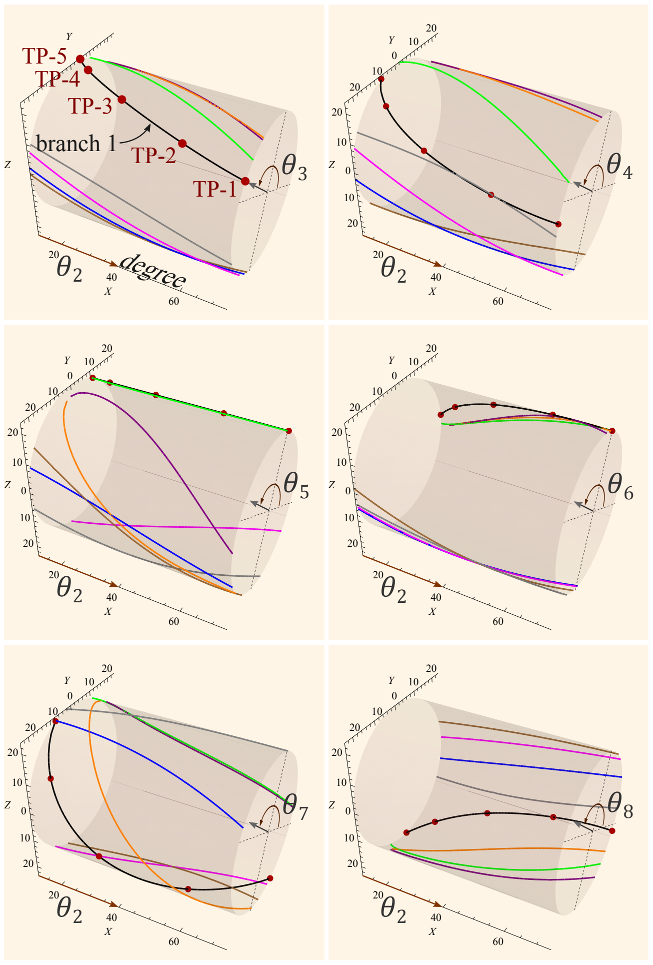


Figure 4.8: Configurations trajectories (branches) for an defect-free eight-bar linkage. All the five task position configurations lie on the same branch 1

the RR solutions yields as many as $3 \times 4 = 12$ candidate designs.

A systematic count for all the 15 linkage graphs with dependent constraints yields as many as 162 eight-bar linkage candidate designs.

Finally, this yields a total of 340 eight-bar designs for the 32 linkage graphs.

4.6 Performance Verification

The performance of a eight-bar linkage design is analyzed to determine the movement of the end-effector for each assembly. The end-effector must pass through the five task positions in a single assembly in order for the candidate linkage to be a successful design.

For the analysis of eight-bar linkages, the automated system developed by [28] is employed. This algorithm reads the location of the pivots of the eight-bar linkage in the first task position and the adjacency matrix of the linkage graph to formulate the three loop equations of the eight-bar linkage. These equations are solved analytically using the Dixon determinant approach in order to obtain all the assemblies of the linkage at each value of the input angle.

A sorting algorithm collects the result of the analysis routine into a maximum of 16 assemblies that define the values of the 10 joint angles for k iterations of the input angle θ_2. Note that θ_1 is the angle made by the ground link which remains constant. The results are the joint trajectories for each of the 16 assemblies,

$$\begin{aligned}
\Theta_1 =&\{\{\theta_{1,1}, \theta_{1,2}, \dots, \theta_{1,10}\}_k\}, \quad k = 1, \dots, n,\\
\Theta_2 =&\{\{\theta_{2,1}, \theta_{2,2}, \dots, \theta_{2,10}\}_k\}, \quad k = 1, \dots, n,\\
&\vdots\\
\Theta_{16} =&\{\{\theta_{16,1}, \theta_{16,2}, \dots, \theta_{16,10}\}_k\}, \quad k = 1, \dots, n. \qquad (4.19)
\end{aligned}$$

For more details about sorting solutions into linkage assemblies to verify performance, see [67].

Figure 5.10 is an example where each assembly trajectory is represented by its joint angle $\theta_3, \dots, \theta_{10}$ trajectories, for the given input joint angle θ_2. In order to meet the performance requirements, all the task positions must lie on one trajectory, or branch, for all the joint angles. In addition to this requirement the designer may also require the link lengths of the linkage to meet certain criteria to be of practical use.

4.7 Tolerance Zones

[47] showed that kinematic synthesis equations regularly fail to yield designs that meet the performance requirements, because the task positions fall on separate branches. This challenge has been overcome by introducing small variations to the task within designer specified tolerance zones. The result is a reliable synthesis procedure that has been implemented in the MechGen series of design systems, [14].

For the example of rectilinear motion linkage discussed later, the location of the ground pivots C_1 and C_6 are randomized, instead of the task positions. The initial pass through the design algorithm, $k = 1$, uses the user defined task positions and 6R loop data, $C_1, \ldots, C_6$. For each subsequent iteration $k = 2, \ldots, q$, where q are the number of iterations specified by the user, the ground pivots are randomized and successful eight-bar linkages are saved. The ground pivots $C_1 = (C_{1x}, C_{1y})$ and $C_6 = (C_{6x}, C_{6y})$ positions are modified by the addition of random variation in the tolerance zones $\Delta C = \{(C_x, C_y) : -\delta < C_x < \delta, -\delta < C_y < \delta\}$, so that

$$C_{1k} = C_1 + \text{Rand}(\Delta C), \quad C_{6k} = C_6 + \text{Rand}(\Delta C), \quad k = 2, \ldots, q. \tag{4.20}$$

where Rand denotes the random selection of a value in the range defined by δ. This introduction of random variations to the ground pivots within user-defined tolerance zones increases the number of successful eight-bar linkage designs.

4.8 Example

In order to demonstrate the design system for eight-bar linkages by constraining the 6R loop, the example with task of an approximate rectilinear motion is presented. Note that in this dissertation, rectilinear motion is referred to as one, where a body moves in a straight line with no change in orientation. A straight line motion linkage, on the other hand, requires only a point on the linkage to move in a straight line, which is a path generation problem. Since the proposed design system can only achieve five task positions, rectilinear motion is only ensured at the five positions, with little control over the motion in between the task positions. This the reason the design system first generates several eight-bar linkage solutions that can achieve these five rectilinear motion task positions and then select the one with minimum deviation from the rectilinear motion. Two cases are shown (i) five task positions that are place symmetrically relative to the 6R loop, and (ii) five task positions that start at the center and extend to one side.

Table 4.1: Five rectilinear motion task positions placed symmetrically about the origin

Task	Orientation (θ) (degrees)	Location(x, y)
1	0°	(−50.0, 0.0)
2	0°	(−25.0, 0.0)
3	0°	(0.0, 0.0)
4	0°	(25.0, 0.0)
5	0°	(50.0, 0.0)

4.8.1 Symmetric rectilinear task positions

The five task positions selected for this example extend over 100mm and are separated by 25mm, see Table 4.1. These task positions are symmetrically positions about the origin. The 6R loop was initially specified as hexagon with 100mm sides, however, this symmetry

Table 4.2: Selected 6R closed chain for symmetric task positions

Pivot	Location(x, y)
C_1	$(-50.00, -150.00)$
C_2	$(-142.11, -110.81)$
C_3	$(-100.00, -20.00)$
C_4	$(0.00, -20.00)$
C_5	$(91.98, -59.24)$
C_6	$(50.00, -150.00)$

Table 4.3: Number of successful eight-bar linkage designs for symmetric task positions

Iterations	Linkage Candidates	Successful Designs	Computation Time
1	46	9	0.737 min
10	411	88	7.084 min
100	4583	1031	68.208 min

does not produce solutions. This was corrected by introducing a 0.1mm addition to the length of the links C_1C_2 and C_2C3, that is $|C_2C_3| = |C_2C_3| = 100.1$mm. The resulting joint coordinates of the 6R closed chain in the first task position is given in Table 4.2.

In order to expand the number of potential designs, a random variation within ± 5mm was introduced for both the (x, y) coordinates of the ground pivots C_1 and C_6. The number of eight-bar designs for $1, 10$ and 100 iterations are shown in Table 4.3. The calculations were performed on an AMD Phenom II, 3.3 GHz, 6 core desktop computer.

Figure 4.9 is an example of a successful design, which in this case is formed by the RR dyads (24)(46). The coordinates of the 10 pivots, $C_1, \ldots, C_{10}$, are listed in Table 4.4. This linkage deviates from the rectilinear task requirements by a maximum of 0.5 micro radians and 26.5 micrometers in the y-direction.

The structure of this eight-bar linkage has the property that none of the links overlap. This lends itself to be manufactured as a single layer with the hinge joints replaced by

Table 4.4: Selected eight-bar linkage solution for symmetric task positions

Pivot	Location(x, y)
C_1	$(-50.00, -150.00)$
C_2	$(-142.11, -110.81)$
C_3	$(-100.00, -20.00)$
C_4	$(0.00, -20.00)$
C_5	$(91.98, -59.24)$
C_6	$(50.00, -150.00)$
C_7	$(-164.50, -13.25)$
C_8	$(-146.05, 3.48)$
C_9	$(46.10, 3.42)$
C_{10}	$(43.42, 28.18)$

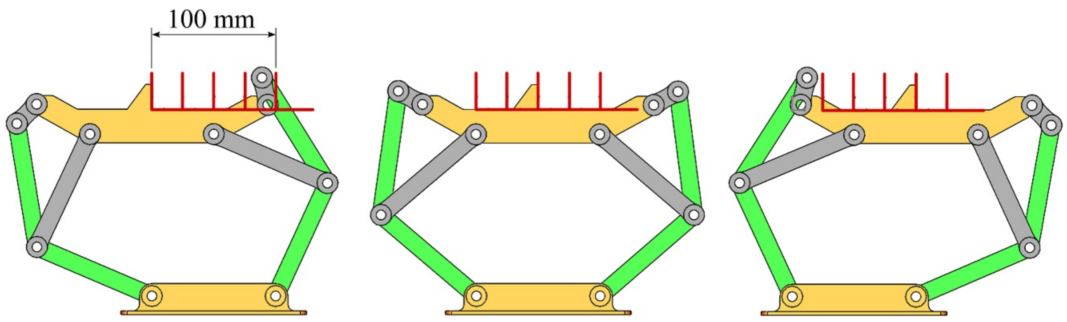


Figure 4.9: Selected eight-bar linkage for symmetric task positions is shown moving through a few positions.

flexure joints, see Howell (2001) [53]. The eight-bar linkage modules can now be stacked and connected together by a four-bar function generator to amplify the approximate rectilinear motion as shown in Figure 4.10. This stacked linkage is a single degree of freedom linkage and provides an approximate rectilinear motion of 300 mm, with maximum deviation in the y direction of 79.5 micrometers.

4.8.2 Offset rectilinear task positions

For this design, each of the task positions in Table 4.1 are shifted in the x direction by the amount of 50mm. The new task positions are given in Table 4.5. Thus, the 6R loop is

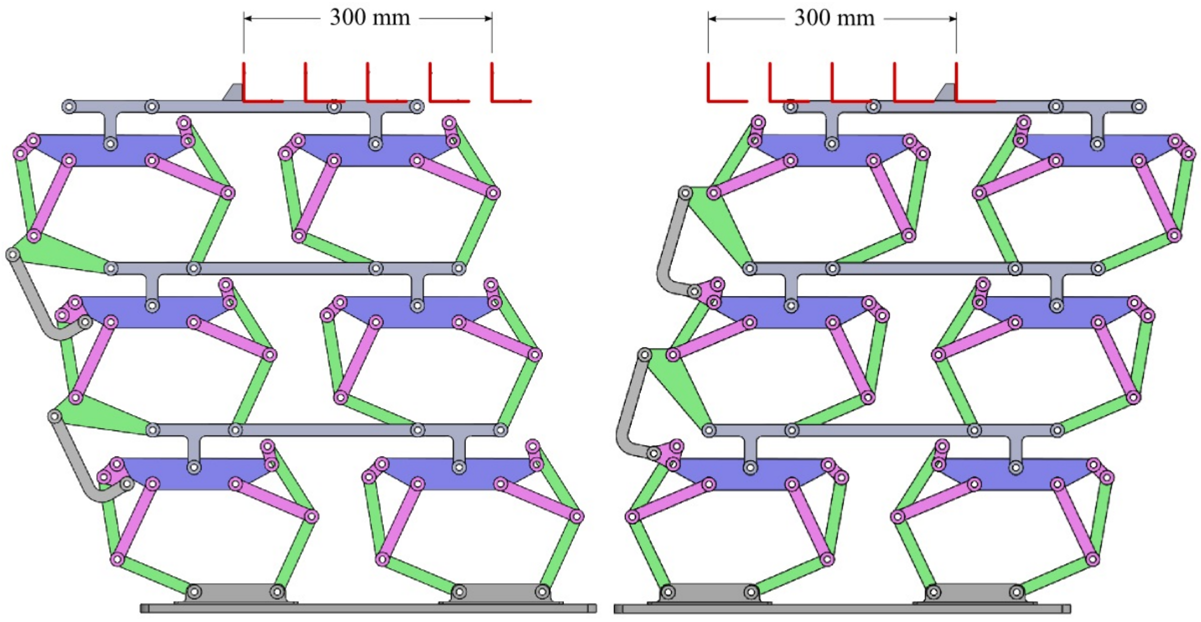


Figure 4.10: The eight-bar linkage obtained for symmetric task positions assembled into three levels connected by a four-bar function generator in order to amplify the rectilinear movement.

centered on the first task position and the remaining are placed over the range of 100mm in the positive x direction. The links of the 6R loop have the same dimensions as in the first example, but the new placement of the task positions yields new coordinates for the joints, $C_1, \ldots, C_6$, Table 4.6. As before, random variations within ±5mm were introduced to the coordinates of the base pivots C_1 and C_6 for each iteration. The results of the calculations by the design system are presented in Table 4.7.

Table 4.5: Five rectilinear motion task positions biased in the positive x axis direction

Task	Orientation (θ) (degrees)	Location(x, y)
1	0°	(0.0, 0.0)
2	0°	(25.0, 0.0)
3	0°	(50.0, 0.0)
4	0°	(75.0, 0.0)
5	0°	(100.0, 0.0)

Table 4.6: Selected 6R closed chain data for offset task positions

Pivot	Location Data (x, y)
C_1	$(50.00, -150.00)$
C_2	$(-126.12, -85.00)$
C_3	$(-50.00, -20.00)$
C_4	$(50.00, -20.00)$
C_5	$(125.99, -85.00)$
C_6	$(50.00, -150.00)$

Table 4.7: Number of successful eight-bar linkage designs for offset task positions

Iterations	Candidate Linkages	Successful Designs	Computation Time
1	60	5	1.74 min
10	641	63	15.61 min
100	6257	584	1 hr 52.80 min

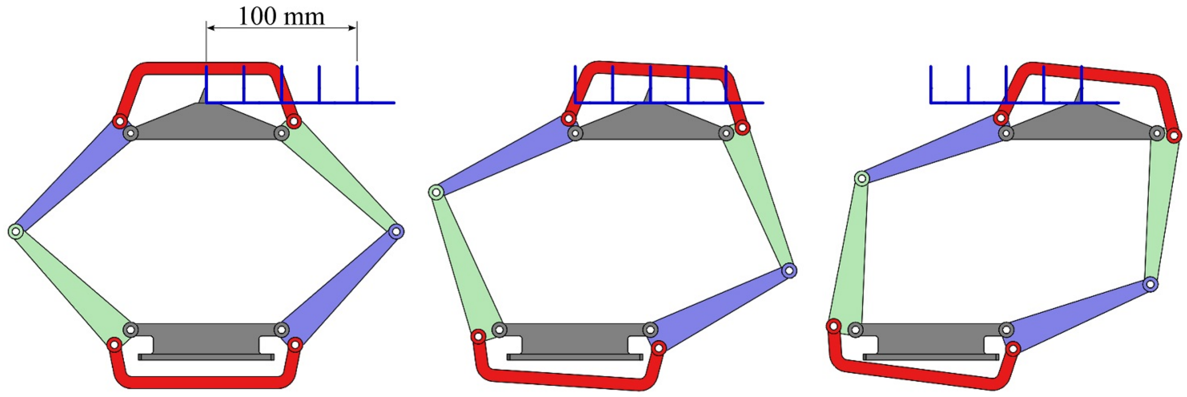


Figure 4.11: Selected eight-bar linkage for offset task positions is shown moving through a few positions

An example eight-bar linkage obtained from this calculation is shown in Figure 4.11. The linkage is obtained by constraining the 6R closed chain robot with two RR constraints (26)(35). The coordinates of its 10 joints, $C_1, \ldots, C_{10}$, are provided in Table 4.8. The maximum angular deviation from horizontal is 5 micro radians and the maximum vertical deviation from the straight line is 5.23 micrometers.

Table 4.8: Selected eight-bar linkage solution for offset task positions

Pivot	Location Data (x, y)
C_1	$(50.00, -150.00)$
C_2	$(-126.12, -85.00)$
C_3	$(-50.00, -20.00)$
C_4	$(50.00, -20.00)$
C_5	$(125.99, -85.00)$
C_6	$(50.00, -150.00)$
C_7	$(-60.80, -159.89)$
C_8	$(59.01, -159.91)$
C_9	$(-57.08, -12.10)$
C_{10}	$(58.18, -12.11)$

4.9 Eight-bar linkage graphs obtained by attaching two RR constraints to a 6R Loop

The 32 linkage graphs $(ij)(kl)$ are separated into (i) independent constraints that have $i, j, k, l \in \{1, 2, 3, 4, 5, 6\}$, Fig4.12, and (ii) dependent constraints that have $i, j, k \in \{1, 2, 3, 4, 5, 6\}$ and $l \in \{7\}$, Fig4.13.

4.10 Summary

This chapter presented the design system for generating defect-free planar eight-bar linkages obtained by adding two RR constraints to the user specified 6R loop. A couple of examples are presented for eight-bar linkages performing approximate rectilinear motion linkage. It is to noted that for a single run of the design algorithm, eight-bar linkage candidates produced are 340, which are significantly higher than six-bar linkage (planar) candidates which are 63. This suggests that as the complexity of the user defined backbone chain increases, significantly more linkages options are available to the designer.

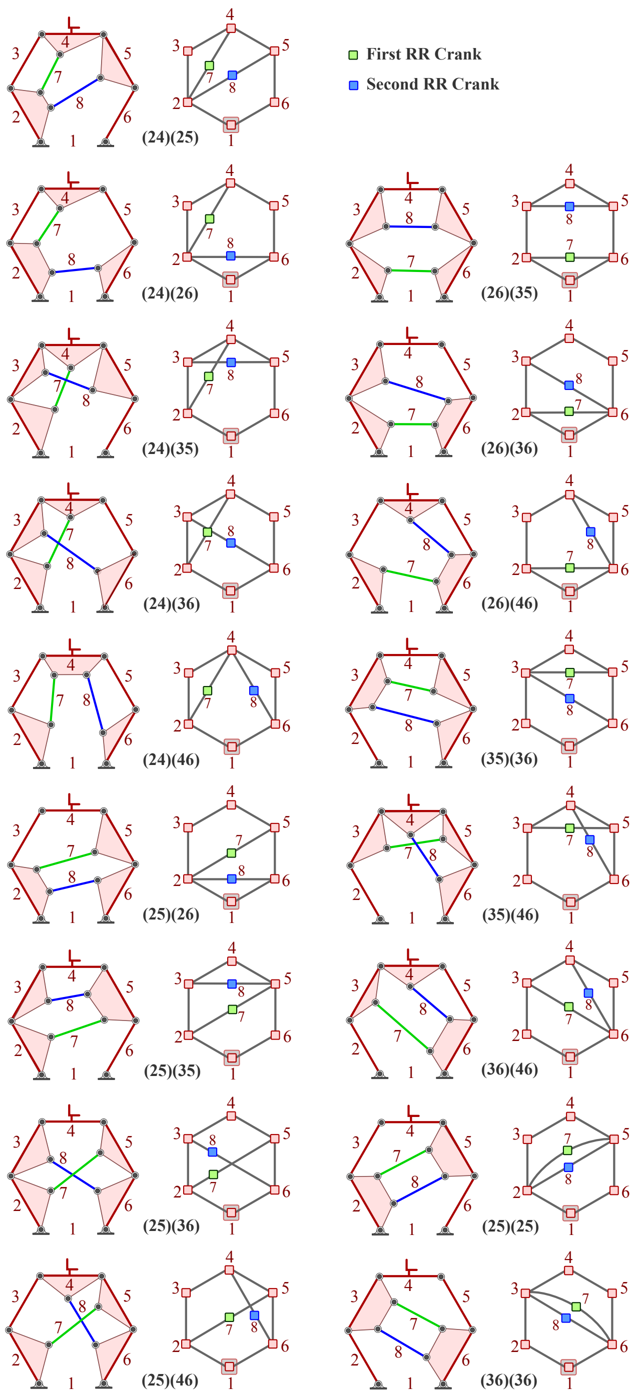


Figure 4.12: The 17 eight-bar linkage graphs $(ij)(kl)$ obtained by adding two independent RR dyads, with $i, j, k, l \in \{1, 2, 3, 4, 5, 6\}$.

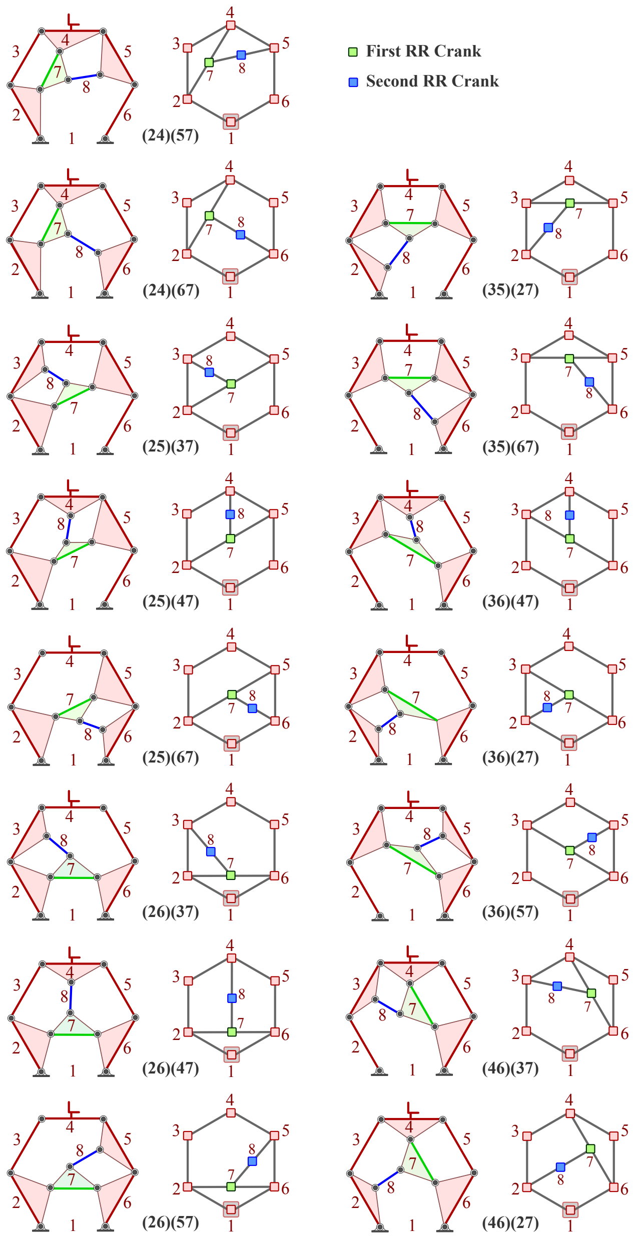


Figure 4.13: The 15 eight-bar linkage graphs $(ij)(kl)$ obtained by adding the second RR dyad to the link of the first RR dyad, that is, with $i, j, k \in \{1, 2, 3, 4, 5, 6\}$ and $l \in \{7\}$.

Chapter 5

Eight-bar Linkages Obtained from a 4R Serial Chain

This chapter presents a design system for eight-bar linkages that begins with a 4R serial chain shown in Fig. 5.1 and obtains the eight-bar linkage by adding three RR constraints. The synthesis procedure is an extension to the technique introduced by Soh et al. (2007) [11], which adds constraining RR chains to a robotic system to obtain a one degree-of-freedom linkage. The design system includes a systematic procedure for finding all possible ways to apply three RR constraints independently and in sequence to the 4R chain. The adjacency matrix of the resulting linkage is used to formulate the analysis equations that verify performance. The system iterates this procedure for variations of the required task within tolerance zones provided by the designer, in order to generate a large number of successful designs. The synthesis of an eight-bar linkage that guides a workpiece along a rectilinear movement is used to demonstrate this design system.

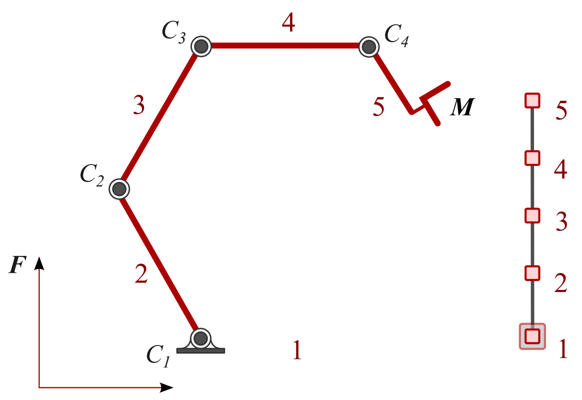


Figure 5.1: 4R open chain serial robot $(C_1, \ldots, C_4)$ with links $\{1, \ldots, 5\}$ along with the graph. Link 1 is the ground link and link 5 is the end-effector link.

5.1 Finding 3 RR constraints using graph theory approach

In this section, a systematic process is presented that identifies attachment locations for a set of three RR constraints that transform a 4R serial chain into an eight-bar linkage. To do this, a linkage graph is introduced for a 4R chain, Tsai (2000) [31] given by $G =< V, E >$, where the list of links V form the vertices of the graph and the list of edges E represents the joints in the linkage. Thus, the linkage graph G of a 4R chain is given by

$$\begin{aligned} G &= < V, E > \\ &= < \{1,2,3,4,5\}, \{\{1,2\},\{2,3\},\{3,4\},\{4,5\}\} > \end{aligned} \tag{5.1}$$

Note that link 1 is the ground and link 5 is the end-effector. See Fig. 5.1.

The goal is to enumerate the linkage graphs L that are obtained from G by adding three subgraphs, each adding a vertex and two edges that represent an RR crank, in a way that reduces the four degrees of freedom of the 4R chain to one degree-of-freedom.

Denote the three RR constraints as the subgraphs,

$$\begin{aligned} A_{ij} &= < \{i, j, 6\}, \{\{i, 6\}, \{j, 6\}\} >, \\ B_{kl} &= < \{k, l, 7\}, \{\{k, 7\}, \{l, 7\}\} >, \\ C_{mn} &= < \{m, n, 8\}, \{\{m, 8\}, \{n, 8\}\} >, \end{aligned} \tag{5.2}$$

where the vertices (i, j), (k, l) and (m, n) are to be enumerated to determine all possible eight-bar linkages. The linkage graph, $L_{(ij)(kl)(mn)}$, obtained by the addition of these RR constraints is given by,

$$\begin{aligned} &L_{(ij)(kl)(mn)} = \\ &G \cup \{\{6, 7, 8\}, \{\{i, 6\}, \{j, 6\}, \{k, 7\}, \{l, 7\}, \{m, 8\}, \{n, 8\}\}\}, \end{aligned} \tag{5.3}$$

In order to enumerate the linkage graphs, $L_{(ij)(kl)(mn)}$, few notations are introduced namely, V_5 for the vertex list of the 4R chain, V_6 for the vertex list of the linkage after A_{ij} is attached, and similarly let V_7 be the vertex list after both A_{ij} and B_{kl} are attached. The lists P_A, P_B and P_C formed from pairs of vertices in the lists V_5, V_6 and V_7 that are available for the attachment of the RR constraints A_{ij}, B_{kl} and C_{mn} are given by

$$\begin{aligned} P_A =& \{(i, j) : i, j \in V_5, i \neq j\}, \ |P_A| = \binom{5}{2} = 10, \\ P_B =& \{(k, l) : k, l \in V_6, k \neq l\}, \ |P_B| = \binom{6}{2} = 15 \\ P_A =& \{(m, n) : m, n \in V_7, m \neq n\}, \ |P_C| = \binom{7}{2} = 21. \end{aligned} \tag{5.4}$$

Thus, the maximum number of linkage graphs, $L_{(ij)(kl)(mn)}$, that can be obtained from the attachment of three RR chains to a 4R chain is given by,

$$|L_{(ij)(kl)(mn)}| = |P_A||P_B||P_C| = 3150. \tag{5.5}$$

This enumeration process yields 3150 sets of three RR attachments that are denoted as $(ij)(kl)(mn)$.

5.1.1 Ordering

In order to eliminate duplicates from the list of linkage graphs, the following convention is introduced,

$$i < j, \quad k < l, \quad m < n. \tag{5.6}$$

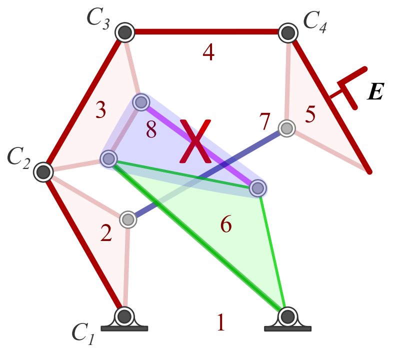
Notice that the three RR constraints, $(ij)(kl)(mn)$, have vertex 6 only in (kl) or (mn) and vertex 7 only in (mn). This leads to an ordering for the linkage graphs given by:

1. if $(ij)(kl)(mn)$ do not include vertices 6 and 7, then order by the first entry followed by the second entry;
2. if $(ij)(kl)(mn)$ includes the vertex 6, insert it as (mn) and sort $(ij)(kl)$ by the first entry followed by the second entry,
3. if $(ij)(kl)(mn)$ includes the vertex 6 twice, insert it as $(kl)(mn)$ and sort by the first entry followed by the second entry.

This ordering makes it possible to identify and eliminate duplicate sets of three RR constraints, which reduces the number of linkage graphs from 3150 to 1275.

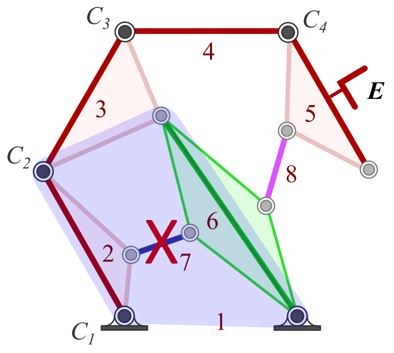
5.1.2 Structure Subgraphs

This list of 1275 linkage graphs reduces further by eliminating the graphs that have the following features:



(13)(25)(36)

Figure 5.2: Application of first and third RR constraint A_{13} and C_{36} causes the links $\{3,6,8\}$ to be connected together by three joints $\{\{3,6\},\{6,8\},\{8,3\}\}$ and forms a structure, hence Linkage graph (13)(25)(36) is invalid.



(13)(26)(56)

Figure 5.3: Application of first RR constraint A_{13} causes the links $\{1,2,3,6\}$ to form a four-bar sub-loop. Application of the second RR constraint B_{26} to this four-bar results in a structure, hence Linkage graph (13)(26)(56) is invalid.

1. Three edges form a triangle, $\{\{i,j\},\{j,k\},\{k,i\}\}$. If three links are connected by three joints, it forms a structure with no relative movement. A typical example is shown in Fig.5.2.

2. A vertex is connected across vertex pairs (i,k) or (j,l) that are opposite sides of a four-bar sub-chain, $\{\{i,j\},\{j,k\},\{k,l\},\{l,i\}\}$. In this case, the five vertices forms a structure with no relative movement. Typical examples include the ones shown in Fig.5.3 and 5.4

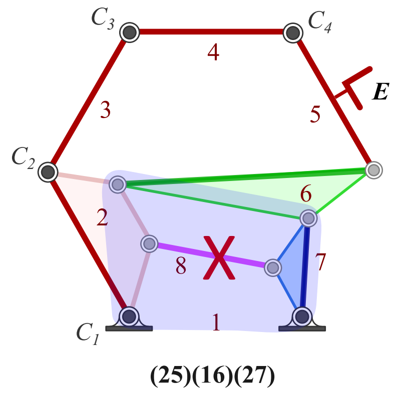


Figure 5.4: Application of the first and second RR constraints A_{25} and B_{16} causes the links $\{1, 2, 6, 7\}$ to form a four-bar sub-loop. Application of the third RR constraint C_{27} to this four-bar results in a structure, hence Linkage graph (25)(16)(27) is invalid.

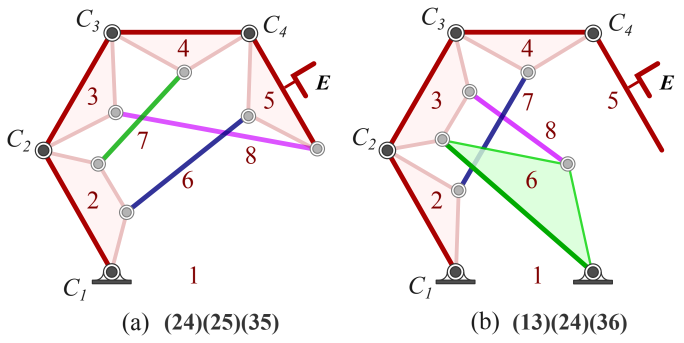


Figure 5.5: The attachment of three RR constraints results in (a) a stucture formed by vertices $\{2, 3, 4, 5, 6, 7, 8\}$ that rotates about a single revolute joint C_1 connected to ground, vertex 1, and (b) vertex 5 is free to rotate about a single revolute joint C_4 connected to a structure formed by vertices $\{1, 2, 3, 4, 6, 7, 8\}$

.

3. Link 1 and link 5 should be connected to the rest of the links by atleast 2 edges. A couple of typical examples of invalid linkage graphs that do not satisfy this are shown in Fig.5.5

The reduces the number of linkage graphs to 152.

5.1.3 Design features

Finally, the constraint that only two edges are connected to the base frame, is introduced. This reduces the demand on the designer to accommodate bearing ground supports for the linkage. The result of these conditions reduces the number of linkage graphs to 100 unique eight-bar linkage graphs.

5.2 Task configurations

Given the linkage graph $L_{(ij)(kl)(mn)}$ for an eight-bar linkage and five task configurations $\nu = 1, \ldots, 5$, the position and orientation of each link $\mu = 1, \ldots, 8$ is defined by the transformation:

$$[K_{\mu,\nu}] = \begin{bmatrix} \cos\theta_{\mu,\nu} & -\sin\theta_{\mu,\nu} & a_{\mu,\nu} \\ \sin\theta_{\mu,\nu} & \cos\theta_{\mu,\nu} & b_{\mu,\nu} \\ 0 & 0 & 1 \end{bmatrix}, \ \mu = 1, \ldots, 8, \ \nu = 1, \ldots, 5. \tag{5.7}$$

The design of an RR constraint say between the vertices $\{i, k\}$ is achieved by making the substitution,

$$[R_\nu] = [K_{i,\nu}], \ [S_\nu] = [K_{k,\nu}], \quad \nu = 1, \ldots, 5, \tag{5.8}$$

and solving the synthesis equations (2.7). Notice that for convenience the symbol ν replaces the symbol j in the synthesis equations.

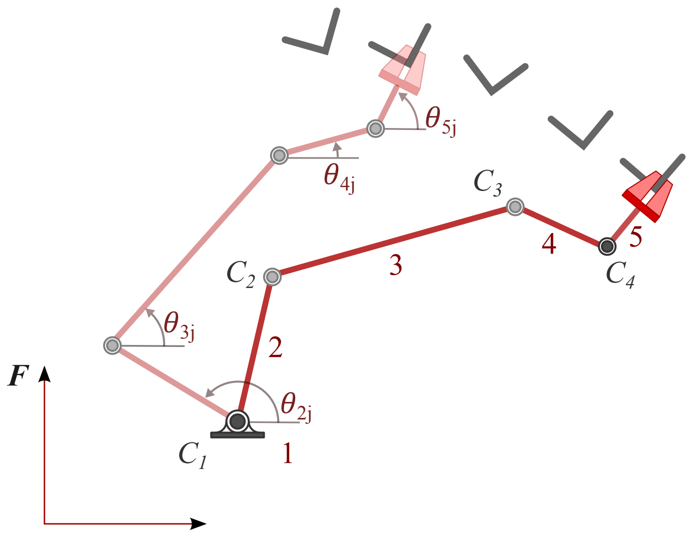


Figure 5.6: Specification of a 4R chain in shown in the first and fourth position.

5.3 Specifying the Design Requirements

The design requirements for the eight-bar linkage are provided by five configurations of a 4R serial chain. This is achieved by specifying:

1. the coordinates $\mathbf{g} = (g_x, g_y, 1)$ of the base joint C_1;
2. the dimensions, (l_2, l_3, l_4), of the four links in the serial chain;
3. the vector $\mathbf{h} = (h_x, h_y, 1)$ from the last joint C_4 to the end-effector frame;
4. the five transformations that define the position $\mathbf{d}_j = \{a_j, b_j\}$ and orientation ϕ_j of the end-effector frame,

$$D_j = \begin{bmatrix} \cos\phi_j & -\sin\phi_j & a_j \\ \sin\phi_j & \cos\phi_j & b_j \\ 0 & 0 & 1 \end{bmatrix}, j = 1, \ldots, 5; \tag{5.9}$$

and,

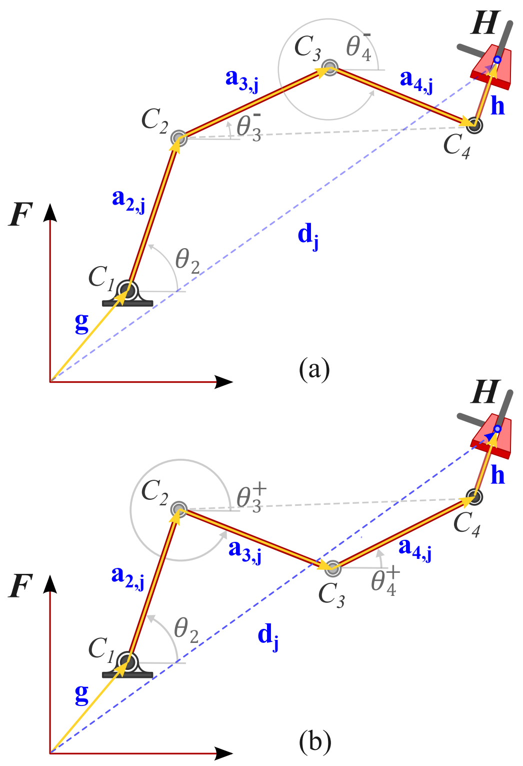


Figure 5.7: For the given input angle θ_2, two different configurations are possible for the 4R chain forms by $\{C_1, C_2, C_3, C_4\}$, namely, (a) elbow up and (b) elbow down.

5. when the end-effector located in each task position, the 4R chain has a degree-of-freedom that the designer is free to specify.

5.4 Inverse Kinematics of the 4R Chain

The kinematics equations of the 4R chain are given by the vector loop equations measured from the origin of the fixed frame F to the origin of the end-effector frame H. Introduce the three link vectors $\mathbf{a}_2 = C_1C_2$, $\mathbf{a}_3 = C_2C_3$, and $\mathbf{a}_4 = C_3C_4$, and compute their coordinates $\mathbf{a}_i = (x_i, y_j)$ in each task position j,

$$\begin{Bmatrix} x_{i,j} \\ y_{i,j} \end{Bmatrix} = \begin{bmatrix} \cos\theta_{i,j} & \sin\theta_{i,j} \\ \sin\theta_{i,j} & \cos\theta_{i,j} \end{bmatrix} \begin{Bmatrix} l_i \\ 0 \end{Bmatrix}, \tag{5.10}$$

or

$$\mathbf{a}_{i,j} = [R(\theta_{i,j}]\mathbf{l}_i, \quad i = 2, 3, 4, \ j = 1, \dots 5. \tag{5.11}$$

The kinematics equations of the 4R chain take the form

$$\mathbf{d}_j = \mathbf{g} + \mathbf{a}_{2,j} + \mathbf{a}_{3,j} + \mathbf{a}_{4,j} + [R(\phi_j)]\mathbf{h}, \ j = 1, \dots, 5. \tag{5.12}$$

When the end-effector is positioned in the task frame, the joints C_1, C_2, C_3 and C_4 form a four-bar linkage. In which case, the angles $\theta_{2,j}$ are free parameters that are specified by the designer. The remaining angles $\theta_{3,j}$ and $\theta_{4,j}$ are related by the condition,

$$|(\mathbf{g} + \mathbf{a}_{2,j}) - (\mathbf{d}_j - [R(\phi_j)]\mathbf{h} - \mathbf{a}_{4,j})| = l_2^2 \tag{5.13}$$

This equation can be solved to determine $\theta_{4,j}$ using the analysis of a four-bar linkage, at which point $\theta_{3,j}$ can also be determined. See McCarthy and Soh (2010) [12].

For a given value of the joint angle $\theta_{2,j}$, the four-bar linkage has two configurations $(\theta_{3,j}, \theta_{4,j})^+$ such that C_3 is below the diagonal C_2C_4, or $(\theta_{3,j}, \theta_{4,j})^-$ that has C_3 above this diagonal. The selection of the joint angle $\theta_{2,j}$ and the configuration of the 4R chain defines the position of each link for the 4R chain in each of the task positions. This defines the transformations $[K_{i,j}]$, $i = 2, 3, 4, 5$ and $j = 1, \dots, 5$.

5.5 4R Chain with Equal Link Lengths

In the formulation of the specifications for the synthesis of an eight-bar linkage it is convenient to select equal values for the link lengths of the 4R chain, that is

$$l = l_2 = l_3 = l_4. \tag{5.14}$$

And Eq. (5.13) can be used to determine the configuration of the 4R chain for given values of $\theta_{2,j}$.

It is tempting to select $\theta_{2,j}$ so that $\theta_{2,j} = 2\pi + 2\psi - \theta_{4,j}$, which creates a symmetric configuration for the 4R chain. Note that ψ is the angle made by C_1C_4 with the horizontal line. However, this symmetry in the task specification results in the synthesis of repeated RR constraints. A small amount of asymmetry between $\theta_{2,j}$ and $\theta_{4,j}$ ensures a complete solution of the synthesis equations.

5.6 Calculating Candidate Designs

The design system solves the synthesis equations (2.7), to calculate the three RR constraints, for each of the of the 100 unique linkage graphs, $L_{(ij)(kl)(mn)}$. The number of solutions must account for RR constraint solutions that satisfy the synthesis equations and are already part of the linkage.

This is done by distinguishing three cases, (i) sets of three RR constraints that include only the vertices, $\{1, 2, 3, 4, 5\}$, of the original 4R chain, (ii) sets of three RR constraints that include one connection to a vertex in the set $\{6, 7\}$, and (iii) sets of three RR constraints that include connections to vertices $\{6, 7\}$.

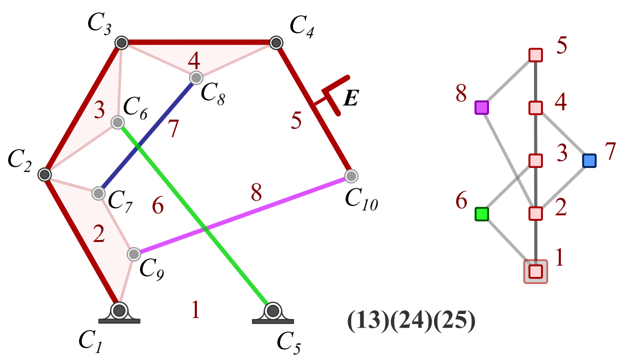


Figure 5.8: Example of linkage graph (13)(24)(25) with independent constraints, which yield as many as $3 \times 3 \times 4 = 36$ candidate designs.

Independent constraints: A set of three RR constraints that connect to the vertices $i, j, k, l, m, n \in \{1, 2, 3, 4, 5\}$ can be applied independent of each other. Of the $\binom{5}{3} = 20$ ways to select these attachments 12 are unique and are shown in Fig.5.14.

In order to count the number of design candidates obtained from the synthesis equations, consider the linkage graph (13)(24)(35) shown in Fig.5.8. Observe that for the attachment of the first RR constraint A_{13}, one solution C_1C_2 already exists, thus the synthesis equations yield at most three new RR constraints. This is true for the second RR constraint B_{24} as well. In contrast, the third RR constraint C_{27} does not bridge an existing constraint so all four RR solutions are available. In this case, the combinations of the RR solutions yields as many as $3 \times 3 \times 4 = 36$ candidate designs.

A systematic count for all the 12 linkage graphs with independent constraints yields as many as 429 eight-bar linkage candidate designs.

Level one constraints: A set of three RR constraints with $i, j, k, l \in \{1, 2, 3, 4, 5\}$, $m \in \{1, 2, 3, 4, 5, 6\}$ and $n \in \{6, 7\}$, is said to have first level dependence because it includes a connection to a previously designed RR constraint. There are 48 first level dependent linkage graphs as shown in Fig.5.15, 5.16 and 5.17. Systematic evaluation of the available solutions

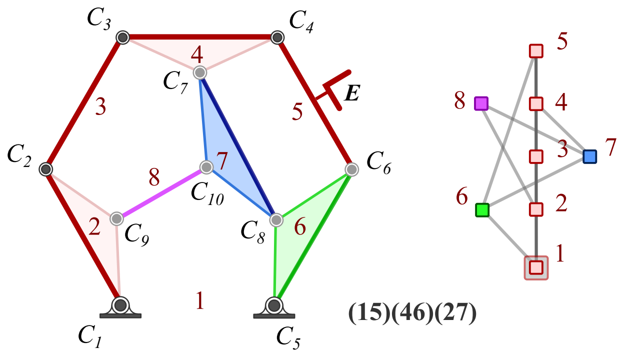


Figure 5.9: Example of linkage graph (15)(46)(27) with dependent RR constraints (second level), which yield as many as $4 \times 3 \times 4 = 48$ candidate designs.

to the synthesis equations for these cases yields 1895 candidate designs.

Level two constraints: A set of three RR constraints with $i, j, k, m \in \{1, 2, 3, 4, 5\}$, $l \in \{6\}$ and $n \in \{6, 7\}$, is said to have second level dependent because it includes connections to both previously designed RR constraints. There are 40 second level dependent linkage graphs as shown in Fig.5.18 and 5.19. Systematic evaluation of the available solutions to the synthesis equations for first level dependence yields 1627 candidate designs.

An example of a set of second level constraints is shown in 5.9. The synthesis equations for the RR constraint A_{15} has as many as four solutions, while the second RR constraint B_{46} has three, since the C_4C_6 already exists. The third RR constraint C_{27} has all the four RR solutions. Thus, there are as many as $4 \times 3 \times 4 = 48$ candidate designs for this case.

Finally, this yields a total of 3951 eight-bar designs for the 100 linkage graphs.

5.7 Performance Verification

The performance of a eight-bar linkage design is analyzed to determine the movement of the end-effector for each assembly. The end-effector must pass through the five task positions in

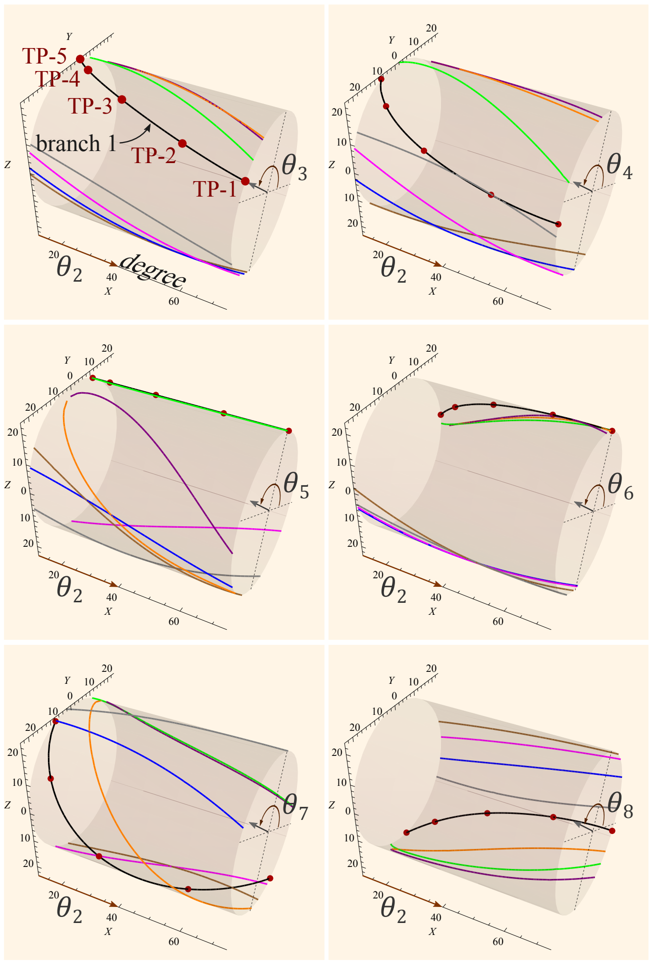


Figure 5.10: Configurations trajectories (branches) for an defect-free eight-bar linkage. All the five task position configurations lie on the same branch 1

a single assembly in order for the candidate linkage to be a successful design.

For the analysis of eight-bar linkages, the automated system developed by Parrish et al. (2014) [28] is employed. This algorithm reads the location of the pivots of the eight-bar linkage in the first task position and the incidence matrix of the linkage graph to formulate the three loop equations of the eight-bar linkage. These equations are solved analytically using the Dixon determinant approach in order to obtain all the assemblies of the linkage at each value of the input angle.

A sorting algorithm collects the result of the analysis routine into a maximum of 16 assemblies that define the values of the 10 joint angles for k iterations of the input angle θ_2. Note that θ_1 is the angle made by the ground link which remains constant. The results are the joint trajectories for each of the 16 assemblies,

$$\begin{aligned}
\Theta_1 =& \{\{\theta_{1,1}, \theta_{1,2}, \ldots, \theta_{1,10}\}_k\}, \quad k = 1, \ldots, n, \\
\Theta_2 =& \{\{\theta_{2,1}, \theta_{2,2}, \ldots, \theta_{2,10}\}_k\}, \quad k = 1, \ldots, n, \\
&\vdots \\
\Theta_{16} =& \{\{\theta_{16,1}, \theta_{16,2}, \ldots, \theta_{16,10}\}_k\}, \quad k = 1, \ldots, n. \qquad (5.15)
\end{aligned}$$

For more details about sorting solutions into linkage assemblies to verify performance, see Plecnik and McCarthy (2013) [67].

Fig.5.10 is an example where each assembly trajectory is represented by its joint angle $\theta_3, \ldots, \theta_{10}$ trajectories, for the given input joint angle θ_2. In order to meet the performance requirements, all the task positions must lie on one trajectory, or branch, for all the joint angles. In addition to this requirement the designer may also require the link lengths of the linkage to meet certain criteria to be of practical use.

5.8 Tolerance Zones

McCarthy and Choe [47] showed that kinematic synthesis equations regularly fail to yield designs that meet the performance requirements, because the task positions fall on separate branches. This challenge has been overcome by introducing small variations to the task within designer specified tolerance zones. The result is a reliable synthesis procedure that has been implemented in the MechGen series of design systems, Sonawale et al. (2013) [**?**].

The initial pass through the design algorithm uses the required task positions, $(\phi_j, \mathbf{d}_j)$,

$j = 1, \ldots, 5$. For each iteration $k = 1, \ldots, q$, where q are the number of iterations specified by the user, the successful design are saved, and the task positions are modified by the addition a random variation from the tolerance zones $(\pm\Delta\phi_j, \pm\Delta\mathbf{d}_j)$, $j = 1, \ldots, 5$, given by

$$\begin{aligned}\{(\phi_i, \mathbf{d}_j), &j = 1, \ldots, 5\}_k = \{(\phi_i, \mathbf{d}_j), j = 1, \ldots, 5\} \\ &+ \{\text{Rand}(\pm\Delta\phi_j, \pm\Delta\mathbf{d}_j), j = 1, \ldots, 5\}, \quad k = 1, \ldots, q. \end{aligned} \tag{5.16}$$

This strategy of introducing variations to the task variations within designer specified tolerance zones yields successful eight-bar linkage designs.

5.9 Example

In this section, the design system for eight-bar linkages by constraining the 4R serial chain is demonstrated by discussing again an approximate rectilinear motion example. The five rectilinear task positions selected are given in Tab.5.1 and the user specified 4R chain information is given in Tab.5.2. The tolerances for each task position $(\Delta\phi_j, \Delta\mathbf{d}_j), j = 1, \ldots, 5$ are given in Tab. 5.3. Notice that since the tolerances are specified on the task positions, the accuracy of rectilinear motion of the linkage solutions is limited to the tolerances specified. A criteria was imposed on the link lengths to find successful linkage solutions. A length metric $l = 200$ was introduced and the minimum allowable link length was chosen to be 1% of l and the maximum allowable link length was chosen to be 100% of l. The results for number of iterations $1, 10$ and100 are shown in Tab.5.4. The calculations were performed on an AMD Phenom II, 3.3 GHz, 6 core Windows machine. The linkage solution with the least deviation from the rectilinear motion is mentioned in Tab. 5.5, as coordinates of the various joints $(C_1, \ldots, C_{10})$. The three RR constraints are applied between the link pairs $(2, 4)(3, 5)(1, 7)$ as shown in Fig.5.11. Figure 5.12 show the linkage movement through the five task positions. A Solidworks rendering of the linkage solution is shown in Fig. 5.13.

Table 5.1: Five rectilinear motion task positions

Task	Orientation (θ) (degrees)	Location $\mathbf{d}$: (x, y)
1	0°	(0.0, 0.0)
2	0°	(22.0, 0.0)
3	0°	(50.0, 0.0)
4	0°	(78.0, 0.0)
5	0°	(100.0, 0.0)

Table 5.2: Selected 4R serial chain data

Ground Pivot (C_1)	$(50.0, -90.0, 0)$
End-effector Pivot (C_4)	$(-3.54, -3.54)$
Common length (a)	45
Elbow Position	up (-1)

5.10 Eight-bar linkage graphs obtained by attaching three RR constraints to a 4R serial chain

Here the 100 linkage graphs $(ij)(kl)(mn)$ are presented, which are obtained by the application of three RR constraints. They are separated into (i) independent constraints that have $i, j, k, l, m, n \in \{1, 2, 3, 4, 5\}$, Fig.5.14, (ii) level one constraints that have $i, j, k, l \in \{1, 2, 3, 4, 5\}$, $m \in \{1, 2, 3, 4, 5, 6\}$ and $n \in \{6, 7\}$, Fig.5.15, 5.16 and 5.17, and (iii) level two constraints with $i, j, k, m \in \{1, 2, 3, 4, 5\}$, $l \in \{6\}$ and $n \in \{6, 7\}$, Fig.5.18 and 5.19 .

5.11 Summary

This chapter presented the design system for generating defect-free planar eight-bar linkages, obtained by adding three RR constraints to the user specified 4R serial chain. It is found that there are 100 different ways to apply the three RR constraints to the 4R serial chain,

Table 5.3: Tolerances on the Five Task Positions

Task	Tolerance Data$(\Delta\phi, \Delta\mathbf{d})$
1	$(0.0°, \{1.0, 0.001\})$
2	$(0.1°, \{5.0, 0.001\})$
3	$(0.2°, \{5.0, 0.001\})$
4	$(0.1°, \{5.0, 0.001\})$
5	$(0.0°, \{1.0, 0.001\})$

Table 5.4: Number of successful eight-bar linkage designs obtained by the design system

No. of Iterations	No. of linkages Synthesized	No. of successful Linkages	Time Taken
1	1577	14	17.8 min
10	12179	108	141.94 min
100	110454	853	43 hr 52 min

to generate as many as 3951 eight-bar linkages. In the earlier chapters, it is observed that adding two RR constraints to a 6R loop generates a maximum of 340 solutions, and adding two RR constraints to a 3R serial chain generates a maximum of 63 planar six-bar linkages. This reinstates the fact that as the complexity of the user defined backbone chain increases, significantly more linkages options are available to the designer.

Table 5.5: Selected Eight-bar Linkage Solution for rectilinear motion

Pivot	Location Data (x, y)
C_1	$(50.0, -90.0, 0)$
C_2	$(62.17, -46.68)$
C_3	$(41.35, -6.78)$
C_4	$(-3.54, -3.54)$
C_5	$(46.69, -39.86)$
C_6	$(56.43, -20.30)$
C_7	$(-4.65, 15.47)$
C_8	$(4.75, 11.67)$
C_9	$(104.60, -78.80)$
C_{10}	$(5.75, -15.33)$

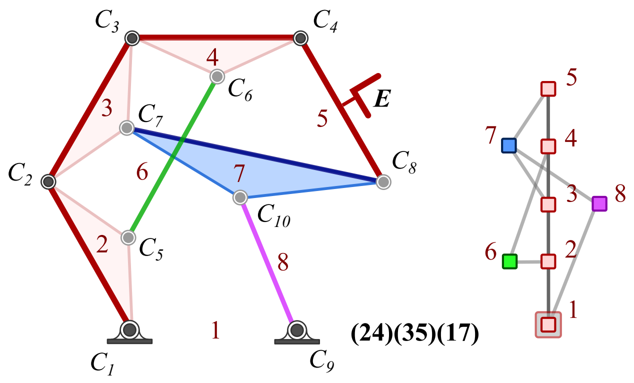


Figure 5.11: Topology of the selected eight-bar linkage graph (24)(35)(17) for rectilinear motion

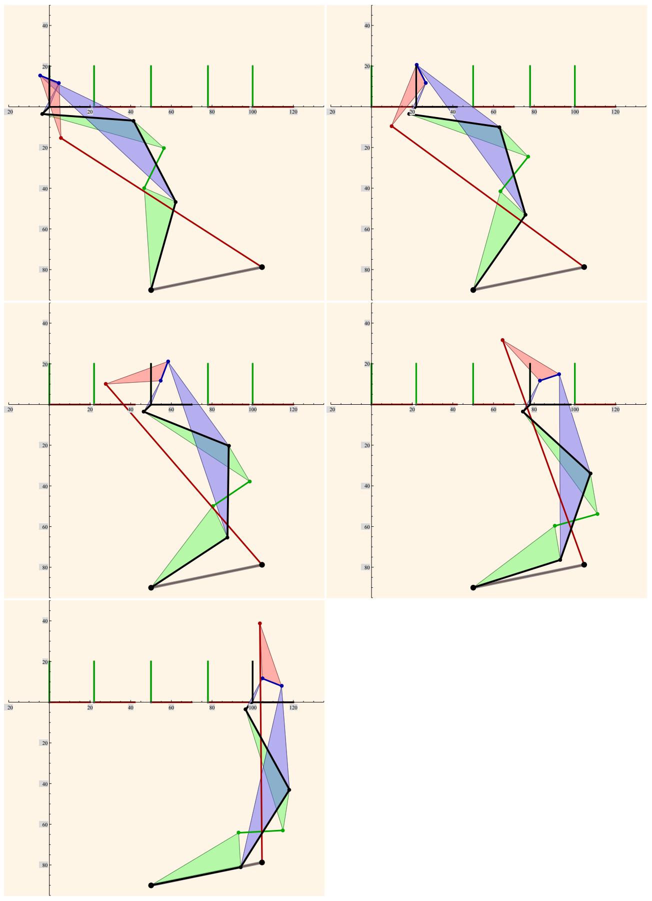


Figure 5.12: Selected Eight-bar linkage for Rectilinear motion is shown moving through the five task positions

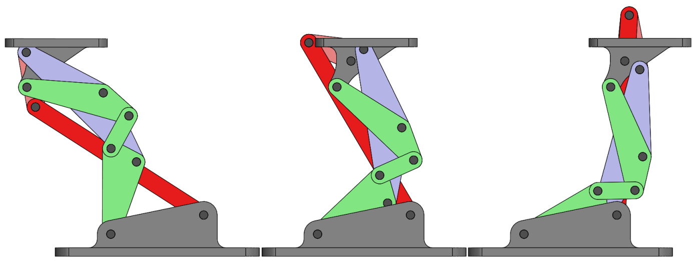


Figure 5.13: Selected linkage solution rendered in SolidWorks is shown moving through a few rectilinear positions

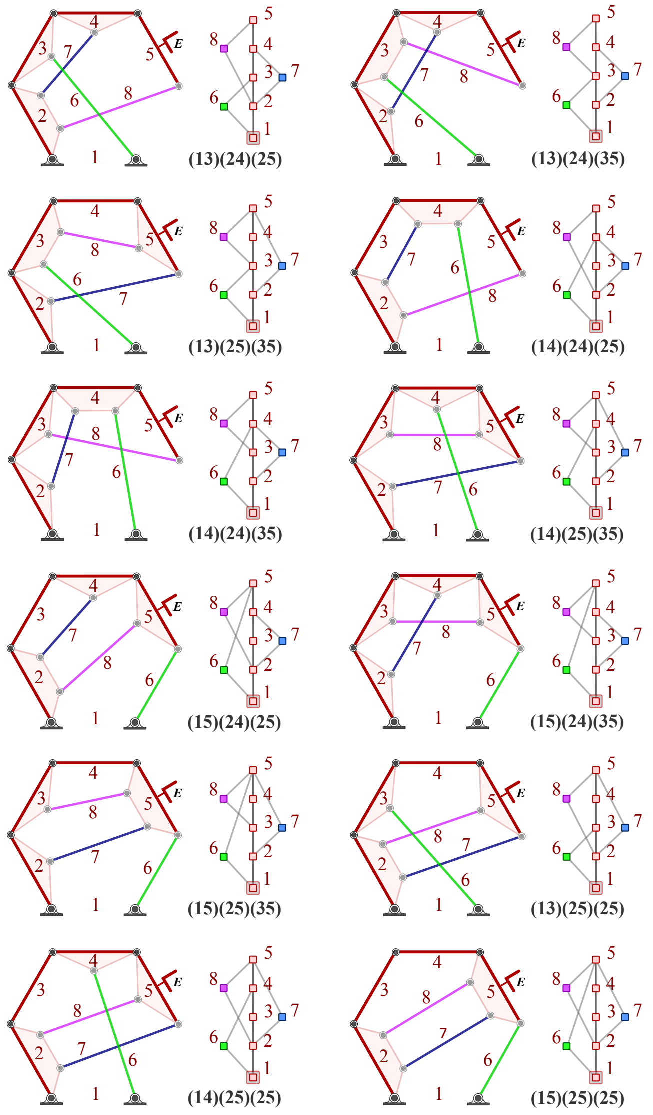


Figure 5.14: Total 12 eight-bar linkage graphs $(ij)(kl)(mn)$ can be obtained by the application of three independent RR constraints, with $i, j, k, l, m, n \in \{1, 2, 3, 4, 5\}$.

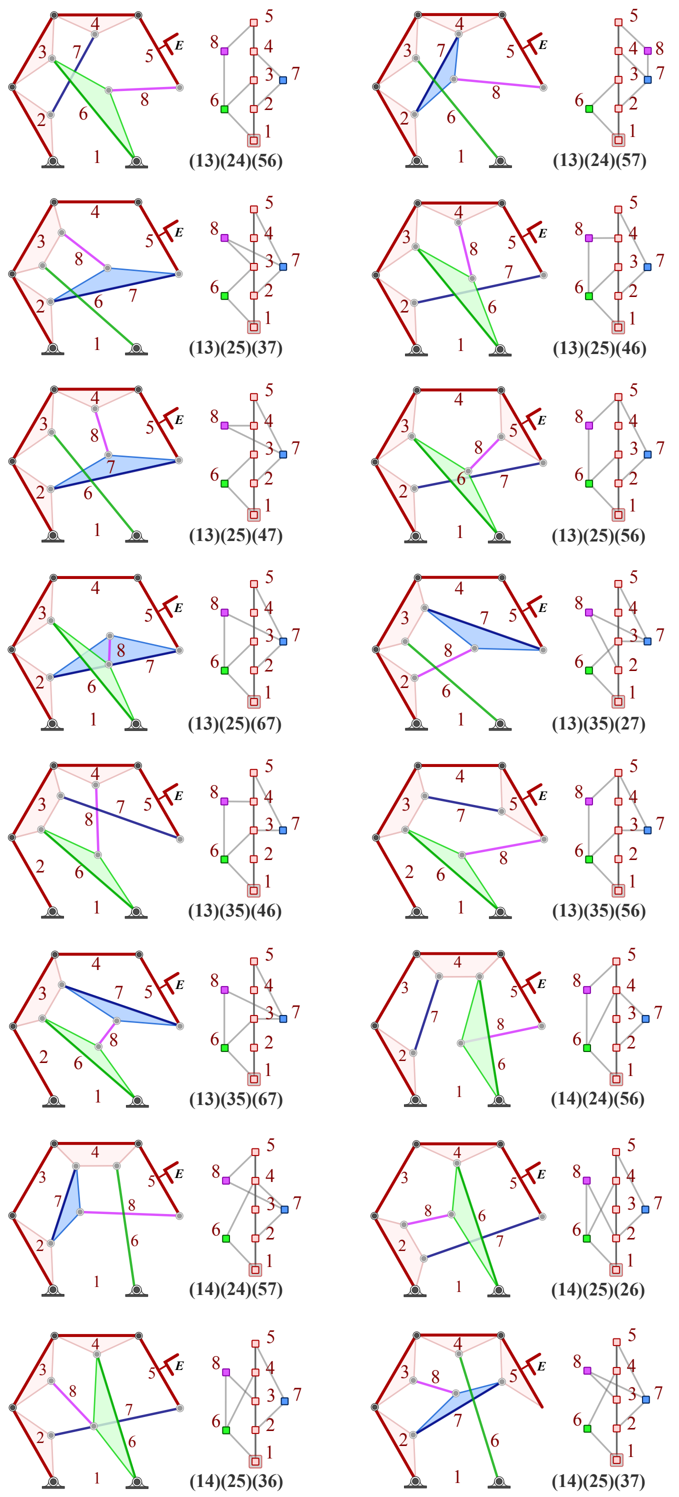


Figure 5.15: Showing $1 - 16$ eight-bar linkage graphs $(ij)(kl)(mn)$ out of 48 which exhibit first level dependency, with $i, j, k, l \in \{1, 2, 3, 4, 5\}$, $m \in \{1, 2, 3, 4, 5, 6\}$ and $n \in \{6, 7\}$.

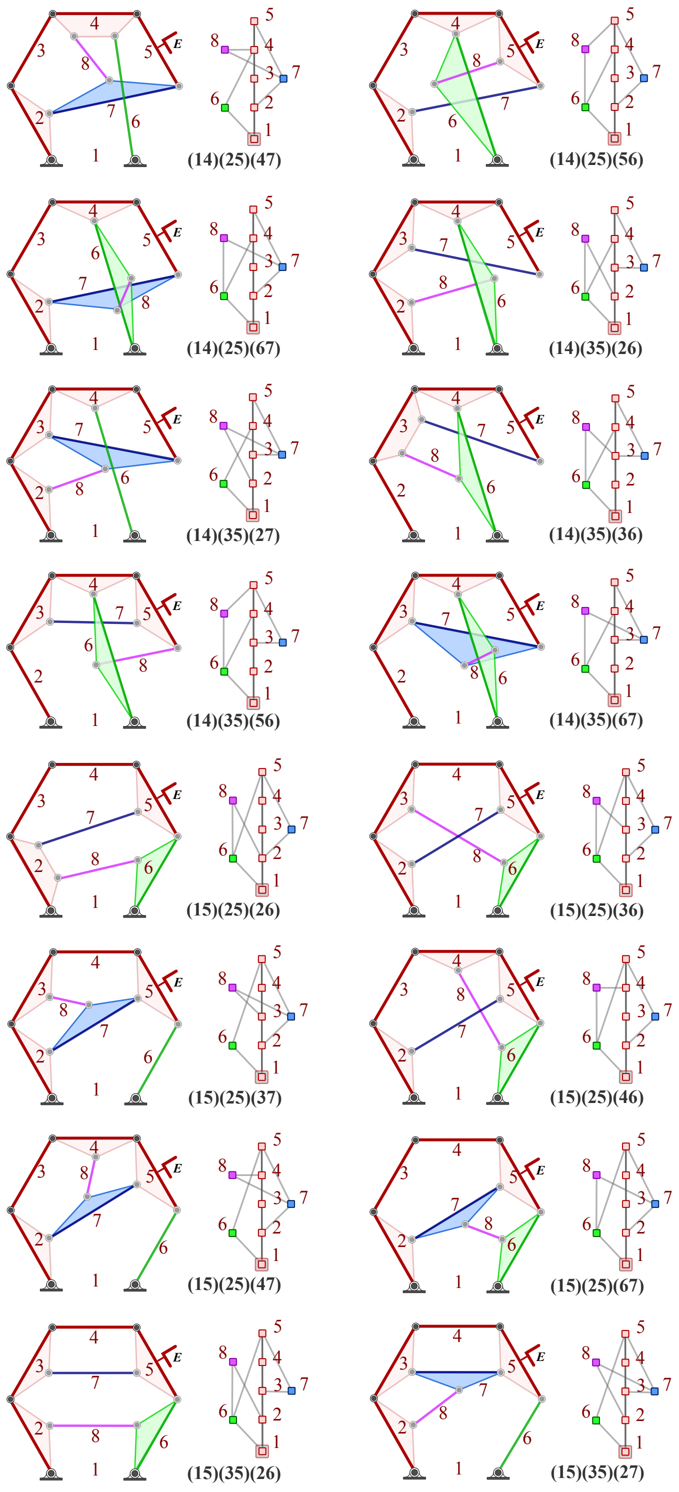


Figure 5.16: Showing $17-32$ eight-bar linkage graphs $(ij)(kl)(mn)$ out of 48 which exhibit first level dependency, with $i, j, k, l \in \{1, 2, 3, 4, 5\}$, $m \in \{1, 2, 3, 4, 5, 6\}$ and $n \in \{6, 7\}$.

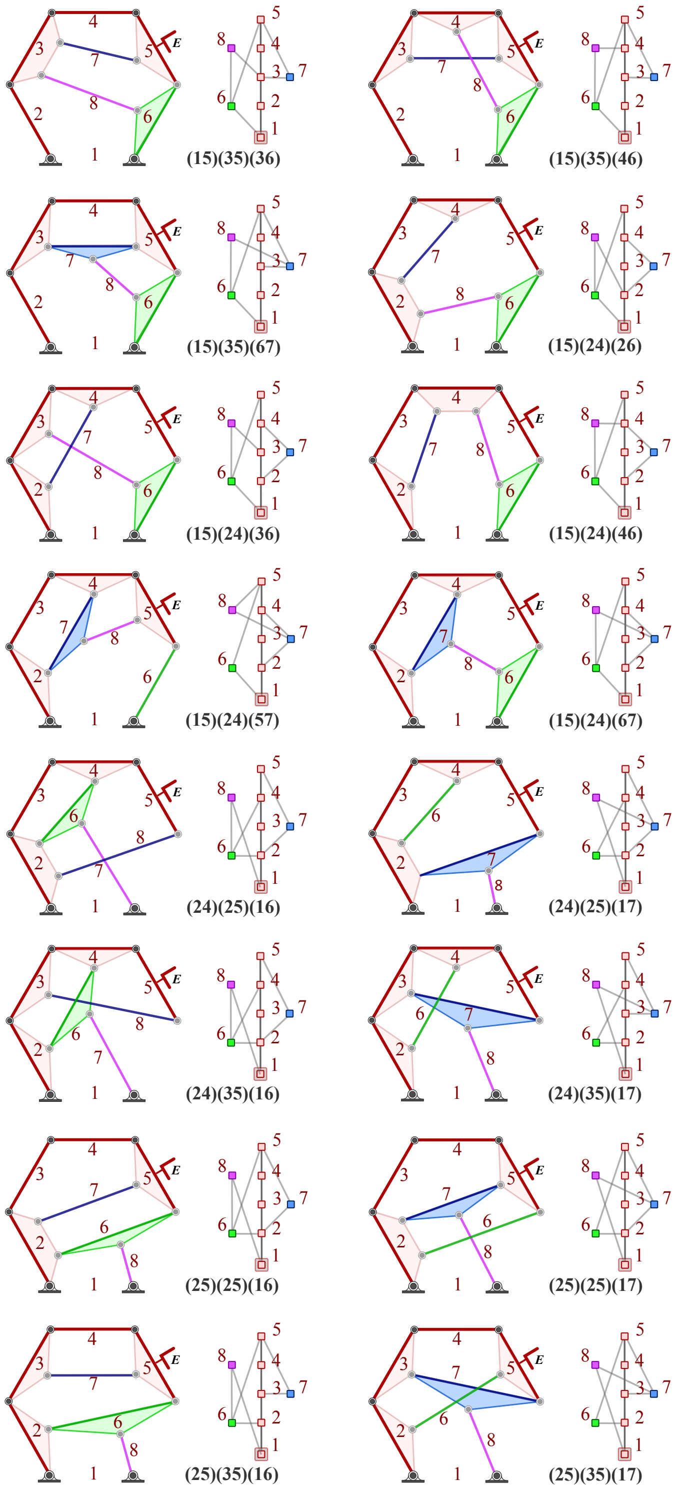


Figure 5.17: Showing $33 - 48$ eight-bar linkage graphs $(ij)(kl)(mn)$ out of 48 which exhibit first level dependency, with $i, j, k, l \in \{1, 2, 3, 4, 5\}$, $m \in \{1, 2, 3, 4, 5, 6\}$ and $n \in \{6, 7\}$.

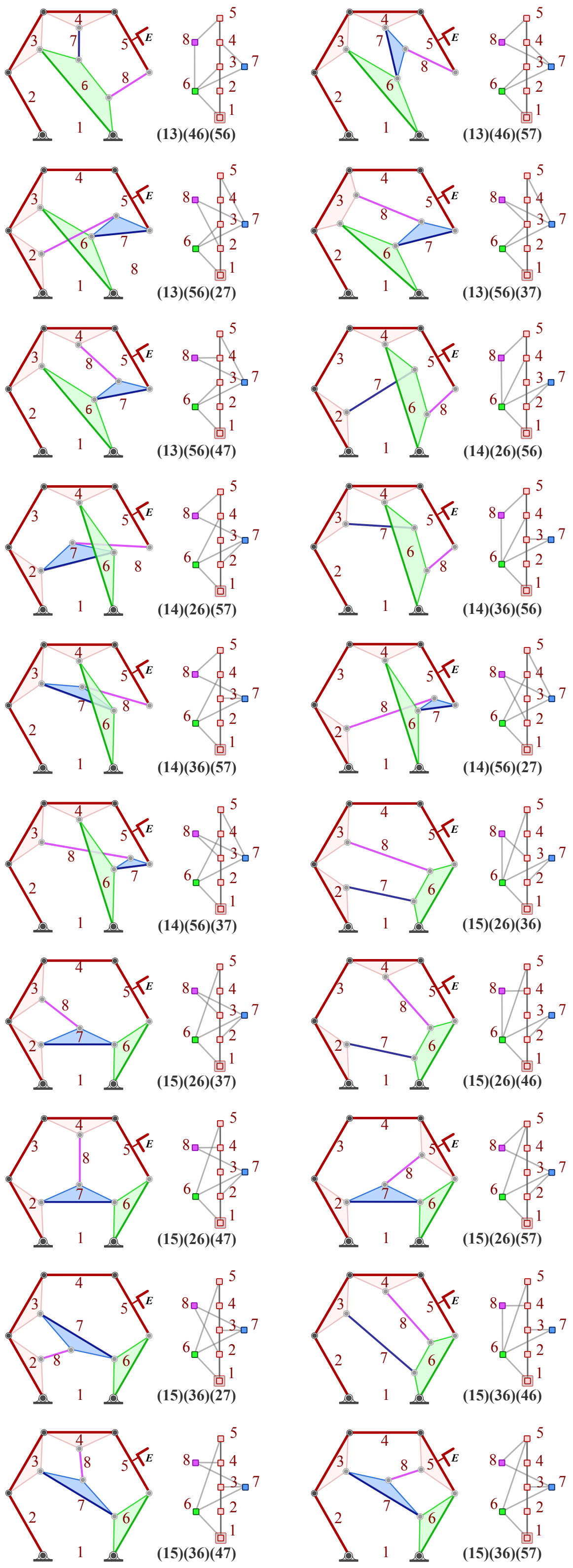


Figure 5.18: Showing $1-20$ eight-bar linkage graphs $(ij)(kl)(mn)$ out of 40 which exhibit second level dependency, with $i,j,k,m \in \{1,2,3,4,5\}$, $l \in \{6\}$, $n \in \{6,7\}$.

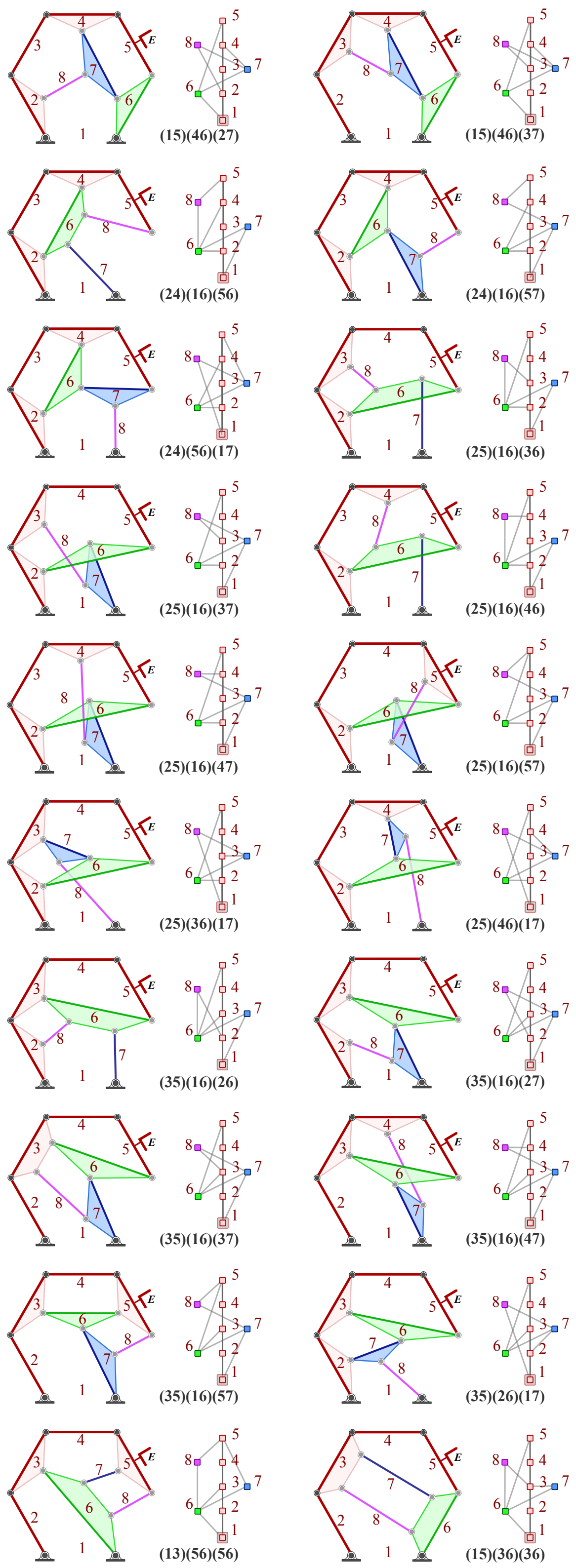


Figure 5.19: Showing $21-40$ eight-bar linkage graphs $(ij)(kl)(mn)$ out of 40 which exhibit second level dependency, with $i, j, k, m \in \{1, 2, 3, 4, 5\}$, $l \in \{6\}$, $n \in \{6, 7\}$.

Chapter 6

Applications

In this section a few additional examples are presented that demonstrates the effectiveness of six-bar and eight-bar linkages to accomplish complex motion. For the six-bar linkage, two folding linkages are presented that can rotate a body with respect to other by 90° and 180° respectively, while performing a complex motion in between the initial and final positions, which a simple hinge cannot accomplish. The application for these linkages could be in folding structures for emergency shelters. For the eight-bar linkage obtained by constraining a 6R loop, a convertible sofa-bed design problem is explored. By specifying the 6R loop, the designer has the freedom to select the location for 6 pivots out of 10 for the synthesized eight-bar. This helps in greater control on the overall size of the linkage. This is very important in a consumer product like sofa-bed, where it can be dangerous to have links projecting outside the body of the sofa frame. For the eight-bar linkage obtained by constraining the 4R chain, a port site closure device is explored. Efforts were made to make the suturing procedure for closing incisions made during Laproscopic surgery convenient, fast and cheap. Linkage solutions with no links overlapping were searched as they could be good candidates for replacement of hinge joints with flexure joints to simplify design and reduce manufacturing cost, see Howell(2001) [53].

6.1 Six-bar Folding Linkage

In this section, a folding structure application is discussed to demonstrate the design system for six-bar linkage. Such a linkage could be used in folding/unfolding structures for emergency shelters. Two types of six-bar folding linkages are explored - one which folds/unfolds stacked panels to 90° and another that folds/unfolds stacked panels to 180°.

Table 6.1: Five task positions for 90° folding linkage

Task	Orientation (θ) (degrees)	Location $\mathbf{d}$: (x, y)
1	0.00°	(−1855.60, 434.85)
2	18.98°	(−1965.60, 499.85)
3	69.51°	(−2284.26, 104.92)
4	80.61°	(−2121.39, −630.86)
5	90°	(−1855.60, −1065.15)

Table 6.2: Selected 3R open chain for 90° folding linkage

Pivot	Location (x, y)
C_1	$(844.40, 634.85)$
C_2	$(1977.63, 2881.16)$
C_3	$(-1513.58, 1374.54)$

6.1.1 90 Degree Folding Linkage

In this example for six-bar linkage, the task positions are given in Tab. 6.1. The 3R serial chain specified by the designer such that the end effector goes though the five positions is given in Tab. 6.2. The selected six-bar solution, mentioned in Tab. 6.3, is shown moving through the five positions as shown in Fig.6.1.

Table 6.3: Selected six-bar linkage solution for 90° folding structure

Pivot	Location (x, y)
C_1	$(844.40, 634.85)$
C_2	$(1977.63, 2881.16)$
C_3	$(-1513.58, 1374.54)$
C_4	$(-882.72, 1162.31)$
C_5	$(-199.75, 2455.55)$
C_6	$(-1795.59, 2389.62)$
C_7	$(-726.07, 3339.88)$

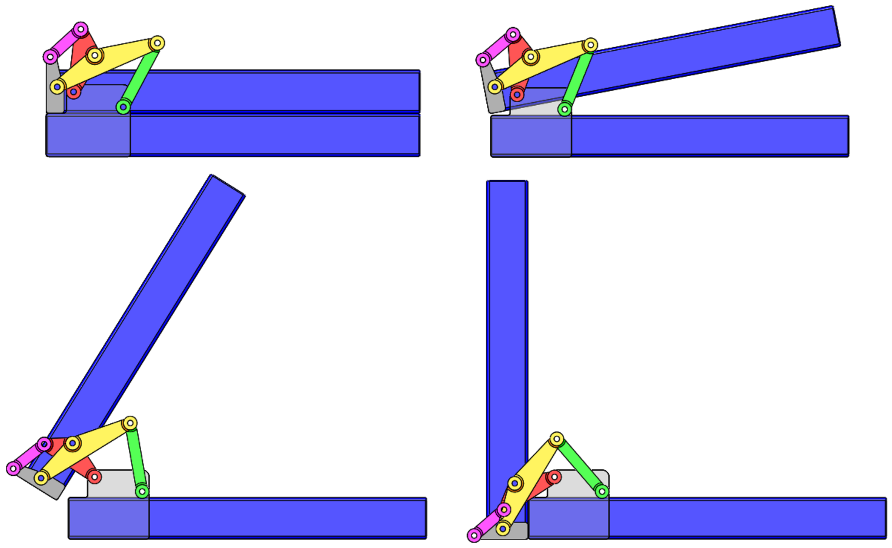


Figure 6.1: Selected six-bar linkage solution for 90° folding structure is shown moving through a few positions.

Table 6.4: Five task positions for 180° folding linkage

Task	Orientation (θ) (degrees)	Location $\mathbf{d}$: (x, y)
1	0.00°	$(0.00,\ 1500.00)$
2	20.60°	$(484.76,\ 2176.91)$
3	65.28°	$(-168.05,\ 2652.91)$
4	117.97°	$(-675.39,\ 2093.02)$
5	180°	$(-40.00,\ 1500.00)$

Table 6.5: Selected 3R open chain for 180° folding linkage

Pivot	Location (x, y)
C_1	$(3512.92, 1741.23)$
C_2	$(2776.73, 5783.97)$
C_3	$(39.19, 4072.91)$

Table 6.6: Selected six-bar linkage solution for 180° folding structure

Pivot	Location (x, y)
C_1	$(3512.92, 1741.23)$
C_2	$(2776.73, 5783.97)$
C_3	$(39.19, 4072.91)$
C_4	$(94.64, 1829.59)$
C_5	$(1105.27, 3674.40)$
C_6	$(-1420.31, 3880.44)$
C_7	$(125.31, 3138.82)$

6.1.2 180 Degree Folding Linkage

In this example for six-bar linkage, the task positions are given in Tab. 6.4. The 3R serial chain specified by the designer such that the end effector goes though the five positions is given in Tab.6.5. The selected six-bar solution, given in Tab.6.6, is shown moving through the five positions as shown in Fig.6.2.

These two six-bar linkages for 90° and 180° motion respectively, serve as building blocks. Different combinations for these building blocks can be used to achieve complex shaped structures. An example of a folding table, using these building blocks, is shown in Fig.6.3.

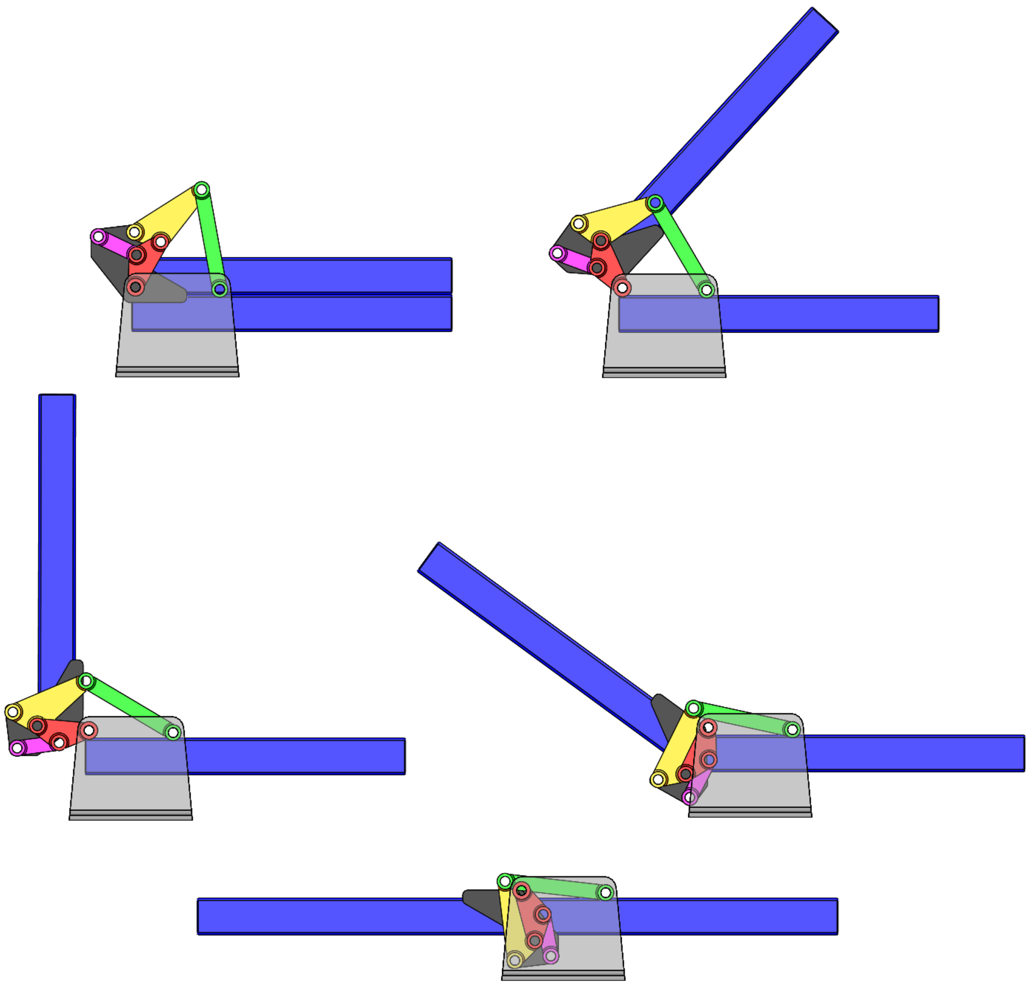


Figure 6.2: Selected six-bar linkage solution for 180° folding structure is shown moving through a few positions

Table 6.7: Five task positions for the convertible sofa-bed problem

Task	Orientation (θ) (degrees)	Location $\mathbf{d}$: (x, y)
1	0.00°	(−4.00, 6.10)
2	−49.00°	(−5.40, 18.50)
3	−53.00°	(−14.00, 24.00)
4	−42.00°	(−20.00, 22.402)
5	0.00°	(−28.60, 13.60)

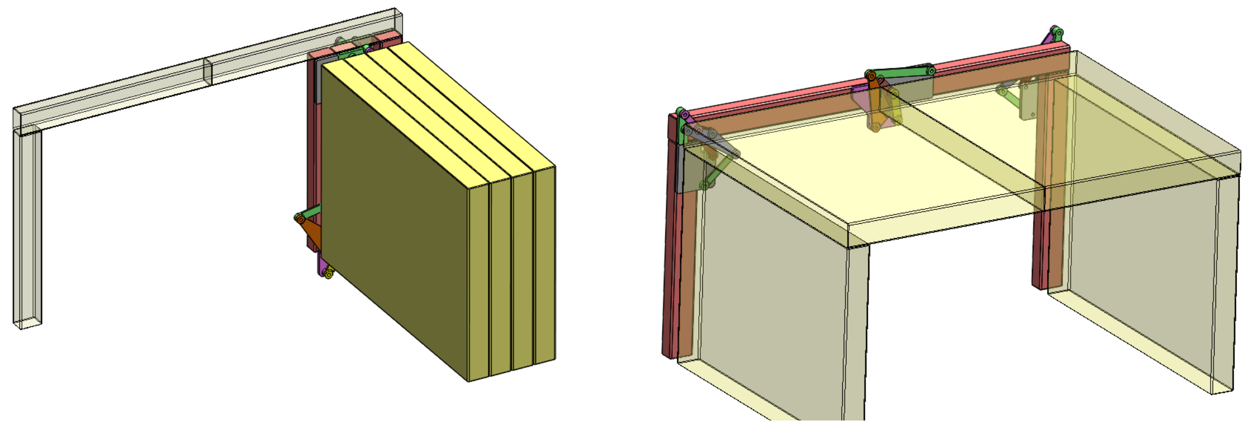


Figure 6.3: Folding table structure designed using a combination of 90° and 180° folding linkages, is shown in the initial stacked configuration and final deployed configuration

Table 6.8: Selected 6R loop coordinates for the sofa-bed problem

Pivot	Location(x, y)
C_1	$(-6.00,\ 0.00)$
C_2	$(8.20,\ 8.00)$
C_3	$(-8.00,\ 6.10)$
C_4	$(-4.00,\ 6.10)$
C_5	$(11.50,\ 15.30)$
C_6	$(2.00,\ 0.00)$

Table 6.9: Five tolerances on the task positions for the sofa-bed problem

Task	Tolerance Data $(\Delta\theta, \Delta x, \Delta y)$
1	$(0°, 1.0, 1.0)$
2	$(5°, 2.0, 2.0)$
3	$(5°, 2.0, 2.0)$
4	$(5°, 2.0, 2.0)$
5	$(0°, 1.0, 1.0)$

6.2 Eight-bar Application: Convertible Sofa-Bed Linkage

In this example the design problem considered is that of a convertible sofa-bed linkage. This example was previously seen in Soh and McCarthy (2007) [11]. The five task positions for

Table 6.10: Linkage solutions for the convertible sofa-bed problem

No. of Iterations	No. of Linkages Synthesized	No. of Useful Linkages	No. of Useful Linkages satisfying Criteria	Time (min)	Notes S: (Serial) P: (Parallel) (Cores/Threads)
1	61	2	1	9.29	$S, Intel$ (4/8)
1	61	2	1	2.75	$P, Intel$ (4/8)
1	61	2	1	2.32	P, AMD (6)
10	559	3	2	19.231	P, AMD (6)
1	61	2	1	1.035	$P, Intel$ (12/24)
10	469	4	2	6.949	$P, Intel$ (12/24)
100	4191	9	4	62.023	$P, Intel$ (12/24)

Table 6.11: Selected eight-bar linkage solution for the convertible sofa-bed problem

Pivot	Location Data (x, y)
C_1	(−6.00, 0.00)
C_2	(8.20, 8.00)
C_3	(−8.00, 6.10)
C_4	(−4.00, 6.10)
C_5	(11.50, 15.30)
C_6	(2.00, 0.00)
C_7	(11.78, 11.19)
C_8	(7.80, 10.48)
C_9	(8.42, 8.56)
C_{10}	(12.62, 12.72)

this problem are listed in Tab.6.7 and the user specified 6R closed chain information is given in Tab.6.8. The tolerance zone for each task position is a list $(\Delta\theta, \Delta x, \Delta y)$, where $\Delta\theta$ is the tolerance on the orientation of the task position and Δx and Δy are the tolerances on the x and y coordinates of the origin for the task position. These are mentioned in Tab. 6.9. During the randomized task position run, a list of numbers will be randomly picked from the range $(\pm\Delta\theta, \pm\Delta x, \pm\Delta y)$ and added to the task position data (θ, x, y). The elbow specification is elbow up, represented by number 1.

This input data was sent to the synthesis routine to add the two RR constraints using the

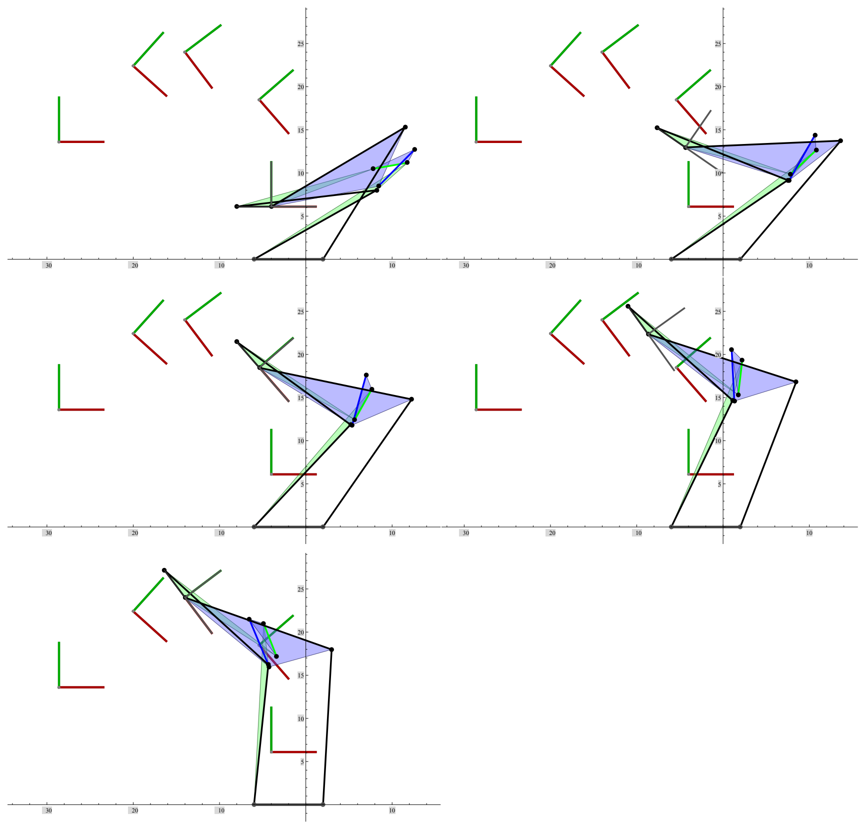


Figure 6.4: Selected eight-bar linkage for convertible sofa-bed is shown moving through the first three task positions along with some intermediate positions

strategies of adding them independently or in sequence, which could be done in 32 different ways. Note that for a single iteration, the program runs only on the original task positions. For multi-iteration run, refer Tab.6.10, the first iteration always runs on the original task positions and successive iterations use randomized task positions, within the tolerance zones. For this example, a total of 61 candidate linkages were synthesized for the original task positions.

These candidate linkages were analyzed and their configuration trajectories were sorted into branches. Figure 5.10 shows a candidate linkage with all the five task position configurations lying on a single branch (branch1). This ensures that the linkage is defect free and will smoothly move through the five task positions. Figure 6.4 and Figure 6.5 show the linkage movement through the five task positions along with few intermediate positions.

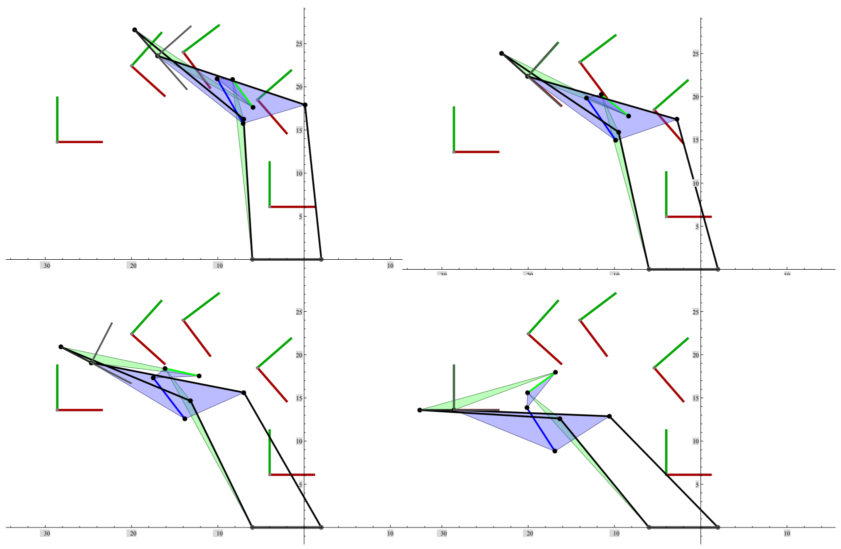


Figure 6.5: Selected eight-bar linkage for convertible sofa-bed is shown moving through the fourth and fifth task positions along with an intermediate position

For a single iteration run, only two linkages were found to be defect-free. In addition to the input data, the user may also add the permissible values for the minimum and maximum allowable link length for the linkage. For the current example, the metric for comparison was derived from the side length of a square fitted around the 5 task positions. This eliminated one solution, thus leaving only one useful linkage. The useful linkage had the RR connections between link pairs $(1, 3)$ and $(4, 6)$, topology. The linkage solution is mentioned in Table 6.11.

The statistics for this example problem, with the design algorithm running on different machines with several iterations, is shown in Tab. 6.10. The program was run in serial and parallel, and substantial speed improvement was reported using Mathematica's automatic parallel computation. The first two rows of the Tab.6.10 show that for a 4-core Intel machine, the time went down from 9.29 min for serial computation to 2.75 min for parallel computation, for one iteration. The program was tested on a 12 core Intel machine for $1, 10$ and 100 iterations and the speed increase was even greater. The next step towards speed improvement is by GPU computing. Mathematica provides an interface called CUDA*Link* to program the GPU efficiently.

Table 6.12: Five Task Positions for the port site closure problem

Task	Orientation (θ) (degrees)	Location $\mathbf{d}$: (x, y)
1	5°	(5.1, −51.5)
2	35°	(19.1, −36.02)
3	75°	(21.0, −21.2)
4	105°	(16.5, −9.8)
5	123°	(7.8, 3.5)

Table 6.13: 4R Chain Data for the port site closure problem

Ground Pivot (C_1)	$(-9.26, 1.35)$
End-effector Pivot (C_4)	$(3.1, -51.7)$
Common length (a)	24
Elbow Position	up (-1)

6.3 Eight-bar Application: Port Site Closure Linkage

Port sites in Laproscopic surgery are the incisions made for inserting surgical tools in to the body. Closing the port sites is an final step in Laproscopic surgery. In a typical port site closing procedure, two incisions are made diametrically opposite to the port site with the help of a suture passer. The suture passer carries the suture with it when the surgeon is making the first incision. The suture passer is then pulled out leaving one end of the suture end inside the peritoneum. A second incision is made diametrically opposite to the first incision around the port site by the suture passer. The gripper on the suture passer is then employed to grabs on the loose end of the suture inside the peritoneum and is pulled out with the suture passer. The two ends of the suture are then tied by the surgeon to complete the closure. A novel suture passer is envisioned here that curves as it makes the first incision and bends, so that a second incision could be made and the suture end can be retrieved from the second incision. The five task positions selected for this example problem are given in Tab. 6.12 and the user specified 4R chain information is given in Tab. 6.13. The tolerances

Table 6.14: Five tolerances on task positions for the port site closure problem

Task	Tolerance Data$(\Delta\theta, \Delta x, \Delta y)$
1	$(5°, 2, 2)$
2	$(5°, 2, 2)$
3	$(5°, 2, 2)$
4	$(5°, 2, 2)$
5	$(5°, 2, 2)$

Table 6.15: Multi-iteration results for the eight-bar design algorithm for the port site closure problem

No. of Iterations	No. of linkages Synthesized	No. of defect-free Linkages	No. of useful Linkages	Time Taken
1	902	7	2	9.9 min
10	9605	1102	108	141.94 min
100	110454	7885	853	43 hr 52 min

are mentioned in Tab. 6.14. The elbow specification is elbow up, denoted by integer -1.

For this example problem, the number of linkage solutions synthesized by running the algorithm on the original task positions, were 902 (max possible 3951). These solutions were analyzed for branch defects and the number of defect-free linkages were found to be 7. A constraint was imposed on the longest link to be no greater than 48 units and the shortest link to be no shorter than 1 unit. This requirement reduced the number of useful solutions to 2. For multi-iteration run, the results are shown in Tab. 6.15.

A sample example eight-bar solution is mentioned in Tab. 6.16, as coordinates of the various joints $(C_1, \ldots, C_{10})$. The three RR connections are applied between the link pairs $(1, 4)$, $(2, 5)$ and $(3, 6)$ respectively. This linkage has four binary and four ternary links and has the topology shown in Fig.6.6. Figure 5.10 shows the configuration trajectories for the link angles $\theta_3, \ldots, \theta_8$ plotted against the input link angle θ_2. Each branch trajectory is denoted by a different color. It can be observed that the linkage has all the five task configurations

Table 6.16: Useful Eight-bar linkage solutions for the port site closure problem

Pivot	Location (x, y)
C_1	$(-9.26, 1.35)$
C_2	$(-22.67, -18.55)$
C_3	$(-18.97, -42.26)$
C_4	$(3.1, -51.7)$
C_5	$(-12.57, 6.85)$
C_6	$(-18.74, -31.32)$
C_7	$(-21.87, -27.86)$
C_8	$(4.43, -51.97)$
C_9	$(-20.11, -22.59)$
C_{10}	$(-24.49, -23.54)$

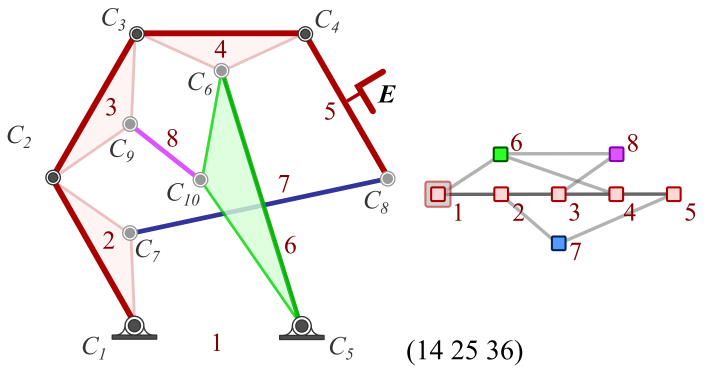


Figure 6.6: Topology of the example eight-bar linkage for port closure is shown along with its graph. The three RR constraints are applied across link pairs $\{(1, 4), (2, 5), (3, 6)\}$

lying on the same branch 1. This ensures that, when the input link is rotated in the prescribed range and the linkage is moving in this branch, it will hit each of the five task positions. This verifies that the linkage is defect free. Figures 6.7 and 6.8 show the linkage movement through the five task positions along with a few intermediate positions. A SolidWorks rendering of the linkage is shown in Fig.6.9.

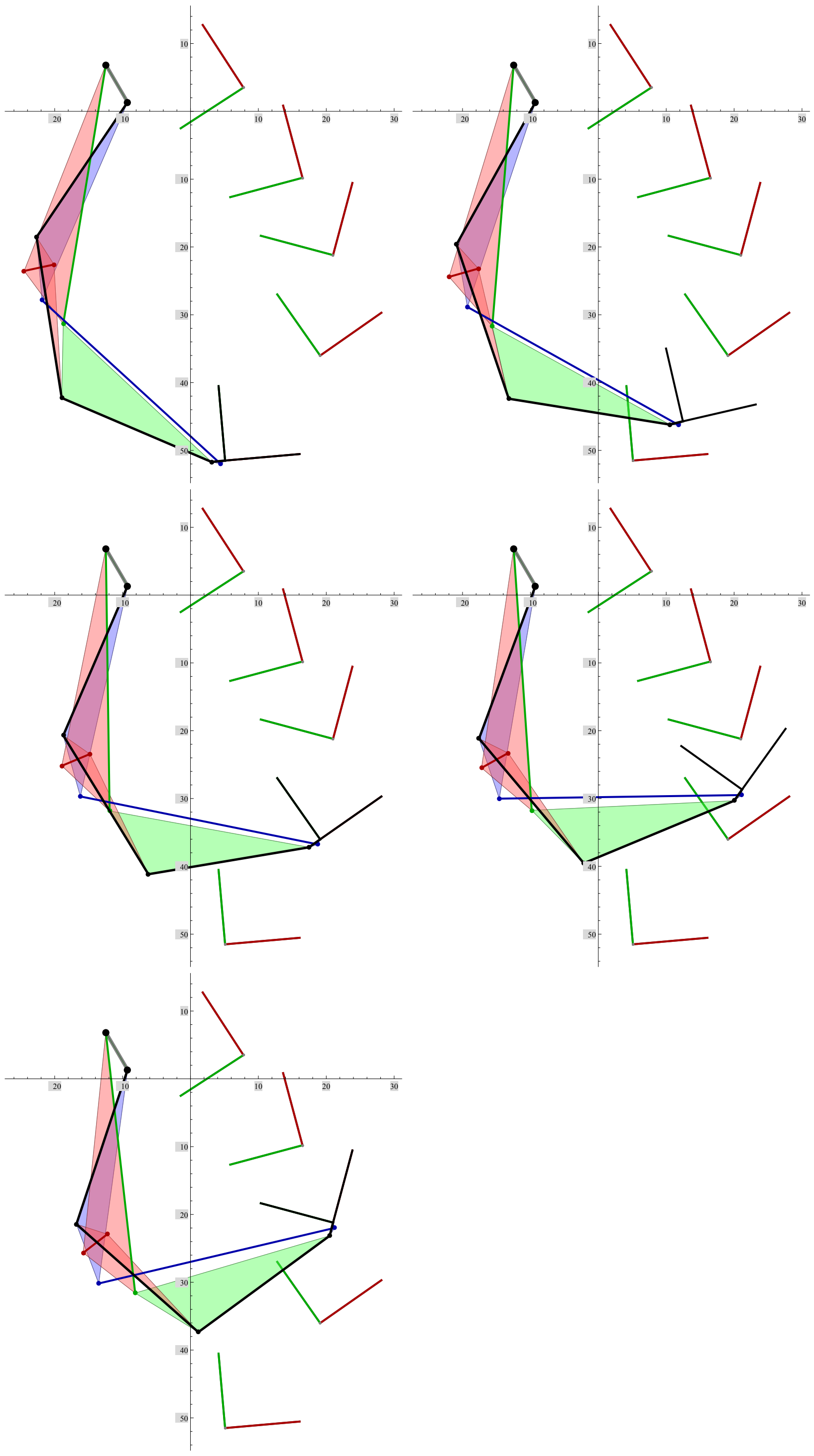


Figure 6.7: Selected eight-bar linkage for port closure is shown moving through the first three task positions along with some intermediate positions

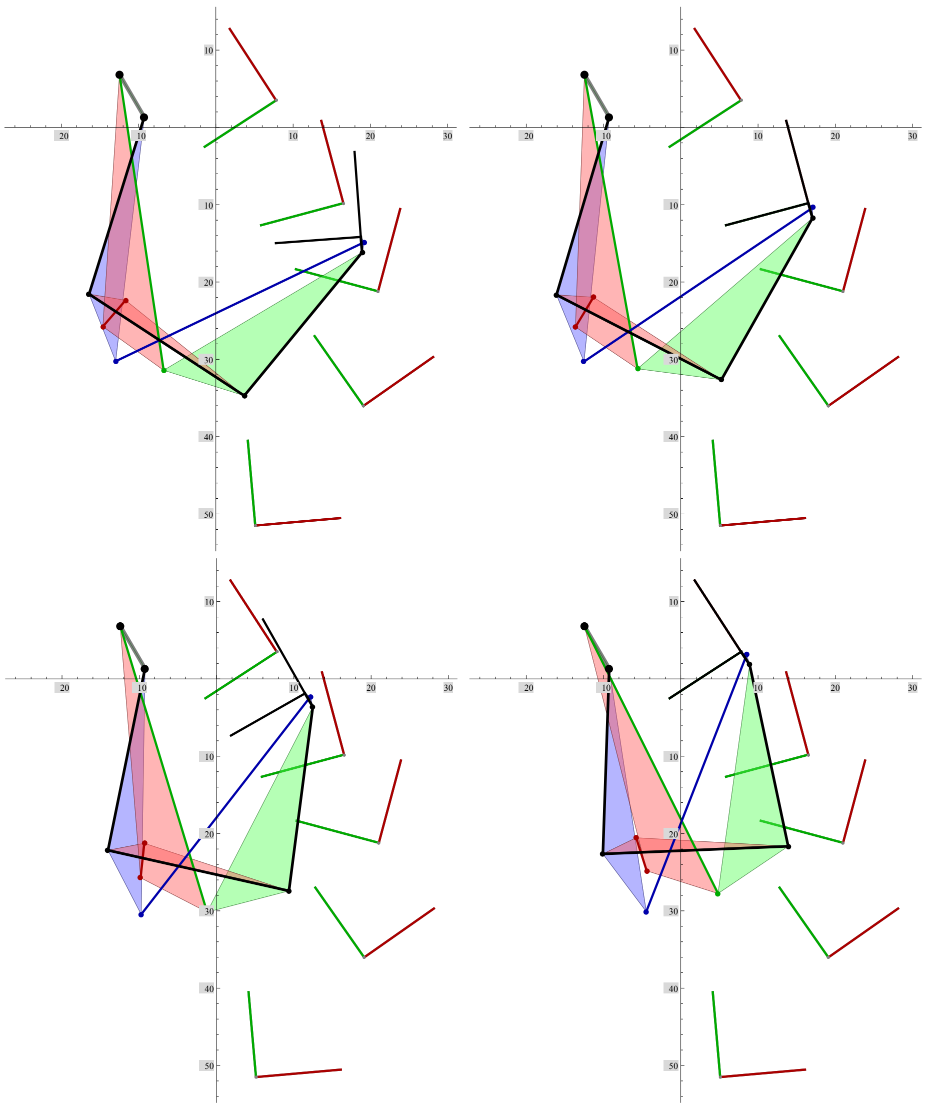


Figure 6.8: Selected eight-bar linkage for port closure is shown moving through the fourth and fifth task positions along with an intermediate position

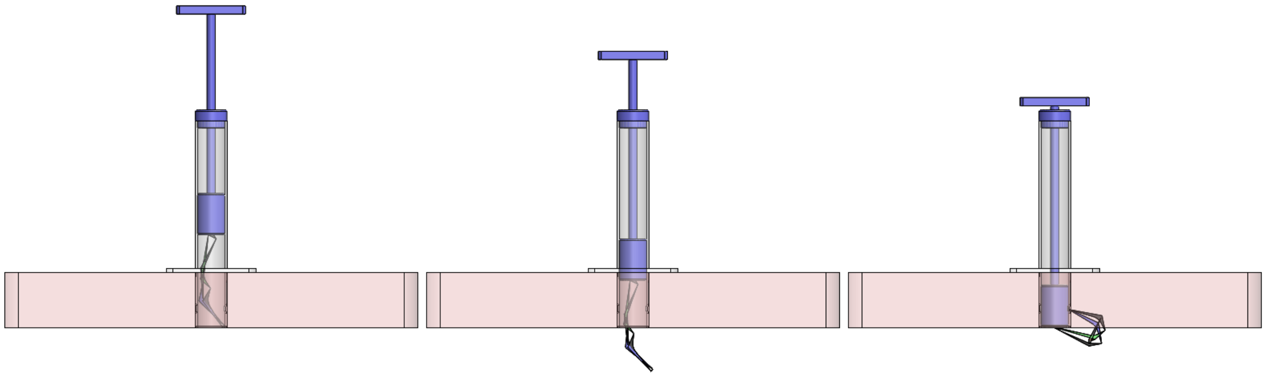


Figure 6.9: SolidWorks rendering of the eight-bar port closure linkage

Chapter 7

Conclusion and Future Work

This dissertation research provides the first systematic design procedure for automated synthesis of eight-bar linkages. The input to the design procedure is a set of five task positions and either the 4R serial chain or the 6R closed chain robot, which can guide the end-effector through the five task positions. The design system uses the graph theory approach to systematically find all the possible ways of constraining this robot using RR constraints, to generate single degree of freedom eight-bar linkages. It has been observed that this method of constraining an open or closed chain allows the user greater control over the synthesized linkage.

In case of the 4R serial chain, which is a 4 DOF robot, three RR constraints are required to constrain it to a single degree of freedom eight-bar linkage. There are in all 100 different ways to apply the three RR constraints to synthesize a maximum of 3951 eight-bar linkage solutions, using various combinations of the RR constraints. It is observed that out of 16 topologies possible for eight-bars, 15 could be synthesized by constraining the 4R chain.

In case of the 6R closed chain, which is a 3 DOF parallel robot, two RR constraints are required to constrain it to a single DOF eight-bar linkage. There are in all 32 different ways to apply the two RR constraints to synthesize a maximum of 340 linkage solutions. It is observed that out of 16 topologies possible for eight-bars, 8 could be synthesized by constraining the 6R closed chain.

A similar process can be employed to synthesize six-bar linkages by constraining a 3 DOF, 3R serial chain using two RR constraints. There are 6 ways to add the two RR constraints to yield a maximum of 63 planar six-bar linkages and 165 spherical six-bar linkages.

The design system performs both type and dimensional synthesis and is topologically independent. This method of constraining can be applied to any serial or closed chain to generate multi-bar linkages for five task positions. A future extension of this design system is its implementation as a rule based expert system, that will determine the correct number of links in an linkage, to produce the required movement.

It is important to note that, each application of an RR constraint reduces the DOF of the serial or closed chain by one. Therefore, if the designer requires a more flexible workspace, he or she may opt for a reduced DOF robot instead of single DOF linkage, at the expense of additional actuators.

Following the synthesis, the candidate linkages are analyzed for performance verification. Dixon determinant elimination is used to find all the forward kinematic solutions for each of the synthesized linkages. For each linkage, these solutions are sorted in to branches representing different linkage configurations (assemblies). A linkage is deemed defect-free if it reaches all the task positions in one configuration. This ensures smooth movement of the end-effector through the five task positions. Defect-free linkages that also satisfy the minimum and maximum allowable link length requirement, specified by the user, are termed as successful linkages.

Once the original task positions are explored for successful linkage solutions, the design algorithm is ran iteratively on randomized task positions within acceptable variations, specified by the designer, to obtain large number of solutions. In order to make this process efficient, the design algorithm runs the various iterations in parallel using Mathematica's auto parallel computation feature. Several examples for six-bar and eight-bar linkages are presented to demonstrate the effectiveness of the design system.

The proposed design system has been successfully integrated in a computer aided linkage design package called MechGen, for designing four-bar and six-bar, planar and spherical six-bar linkages. MechGen works as an add-in for SolidWorks. The user specifies the five task positions and the backbone chain robot in the sketch environment of SolidWorks. MechGen reads the sketch and generates linkage solutions as animations in its user interface. For the selected linkage MechGen generates parts and assembly files in SolidWorks.

With regards to future work, MechGen could be extended to support eight-bar linkages. Efforts could be made to handle task positions greater than five. Currently only five task positions are used because an RR constraint can be synthesized for a maximum of five positions. Different methods like loop closure could be explored to generate synthesis equations for multi-bar linkages to handle more than five task positions. The challenge associated with these methods would be the computation time. Problems like these require the use of polynomial homotopy solvers, see Wampler(2005) [49], which can take days to calculate solutions.

Synthesis of 10-bars and 12-bars can be explored that can literally yield thousands of new inventions for very specific design goals. The increase in the number of links provides greater capability for the designer to shape or rather package the overall motion of the linkage and can lead to designs with less relative motion between the various links. This is a significant benefit that offsets the increase in complexity. Overall, this is a very interesting research pursuit and will effectively contribute to computer aided innovation in mechanical design.

Bibliography

[1] Wunderlich, W., "Hahere Koppelkurven.", *Osterreichisches Ingenieur Archiv*, XVll(3), p.162-165, 1963.

[2] Primrose, E. J., Freudenstein, F., and Roth, B., "Six-Bar Motion III. Extension of the Six-Bar Techniques to Eight-Bar and 2n-Bar Mechanisms.", *Archive for Rational Mechanics and Analysis*, 24.1, p.73-77, 1967.

[3] Olson, D. G., Erdman, A. G., and Riley, D. R., "A systematic procedure for type synthesis of mechanisms with literature review literaturbe-sprechung.", *Mechanism and Machine Theory*, 20(4), 285-295, 1985.

[4] Mueller, J., Dissertation: "Design Procedures for the Determination of Dimensions of Eight-bar and Ten-bar Linkages.", Technische Universitat Dresden, Germany, 1954.

[5] Hain, K., "The Simultaneous Production of Two Rectilinear Translations by Means of Eight-link Mechanisms.", *Journal of Mechanisms*, Vol. 2, No. 2, pp. 185-191, 1967.

[6] Hamid, S., and Soni A. H., "Synthesis of an Eight-Link Mechanism for Varieties of Motion Programs.", *Journal of Engineering for Industry*, 95, no. 3, pp. 744-750, 1973.

[7] Soni, A. H., "Coupler Cognates of Eight-Link Mechanisms With Ternary and Quaternary Links - Part 1.", *Journal of Manufacturing Science and Engineering*, 93, no. 1, 294-298, 1971.

[8] Pennock, G. R., and Raje, N. N., "Coupler cognates for the double flier eight-bar linkage.", *Journal of Mechanical Design*, 127(6), 1145-1151, 2005.

[9] Chen, C., Angeles, J., and Univerisity, M., "Kinematic synthesis of an eight-bar linkage to visit eleven poses exactly.", *In Proc. CDEN/C2E2 2007 Conference*, 2007.

[10] Chen, C., and Angeles, J., "A novel family of linkages for advanced motion synthesis.", *Mechanism and Machine Theory*, 43(7), 882-890, 2008.

[11] Soh, G. S., and McCarthy, J. M., "Synthesis of Eight-Bar Linkages as Mechanically Constrained Parallel Robots.", *12th IFToMM world congress A*, Vol. 653, 2007.

[12] McCarthy, J. M., and Soh, G. S., *Geometric Design of Linkages. 2nd Ed.*, Springer-Verlag, 2010.

[13] Soh, G. S., and Ying, F., "Dimensional Synthesis of Planar Eight-Bar Linkages based on a Parallel Robot with a Prismatic Base Joint.", *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Paper No. DETC2013-12799, August 4-7, 2013, Portland, Oregon, USA.

[14] Sonawale, K. H., Arredondo, A., and McCarthy, J. M., "Computer Aided Design of Useful Spherical Watt I Six-bar Linkages.", *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DET2013-13454, August 4-7, 2013, Portland, Oregon USA.

[15] Sonawale, K. H., and McCarthy, J. M., "Synthesis of Useful Eight-bar Linkages as Constrained 6R loops.", *Proceedings of the ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DET2014-35523, August 17-20, 2014, Buffalo, New York, USA.

[16] Sonawale, K. H., and McCarthy, J. M., "Design of Eight-bar Linkages with an Application to Rectilinear Motion.", *Proceedings of the ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DET2015-47804, August 3-5, 2015, Boston Massachusetts, USA.

[17] Sonawale, K. H., and McCarthy, J. M., "Synthesis of Eight-bar Linkages by Constraining a 6R Loop.", *Mechanism and Machine Theory*, 2015.

[18] Sonawale, K. H., and McCarthy, J. M., "A Design System for Eight-bar Linkages as Constrained 4R Serial Chains.", *Journal of Mechanisms and Robotics*, 2015.

[19] Soh, G. S., Thesis: "Kinematic Synthesis of Six and Eight-Bar Planar Linkages.", University of California, Irvine, CA, USA, 2006.

[20] Burmester, L., *Lehrbuch der Kinematik, Vol. 1, Die ebene Bewegung*, Leipzig, 1888.

[21] Sandor, G. N., and Erdman, A. G., *Advanced Mechanism Design: Analysis and Synthesis, Vol. 2.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[22] Koetsier, T., "The centenary of Ludwig Burmeister's Lehrbuch der Kinematik.", *Mechanism and Machine Theory*, 1:37-38, 1989.

[23] Nielsen, J., and Roth, B., "Solving the input/output problem for planar mechanisms.", *Journal of Mechanical Design*, 121, 206, 1999.

[24] Wampler C. W., "Solving the Kinematics of Planar Mechanisms by Dixon Determinant and a Complex Plane Formulation.", *ASME Journal of Mechanical Design*, 123(3), pp. 382-387, 2001.

[25] Dixon, A. L. "The eliminant of three quantics in two independent variables.", *Proceedings of the London Mathematical Society*, 2.1, 49-69, 1909.

[26] Dixon, A. L., "The Eliminant of Three Quantics in two Independent Variables:(Second Paper).", *Proceedings of the London Mathematical Society*, 2.1, 473-492, 1909.

[27] Dhingra A. K., Almadi A. N.,l and D. Kohli D., "Closed-form displacement analysis of 8, 9, and 10-link mechanisms, Part I: 8-link 1-DOF mechanisms.", *Mechanism and Machine Theory*, 35:821-850, 2000.

[28] Parrish, B. E., McCarthy, J. M., and Eppstein, D., "Automated Generation of Linkage Loop Equations for Planar 1-DOF Linkages, Demonstrated up to 8-bar.", *Proceedings of the ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Paper No. DETC2014-35263, August 17-20, 2014, Buffalo, New York, USA.

[29] Parrish, B. E, McCarthy, J. M., and Eppstein, D., "Automated Generation of Linkage Loop Equations for Planar 1-DoF Linkages, Demonstrated up to 8-bar," *ASME Journal of Mechanisms and Robotics*, Vol. 6, December 2014.

[30] Parrish, B. E., Dissertation: "Automated Configuration Analysis of Planar Eight-Bar Linkages", University of California, Irvine, CA, USA, 2014.

[31] Tsai L. W., *Mechanism Design: Enumeration of Kinematic Structures According to Function*, CRC Press, 2000.

[32] F. Freudenstein and G. N. Sandor, "Synthesis of Path Generating Mechanisms by Means of a Programmed Digital Computer," *ASME Journal of Engineering for Industry*, 81:159-168, 1959.

[33] Kaufman, R. E., and Maurer, W. G., "Interactive Linkage Synthesis on a Small Computer.", *ACM National Conference*, Aug.3-5, 1971.

[34] Rubel, A. J., and Kaufman, R. E., "KINSYN III: A New Human-Engineered System for Interactive Computer-aided Design of Planar Linkages.", *ASME Transactions, Journal of Engineering for Industry*, May, 1977.

[35] Erdman, A. G., and Gustafson, J., "LINCAGES-A Linkage Interactive Computer Analysis and Graphically Enhanced Synthesis Package.", ASME Paper No. 77-DTC-5, Chicago, Illinois, 1977.

[36] Hunt, L., Erdman, A. G., and Riley, D. R., "MicroLINCAGES: Microcomputer Synthesis and Analysis of Planar Linkages.", *Proceedings of the Seventh OSU Applied Mechanisms Conference*, Dec, 1981.

[37] Chuang, J. C., Strong, R. T., and Waldron. K. J., "Implementation of Solution Rectification Techniques in an Interactive Linkage Synthesis Program.", *ASME Journal of Mechanical Design*, 103:657-664.

[38] Ruth, D. A., and McCarthy, J. M., "SphinxPC: An Implementation of Four Position Synthesis for Planar and Spherical Linkages.", *Proceedings of the ASME Design Engineering Technical Conferences*, Sacramento, CA, Sept. 14-17, 1997.

[39] Furlong, T. J., Vance, J. M., and Larochelle, P. M., "Spherical Mechanism Synthesis in Virtual Reality.", *ASME Journal of Mechanical Design*, 121:515, 1999.

[40] Collins, C. L., McCarthy, J. M., Perez, A., and Su, H., "The Structure of an Extensible Java Applet for Spatial Linkage Synthesis.", *ASME Journal of Computing and Information Science and Engineering*, 2(1):45-49, March, 2002.

[41] Perez, A., Su, H., and McCarthy, J. M., "Synthetica 2.0: Software for the Synthesis of Constrained Serial Chains.", *Proceedings of the ASME 2004 Design Engineering Technical Conferences*, Sept. 28-Oct. 2, 2004, Salt Lake City, Utah.

[42] Kinzel, E. C., Schmiedeler, J. P., and Pennock, G. R., "Kinematic Synthesis for Finitely Separated Positions Using Geometric Constraint Programming.", *ASME Journal of Mechanical Design*, Vol 128:1070-1079, 2006.

[43] Kinzel, E. C., Schmiedeler, J. P., and Pennock, G. R., "Function Generation With Finitely Separated Precision Points Using Geometric Constraint Programming.", *Journal of Mechanical Design*, Vol 129:1185-1190, 2007.

[44] Soni, A. H., Dado, M. H., and Weng, Y., "An automated procedure for intelligent mechanism selection and dimensional synthesis.", *Journal of Mechanical Design*, 110(2), 130-137, 1988.

[45] Hansen, M. R., "An automated procedure for dimensional synthesis of mechanisms.", *Structural optimization*, 5(3), 145-151, 1993.

[46] Liu, Y., and McPhee, J., "Automated Kinematic Synthesis of Planar Mechanisms with Revolute Joints.", *Mechanics Based Design of Structures and Machines*, 35(4), 405-445, 2007.

[47] McCarthy, J. M., and Choe, J., "Difficulty of Kinematic Synthesis of Usable Constrained Planar 6R Robots," *Advances in Robot Kinematics*, 12th International Symposium, June 28 - July 1, 2010, Portoroz, Slovenia.

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

[49] Sommese, A. J., and Wampler, C. W. "The Numerical solution of systems of polynomials arising in engineering and science", (Vol. 99), Singapore: World Scientific, 2005.

[50] Wampler, C. W., "Isotropic coordinates, circularity and Bezout numbers: planar kinematics from a new perspective.", *Proceedings of the 1996 ASME Design Engineering Technical Conference, Irvine, California August, 18-22*, 1966

[51] Plecnik, M. M. and McCarthy, J. M., "Design of a 5-SS Spatial Steering Linkage," DETC 2012-71405, *Proceedings of the ASME 2012 International Design Engineering*

Technical Conferences and Computers and Information in Engineering Conferences, August 12-15, 2012, Chicago, IL, USA.

[52] Plecnik, M. M. and McCarthy, J. M., "Numerical Synthesis of Six-bar Linkages for Mechanical Computation.", *Journal of Mechanisms and Robotics*, 2013.

[53] Howell, L. L., "Compliant mechanisms.", John Wiley & Sons ,2001.

[54] Myszka, D. H., Murray, A. P., and Wampler, C. W., "Mechanism branches, turning curves, and critical points.", *In ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1513-1525, American Society of Mechanical Engineers, August, 2012.

[55] Roth, B., and Freudenstein, F., "Synthesis of path-generating mechanisms by numerical methods.", emphJournal of Manufacturing Science and Engineering, 85(3), 298-304, 1963

[56] Almadi, A. N., Dhingra, A. K., and Kohli, D., "A framework for closed-form displacement analysis of planar mechanisms.", *Journal of Mechanical Design*, 121(3), 392-401, 1999.

[57] Balli, S. S., and Chand, S., "Defects in link mechanisms and solution rectification.", *Mechanism and Machine Theory*, 37(9), 851-876, 2002.

[58] Filemon, E., "Useful ranges of centerpoint curves for design of crank-and-rocker linkages.", *Mechanism and Machine Theory*, 7(1), 47-53, 1972.

[59] Bawab, S., Kinzel, G. L., and Waldron, K. J., "Rectified synthesis of six-bar mechanisms with well-defined transmission angles for four-position motion generation.", *Journal of Mechanical Design*, 118(3), 377-383, 1996.

[60] Barker, C. R., "A complete classification of planar four-bar linkages.", *Mechanism and Machine Theory*, 20(6), 535-554, 1995.

[61] Beloiu, A. S., and Gupta, K. C., “A unified approach for the investigation of branch and circuit defects.”, *Mechanism and machine theory*, 32(5), 539-557, 1997.

[62] Chase, T. R., and Mirth, J. A., “Circuits and branches of single-degree-of-freedom planar linkages.”, *Journal of mechanical design*, 115(2), 223-230, 1993.

[63] Mirth, J. A., and Chase, T. R., “Circuit analysis of Watt chain six-bar mechanisms.”, *Journal of Mechanical Design*, 115(2), 214-222, 1990.

[64] Primrose, E. J. F., Freudenstein, F., and Roth, B., “Six-bar motion I.”. *The Watt mechanism.* Arc, 1967.

[65] Kempe, A. B. , *How to Draw a Straight Line; A Lecture on Linkages*, McMillan and Co., London, 1877.

[66] Murray, R. M., Li, Z., and Sastry, S. S., *A Mathematical Introduction to Robotics Manipulators*, CRC Press, 1994, 456 pp.

[67] Plecnik, M. M. and McCarthy, J. M., “Numerical Synthesis of Six-bar Linkages for Mechanical Computation,” *Journal of Mechanisms and Robotics*, 2013.

Appendices

A Mathematica code for Spherical Watt-I Six-bar Linkage Design

```
(*General Functions*)

npos = 5;
MakeUnitV[a_] := If[Norm[a] == 0, {0, 0, 0}, Chop[N[a/Norm[a]]]];
Make4by4[pos_] :=
  Append[Table[Append[pos[[i]], 0], {i, 3}], {0, 0, 0, 1}];
LinkLength[r_, s_] := N[Sqrt[Dot[s - r, s - r]]];
LinkAngle[a_, b_] := N[ArcCos[Dot[MakeUnitV[a], MakeUnitV[b]]]];
JointAngle[a_, b_, c_] :=
  If[Norm[a] == 0 || Norm[b] == 0 || Norm[c] == 0, 0,
   ArcTan[
    Dot[Cross[MakeUnitV[a], MakeUnitV[b]],
     Cross[MakeUnitV[a], MakeUnitV[c]]],
    Det[{MakeUnitV[a], Cross[MakeUnitV[a], MakeUnitV[b]],
      Cross[MakeUnitV[a], MakeUnitV[c]]}]]];
vectAngle[b_, c_] := ArcTan[Dot[b, c], Det[{b, c}]];
```

```
RotZ[t_] := {{Cos[t], -Sin[t], 0}, {Sin[t], Cos[t], 0}, {0, 0, 1}};
RotY[s_] := {{Cos[s], 0, Sin[s]}, {0, 1, 0}, {-Sin[s], 0, Cos[s]}};
RotX[u_] := {{1, 0, 0}, {0, Cos[u], -Sin[u]}, {0, Sin[u], Cos[u]}};
Zdisplace[
   a_] := {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, a}, {0, 0, 0, 1}};
rem = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}};
hom[v_] := {v[[1]], v[[2]], v[[3]], 1};
Disp[x_] := Chop[N[RotY[x[[1]]].RotX[-x[[2]]].RotZ[x[[3]]]]];

RandomVariables[RangeTheta_, RangePhi_, RangePsi_, FivePos_] :=
  Module[{Thetav, Phiv, Psiv, datanum, positionX, K},
   datanum = 5;
   Thetav = RandomReal[RangeTheta, datanum];
   Phiv = RandomReal[RangePhi, datanum];
   Psiv = RandomReal[RangePsi, datanum];
   (*Print["Tolerances on three angles ",{Thetav,Phiv,
   Psiv}*180/\[Pi]];*)
   K = FivePos;
   positionX =
    Table[K[[i]].RotY[Thetav[[i]]].RotX[-Phiv[[i]]].RotZ[
       Psiv[[i]]], {i, 5}];
   positionX];

Orthodrome[p1_, p2_, r_] := Module[{p, sol, t, pts, k},
   p = p1 + t (p2 - p1);
   sol = Flatten[Solve[k[t] Norm[p1 + t (p2 - p1)] == r, k[t]]][[1]];
   pts = DeleteCases[
     Quiet@Table[(k[t]*p) /. sol /. t -> t2, {t2,
        0, .995, .005}], {Indeterminate, 0, 0}];
   pts];

DisplayTP[position_, color_, color2_, color3_, colordot_, lengthco_,
   lengthex_, lengthey_] := Module[{orig, ex, ez, ey, frame},
   orig =
    Table[rem.position[[i]].Zdisplace[
      1.02*lengthco].hom[{0, 0, 0}], {i, 5}];
   ex = Table[
     rem.position[[i]].Zdisplace[
      1.02*lengthco].hom[{lengthex, 0, 0}], {i, 5}];
   ey = Table[
     rem.position[[i]].Zdisplace[
      1.02*lengthco].hom[{0, lengthey, 0}], {i, 5}];
   ez = Table[
     rem.position[[i]].Zdisplace[
      1.02*lengthco].hom[{0, 0, lengthey}], {i, 5}];
   frame = Table[{
      Thickness[0.004],
      color, Line[{orig[[i]], ex[[i]]}], color2,
      Line[{orig[[i]], ez[[i]]}], color3, Line[{orig[[i]], ey[[i]]}],
      (*Thickness[0.001],Black,Line[{ex[[i]],ey[[i]]}],*)
      PointSize[0.01], colordot, Point[orig[[i]]]}, {i, 5}];
   {frame}];
```

```
RRRInverseKinematicsSP[hframestogframe_, hFramesOriginsinGlobalframe_,
   ptO_, FrameptO_, A1_, A2_, Elbow3Rchain_] :=
 Module[{npos, hFramesOrigIngFrame,
   s2, \[Theta]1, \[Theta]2, \[Theta]3, \[Psi], \[Alpha], psis, x, z,
   kx, ptOXaxisrotatedtoA, gFrame, zhFrameingFrame, endeffs,
   xhFrameinglobalFrame, zhFrameinglobalFrame, checkimg},

  npos = 5;
  s2 = Table[
    LinkAngle[ptO, hFramesOriginsinGlobalframe [[i]]] , {i, npos}];
  \[Theta]2 =
   Mod[Elbow3Rchain*(\[Pi] -
       Mod[Table[
          ArcCos[(-Cos[A1]*Cos[A2] + Cos[s2[[i]]])/(Sin[A1]*
              Sin[A2])], {i, 5}], 2 \[Pi]]), 2 \[Pi]];
  (*Print["\[Theta]2*180/\[Pi] ", \[Theta]2*180/\[Pi]];*)

  psis = Elbow3Rchain*(Mod[
      Table[ArcCos[(-Cos[A1]*Cos[s2[[i]]] + Cos[A2])/(Sin[A1]*
            Sin[s2[[i]]])], {i, 5}], 2 \[Pi]]);
  (*Print[" psis*180/\[Pi] ",psis*180/\[Pi]];*)

  x = FrameptO.{1, 0, 0};
  z = FrameptO.{0, 0, 1};

  kx = Table[
    JointAngle[z, x, hFramesOriginsinGlobalframe [[i]]] , {i, npos}];
  (*Print["kx*180/\[Pi] ",kx*180/\[Pi]];*)

  \[Theta]1 = Mod[kx - psis, 2 \[Pi]];
  (*Print["\[Theta]1*180/\[Pi] ",\[Theta]1*180/\[Pi]];*)

  gFrame = FrameptO;
  ptOXaxisrotatedtoA =
   Table[gFrame.RotZ[\[Theta]1[[i]]].RotY[
      A1].RotZ[\[Theta]2[[i]]].RotY[A2].{1, 0, 0}, {i, 5}];
  zhFrameinglobalFrame =
   Table[gFrame.hframestogframe[[i]].{0, 0, 1}, {i, 5}];
  xhFrameinglobalFrame =
   Table[gFrame.hframestogframe[[i]].{1, 0, 0}, {i, 5}];

  \[Theta]3 =
   Mod[Table[
     JointAngle[zhFrameinglobalFrame[[i]], ptOXaxisrotatedtoA[[i]],
      xhFrameinglobalFrame[[i]]] , {i, npos}], 2 \[Pi]];
  checkimg = Flatten[{\[Theta]1, \[Theta]2, \[Theta]3}];
  Table[If[
    Im[checkimg[[i]]] != 0, \[Theta]1 = 0; \[Theta]2 = 0; \[Theta]3 =
     0], {i, Length[checkimg]}];
  (*Print["\[Theta]3*180/\[Pi] ",\[Theta]3*180/\[Pi]];*)

  {\[Theta]1, \[Theta]2, \[Theta]3}] (* \
\[Theta]1+\[Theta]2+\[Theta]3=\[Alpha] thus \[Theta]3=\[Alpha]-\
\[Theta]1-\[Theta]2   *)
```

```
(*Converting the user defined data to inout data for the program*)

LengthCO = Chop[LengthCO, 10^-4];
LengthEX = Chop[LengthEX, 10^-4];
LengthEY = Chop[LengthEX, 10^-4];
FrameptO = Chop[FrameptO, 10^-4];
FramesptA = Chop[FramesptA, 10^-4];
PositionOrig = Chop[PositionOrig, 10^-4];
a1 = Chop[a1, 10^-4];
a2 = Chop[a2, 10^-4];
FramesEndEff = Chop[FramesEndEff, 10^-4];



gFrame = Chop[FrameptO, 10^-4];
tFrame = Chop[Inverse[FramesEndEff[[1]]].PositionOrig[[1]], 10^-4];
hframestogframe =
  Table[Inverse[gFrame].PositionOrig[[i]].Inverse[tFrame], {i, npos}];
hFramesOriginsinGlobalframe =
  Table[FramesEndEff[[i]].{0, 0, 1}, {i, npos}];
```

```
s2 = Table[LinkAngle[ptO, hFramesOriginsinGlobalframe[[i]]], {i, 5}];
l1 = ptO;
l2 = FramesptA[[1]].{0, 0, 1};
l3 = FramesEndEff[[1]].{0, 0, 1};
elbowAngle = JointAngle[l2, l1, l3];
If[elbowAngle <= \[Pi], Elbow3Rchain = -1, Elbow3Rchain = 1];
```

```
(*Synthesis and Analysis functions for the two fourbars *)

CheckInput[\[Alpha]_, \[Beta]_, \[Gamma]_, \[Eta]_, Theta_] :=
  Module[{Cmin, Cmax, limit, Prob, end, range, npos},

   npos = Length[Theta];
   If[Chop[\[Alpha]] == 0 || Chop[\[Beta]] == 0 ||
     Chop[\[Gamma]] == 0 || Chop[\[Eta]] == 0, Prob = 1; Goto[end]];
   Cmin =
    ArcCos[(Cos[\[Eta] - \[Beta]] -
        Cos[\[Alpha]]*Cos[\[Gamma]])/(Sin[\[Alpha]]*Sin[\[Gamma]])];
   Cmax =
    ArcCos[(Cos[\[Eta] + \[Beta]] -
        Cos[\[Alpha]]*Cos[\[Gamma]])/(Sin[\[Alpha]]*Sin[\[Gamma]])];
   (*Print["Thetas  ",Theta *180/\[Pi] ];
   Print["Cmin Cmax ",{Cmin*180/\[Pi] ,Cmax*180/\[Pi] }];*)
   (*Print[""];*)
   Prob = 0;
   If[(Im[Cmin] != 0) && (Im[Cmax] != 0), Prob = 0; Goto[end]];

   If[(Im[Cmin] == 0) && (Im[Cmax] != 0),
    range = 2 \[Pi] - 2*Cmin;
    If[Table[
       If[Mod[vectAngle[{Cos[Cmin], Sin[Cmin]}, {Cos[Theta[[i]]],
            Sin[Theta[[i]]]}], 2*\[Pi]] <= range, 1, 0], {i,
        npos}] != {1, 1, 1, 1, 1}, Prob = 1; Goto[end]]];

   If[(Im[Cmin] != 0) && (Im[Cmax] == 0),
    range = 2*Cmax;
    If[Table[
       If[Mod[vectAngle[{Cos[-Cmax], Sin[-Cmax]}, {Cos[Theta[[i]]],
            Sin[Theta[[i]]]}], 2*\[Pi]] <= range, 1, 0], {i,
        npos}] != {1, 1, 1, 1, 1}, Prob = 1; Goto[end]]];

   If[(Im[Cmin] == 0) && (Im[Cmax] == 0),
    range = Cmax - Cmin;
    If[Table[
       If[Mod[vectAngle[{Cos[Cmin], Sin[Cmin]}, {Cos[Theta[[i]]],
            Sin[Theta[[i]]]}], 2*\[Pi]] <= range, 1, 0], {i,
        npos}] != {1, 1, 1, 1, 1},
       If[Table[
           If[Mod[vectAngle[{Cos[-Cmax], Sin[-Cmax]}, {Cos[Theta[[i]]],
               Sin[Theta[[i]]]}], 2*\[Pi]] <= range, 1, 0], {i,
           npos}] != {1, 1, 1, 1, 1}, Prob = 1; Goto[end]];
       ]];
     (*Print["range ",range];*)
     Label[end];
     Prob];




   FirstFourBarSynthesis[position_, ptO_] :=
     Module[{npos, W1, G, Sdisp, ConstraintEqn, DesignEQN, numsol1, sol,
       solp, solp1, solp2, solp3, Var, Var1, Var2, ptsO, ptsA, ptsC,
       ptsB, x, y, u, v, R, flag, CrankLength, ptStore, solution, order,
       UserLinka1, noSol, EndFunction, ptOz, ptA, ptAz, k, solp4},

     npos = 5;
     ptOz = ptO/ptO[[3]];
     ptA = position[[1]].{0, 0, 1};
           ptAz = ptA/ptA[[3]];
     W1 = {x, y, 1};
     G = {u, v, 1};



     UserLinka1 = Flatten[{ptOz[[1]], ptOz[[2]], ptAz[[1]], ptAz[[2]]}];
```

```
(*Print["  "];*)
(*Print["User Specified Link  ",UserLinka1 ];*)

Sdisp =
 Table[Chop[position[[i]].Inverse[position[[1]]]], {i, npos}];

ConstraintEqn =
 Table[Expand[Dot[Sdisp[[i]].W1 − G, Sdisp[[i]].W1 − G]] − R^2, {i,
    npos}];

DesignEQN =
 Table[Chop[ConstraintEqn[[i + 1]] − ConstraintEqn[[1]]], {i,
   npos − 1}];
       numsol1 = NSolve[DesignEQN == {0, 0, 0, 0}, {u, v, x, y}];
If[Length[numsol1] == 0, Goto[noSol]];

sol = Sort[{u, v, x, y} /. numsol1];
solp =
 Table[Table[If[Im[sol[[i, j]]] != 0, 0, sol[[i, j]]], {j, 4}], {i,
    Length[sol]}];
solp2 = Cases[solp, Except[{0, 0, 0, 0}]];
If[Length[solp2] == 0, Goto [noSol]];
order =
 Ordering[Table[Norm[solp2[[i]] − UserLinka1], {i, Length[solp2]}],
   All, Less];
solp3 = solp2[[order]];
```

```
    (*Print["Solutions to the first fourbar synthesis    ",solp3];
    Print["Norm (solp3 - UserLinka1)   ",Table[Chop[Norm[solp3[[i]]-
    UserLinka1],10^-3],{i,Length[solp3]}]];*)
    If[Chop[Norm[solp3[[1]] - UserLinka1], 10^-3] > 10^-3, Goto[noSol]];
    (* Gives the position of the nos of the list so as to sort it eg \
a={101,23,84,29}   order = Ordering[a] is {2,4,3,
    1} that means the 2nd element from the list a should be placed in \
1st spot, 4th in 2nd,
    3rd in 3rd and 1st in 4th respectively to sort a & a[[
    order]] sorts it *)

    (*Print["Length of solp3 >>>>>>",Length[solp3]];*)




    (*Neglect solutions that are huge >>>>>>>>>>>>>>>>>*)
    k = ConstantArray[0, Length[solp3]];
    Table[If[Norm[solp3[[i]]] > 100, k[[i]] = 1], {i, Length[solp3]}];
    Table[If[k[[i]] != 0, solp3[[i]] = {0}], {i, Length[solp3]}];
    solp4 = Cases[solp3, Except[{0}]];
    (*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*)

    (*Print["Length of solp4 >>>>>>",Length[solp4]];*)


    (*Var=Which[Length[solp4]==2,{{1,2}},
Length[solp4]==3,{{1,2},{1,3}},
Length[solp4]==4,{{1,2},{1,3},{1,4}},
Length[solp4]==5,{{1,2},{1,3},{1,4}},{{1,5}},
Length[solp4]==6,{{1,2},{1,3},{1,4}},{{1,5}},{{1,6}},
Length[solp4]==7,{{1,2},{1,3},{1,4}},{{1,5}},{{1,6}},{{1,7}}];*)

If[Length[solp4] == 2, Var = {{1, 2}}];
If[Length[solp4] == 3, Var = {{1, 2}, {1, 3}}];
If[Length[solp4] == 4, Var = {{1, 2}, {1, 3}, {1, 4}}];
If[Length[solp4] == 5, Var = {{1, 2}, {1, 3}, {1, 4}, {1, 5}}];
If[Length[solp4] == 6,
 Var = {{1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}}];
If[Length[solp4] == 7,
 Var = {{1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}}];




Var1 = Table[Var[[i, 1]], {i, Length[Var]}];
Var2 = Table[Var[[i, 2]], {i, Length[Var]}];
ptsO =
 Table[{solp4[[Var1[[i]], 1]], solp4[[Var1[[i]], 2]], 1}, {i,
   Length[Var]}];
ptsA =
 Table[{solp4[[Var1[[i]], 3]], solp4[[Var1[[i]], 4]], 1}, {i,
   Length[Var]}];
ptsB =
   Table[{solp4[[Var2[[i]], 3]], solp4[[Var2[[i]], 4]], 1}, {i,
     Length[Var]}];
  ptsC =
   Table[{solp4[[Var2[[i]], 1]], solp4[[Var2[[i]], 2]], 1}, {i,
     Length[Var]}];
  solution = Chop[{ptsO, ptsA, ptsB, ptsC}];

  Goto[EndFunction];

  Label [noSol];
  solution = {0};

  Label[EndFunction];
  solution];




LinkAnalysis[t_, \[Alpha]_, \[Beta]_, \[Gamma]_, \[Eta]_] :=
 Module[{Acoef, Bcoef, Ccoef, Psi, Phi, end, aout},
  If[Chop[\[Alpha]] == 0 || Chop[\[Beta]] == 0 ||
    Chop[\[Gamma]] == 0 || Chop[\[Eta]] == 0, Psi = {0, 0};
   Phi = {0, 0}; Goto[end]];

  Acoef =
   Cos[t]*Sin[\[Alpha]]*Cos[\[Gamma]]*Sin[\[Beta]] -
    Cos[\[Alpha]]*Sin[\[Gamma]]*Sin[\[Beta]];
```

```
Bcoef = Sin[t]*Sin[\[Alpha]]*Sin[\[Beta]];
Ccoef =
 Cos[\[Eta]] - Cos[t]*Sin[\[Alpha]]*Sin[\[Gamma]]*Cos[\[Beta]] -
  Cos[\[Alpha]]*Cos[\[Gamma]]*Cos[\[Beta]];
Psi = {ArcTan[Acoef, Bcoef] -
   ArcCos[Ccoef/Sqrt[Acoef^2 + Bcoef^2]],
  ArcTan[Acoef, Bcoef] + ArcCos[Ccoef/Sqrt[Acoef^2 + Bcoef^2]]};
Phi = Table[
  ArcTan[(Sin[\[Beta]]*Cos[t]*Cos[\[Gamma]]*Cos[Psi[[i]]] +
      Cos[t]*Sin[\[Gamma]]*Cos[\[Beta]] +
      Sin[\[Beta]]*Sin[t]*Sin[Psi[[i]]] -
      Sin[\[Alpha]]*Cos[\[Eta]])/(Cos[\[Alpha]]*
      Sin[\[Eta]]), (-Sin[\[Beta]]*Sin[t]*Cos[\[Gamma]]*
       Cos[Psi[[i]]] - Sin[t]*Sin[\[Gamma]]*Cos[\[Beta]] +
      Sin[\[Beta]]*Cos[t]*Sin[Psi[[i]]])/Sin[\[Eta]]], {i,
   Length[Psi]}];

Label[end];
aout = {Psi, Phi}];




FirstFourBarAnalysis[ptO_, ptA_, ptB_, ptC_, position_] :=
Module[{\[Alpha], \[Beta], \[Gamma], \[Eta], Sdisp, ptsA,
 ptsB, \[CapitalTheta], \[CapitalPsi], psiout, phiout, modes, end,
 aout, Prob, Prob1, Prob2, s1, npos, Proboutofrange, m, c1, c2,
 wptsO, wptsA, wptsB, wptsC, n, solution},



npos = 5;
m = Length[ptO];
Proboutofrange = ConstantArray[0, m];
s1 = ConstantArray[10, m];
c1 = ConstantArray[10, {m, 5}];
c2 = ConstantArray[10, {m, 5}];
Prob = Prob1 = Prob2 = 0;

wptsO = ptO; wptsA = ptA; wptsB = ptB; wptsC = ptC;



Table[
 Sdisp =
  Table[Chop[position[[i]].Inverse[position[[1]]]], {i, npos}];
 \[Alpha] = LinkAngle[ptO[[n]], ptA[[n]]]  ;
 \[Beta] = LinkAngle[ptB[[n]], ptC[[n]]]  ;
 \[Gamma] = LinkAngle[ptO[[n]], ptC[[n]]];
 \[Eta] = LinkAngle[ptB[[n]], ptA[[n]]];
 (*If[ Chop[\[Alpha]]==0||Chop[\[Beta]]==0||Chop[\[Gamma]]==0||
 Chop[\[Eta]]==0,\[CapitalTheta]=0;Prob=1;Goto[nextFourbar]];*)
 (*Print[{\[Alpha],\[Beta],\[Gamma],\[Eta]}];*)

 ptsA = Table[Sdisp[[i]].MakeUnitV[ptA[[n]]], {i, npos}];
 ptsB = Table[Sdisp[[i]].MakeUnitV[ptB[[n]]], {i, npos}];
```

```
    \[CapitalTheta] =
     Table[JointAngle[ptO[[n]], ptC[[n]], ptsA[[i]]], {i, npos}];
    \[CapitalPsi] =
     Table[If[Chop[ptsB[[i]]] == Chop[MakeUnitV[ptC[[n]]]], 0,
       Mod[Pi - JointAngle[ptC[[n]], ptsB[[i]], ptO[[n]]],
        2 \[Pi]]], {i, npos}];

    (*Print[""];
    Print["Output angles \[Psi] first fourbar ",\[CapitalPsi]];*)

    (*Print["Five crank angles ",Mod[\[CapitalTheta],
    2\[Pi]]*180/\[Pi] ];
    Print["Five output angles ",Mod[\[CapitalPsi],
    2\[Pi]] *180/\[Pi]];*)
    aout =
     Table[LinkAnalysis[\[CapitalTheta][[
        i]], \[Alpha], \[Beta], \[Gamma], \[Eta]], {i, npos}];
    psiout = Table[Mod[aout[[i, 1]], 2 \[Pi]], {i, npos}];
    (*Print["Output angles \[Psi] elbow up/ down first fourbar ",
    psiout];*)
    (*Print[
    "Five follower angles using LinkAnalysis 2 values hence c1 or c2 \
",psiout*180/\[Pi] ];*)
    (*phiout=Table[Mod[aout[[i,2]],2\[Pi]],{i,npos}];*)

    c1[[n]] =
     Chop[Table[psiout[[i, 1]] - \[CapitalPsi][[i]], {i, 5}],
       10^-3]; (*Print["c1 ",c1[[n]]];*)
    c2[[n]] =
     Chop[Table[psiout[[i, 2]] - \[CapitalPsi][[i]], {i, 5}],
       10^-3]; (*Print["c2 ",c2[[n]]];*)
    Proboutofrange[[n]] =
     CheckInput[\[Alpha], \[Beta], \[Gamma], \[Eta], \
\[CapitalTheta]];(* Print["proboutofrange ", Proboutofrange[[n]]];*)


    , {n, m}];


   (*Print[""];Print[""];
   Print["The first fourbars to be analyzed have if crank moving out \
of range 1 or 0 ",Proboutofrange];
   Print["C1 ",c1];
   Print["C2 ",c2];*)

   Table[If[Chop[c1[[i]]] == {0, 0, 0, 0, 0}, s1[[i]] = -1], {i,
     m} ];(* elbow up *)
   Table[If[Chop[c2[[i]]] == {0, 0, 0, 0, 0}, s1[[i]] = 1], {i,
     m}];  (* elbow down *)



   Table[If[(Chop[c1[[i]]] != {0, 0, 0, 0, 0} &&
       Chop[c2[[i]]] != {0, 0, 0, 0, 0})  ,
     s1[[i]] = 0; wptsO[[i]] = {0, 0}; wptsA[[i]] = {0, 0};
     wptsB[[i]] = {0, 0}; wptsC[[i]] = {0, 0}], {i, m}];
   (*Checking cmin cmax here*)
   Table[If[Proboutofrange[[i]] == 1,
     s1[[i]] = 0; wptsO[[i]] = {0, 0}; wptsA[[i]] = {0, 0};
     wptsB[[i]] = {0, 0}; wptsC[[i]] = {0, 0}], {i, m}];



   wptsO = Cases[wptsO, Except[{0, 0}]];
   wptsA = Cases[wptsA, Except[{0, 0}]];
   wptsB = Cases[wptsB, Except[{0, 0}]];
   wptsC = Cases[wptsC, Except[{0, 0}]];
   s1 = Cases[s1, Except[0]];

   If[Length[wptsO] == 0, s1 = {0}; solution = {0},
    solution = Chop[{wptsO, wptsA, wptsB, wptsC}]];

   {solution, s1}];





SecondFourBarSynthesis[gFrame_, NEWPTO_, NEWPTA_, NEWPTB_, NEWPTC_,
   B2_, B3_, s1_] :=
 Module[{npos, B2Sdisp, NEW5PTSA, NEW5PTSB, \[Psi]pos, B4, m, W1, G,
   Edisp, Fdisp, ConstraintEqn, DesignEQN, numsol1, Case1, Case2,
   Case3, sol, solp, solp2, solp3, solp4, x, y, u, v, R, sflag,
   CrankLength, NEWPTF, NEWPTE, NewPtD, UserLinka2, order, ptBz, ptDz,
    ptcFrameZaxis , ptcFrameXaxis , ptcFrameYaxis , FrameptC , xgFrame ,
   zgFrame , groundAngle , k , k2} ,



  (*Print[
  "Second Synthesis ************************************"];*)

  npos = 5;
  NewPtD = B3[[1]].{0 , 0, 1};   ptDz = NewPtD/NewPtD[[3]];
  m = Length[NEWPTO];
  NEWPTF = ConstantArray[0 , m];
  NEWPTE = ConstantArray[0 , m];

  Table[



   (*Print["_____________",n,
   " Linkage is being synthesized for second \
fourbar_______________"];*)
   ptBz = NEWPTB[[n]]/NEWPTB[[n, 3]];
   UserLinka2 = Flatten[{ptBz[[1]] , ptBz[[2]] , ptDz[[1]] , ptDz[[2]]}];

   (*Print["UserLink 2 = ",UserLinka2];*)

   B2Sdisp = Table[Chop[B2[[i]].Inverse[B2[[1]]]] , {i , npos}];
   NEW5PTSB = Table[B2Sdisp[[i]].MakeUnitV[NEWPTB[[n]]] , {i , npos}];
```

```
xgFrame = gFrame.{1, 0, 0};
zgFrame = gFrame.{0, 0, 1};
groundAngle =
 Mod[JointAngle[MakeUnitV[NEWPTO[[n]]], xgFrame,
   MakeUnitV[NEWPTC[[n]]]], 2 \[Pi]];
k = LinkAngle[NEWPTO[[n]], NEWPTC[[n]]] ;
FrameptC = gFrame.RotZ[groundAngle].RotY[k];
ptcFrameZaxis = FrameptC.{0, 0, 1};
ptcFrameXaxis = FrameptC.{1, 0, 0};



(*ptcFrameZaxis=Chop[MakeUnitV[NEWPTC[[n]]]];
ptcFrameXaxis=Chop[MakeUnitV[Cross[MakeUnitV[NEWPTC[[n]]],
MakeUnitV[{NEWPTC[[n,1]],0,NEWPTC[[n,3]]}]]]];
ptcFrameYaxis=Chop[MakeUnitV[Cross[ptcFrameZaxis,ptcFrameXaxis]]];
FrameptC=Transpose[{ptcFrameXaxis,ptcFrameYaxis,ptcFrameZaxis}]; *)

\[Psi]pos =
 Table[JointAngle[ptcFrameZaxis, ptcFrameXaxis, NEW5PTSB[[i]]], {i,
    npos}];

B4 = Table[FrameptC.RotZ[\[Psi]pos[[i]]], {i, npos}];

W1 = {x, y, 1};
G = {u, v, 1};

Edisp = Table[Chop[B3[[i]].Inverse[B3[[1]]]], {i, npos}];(*x,y*)
```

```
Fdisp = Table[Chop[B4[[i]].Inverse[B4[[1]]]], {i, npos}];  (*u,
v*)

ConstraintEqn =
 Table[Expand[
    Dot[Edisp[[i]].W1 - Fdisp[[i]].G,
     Edisp[[i]].W1 - Fdisp[[i]].G]] - R^2, {i, npos}];
DesignEQN =
 Table[Chop[ConstraintEqn[[i + 1]] - ConstraintEqn[[1]]], {i,
   npos - 1}];
numsol1 = NSolve[DesignEQN == {0, 0, 0, 0}, {u, v, x, y}];
sol = Sort[{u, v, x, y} /. numsol1];
solp =
 Table[Table[If[Im[sol[[i, j]]] != 0, 0, sol[[i, j]]], {j, 4}], {i,
    Length[sol]}];
solp2 = Cases[solp, Except[{0, 0, 0, 0}]];
order =
 Ordering[
  Table[Chop[Norm[solp2[[i]] - UserLinka2], 10^-3], {i,
    Length[solp2]}], All, Less];
solp3 = solp2[[order]];



(*Neglect solutions that are huge >>>>>>>>>>>>>>>>>*)
k2 = ConstantArray[0, Length[solp3]];
```

```
    Table[If[Norm[solp3[[i]]] > 100, k2[[i]] = 1], {i, Length[solp3]}];
    Table[If[k2[[i]] != 0, solp3[[i]] = {0}], {i, Length[solp3]}];
    solp3 = Cases[solp3, Except[{0}]];
    (*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*)


    (*Print["Second foutbar synthesis ",MatrixForm[solp3]];
    Print["Norm[solp3[[i]]-UserLink 2] ",Table[Chop[Norm[solp3[[i]]-
    UserLinka2],10^-2],{i,Length[solp3]}]];
    *)

    solp4 = Table[solp3[[i]], {i, 2, Length[solp3]}];
    (*Print["Crank options apart from the given crank ",MatrixForm[
    solp4]];*)

    NEWPTF[[n]] =
     Chop[Table[{solp4[[i, 1]], solp4[[i, 2]], 1}, {i, Length[solp4]}]];
    NEWPTE[[n]] =
     Chop[Table[{solp4[[i, 3]], solp4[[i, 4]], 1}, {i, Length[solp4]}]];

    (*Table[Print["{\[Alpha],\[Beta],\[Gamma],\[Eta]} ",{LinkAngle[
    NewPtD,NEWPTB[[n]]],LinkAngle[NEWPTF[[n,i]],NEWPTE[[n,i]]],
    LinkAngle[NEWPTB[[n]],NEWPTF[[n,i]]],LinkAngle[NewPtD,NEWPTE[[n,
    i]]]}],{i,Length[NEWPTF[[n]]]}];
    Print["_______________END of Linkage being \
synthesized__________________"];*)
    , {n, m}];
```

```
  (*Print["PtsF  ",NEWPTF]; Print["PtsE  ",NEWPTE];*)
  (*Print[
  "{candidates for the second fourbars \
NEWPTO,NEWPTA,NEWPTB,NEWPTC,NEWPTF,NEWPTE}  ",{NEWPTO,NEWPTA,NEWPTB,
  NEWPTC,NEWPTF,NEWPTE}];*)


  (*Print[
  "END Second Synthesis ***************************************"];
  Print[""];
  Print[""];*)


  {m, NEWPTO, NEWPTA, NEWPTB, NEWPTC, NEWPTF, NEWPTE, s1}]




SecondFourBarAnalysis[gFrame_, NewPtO_, NewPtA_, NewPtB_, NewPtC_,
  NewPtF_, NewPtE_, B2_, B3_, s1_] :=
 Module[{npos, m, B2Sdisp, B3Sdisp, B4, B4Sdisp, ptsA, ptsB, ptsD,
   ptsE, ptsF, \[Theta], psi, aout, psiout, c1, c2, wptsF, wptsE,
   sum1, sum2, sum3, sum4, NewPtD, \[Psi]pos, solution, sg1, s2,
   Proboutofrange, \[Theta]seccranklocal, GroundAngles, ptcFrameZaxis,
    ptcFrameYaxis, ptcFrameXaxis,
   FrameptC, \[Alpha], \[Beta], \[Gamma], \[Eta], xgFrame, zgFrame,
 groundAngle, k},



(*Print[
"SolutionSet Second Analysis >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> "];*)
\


npos = 5;
NewPtD = B3[[1]].{0, 0, 1};

(*Print["OABCDFE ",{NewPtO,NewPtA,NewPtB,NewPtC,NewPtD,NewPtF,
NewPtE}];*)

m = Length[NewPtF];
solution = ConstantArray[0, m];
sg1 = ConstantArray[s1, m];
s2 = ConstantArray[2, m];
Proboutofrange = ConstantArray[0, m];
c1 = ConstantArray[1, {m, 4}];
c2 = ConstantArray[1, {m, 4}];
\[Theta]seccranklocal = ConstantArray[10, {m, 5}];
(*GroundAngles=ConstantArray[10,{m,5}];*)

wptsF = NewPtF; wptsE = NewPtE;
```

```
xgFrame = gFrame.{1, 0, 0};
zgFrame = gFrame.{0, 0, 1};
groundAngle =
 Mod[JointAngle[MakeUnitV[NewPtO], xgFrame, MakeUnitV[NewPtC]],
   2 \[Pi]];
k = LinkAngle[NewPtO, NewPtC] ;
FrameptC = gFrame.RotZ[groundAngle].RotY[k];
(*Print[FrameptC];*)
ptcFrameZaxis = FrameptC.{0, 0, 1};
ptcFrameXaxis = FrameptC.{1, 0, 0};



Table[
 (*Print[""];
 Print["//////////////////////",n,
 "Linkage being checked \
{NewPtO_,NewPtA_,NewPtB_,NewPtC_,NewPtF_,NewPtE} ",{NewPtO,NewPtA,
 NewPtB,NewPtC,NewPtF[[n]],NewPtE[[n]]},"//////////////////"];*)

 B2Sdisp = Table[Chop[B2[[i]].Inverse[B2[[1]]]], {i, npos}];
 ptsB = Table[B2Sdisp[[i]].MakeUnitV[NewPtB], {i, npos}];

 (*ptcFrameZaxis=Chop[MakeUnitV[NewPtC]];
 ptcFrameXaxis=Chop[MakeUnitV[Cross[MakeUnitV[NewPtC],
 MakeUnitV[{NewPtC[[1]],0,NewPtC[[3]]}]]]];
 ptcFrameYaxis=Chop[MakeUnitV[Cross[ptcFrameZaxis,ptcFrameXaxis]]];
```

```
    FrameptC=Transpose[{ptcFrameXaxis,ptcFrameYaxis,
    ptcFrameZaxis}];   *)


    \[Psi]pos =
     Table[JointAngle[ptcFrameZaxis, ptcFrameXaxis, ptsB[[i]]], {i,
       npos}];
    B4 = Table[FrameptC.RotZ[\[Psi]pos[[i]]], {i, npos}];

    B3Sdisp = Table[Chop[B3[[i]].Inverse[B3[[1]]]], {i, npos}];
    B4Sdisp = Table[Chop[B4[[i]].Inverse[B4[[1]]]], {i, npos}];

    \[Alpha] = LinkAngle[NewPtB, NewPtD]  ;
    \[Beta] = LinkAngle[NewPtF[[n]], NewPtE[[n]]]  ;
    \[Gamma] = LinkAngle[NewPtB, NewPtF[[n]]];
    \[Eta] = LinkAngle[NewPtD, NewPtE[[n]]];

    (*Print[
    "(\[Alpha],\[Beta],\[Gamma],\[Eta])"  ,{\[Alpha],\[Beta],\[Gamma],\
\[Eta]}];*)
    ptsD = Table[B3[[i]].{0, 0, 1}, {i, npos}];
    ptsE = Table[B3Sdisp[[i]].MakeUnitV[NewPtE[[n]]], {i, npos}];
    ptsF = Table[B4Sdisp[[i]].MakeUnitV[NewPtF[[n]]], {i, npos}];


    (*GroundAngles[[n]]=Table[JointAngle[{1,0},ptsF[[i]]-ptsB[[i]]],{i,
    npos}];*)
```

```
\[Theta] =
 Table[JointAngle[ptsB[[i]], ptsF[[i]], ptsD[[i]]], {i, npos}];
(*Print[" theta ",\[Theta]];*)
psi = Table[
  Mod[\[Pi] - JointAngle[ptsF[[i]], ptsE[[i]], ptsB[[i]]],
   2 \[Pi]], {i, npos}];




(*Print["second fourbar  Output angles \[Psi] ",psi];*)
aout =
 Table[LinkAnalysis[\[Theta][[
    i]], \[Alpha], \[Beta], \[Gamma], \[Eta]], {i, npos}];

psiout = Table[Mod[aout[[i, 1]], 2 \[Pi]], {i, npos}];
(*Print[
"Output angles \[Psi] elbow up/ down second fourbar ",{psiout}];*)

c1[[n]] = Chop[Table[psiout[[i, 1]] - psi[[i]], {i, npos}], 10^-2];
c2[[n]] =
 Chop[Table[psiout[[i, 2]] - psi[[i]], {i, npos}], 10^-2];
Proboutofrange[[n]] =
 CheckInput[\[Alpha], \[Beta], \[Gamma], \[Eta], \[Theta]];
(*Print["c1 in 2 ",c1[[n]]];
Print["c2 in 2 ",c2[[n]]];
Print["second crank out ofrange prob: ",Proboutofrange];*)
\[Theta]seccranklocal[[n]] = \[Theta];
```

```
   (*Print[
   "//////////////////END of Linkage being \
analyzed///////////////////"];*)

   , {n, m}];

   Table[If[Chop[c1[[i]]] == {0, 0, 0, 0, 0}, s2[[i]] = -1], {i,
    m} ];
  Table[If[Chop[c2[[i]]] == {0, 0, 0, 0, 0}, s2[[i]] = 1], {i, m}];

  (*Print["second crank out ofrange prob: ",Proboutofrange];
  Print["c1 in second analysis ",MatrixForm[c1]];
  Print["c2 in second analysis ",MatrixForm[c2]];*)



  Table[If[(Chop[c1[[i]]] != {0, 0, 0, 0, 0} &&
      Chop[c2[[i]]] != {0, 0, 0, 0, 0}) ,
    s2[[i]] = 0; sg1[[i]] = 0; wptsF[[i]] = {0, 0};
    wptsE[[i]] = {0, 0}; \[Theta]seccranklocal[[i]] = {0, 0, 0, 0,
      0};(*GroundAngles[[i]]={0,0,0,0,0}*)], {i, m}];



  (*Checking cmin cmax here*)
  Table[If[Proboutofrange[[i]] == 1 ,
    s2[[i]] = 0; sg1[[i]] = 0; wptsF[[i]] = {0, 0};
    wptsE[[i]] = {0, 0}; \[Theta]seccranklocal[[i]] = {0, 0, 0, 0,
      0};(*GroundAngles[[i]]={0,0,0,0,0}*)], {i, m}];
  wptsF = Cases[wptsF, Except[{0, 0}]];
  wptsE = Cases[wptsE, Except[{0, 0}]];
  sg1 = Cases[sg1, Except[0]];
  s2 = Cases[s2, Except[0]];
  \[Theta]seccranklocal =
   Cases[\[Theta]seccranklocal, Except[{0, 0, 0, 0, 0}]];
  (*GroundAngles=Cases[GroundAngles,Except[{0,0,0,0,0}]];*)

  If[Length[wptsF] == 0, s2 = {0}; solution = {0};
   sg1 = {0}; \[Theta]seccranklocal = {0};(*GroundAngles={0}*),
   Table[solution[[i]] =
     Chop[{NewPtO, NewPtA, NewPtB, NewPtC, NewPtD, wptsF[[i]],
       wptsE[[i]]}], {i, Length[wptsF]}]];

  (*Print["\[Theta]seccranklocal at end",\[Theta]seccranklocal ];
  Print["GroundAngles at end",GroundAngles];*)
  (*solution=Drop[solution,-(m-Length[wptsF])];*)
  solution = Cases[solution, Except[0]];
  If[solution == {}, solution = {0}];

  (*Print[
  "END of SolutionSet Second Analysis >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> \
"];
  Print[""];
  Print[""];*)
  {solution, B2, B3, B4, sg1, s2, \[Theta]seccranklocal}]
```

```
AnimateSphSixBar[pos_, posOriginal_, LengthCO_, LengthEX_, LengthEY_,
   ptO_, a1_, a2_, Solution_, \[Theta]1_, sgn1_, sgn2_, gFrame_,
   tFrame_] :=
  Module[{nu, Origin, Coordframe, ReqStuff, position4by4,
    positionOriginal4by4, taskframesnew, taskframesOriginal, po, pa,
    pb, pc, pd, pf, pe, pto, pta, ptb, ptc, ptd, ptf,
    pte, \[Alpha], \[Beta], \[Gamma], \[Eta], orig, ex, ey, ez, eeEX,
    eeEY, eeEZ, FrameDE, toolOrigwrtFrameDE, toolXwrtFrameDE,
    toolYwrtFrameDE, toolZwrtFrameDE, FrameAB, ptDinFrameAB, xgFrame,
    zgframe, groundAngle, FrameC, xFrameC, anglegroundtoCB, FrameCB,
    ptFinFrameCB, \[Theta]min, \[Theta]max, \[Theta]1mod, ktemp,
    thetaX, kpointer, \[Theta]s, \[Theta]e, \[Theta]lowerboundp, \
\[Theta]lowerboundm, \[Theta]upperboundp, \[Theta]upperboundm,
    rangepm, inc, thm, \[Theta]in, length, Acoef, Bcoef, Ccoef, Psi,
    Psiout, t, r, PtsA, PtsB, PtsD, PtsF, PtsE, FramesonAB,
    FramesonCB, EXs, EYs, EZs, origins, exs, eys, ezs,
    SecondCrankAngles, a, b, g, h, Acoef2, Bcoef2, Ccoef2, Psi2,
    Psiout2, FramesonBF, FramesonDE, toolOrigins, toolXaxiss,
    toolYaxiss, toolZaxiss, k1, k2, imaginaryval1, imaginaryval2,
    imaginaryval3, imaginaryval4, imaginaryval5, DisplayAnimation,
    DiscardLinkage, linksOA, linksAB, linksBC, linksOC, linksAD,
    linksBD, linksCF, linksBF, linksDE, linksEF, linksDTorig,
    linksETorig, FullLinkage, Fullendeffpackage, SpLinkage,
 SpLinkageExtra},



(*nu=19;*)
Origin = {0, 0, 0};
Coordframe = {Thickness[.004], Brown,
  Line[{-5*{LengthCO, 0, 0}, Origin}], Darker[Brown],
  Line[{Origin, 5*{LengthCO, 0, 0}}], Purple,
  Line[{-5*{0, LengthCO, 0}, Origin}], Darker[Purple],
  Line[{Origin, 5*{0, LengthCO, 0}}], Pink,
  Line[{-5*{0, 0, LengthCO}, Origin}], Darker[Pink],
  Line[{Origin, 5*{0, 0, LengthCO}}]};
ReqStuff = {AspectRatio -> Automatic, Axes -> True,
  AxesLabel -> {x, y, z}, ImageSize -> 600,
  PlotRange -> {{-1.5, 1.5}, {-1.5, 1.5}, {-1.5, 1.5}}*0.85*
    LengthCO, ViewPoint -> {4, 2, 3}*1.5,
  ViewVertical -> {0, 1, 0}, ImageSize -> 900,
  SphericalRegion -> True};
position4by4 = Table[Make4by4[pos[[i]]], {i, Length[pos]}];
positionOriginal4by4 =
 Table[Make4by4[posOriginal[[i]]], {i, Length[posOriginal]}];
taskframesnew =
 DisplayTP[position4by4, Black, Black, Black, Black, LengthCO,
  LengthEX, LengthEY];
taskframesOriginal =
 DisplayTP[positionOriginal4by4, Gray, Gray, Gray, Gray, LengthCO,
     LengthEX, LengthEY];




   imaginaryval1 =
    imaginaryval2 = imaginaryval3 = imaginaryval4 = imaginaryval5 = 0;
   (*Print[" "];
   Print[" "];
   Print["++++++++++++++++++++++in animation \
routine++++++++++++++++++++++++++++++++++++++"];*)


   (*Making all points of length LengthCO and finding link lengths \
(angular)*)
   {po, pa, pb, pc, pd, pf, pe} = Solution;
   pto = (po/Norm[po])*LengthCO;
   pta = (pa/Norm[pa])*LengthCO;
   ptb = (pb/Norm[pb])*LengthCO;
   ptc = (pc/Norm[pc])*LengthCO;
   ptd = (pd/Norm[pd])*LengthCO;
   ptf = (pf/Norm[pf])*LengthCO;
   pte = (pe/Norm[pe])*LengthCO;
   {\[Alpha], \[Beta], \[Gamma], \[Eta]} = {LinkAngle[pto, pta],
     LinkAngle[ptb, ptc], LinkAngle[pto, ptc], LinkAngle[ptb, pta]};
```

```
(* Finding the groundAngle *)
xgFrame = gFrame.{1, 0, 0};
zgframe = gFrame.{0, 0, 1};
groundAngle = Mod[JointAngle[po, xgFrame, pc], 2 \[Pi]];
(*Print["groundAngle  ",groundAngle*180/\[Pi]];*)



(*Check Range for first crank*)
imaginaryval1 =
 CheckInput[\[Alpha], \[Beta], \[Gamma], \[Eta],
  Mod[\[Theta]1 - groundAngle, 2 \[Pi]]];
(*Print["Checkinput range of first crank; prob ",imaginaryval1];*)

If[imaginaryval1 == 1, Goto [DiscardLinkage]];



(* Finding the ptD in frame AB*)
eeEZ = MakeUnitV[pa];
eeEY = MakeUnitV[Cross[pa, pb]];
eeEX = MakeUnitV[Cross[eeEY, eeEZ]];
FrameAB = Transpose[{eeEX, eeEY, eeEZ}];
ptDinFrameAB = Chop[Inverse[FrameAB].ptd];



(* Finding the Frame at C *)
FrameC = gFrame.RotZ[groundAngle].RotY[\[Gamma]];
```

```
(* Finding the ptF in frameCB *)
xFrameC = FrameC.{1, 0, 0};
anglegroundtoCB = JointAngle[pc, xFrameC, pb];
FrameCB = FrameC.RotZ[anglegroundtoCB];
ptFinFrameCB = Inverse[FrameCB].ptf;



(*\[Theta]min and \[Theta]max are always from the line OC of the \
four bar such that line OC is flat on the ground along the x axis.*)
(*\[Theta]min=ArcCos[(Cos[\[Eta]-\[Beta]]-Cos[\[Alpha]]*
Cos[\[Gamma]])/(Sin[\[Alpha]]*Sin[\[Gamma]])]; Print[
"\[Theta]min ", \[Theta]min*180/\[Pi]];
\[Theta]max=ArcCos[(Cos[\[Eta]+\[Beta]]-Cos[\[Alpha]]*
Cos[\[Gamma]])/(Sin[\[Alpha]]*Sin[\[Gamma]])];  Print[
"\[Theta]max ",  \[Theta]max*180/\[Pi]];*)

(*If[(Im[\[Theta]min]!=0)&&(Im[\[Theta]max]!=0), Print[
"Range allowed is from 0 to 2\[Pi] allowed"]];
If[(Im[\[Theta]min]==0)&&(Im[\[Theta]max]!=0), Print[
"Range allowed \[Theta]min to 2\[Pi]-\[Theta]min that is is from ",
Mod[\[Theta]min,2\[Pi]]*180/\[Pi]," to ",Mod[(2\[Pi]-\[Theta]min),
2\[Pi]]*180/\[Pi]]];
If[(Im[\[Theta]min]!= 0)&&(Im[\[Theta]max]==0), Print[
"Range allowed -\[Theta]max to \[Theta]max that is from ",
Mod[(2\[Pi]-\[Theta]max),2\[Pi]]*180/\[Pi]," to ",Mod[\[Theta]max,
2\[Pi]]*180/\[Pi]]];
If[(Im[\[Theta]min]== 0)&&(Im[\[Theta]max]==0), Print[
"Range one \[Theta]min to \[Theta]max allowed is from ",
Mod[\[Theta]min,2\[Pi]]*180/\[Pi]," to ",Mod[\[Theta]max ,
2\[Pi]]*180/\[Pi]];
Print["Range two -\[Theta]max to -\[Theta]min allowed is from ",
Mod[(-\[Theta]max),2\[Pi]]*180/\[Pi]," to ",Mod[(-\[Theta]min),
2\[Pi]]*180/\[Pi]];];*)

\[Theta]1mod = Mod[\[Theta]1 - groundAngle, 2 \[Pi]];    (*Print[
"First Crank angles ",\[Theta]1mod*180/\[Pi]];*)



(*Defining first crank angles and lence finding PtsA and PtsB*)
(*thm=ConstantArray[0,nu+1];
Table[thm[[i]]=\[Theta]s+(i-1)*inc,{i,nu+1}];     *)
\[Theta]in = Flatten[Append[\[Theta]1mod, Reverse[\[Theta]1mod]]];
length = Length[\[Theta]in];
Acoef =
 Cos[t]*Sin[\[Alpha]]*Cos[\[Gamma]]*Sin[\[Beta]] -
  Cos[\[Alpha]]*Sin[\[Gamma]]*Sin[\[Beta]];
Bcoef = Sin[t]*Sin[\[Alpha]]*Sin[\[Beta]];
Ccoef =
 Cos[\[Eta]] - Cos[t]*Sin[\[Alpha]]*Sin[\[Gamma]]*Cos[\[Beta]] -
  Cos[\[Alpha]]*Cos[\[Gamma]]*Cos[\[Beta]];
Psi = ArcTan[Acoef, Bcoef] +
  sgn1*ArcCos[Ccoef/Sqrt[Acoef^2 + Bcoef^2]];
Psiout =
 Mod[Table[Psi /. t -> \[Theta]in[[i]], {i, Length[\[Theta]in]}],
   2 \[Pi]];

PtsA =
  Table[rem.Make4by4[
     gFrame.RotZ[
        groundAngle].RotZ[\[Theta]in[[i]]].RotY[\[Alpha]]].Zdisplace[
     LengthCO].{0, 0, 0, 1}, {i, length}];
PtsB =
  Table[rem.Make4by4[
     FrameC.RotZ[Psiout[[i]]].RotY[\[Beta]]].Zdisplace[LengthCO].{0,
       0, 0, 1}, {i, length}];
FramesonAB =
  FramesonCB = FramesonBF = FramesonDE = ConstantArray[0, length];



(* Finding the framesonAB to find PtsD *)
Table[
  EZs = MakeUnitV[PtsA[[i]]];
  EYs = MakeUnitV[Cross[PtsA[[i]], PtsB[[i]]]];
  EXs = MakeUnitV[Cross[EYs, EZs]];
  FramesonAB[[i]] = Transpose[{EXs, EYs, EZs}];
  , {i, length}];
PtsD = Table[FramesonAB[[i]].ptDinFrameAB, {i, length}];
```

```
(* Finding the framesonCB to find PtsF *)
Table[
 FramesonCB[[i]] = FrameC.RotZ[Psiout[[i]]];
  , {i, length}];
PtsF = Table[FramesonCB[[i]].ptFinFrameCB, {i, length}];



(*Check whether any points are imaginary*)
k1 = Flatten[
   Join[pto, pta, ptb, ptc, ptd, ptf, pte, PtsA, PtsB, PtsD, PtsF]];
imaginaryval1 = 0;
Table[If[Im[k1[[i]]] != 0, imaginaryval2 = 1], {i, Length[k1]}];
(*Print[
"Some points are imaginary in the list ptO, ptsA, ptsB, ptC, ptsD, \
ptsF: prob ",imaginaryval2];*)
If[imaginaryval2 == 1, Goto [DiscardLinkage]];




(*Defining second crank angles and hence PtsE*)
SecondCrankAngles =
 Mod[Table[
    JointAngle[PtsB[[i]], PtsF[[i]], PtsD[[i]]], {i, length}],
   2 \[Pi]];
(*SecondCrankAngles*180/\[Pi];*)
{a, b, g, h} = {LinkAngle[ptb, ptd], LinkAngle[pte, ptf],
   LinkAngle[ptb, ptf], LinkAngle[ptd, pte]};
```

```
Acoef2 = Cos[r]*Sin[a]*Cos[g]*Sin[b] - Cos[a]*Sin[g]*Sin[b];
Bcoef2 = Sin[r]*Sin[a]*Sin[b];
Ccoef2 =
 Cos[h] - Cos[r]*Sin[a]*Sin[g]*Cos[b] - Cos[a]*Cos[g]*Cos[b];
Psi2 =
 ArcTan[Acoef2, Bcoef2] +
  sgn2*ArcCos[Ccoef2/Sqrt[Acoef2^2 + Bcoef2^2]];
Psiout2 =
 Mod[Table[Psi2 /. r -> SecondCrankAngles[[i]], {i, length}],
  2 \[Pi]];
(*Print[" Psiout2 ",Psiout2];*)



(*Check Range for second crank*)
(*imaginaryval3=CheckInput[a,b,g,h,Mod[SecondCrankAngles,2\[Pi]]];
Print["Checkinput range of second crank; prob ",imaginaryval3];
If[imaginaryval3==1,Goto [DiscardLinkage]];*)



(*Check if any Psiout angles are imaginary*)
(*imaginaryval2=0;*)
Table[If[Im[Psiout2[[i]]] != 0, imaginaryval4 = 1], {i, length}];
(*Print["Second output angles are imaginary: Prob ",
imaginaryval4];*)
If[imaginaryval4 == 1, Goto [DiscardLinkage]];
```

```
(* Finding the framesonBF to find PtsE*)
Table[
 EZs = MakeUnitV[PtsB[[i]]];
 EYs = MakeUnitV[Cross[PtsB[[i]], PtsF[[i]]]];
 EXs = MakeUnitV[Cross[EYs, EZs]];
 FramesonBF[[i]] = Transpose[{EXs, EYs, EZs}];
 , {i, length}];
PtsE =
 Table[FramesonBF[[i]].RotY[g].RotZ[Psiout2[[i]]].RotY[b].{0, 0,
    LengthCO}, {i, length}];



(* Finding the framesonDE to find the toolOrigins,
toolXaxes and toolYaxes *)
EZs = EYs = EXs = 0;
Table[
 EZs = MakeUnitV[PtsD[[i]]];
 EYs = MakeUnitV[Cross[PtsD[[i]], PtsE[[i]]]];
 EXs = MakeUnitV[Cross[EYs, EZs]];
 FramesonDE[[i]] = Transpose[{EXs, EYs, EZs}];
 , {i, length}];


(* Finding the toolframe origin and axis information in frame DE*)
```

```
orig = rem.Make4by4[pos[[1]]].Zdisplace[LengthCO].hom[{0, 0, 0}];
ex = rem.Make4by4[pos[[1]]].Zdisplace[
   LengthCO].hom[{1.2*LengthEX, 0, 0}];
ey = rem.Make4by4[pos[[1]]].Zdisplace[
   LengthCO].hom[{0, 1.2*LengthEY, 0}];
ez = rem.Make4by4[pos[[1]]].Zdisplace[
   LengthCO].hom[{0, 0, 1.2*LengthEY}];
eeEZ = MakeUnitV[pd];
eeEY = MakeUnitV[Cross[pd, pe]];
eeEX = MakeUnitV[Cross[eeEY, eeEZ]];
FrameDE = Transpose[{eeEX, eeEY, eeEZ}];
toolOrigwrtFrameDE = Inverse[FrameDE].orig;
toolXwrtFrameDE = Inverse[FrameDE].ex;
toolYwrtFrameDE = Inverse[FrameDE].ey;
toolZwrtFrameDE = Inverse[FrameDE].ez;


(* Finding toolOrigins, toolXaxes and toolYaxes in global frame *)

toolOrigins =
 Chop[Table[FramesonDE[[i]].toolOrigwrtFrameDE , {i, length}],
  10^-5]; toolXaxiss =
 Chop[Table[FramesonDE[[i]].toolXwrtFrameDE , {i, length}], 10^-5];
 toolYaxiss =
 Chop[Table[FramesonDE[[i]].toolYwrtFrameDE , {i, length}], 10^-5];
toolZaxiss =
 Chop[Table[FramesonDE[[i]].toolZwrtFrameDE , {i, length}], 10^-5];
```

```
(*Check again if any points are imaginary*)
k2 = Flatten[
   Join[pto, ptc, PtsA, PtsB, PtsD, PtsF, PtsE, toolOrigins,
    toolXaxiss, toolYaxiss]];
(*imaginaryval5=0;*)
Table[If[Im[k2[[i]]] != 0, imaginaryval5 = 1], {i, Length[k2]}];
(*Print[
"Some of the ptsA,ptsB,ptsD,ptsF,ptsE, toolOrigs, toolXaxes or \
toolYaxes are imaginary ",imaginaryval5];*)
If[imaginaryval5 == 1, Goto [DiscardLinkage]];



(*Defining all the circular links*)
linksOA = Table[Orthodrome[pto, PtsA[[i]], LengthCO], {i, length}];
linksAB =
  Table[Orthodrome[PtsA[[i]], PtsB[[i]], LengthCO], {i, length}];
linksBC = Table[Orthodrome[PtsB[[i]], ptc, LengthCO], {i, length}];
linksOC = Table[Orthodrome[pto, ptc, LengthCO], {i, length}];
linksAD =
  Table[Orthodrome[PtsA[[i]], PtsD[[i]], LengthCO], {i, length}];
linksBD =
  Table[Orthodrome[PtsB[[i]], PtsD[[i]], LengthCO], {i, length}];
```

```
linksCF = Table[Orthodrome[ptc, PtsF[[i]], LengthCO], {i, length}];
linksBF =
 Table[Orthodrome[PtsB[[i]], PtsF[[i]], LengthCO], {i, length}];
linksDE =
 Table[Orthodrome[PtsD[[i]], PtsE[[i]], LengthCO], {i, length}];
linksEF =
 Table[Orthodrome[PtsE[[i]], PtsF[[i]], LengthCO], {i, length}];
linksDTorig =
 Table[Orthodrome[PtsD[[i]], toolOrigins[[i]], LengthCO], {i,
   length}];
linksETorig =
 Table[Orthodrome[PtsE[[i]], toolOrigins[[i]], LengthCO], {i,
   length}];

FullLinkage =
 Table[{Thickness[0.005], Black, Line[{linksOC[[i]]}], Orange,
   Line[{linksAB[[i]]}], Line[{linksBD[[i]]}], Green,
   Line[{linksOA[[i]]}], Line[{linksAD[[i]]}], Blue,
   Line[{linksBC[[i]]}], Line[{linksCF[[i]]}],
   Line[{linksBF[[i]]}], Red, Line[{linksEF[[i]]}], Gray,
   Line[{linksDE[[i]]}]}, {i, length}];

Fullendeffpackage =
 Table[{Thickness[0.007], Darker[Green], Line[{linksDTorig[[i]]}],
   Line[{linksETorig[[i]]}], Darker[Cyan],
   Line[{toolOrigins[[i]], toolXaxiss[[i]]}],
   Line[{toolOrigins[[i]], toolYaxiss[[i]]}],
  Line[{toolOrigins[[i]], toolZaxiss[[i]]}]}, {i, length}];




SpLinkage = Table[
  Show[Graphics3D[
    {FullLinkage[[i]], Fullendeffpackage[[i]], taskframesOriginal,
     taskframesnew,
     Coordframe, {Gray, Opacity[0.2],
      Sphere[{0, 0, 0}, LengthCO]}}, ReqStuff],

   ParametricPlot3D[r {Cos[t], 0, Sin[t]}, {t, 0, 2 Pi},
    PlotStyle -> {Lighter[Black]}],
   ParametricPlot3D[r {0, Cos[n], Sin[n]}, {n, 0, 2 Pi},
    PlotStyle -> {Lighter[Gray]}],
   (*ParametricPlot3D[r{Cos[n],Sin[n],0},{n,0,2Pi},
   PlotStyle->{Lighter[Gray]}]*)
   Graphics3D[{Darker[Gray], Cylinder[{0.97*pto, 1.03*pto}, 4]}],
   Graphics3D[{Orange,
     Cylinder[{0.97*PtsA[[i]], 1.03*PtsA[[i]]}, 4]}],
   Graphics3D[{Orange,
     Cylinder[{0.97*PtsB[[i]], 1.03*PtsB[[i]]}, 4]}],
   Graphics3D[{Darker[Gray], Cylinder[{0.97*ptc, 1.03*ptc}, 4]}],
   Graphics3D[{Darker[Gray],
     Cylinder[{0.97*PtsD[[i]], 1.03*PtsD[[i]]}, 4]}],
    Graphics3D[{Red, Cylinder[{0.97*PtsF[[i]], 1.03*PtsF[[i]]}, 4]}],
    Graphics3D[{Darker[Gray],
      Cylinder[{0.97*PtsE[[i]], 1.03*PtsE[[i]]}, 4]}],
    Graphics3D[{Darker[Gray],
      Cylinder[{0.97*toolOrigins[[i]], 1.03*toolOrigins[[i]]}, 4]}]

    ]
   , {i, length}];



(*Print["length of the the animation frames to and fro ",
length];*)

SpLinkageExtra = ConstantArray[0, 100];
Table[
 Table[
  SpLinkageExtra[[10*(i - 1) + j]] = SpLinkage[[i]];
   , {j, 10}]
  , {i, 10}];








Goto [DisplayAnimation];
Label[DiscardLinkage];
SpLinkageExtra = 0;
```

```
    Label [DisplayAnimation];
    (*Print[
   "++++++++++++++++++++++out of animation \
routine+++++++++++++++++++++++++++++++++++++++"];
    Print[" "];
    Print[" "];*)

    {{imaginaryval1, imaginaryval2, imaginaryval3, imaginaryval4,
      imaginaryval5}, SpLinkageExtra}
    ];









(*Main Loop*)

pos = posOriginal = PositionOrig;
(*Print["Original data ",N[PositionOrig]];*)
n = 1;
A1 = a1; A2 = a2;
(*Solutions=ConstantArray[1,{4*Iterations,9}];        *)
\
Solutions = ConstantArray[1, {4*Iterations, 3}];
pp = count = 0;



positionOriginal4by4 =
  Table[Make4by4[posOriginal[[i]]], {i, Length[posOriginal]}];
origOriginal =
  Table[rem.positionOriginal4by4[[i]].Zdisplace[
     LengthCO].hom[{0, 0, 0}], {i, 5}];
exOriginal =
  Table[rem.positionOriginal4by4[[i]].Zdisplace[
     LengthCO].hom[{LengthEX, 0, 0}], {i, 5}];
eyOriginal =
  Table[rem.positionOriginal4by4[[i]].Zdisplace[
     LengthCO].hom[{0, LengthEY, 0}], {i, 5}];
ezOriginal =
  Table[rem.positionOriginal4by4[[i]].Zdisplace[
     LengthCO].hom[{0, 0, LengthEY}], {i, 5}];
tpsOriginal =
  Table[{origOriginal[[i]], exOriginal[[i]], eyOriginal[[i]],
    ezOriginal[[i]]}, {i, 5}];



(*Graphics3D[{DisplayTP[positionOriginal4by4,Green,Black,Red,Black,\
LengthCO,LengthEX,LengthEY]},{AspectRatio->Automatic,Axes->True,\
AxesLabel->{x,y,z},ImageSize->600,PlotRange->{{-1.5,1.5},{-1.5,1.5},{-\
1.5,1.5}}*0.85*LengthCO,ViewPoint->{4,2,3}*1.5,ViewVertical->{0,1,0},\
ImageSize->900,SphericalRegion->True}]*)


Tol\[CapitalTheta] = {-N[TolThetav* \[Pi]/180],
   N[TolThetav* \[Pi]/180]};
Tol\[CapitalPhi] = {-N[TolPhiv* \[Pi]/180], N[TolPhiv* \[Pi]/180]};
Tol\[CapitalPsi] = {-N[TolPsiv \[Pi]/180], N[TolPsiv* \[Pi]/180]};
(*Print["Tolerance in \[Theta] ",Tol\[CapitalTheta]*180/\[Pi]];
Print["Tolerance in \[CapitalPhi] ",Tol\[CapitalPhi]*180/\[Pi]];
Print["Tolerance in \[CapitalPsi] ",Tol\[CapitalPsi]*180/\[Pi]];*)
temppos =
  RandomVariables[Tol\[CapitalTheta], Tol\[CapitalPhi],
   Tol\[CapitalPsi], posOriginal];
(*Print["temppos ",temppos];
Print[" "]; Print[" "];*)


(*temppos4by4=Table[Make4by4[temppos[[i]]],{i,Length[temppos]}];
Show[Graphics3D[{DisplayTP[temppos4by4,Gray,Gray,Gray,Gray,LengthCO,\
LengthEX,LengthEY]},{AspectRatio->Automatic,Axes->True,AxesLabel->{x,\
y,z},ImageSize->600,
PlotRange->{{-1.5,1.5},{-1.5,1.5},{-1.5,1.5}}*0.85*LengthCO,ViewPoint-\
>{4,2,3}*1.5,ViewVertical->{0,1,0},ImageSize->900,SphericalRegion->\
True}]]*)
```

```
(*Loop starts here. For every iteration we can get a max of 3*3 that \
is 9 linkages and as few as zero*)
While[n <= Iterations,

 hframestogframe =
  Table[Inverse[gFrame].pos[[i]].Inverse[tFrame], {i, npos}];
 hFramesOriginsinGlobalframe =
  Table[pos[[i]].Inverse[tFrame].{0, 0, 1}, {i, npos}];
 Table[If[
   LinkAngle[ptO,
     hFramesOriginsinGlobalframe [[i]]] >= (a1 + a2),(*Print[
   "Link lengths a1 & a2 unable to reach all task positions"];*)
   Goto[RandomizeTP]], {i, npos}];

 {\[Theta]1, \[Theta]2, \[Theta]3} =
  N[RRRInverseKinematicsSP[hframestogframe,
    hFramesOriginsinGlobalframe , ptO, FrameptO, A1, A2,
    Elbow3Rchain]];
 If[Norm[{\[Theta]1, \[Theta]2, \[Theta]3}] == 0, Goto[RandomizeTP]];



 B2 = Table[
   gFrame.RotZ[\[Theta]1[[i]]].RotY[A1].RotZ[\[Theta]2[[i]]], {i,
    5}];    (*Frames attached at ptA*)
 B3 = Table[
   gFrame.RotZ[\[Theta]1[[i]]].RotY[A1].RotZ[\[Theta]2[[i]]].RotY[
     A2].RotZ[\[Theta]3[[i]]], {i, 5}];  (*Frames attached at ptD*)



 (*Print[" "];
 Print["-------------------------------Iteration no \
",n,"------------------------------------"];
 Print[" "];*)
 (*Print["{\[Theta]1,\[Theta]2,\[Theta]3} ",{\[Theta]1,\[Theta]2,\
\[Theta]3}*180/\[Pi]];*)
 (*Print["Randomized position ",N[pos]];*)

 (*First Four Bar Synthesis*)
 (*Print["B2 ",B2];*)
 As = FirstFourBarSynthesis[B2,
   ptO];          (* As = solution = {ptsO,ptsA,ptsB,ptsC}. At max \
three candidates for the first four bar*)
 solFirstSyn = N[As];
 (*Print[" "];*)
 (*Print["Result of the first four bar synthesis    ",solFirstSyn];*)

 If[solFirstSyn == {0}, Goto[RandomizeTP]];
 (*Print["Length of solution ", Length[solFirstSyn[[1]]]];*)




 (*First Four Bar Analysis*)
 Bs = FirstFourBarAnalysis[solFirstSyn[[1]], solFirstSyn[[2]],
   solFirstSyn[[3]], solFirstSyn[[4]],
   B2];    (* Bs = {solution,s1} . solution = \
{wptsO,wptsA,wptsB,wptsC} . At max three good first four-bar linkages*)

 solFirstAna = N[Bs[[1]]];
 (*Print[" "];*)
 (*Print["Result of the first four bar analysis \
",MatrixForm[solFirstAna]];*)
 If[solFirstAna == {0}, Goto[RandomizeTP]];
 elbowPOSupdown = Bs[[2]];    (*Gives info for elbow up or down*)
 (*Print[" elbowPOSupdown ",elbowPOSupdown];*)




 (*Second Four Bar Synthesis*)
 Cs = SecondFourBarSynthesis[gFrame, solFirstAna[[1]],
   solFirstAna[[2]], solFirstAna[[3]], solFirstAna[[4]], B2, B3,
   elbowPOSupdown];
 (* Cs = {m,NewptO,NewptA,NewptB,NewptC,NewptF,NewptE,s1} . At max \
three candidates for second four-bar for each of the 'm' good first \
four-bar linkages *)
 solSecondSyn =
  N[{Cs[[2]], Cs[[3]], Cs[[4]], Cs[[5]], Cs[[6]], Cs[[7]]}];
 (*Print[" "];*)
 (*Print["Result of the second four bar Synthesis ",solSecondSyn];*)
 elbowPOSupdown = Cs[[8]];
```

```
(*Second Four Bar Analysis*)
(*Analyse each of the (max three) first good four-bar's max three \
candidates for second four-bar in the table*)
(*Print[" "];*)
(*Print[" "];
Print["Result of Second Four Bar Analysis"];
Print[" "];*)


Table[
 Ds = SecondFourBarAnalysis[gFrame, solSecondSyn[[1, i]],
   solSecondSyn[[2, i]], solSecondSyn[[3, i]], solSecondSyn[[4, i]],
   solSecondSyn[[5, i]], solSecondSyn[[6, i]], B2, B3,
   elbowPOSupdown[[i]]];
 (* Ds = {solution,B2,B3,B4,firstFourBar_elbowupdown,
 secondFourBar_elbowupdown,\[Theta]seccranklocal} *)
 (* solution[[i]] = {NewPtO,NewPtA,NewPtB,NewPtC,NewPtD,wptsF,
 wptsE} max 3*)
 SixBarLinkage =
  Chop[N[Ds[[1]]],
   10^-5];   (*Max three good second four bars and hence three good \
six bars for this particular good first four bar*)
 (*Print["SixBarLinkage ",MatrixForm[SixBarLinkage]];*)
 If[SixBarLinkage == {0}, Goto[NextGoodFirstFourBar]];
```

```
  B4 = Ds[[4]];
  FirstElbow = Ds[[5]];
  SecondElbow = Ds[[6]];
  \[Theta]cranksec4barlocal = N[Ds[[7]]];
  (*GrAngsec4bar=N[Ds[[8]]]; *)
  (*Check whether each of the max 3 good sixbar mechanism is within \
the RangeX. If not go to the next good second four bar (six bar).
  Then save it and animate it*)

  badgoodsecfourbar = 0;
  Table[
   (*{imaginaryval1, imaginaryval2, imaginaryval3}=Animation[[1]];
   If[(imaginaryval1+imaginaryval2+imaginaryval3)>0,Goto [
   NextGoodSecondFourBar]];*)

   Animation =
    AnimateSphSixBar[pos, posOriginal, LengthCO, LengthEX, LengthEY,
     ptO, A1, A2, Chop[N[SixBarLinkage[[k]]]], \[Theta]1,
     FirstElbow[[k]], SecondElbow[[k]], gFrame, tFrame];
   (*Print[
   "{imaginaryval1, \
imaginaryval2,imaginaryval3,imaginaryval4,imaginaryval5} ",Animation[[
   1]]];*)

   If[Norm[Animation[[1]]] != 0,
    badgoodsecfourbar = badgoodsecfourbar + 1;
    Goto [NextGoodSecondFourBar]];
    Input3 = Animation[[2]];
    (*Print[ListAnimate[Input3]];*)
    (*Print["badgoodsecfourbar = ",badgoodsecfourbar];
    Print["  "];
    Print["  "];
    Print[">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 "];
    Print["Useful Six Bar Linkage no. ",count+k-badgoodsecfourbar];
    Print[SixBarLinkage[[k]]];
    Print["First Elbow positions sgn1 = ",FirstElbow[[k]]];
    Print["Second Elbow positions sgn2 = ",SecondElbow[[k]]];
    Print["First fourbar 5 crank angles local from ground and not from \
 OC ",\[Theta]1*180/\[Pi]];
    Print["Second fourbar 5 crank angles local \
 ",\[Theta]cranksec4barlocal[[k]]*180/\[Pi]];
    Print["pos current ",pos];*)
    (*Print["B2 ",B2];
    Print["B3 ",B3];*)
    (*Print[
    "{\[Theta]1,\[Theta]2,\[Theta]3} \
 ",{\[Theta]1,\[Theta]2,\[Theta]3}];*)
    (*Print["Second fourbar 5 groundangles ",GrAngsec4bar[[k]]];*)
    (*Print["ptO  ",pto];
    Print["a1 ",A1];
    Print["a2 ",A2];*)
    (*Print["\[Theta]1,\[Theta]2,\[Theta]3 ",
N[{\[Theta]1,\[Theta]2,\[Theta]3}]];
Print["\[Theta]1 crank angles in degrees ",Mod[
N[\[Theta]1*180/\[Pi]],360]];*)


(*If[Animation[[2]]!= 0,Print[ListAnimate[Animation[[2]]]]];*)
(*Print[ListAnimate[Animation[[2]]]];*)

{po, pa, pb, pc, pd, pf, pe} = SixBarLinkage[[k]];
Input1 = {po*LengthCO/Norm[po], pa*LengthCO/Norm[pa],
  pb*LengthCO/Norm[pb], pc*LengthCO/Norm[pc], pd*LengthCO/Norm[pd],
   pf*LengthCO/Norm[pf], pe*LengthCO/Norm[pe]};


position4by4 = Table[Make4by4[pos[[i]]], {i, Length[pos]}];
orig =
 Table[rem.position4by4[[i]].Zdisplace[
    LengthCO].hom[{0, 0, 0}], {i, 5}];
ex = Table[
  rem.position4by4[[i]].Zdisplace[
    LengthCO].hom[{1.2*LengthEX, 0, 0}], {i, 5}];
ey = Table[
  rem.position4by4[[i]].Zdisplace[
    LengthCO].hom[{0, 1.2*LengthEY, 0}], {i, 5}];
ez = Table[
  rem.position4by4[[i]].Zdisplace[
    LengthCO].hom[{0, 0, 1.2*LengthEY}], {i, 5}];
```

```
tpsNEW = Table[{orig[[i]], ex[[i]], ey[[i]], ez[[i]]}, {i, 5}];
Input2 = tpsNEW;




(*Print["gFrame ",gFrame];
Print["tFrame ",tFrame];*)
(*Save the SixBar Solutions*)
pp = count + k − badgoodsecfourbar;
Solutions[[pp]] = {Input1, Input2, Input3};

(*Solutions[[pp,1]]=Chop[N[SixBarLinkage[[
k]]]]; *)                    (* Input1: {NewPtO,NewPtA,NewPtB,NewPtC,
NewPtD,wptsF,wptsE} *)
(*Solutions[[pp,2]]= pos;*)(*Table[N[{{data2[[t,2]]+Scaler*−Sin[
data2[[t,1]]],data2[[t,3]]+Scaler*Cos[data2[[t,1]]]},{data2[[t,2]],
data2[[t,3]]},{data2[[t,2]]+Scaler*Cos[data2[[t,1]]],data2[[t,3]]+
Scaler*Sin[data2[[t,1]]]}}],{t,
5}]; *)
\
            (* Input2: Current randomized task position *)
(*Solutions[[pp,3]]=
0;
\
            (* Input3: original task position *)
Solutions[[pp,4]]=
0;                                                          (*
Input4: Frames attached at ptA*)
Solutions[[pp,5]]=
0;                                                          (*
Input5: Frames attached at ptD*)
Solutions[[pp,6]]=
0;                                                          (*
Input6: Frames attached at ptC*)
Solutions[[pp,7]]=
0;                                                 (* Input7:
Elbow up/ down for first four-bar*)
Solutions[[pp,8]]=
0;                                               (* Input8:
Elbow up/ down for second four-bar*)
Solutions[[pp,9]]=
0;
\
          (*Input9: 5 input crank angles*)*)

(*Print[TripleRchainAnimation[\[Theta]1,\[Theta]2,\[Theta]3,gFrame,
tFrame,A1,A2,position]];*)



Label[NextGoodSecondFourBar];
, {k, Length[
  SixBarLinkage]}]; (*k can be max 3 as max three good second \
four bars for one good first four-bar*)
```

```
   count = count + Length[SixBarLinkage] - badgoodsecfourbar;

   Label[NextGoodFirstFourBar];
   , {i, Length[solSecondSyn[[1]]]}];



  Label[RandomizeTP];
  pos = RandomVariables[Tol\[CapitalTheta], Tol\[CapitalPhi],
     Tol\[CapitalPsi], posOriginal];
  (*pto=RandomPTO[TolptOx,TolptOy,ptO];*)
  (*A1=RandomA1[TolA1,a1];
  A2=RandomA1[TolA1,a2];*)

  n++]


LinkageSolutions = Table[Solutions[[i]], {i, count}];
ClearAll  [Solutions, Input2, Input3];


LinkageSolutions2 =
   Sort[LinkageSolutions,
    Max[{LinkAngle [#1[[1, 1]], #1[[1, 2]]],
         LinkAngle  [#1[[1, 2]], #1[[1, 3]]],
         LinkAngle   [#1[[1, 3]], #1[[1, 4]]],
         LinkAngle  [#1[[1, 1]], #1[[1, 4]]],
         LinkAngle  [#1[[1, 2]], #1[[1, 5]]],
         LinkAngle  [#1[[1, 3]], #1[[1, 5]]],
         LinkAngle  [#1[[1, 3]], #1[[1, 7]]],
         LinkAngle  [#1[[1, 5]], #1[[1, 6]]],
         LinkAngle  [#1[[1, 6]], #1[[1, 7]]],
         LinkAngle [{#1[[2, 1, 1, 3]], #1[[2, 1, 2, 3]], #1[[2, 1, 3,
            3]]}, #1[[1, 5]]],
         LinkAngle [{#1[[2, 1, 1, 3]], #1[[2, 1, 2, 3]], #1[[2, 1, 3,
            3]]}, #1[[1, 6]]]}]/

       Min[{LinkAngle   [#1[[1, 1]], #1[[1, 2]]],
         LinkAngle  [#1[[1, 2]], #1[[1, 3]]],
         LinkAngle  [#1[[1, 3]], #1[[1, 4]]],
         LinkAngle   [#1[[1, 1]], #1[[1, 4]]],
         LinkAngle  [#1[[1, 2]], #1[[1, 5]]],
         LinkAngle  [#1[[1, 3]], #1[[1, 5]]],
         LinkAngle  [#1[[1, 3]], #1[[1, 7]]],
         LinkAngle  [#1[[1, 5]], #1[[1, 6]]],
         LinkAngle  [#1[[1, 6]], #1[[1, 7]]],
         LinkAngle [{#1[[2, 1, 1, 3]], #1[[2, 1, 2, 3]], #1[[2, 1, 3,
            3]]}, #1[[1, 5]]],
         LinkAngle [{#1[[2, 1, 1, 3]], #1[[2, 1, 2, 3]], #1[[2, 1, 3,
            3]]}, #1[[1, 6]]]}]   <

      Max[{LinkAngle   [#2[[1, 1]], #2[[1, 2]]],
         LinkAngle  [#2[[1, 2]], #2[[1, 3]]],
         LinkAngle  [#2[[1, 3]], #2[[1, 4]]],
         LinkAngle   [#2[[1, 1]], #2[[1, 4]]],
         LinkAngle  [#2[[1, 2]], #2[[1, 5]]],
         LinkAngle  [#2[[1, 3]], #2[[1, 5]]],
         LinkAngle  [#2[[1, 3]], #2[[1, 7]]],
         LinkAngle  [#2[[1, 5]], #2[[1, 6]]],
         LinkAngle  [#2[[1, 6]], #2[[1, 7]]],
         LinkAngle [{#2[[2, 1, 1, 3]], #2[[2, 1, 2, 3]], #2[[2, 1, 3,
            3]]}, #2[[1, 5]]],
         LinkAngle [{#2[[2, 1, 1, 3]], #2[[2, 1, 2, 3]], #2[[2, 1, 3,
            3]]}, #2[[1, 6]]]}]/

       Min[{LinkAngle  [#2[[1, 1]], #2[[1, 2]]],
         LinkAngle  [#2[[1, 2]], #2[[1, 3]]],
         LinkAngle   [#2[[1, 3]], #2[[1, 4]]],
         LinkAngle   [#2[[1, 1]], #2[[1, 4]]],
         LinkAngle  [#2[[1, 2]], #2[[1, 5]]],
         LinkAngle  [#2[[1, 3]], #2[[1, 5]]],
         LinkAngle  [#2[[1, 3]], #2[[1, 7]]],
         LinkAngle  [#2[[1, 5]], #2[[1, 6]]],
         LinkAngle  [#2[[1, 6]], #2[[1, 7]]],
         LinkAngle [{#2[[2, 1, 1, 3]], #2[[2, 1, 2, 3]], #2[[2, 1, 3,
            3]]}, #2[[1, 5]]],
         LinkAngle [{#2[[2, 1, 1, 3]], #2[[2, 1, 2, 3]], #2[[2, 1, 3,
            3]]}, #2[[1, 6]]]}] &];
```

```
s = LinkageSolutions2;
k1 = ConstantArray[0, Length[s]];
Table[

  Lmax = Max[{LinkAngle [s[[i, 1, 1]], s[[i, 1, 2]]],
      LinkAngle  [s[[i, 1, 2]], s[[i, 1, 3]]],
      LinkAngle   [s[[i, 1, 3]], s[[i, 1, 4]]],
      LinkAngle  [s[[i, 1, 1]], s[[i, 1, 4]]],
      LinkAngle [ s[[i, 1, 2]], s[[i, 1, 5]]],
      LinkAngle  [s[[i, 1, 3]], s[[i, 1, 5]]],
      LinkAngle  [s[[i, 1, 3]], s[[i, 1, 7]]],
      LinkAngle  [s[[i, 1, 5]], s[[i, 1, 6]]],
      LinkAngle  [s[[i, 1, 6]], s[[i, 1, 7]]],
      LinkAngle [{s[[i, 2, 1, 1, 3]], s[[i, 2, 1, 2, 3]],
        s[[i, 2, 1, 3, 3]]}, s[[i, 1, 5]]],
      LinkAngle [{s[[i, 2, 1, 1, 3]], s[[i, 2, 1, 2, 3]],
        s[[i, 2, 1, 3, 3]]}, s[[i, 1, 6]]]}];

  Lmin = Min[{LinkAngle [s[[i, 1, 1]], s[[i, 1, 2]]],
      LinkAngle  [s[[i, 1, 2]], s[[i, 1, 3]]],
      LinkAngle   [s[[i, 1, 3]], s[[i, 1, 4]]],
      LinkAngle  [s[[i, 1, 1]], s[[i, 1, 4]]],
      LinkAngle [ s[[i, 1, 2]], s[[i, 1, 5]]],
      LinkAngle  [s[[i, 1, 3]], s[[i, 1, 5]]],
      LinkAngle  [s[[i, 1, 3]], s[[i, 1, 7]]],
      LinkAngle  [s[[i, 1, 5]], s[[i, 1, 6]]],
    LinkAngle  [s[[i, 1, 6]], s[[i, 1, 7]]],
    LinkAngle [{s[[i, 2, 1, 1, 3]], s[[i, 2, 1, 2, 3]],
      s[[i, 2, 1, 3, 3]]}, s[[i, 1, 5]]],
    LinkAngle [{s[[i, 2, 1, 1, 3]], s[[i, 2, 1, 2, 3]],
      s[[i, 2, 1, 3, 3]]}, s[[i, 1, 6]]]}] ;

  k1[[i]] = Lmax/Lmin;
  , {i, Length[s]}];



Sol1 = Table[{LinkageSolutions2[[i, 1]],
    LinkageSolutions2[[i, 2]]}, {i, Length[LinkageSolutions2]}];

Sollinkages = Flatten[{tpsOriginal, Sol1}];

Solimages =
  Table[LinkageSolutions2[[i, 3]], {i, Length[LinkageSolutions2]}];
```

B Mathematica code for Eight-bar Linkage Design obtained from a 6R Loop

```
LinkLength[a_, b_] :=
  If[Norm[a] == 0 && Norm[b] == 0, 0, N[Sqrt[Dot[b - a, b - a]]]];

JointAngle[b_, c_] :=
  If[Norm[b] == 0 || Norm[c] == 0, 0,
   Mod[ArcTan[Dot[b, c], Det[{b, c}]], 2 \[Pi]]];

JointAnglewithoutMod[b_, c_] :=
  If[Norm[b] == 0 || Norm[c] == 0, 0, ArcTan[Dot[b, c], Det[{b, c}]]];

Zmat[\[Theta]_] := {{Cos[\[Theta]], -Sin[\[Theta]],
    0}, {Sin[\[Theta]], Cos[\[Theta]], 0}, {0, 0, 1}};

Xmat[a_] := {{1, 0, a}, {0, 1, 0}, {0, 0, 1}};

(*rem = {{1,0,0},{0,1,0}};*)

hom[v_] := {v[[1]], v[[2]], 1};

Disp[x_] := {{Cos[x[[1]]], -Sin[x[[1]]], x[[2]]}, {Sin[x[[1]]],
    Cos[x[[1]]], x[[3]]}, {0, 0, 1}};

RRRInverseKinematics[hframestogframe_, a1_, a2_, sgn_] :=
 Module[{npos, hFramesOrigIngFrame,
   s2, \[Theta]1, \[Theta]2, \[Theta]3, \[Psi], \[Alpha]},
  npos = Length[hframestogframe];
  hFramesOrigIngFrame =
   Table[{hframestogframe[[i, 1, 3]], hframestogframe[[i, 2, 3]]}, {i,
     npos}];
  s2 = Table[Norm[hFramesOrigIngFrame[[i]]]^2 , {i, npos}];
  \[Theta]2 =
   Table[ sgn*ArcCos[(s2[[i]] - a1^2 - a2^2)/(2 a1 a2)], {i, npos}];
  \[Psi] =
   Table[ArcTan[a1 + a2 Cos[\[Theta]2[[i]]],
     a2 Sin[\[Theta]2[[i]]]], {i, npos}];
  \[Theta]1 =
   Table[ArcTan[hFramesOrigIngFrame[[i, 1]],
     hFramesOrigIngFrame[[i, 2]]] - \[Psi][[i]], {i,
     npos}]; \[Alpha] =
   Table[ArcTan[hframestogframe[[i, 1, 1]],
     hframestogframe[[i, 2, 1]]], {i, npos}];
  \[Theta]3 =
   Table[\[Alpha][[i]] - \[Theta]1[[i]] - \[Theta]2[[i]], {i, npos}];
  {\[Theta]1, \[Theta]2, \[Theta]3}] (*  \
\[Theta]1+\[Theta]2+\[Theta]3=\[Alpha] thus \[Theta]3=\[Alpha]-\
\[Theta]1-\[Theta]2  *)

RandomVariables5[AngleLimits_, xLimits_, yLimits_, data_] :=
 Module[{xTemp, yTemp, angleTemp, dataTemp, positionTemp},

  dataTemp = positionTemp = ConstantArray[0, 5];

  Table[
   angleTemp = RandomReal[AngleLimits[[i]]]*Degree;
   xTemp = RandomReal[xLimits[[i]]];
   yTemp = RandomReal[yLimits[[i]]];
   (*Print[{angleTemp,xTemp,yTemp}];*)
   dataTemp[[i]] = {(data[[i, 1]] + angleTemp), data[[i, 2]] + xTemp,
     data[[i, 3]] + yTemp};
   positionTemp[[i]] = Disp[dataTemp[[i]]];
   , {i, 5}];

   {dataTemp, positionTemp}]

RandomVariables5onlydata[AngleLimits_, xLimits_, yLimits_, data_] :=
 Module[{xTemp, yTemp, angleTemp, dataTemp},

   dataTemp = ConstantArray[0, 5];

   Table[
   angleTemp = RandomReal[AngleLimits[[i]]]*Degree;
   xTemp = RandomReal[xLimits[[i]]];
   yTemp = RandomReal[yLimits[[i]]];
   (*Print[{angleTemp,xTemp,yTemp}];*)
   dataTemp[[i]] = {(data[[i, 1]] + angleTemp), data[[i, 2]] + xTemp,
     data[[i, 3]] + yTemp};
   , {i, 5}];

   dataTemp]

GenerateRandomizedData[data_, noOfLoops_, TolTheta5_, TolXfive_,
   TolYfive_] := Block[{dataArray, TolTheta, TolX, TolY},
   TolTheta = MapThread[{#1, #2} &, {-TolTheta5, TolTheta5}];
   TolX = MapThread[{#1, #2} &, {-TolXfive, TolXfive}];
   TolY = MapThread[{#1, #2} &, {-TolYfive, TolYfive}];
   dataArray =
   ParallelTable[
    RandomVariables5onlydata[TolTheta, TolX, TolY, data], {i,
     noOfLoops}];
   dataArray = Take[Join[{data}, dataArray], noOfLoops];
   dataArray]

GenerateRandomizedSixRloop[SixRloop_, noOfLoops_, TolXC1_, TolYC1_,
   TolXC6_, TolYC6_] :=
 Block[{dataArray, TolTheta, TolX, TolY, SixRloopArray, c1, c2, c3,
   c4, c5, c6},

   (*TolTheta=MapThread[{#1,#2}&,{-TolTheta5,TolTheta5}];
   TolX=MapThread[{#1,#2}&,{-TolXfive,TolXfive}];
   TolY=MapThread[{#1,#2}&,{-TolYfive,TolYfive}];*)
   {c1, c2, c3, c4, c5, c6} = SixRloop;
   SixRloopArray =
   Table[{c1 + {RandomReal[{-TolXC1, TolXC1}],
       RandomReal[{-TolYC1, TolYC1}]}, c2, c3, c4, c5,
     c6 + {RandomReal[{-TolXC6, TolXC6}],
       RandomReal[{-TolYC6, TolYC6}]}}, {i, noOfLoops}];
   SixRloopArray = Take[Join[{SixRloop}, SixRloopArray], noOfLoops];
   (*Print["SixRloopArray = ",MatrixForm[SixRloopArray]];*)
   SixRloopArray]

MakeDivisionsinTps2[angSequence_, inc_] :=
 Module[{theta1, theta2, theta3, theta4, theta5, Thetas, minidivs,
   ipAngles, angstemp, FiveTpsLocs, miniRangeAntiClockwise,
   miniRangeClockwise , rangeOfMotionforLink1 },

 Thetas = {theta1 , theta2 , theta3 , theta4 , theta5} = angSequence;
 (*Print[" five theta1 = ",Thetas/Degree];*)
 ipAngles = 0; rangeOfMotionforLink1 = 0;
 FiveTpsLocs = {1, 0, 0, 0, 0};

 Table[
   miniRangeAntiClockwise =
   N[JointAngle[{Cos[Thetas[[i]]],
       Sin[Thetas[[i]]]}, {Cos[Thetas[[i + 1]]],
       Sin[Thetas[[i + 1]]]}]];
   (*Print["miniRange = ",N[miniRange/Degree]];*)
   miniRangeClockwise =
   N[2 \[Pi] -
     JointAngle[{Cos[Thetas[[i]]],
       Sin[Thetas[[i]]]}, {Cos[Thetas[[i + 1]]],
       Sin[Thetas[[i + 1]]]}]];
   (*Print[
   "{miniRangeAntiClockwise,miniRangeClockwise} = \
",{miniRangeAntiClockwise,miniRangeClockwise}/Degree];*)

   If[miniRangeAntiClockwise <= miniRangeClockwise,
   rangeOfMotionforLink1 =
   rangeOfMotionforLink1 + miniRangeAntiClockwise;
   minidivs = Floor[miniRangeAntiClockwise/inc];
   (*Print["minidivs = ",N[minidivs]];*)
```

```
    angstemp =
     Table[Mod[Thetas[[i]] + j*inc, 2 \[Pi]], {j, 0, minidivs}],

    rangeOfMotionforLink1 = rangeOfMotionforLink1 + miniRangeClockwise;
    minidivs = Floor[miniRangeClockwise/inc];
    (*Print["minidivs = ",N[minidivs]];*)
    angstemp =
     Table[Mod[Thetas[[i]] - j*inc, 2 \[Pi]], {j, 0, minidivs}]];

   FiveTpsLocs[[i + 1]] = FiveTpsLocs[[i]] + minidivs + 1;
   If[Length[ipAngles] == 0, ipAngles = angstemp,
    ipAngles = Flatten[Append[ipAngles, angstemp]]];
   , {i, 4}];

  (*Print["FiveTpsLocs = ",FiveTpsLocs];*)
  ipAngles = N[Flatten[Append[ipAngles, theta5]]];
  (*Print["rangeOfMotionforLink1 in deg = ",rangeOfMotionforLink1/
  Degree];*)
  {ipAngles, FiveTpsLocs, rangeOfMotionforLink1}]

      (*Difference between angle vectors*)
NormofDiffBetAngVectors[a_, b_] :=
  Module[{VecA, VecB, diff1, diff2, finalDiff},
   VecA = Table[{Cos[a[[i]]], Sin[a[[i]]]}, {i, Length[a]}];
   VecB = Table[{Cos[b[[i]]], Sin[b[[i]]]}, {i, Length[b]}];

   diff1 = Table[
     If[Mod[JointAngle[VecA[[i]], VecB[[i]]], 2 \[Pi]] <
       Mod[JointAngle[VecB[[i]], VecA[[i]]], 2 \[Pi]],
      Mod[JointAngle[VecA[[i]], VecB[[i]]], 2 \[Pi]],
      Mod[JointAngle[VecB[[i]], VecA[[i]]], 2 \[Pi]]]
     , {i, Length[a]}];
   (*Print[diff1];*)

   N[Norm[diff1]]];

MakeCylindrical[list_, rad_] := Block[{newList, list2},
  list2 = list Degree;
  newList = Table[
    {list[[i, 1]], rad*Cos[list2[[i, 2]]], rad*Sin[list2[[i, 2]]]}
    , {i, Length[list]}];
  (*newList=Table[
  Table[
  {list[[i,j,1]],rad*Cos[list2[[i,j,2]]],rad*Sin[list2[[i,j,2]]]}
  ,{j,Length[list[[i]]]}]
  ,{i,Length[list]}];*)
  newList]

DiffBetAngs[a_, b_] := Module[{VecA, VecB, diff1, diff2, finalDiff},
  VecA = {Cos[a], Sin[a]};
  VecB = {Cos[b], Sin[b]};
  diff1 = Mod[Abs[JointAngle[VecA, VecB]], 6.28];
  diff2 = Mod[Abs[JointAngle[VecB, VecA]], 6.28];
  If[diff1 <= diff2, finalDiff = diff1, finalDiff = diff2];
```

```
 N[Abs[finalDiff]]]


 LinkageLinkLenBoundCheck[linkage_, RRcons_, MaxLen_, MinLen_] :=
Block[{lenList, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, linkList,
   linkList2, linkLengthArray, maxlen, minlen, linkageCheck},
 (*Print["linkage = ",linkage];
 Print["RRcons = ",RRcons];*)
 (*Print["{Maxlen,Minlen} = ",{MaxLen,MinLen}];*)

 linkageCheck = 1;
 lenList = {};
 {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10} = linkage;
 linkList = {{c1, c2}, {c2, c3}, {c3, c4}, {c4, c5}, {c5, c6}, {c7,
    c8}};
 linkList2 =
   Join[{{linkList[[RRcons[[1, 1]], 1]],
      c7}, {linkList[[RRcons[[1, 1]], 2]],
      c7}, {linkList[[RRcons[[1, 2]], 1]],
      c8}, {linkList[[RRcons[[1, 2]], 2]], c8},
    {linkList[[RRcons[[2, 1]], 1]],
      c9}, {linkList[[RRcons[[2, 1]], 2]],
      c9}, {linkList[[RRcons[[2, 2]], 1]],
      c10}, {linkList[[RRcons[[2, 2]], 2]], c10}}, {{c7, c8}, {c9,
      c10}}];
 (*Print[linkList2];*)
 linkLengthArray = ConstantArray[0, Length[linkList2]];
 Table[linkLengthArray[[i]] =
  LinkLength[linkList2[[i, 1]], linkList2[[i, 2]]], {i,
  Length[linkList2]}];
(*Print[MatrixForm[linkLengthArray]];*)
{maxlen, minlen} = {Max[linkLengthArray], Min[linkLengthArray]};
(*Print["Current {maxlen,minlen} = ",{maxlen,minlen}];*)
If[maxlen > MaxLen || minlen < MinLen, linkageCheck = 0];

{linkageCheck, maxlen/minlen}]

FindModifiedAdjacencyMatrix3[linkage_, RRconnections_] :=
Block[{LinkageGraph, LinkageGraphLocs, NewConnections,
  SixRloopConnections, AllCons, AllConsLocations,
  LinkageGraphLocsSym, SixRloopConnectionsLocationsSym,
  AllConsLocationsSym, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10},

 LinkageGraph = ConstantArray[ConstantArray[0, 8], 8];
 LinkageGraphLocs = ConstantArray[ConstantArray[0, 8], 8];
 LinkageGraphLocsSym = ConstantArray[ConstantArray[0, 8], 8];

 SixRloopConnections = {{1, 8}, {1, 2}, {2, 3}, {3, 4}, {4, 5}, {5,
    8}};
 NewConnections = {{RRconnections[[1, 1]], 6}, {6,
    RRconnections[[1, 2]]}, {RRconnections[[2, 1]], 7}, {7,
    RRconnections[[2, 2]]}};
 AllCons = Join[SixRloopConnections, NewConnections];
 AllConsLocations = linkage;
 AllConsLocationsSym =
  Flatten[{c1, c2, c3, c4, c5, c6, c7, c8, c9, c10}];

 Table[
  LinkageGraph[[AllCons[[j, 1]], AllCons[[j, 2]]]] = 1;
  LinkageGraph[[AllCons[[j, 2]], AllCons[[j, 1]]]] = 1;

  LinkageGraphLocs[[AllCons[[j, 1]], AllCons[[j, 2]]]] =
   AllConsLocations[[j]];
  LinkageGraphLocs[[AllCons[[j, 2]], AllCons[[j, 1]]]] =
   AllConsLocations[[j]];

  LinkageGraphLocsSym[[AllCons[[j, 1]], AllCons[[j, 2]]]] =
   AllConsLocationsSym[[j]];
  LinkageGraphLocsSym[[AllCons[[j, 2]], AllCons[[j, 1]]]] =
   AllConsLocationsSym[[j]];

  , {j, Length[AllCons]}];



 {LinkageGraph, LinkageGraphLocs, LinkageGraphLocsSym}];

 FindRRConstraint[A_, B_, LinkageVertexListTP1_, data_] :=
Module[{npos, ASdisp, BSdisp, W, x, y, G, u, v, R, ConstraintEqn,
 DesignEQN, numsol1, RawSolution, sflag, CrankLength, RealSolution,
 Userlnk1, Userlnk2, W1, G1, n, order, kot, OneSolution,
 EndFunction, c1, c2, c3, c4, c5, c6, c7, c8, Userlinks,
 CheckNextSolution, FilteredSolution, maxDist, kot2},
maxDist =
 Max[LinkLength[{data[[1, 2]], data[[1, 3]]}, {data[[#, 2]],
       data[[#, 3]]}] & /@ {2, 3, 4, 5}];

{c1, c2, c3, c4, c5, c6, c7, c8} = LinkageVertexListTP1;
Userlinks =
 Join[{Flatten[{c1, c6}], Flatten[{c2, c3}], Flatten[{c3, c4}],
   Flatten[{c4, c5}]},
  Flatten[Table[{Flatten[{LinkageVertexListTP1[[i]], c7}],
     Flatten[{LinkageVertexListTP1[[i]], c8}]}, {i,
     Length[LinkageVertexListTP1]}], 1]];
Table[If[
  Userlinks[[i, 1]] - Userlinks[[i, 3]] == 0 &&
   Userlinks[[i, 2]] - Userlinks[[i, 4]] == 0,
  Userlinks[[i]] = {0, 0, 0, 0}], {i, Length[Userlinks]}];
Userlinks = Cases[Userlinks, Except[{0, 0, 0, 0}]];
(*Print["Userlinks = ",MatrixForm[Userlinks]];*)

npos = 5;
OneSolution = 0;
FilteredSolution = 0;
(*Userlnk1 = Flatten[{ptc1,ptc3}];
Userlnk2=Flatten[{ptc2,ptc4}];*)

ASdisp = Table[Chop[A[[i]].Inverse[A[[1]]]], {i, npos}];
BSdisp = Table[Chop[B[[i]].Inverse[B[[1]]]], {i, npos}];
```

```
G = {u, v, 1};
W = {x, y, 1};
ConstraintEqn =
  Table[Expand[
      Dot[ASdisp[[i]].G - BSdisp[[i]].W,
        ASdisp[[i]].G - BSdisp[[i]].W]] - R^2, {i, npos}];
DesignEQN =
  Table[Chop[ConstraintEqn[[i + 1]] - ConstraintEqn[[1]]], {i,
    npos - 1}];
numsol1 = NSolve[DesignEQN == {0, 0, 0, 0}, {u, v, x, y}];
RawSolution = Sort[{u, v, x, y} /. numsol1];
(*Print["Numsol Solution = ",MatrixForm[RawSolution]];*)

kot = kot2 = ConstantArray[0, Length[RawSolution]];
Table[Table[
   If[Im[RawSolution[[i, j]]] != 0, kot[[i]] = 1], {j, 4}], {i,
   Length[RawSolution]}];
(*Print["kot = ",kot];*)
Table[Table[
   If[Abs[RawSolution[[i, j]]] > maxDist*10, kot2[[i]] = 1], {j,
    4}], {i, Length[RawSolution]}];
(*Print["kot2 = ",kot2];*)

RealSolution = RawSolution;
Table[If[kot[[i]] == 1, RealSolution[[i, 1]] = 0;
   RealSolution[[i, 2]] = 0; RealSolution[[i, 3]] = 0;
   RealSolution[[i, 4]] = 0], {i, Length[RealSolution]}];
```

```
Table[If[kot2[[i]] == 1, RealSolution[[i, 1]] = 0;
  RealSolution[[i, 2]] = 0; RealSolution[[i, 3]] = 0;
  RealSolution[[i, 4]] = 0], {i, Length[RealSolution]}];



RealSolution = Chop[Cases[RealSolution, Except[{0, 0, 0, 0}]]];
If[Length[RealSolution] == 0, OneSolution = 0; FilteredSolution = 0;
  Goto [EndFunction];];

(*Print["Real Solutions ",MatrixForm[RealSolution]];*)
(*Print[""];*)

Table[
 Table[
  If[Chop[Norm[RealSolution[[j]] - Userlinks[[i]]]] <= 10^-3 ||
     Chop[Norm[Reverse[RealSolution[[j]]] - Userlinks[[i]]]] <= 10^-3
    , RealSolution[[j]] = {0, 0, 0, 0}; Goto [CheckNextSolution]];

  , {i, Length[Userlinks]}];

 Label [CheckNextSolution];
 , {j, Length[RealSolution]}];



(*order=Ordering[Table[Norm[solp2[[i]]-Userlnk1],{i,Length[solp2]}],
All,Less];            *)                    (*
Gives the position of the nos of the list so as to sort it eg \
a={101,23,84,29}  order = Ordering[a] is {2,4,3,
  1} that means the 2nd element from the list a should be placed in \
1st spot, 4th in 2nd,
  3rd in 3rd and 1st in 4th respectively to sort a & a[[
  order]] sorts it *)

  (*Print["########################## solp2 ",MatrixForm[solp2]];*)
  FilteredSolution = Cases[RealSolution, Except[{0, 0, 0, 0}]];
  (*Print[" Filtered Solution ",MatrixForm[FilteredSolution]];*)
  (*If[Norm[solp3[[1]]-Flatten[{ptc1,ptc4}]]>10^-3,Goto EndFunction];
  Print[Norm[solp3[[1]]-Flatten[{ptc1,ptc4}]]];
  (*solp4=solp3;*)
  solp4 =Table[solp3[[i]],{i,2,Length[solp3]}];*)

  If[Length[FilteredSolution] == 0, OneSolution = 0;
   FilteredSolution = 0; Goto [EndFunction]];

  OneSolution = FilteredSolution[[1]];

  Label[EndFunction];
  {OneSolution, FilteredSolution}]

IndependentConstraints[data_, SixRloop_, AFrames_, Connections_] :=
  Block[{A1frm, B1frm, Linkage, RR1oneSol, RR1Sols, A2frm, B2frm,
    RR2oneSol, RR2Sols, DegenerateSolution, kcount, Solutions,
    BadLinkage, lenTemp, ptsc7, ptsc8, ptsc9, ptsc10, ptc7, ptc8,
    ptc9, ptc10, thetas6, thetas7, rem},
   rem = {{1, 0, 0}, {0, 1, 0}};
   (*Print[AFrames[[6]]];*)
   Linkage = Join[SixRloop, {{0, 0}, {0, 0}}];
   (*Print["linkage = ",Linkage];
   Print["Current Connection = ",Connections];*)

   (*Print[
   "/////////     Synthesizing first RR constraint (link6)   \
/////////"];*)
   A1frm = Chop[AFrames[[Connections[[1, 1]]]], 10^-4];
   B1frm = Chop[AFrames[[Connections[[1, 2]]]], 10^-4];
   {RR1oneSol, RR1Sols} =
    FindRRConstraint[A1frm, B1frm, Linkage, data];
   If[RR1Sols == 0,(*Print["No linkage exists for this type!!"];*)
    Solutions = 0; Goto [BadLinkage]];
   (*Print[""];*)
   (*Print["RR1 result = ",{MatrixForm[RR1Sols],Length[RR1Sols]}];*)



   (*Print[""];*)
   (*Print[
   "/////////     Synthesizing second RR constraint (link7)   \
/////////"];*)
   A2frm = Chop[AFrames[[Connections[[2, 1]]]], 10^-4];
   B2frm = Chop[AFrames[[Connections[[2, 2]]]], 10^-4];
   {RR2oneSol, RR2Sols} =
    FindRRConstraint[A2frm, B2frm, Linkage, data];
   If[RR2Sols == 0,(*Print["No linkage exists for this type!!"];*)
    Solutions = 0; Goto [BadLinkage]];
   (*Print[""];*)
   (*Print["RR2 result = ",{MatrixForm[RR2Sols],Length[RR2Sols]}];*)


   (*Special case for same RR connections*)
   If[Connections[[1]] == Connections[[2]],
    lenTemp = Length[RR1Sols];
    RR2Sols = {RR1Sols[[lenTemp]]};
    RR1Sols = Take[RR1Sols, lenTemp - 1];
    (*Print[""];
    Print["**** Special Case. Both connections are same."];
    Print["New RR1Sols = ",{MatrixForm[RR1Sols],Length[RR1Sols]}];
    Print["New RR2Sols = ",{MatrixForm[RR2Sols],Length[RR2Sols]}];*)
    ];


   (*Saving distinct Solutions for this type*)
   kcount = 1;
   Solutions = ConstantArray[0, Length[RR1Sols]*Length[RR2Sols]];
   (*Print["Solutions = ",Solutions];*)
   Table[
    Table[
      If[Norm[RR1Sols[[x]] - RR2Sols[[y]]] <= 10^-3,
       Goto [DegenerateSolution]];
```

```
{ptc7,
  ptc8} = {{RR1Sols[[x, 1]], RR1Sols[[x, 2]]}, {RR1Sols[[x, 3]],
    RR1Sols[[x, 4]]}};
{ptc9,
  ptc10} = {{RR2Sols[[y, 1]],
   RR2Sols[[y, 2]]}, {RR2Sols[[y, 3]], RR2Sols[[y, 4]]}};
(*Print["{ptc7,ptc8,ptc9,ptc10} = ",{ptc7,ptc8,ptc9,ptc10}];*)
ptsc7 =
 Chop[
  Table[rem.A1frm[[i]].Inverse[A1frm[[1]]].hom[ptc7], {i, 5}]];
ptsc8 =
 Chop[Table[
   rem.B1frm[[i]].Inverse[B1frm[[1]]].hom[ptc8], {i, 5}]];
ptsc9 =
 Chop[Table[
   rem.A2frm[[i]].Inverse[A2frm[[1]]].hom[ptc9], {i, 5}]];
ptsc10 =
 Chop[Table[
   rem.B2frm[[i]].Inverse[B2frm[[1]]].hom[ptc10], {i, 5}]];
thetas6 =
 Table[JointAngle[{1, 0}, ptsc8[[i]] - ptsc7[[i]]], {i, 5}];
thetas7 =
 Table[JointAngle[{1, 0}, ptsc10[[i]] - ptsc9[[i]]], {i, 5}];
Solutions[[
  kcount]] = {Join[
   SixRloop, {ptc7, ptc8}, {ptc9, ptc10}], {RR1Sols[[x]],
   RR2Sols[[y]]}, Connections, {thetas6, thetas7}};
```

```
      kcount++;
      Label [DegenerateSolution];
      , {y, Length[RR2Sols]}];
    , {x, Length[RR1Sols]}];



   Solutions = Cases[Solutions, Except[0]];
   If[Solutions == {},
    (*Print["No solution found for this connection!!"];*)
    Solutions = 0; Goto [BadLinkage]];
   (*Print[""];*)
   (*Print["Final Solutions = ",MatrixForm[Solutions]];*)



   (*Print[LinkageDisplay[dataTemp,linkage,Connections,RangeXY]];*)
   Label [BadLinkage];

   Solutions];

DependentConstraints[data_, SixRloop_, AFrames_, Connections_] :=
  Block[{A1frm, B1frm, Linkage, RR1oneSol, RR1Sols, A2frm, B2frm,
    RR2oneSol, RR2Sols, DegenerateSolution, kcount, Solutions,
    BadLinkage, lenTemp, ptc7, ptc8, Fiveptsc7, Fiveptsc8,
    FrameAttachedtoLink6, NextRR1Sol, AFrames2, thetas6, thetas7,
    ptc9, ptc10, Fiveptsc9, Fiveptsc10, rem},
   rem = {{1, 0, 0}, {0, 1, 0}};
   Linkage = Join[SixRloop, {{0, 0}, {0, 0}}];
   (*Print["linkage = ",Linkage];*)
   (*Print["Current Connection = ",Connections];*)

   (*Print[
   "/////////      Synthesizing first RR constraint (link6)    \
/////////"];*)
   A1frm = Chop[AFrames[[Connections[[1, 1]]]], 10^-4];
   B1frm = Chop[AFrames[[Connections[[1, 2]]]], 10^-4];
   {RR1oneSol, RR1Sols} =
    FindRRConstraint[A1frm, B1frm, Linkage, data];
   If[RR1Sols == 0,(*Print["No linkage exists for this type!!"];*)
    Solutions = 0; Goto [BadLinkage]];
   (*Print[""];*)
   (*Print["RR1 result = ",{MatrixForm[RR1Sols],Length[RR1Sols]}];*)



   kcount = 1;
   Solutions = ConstantArray[0, 16];
   Table[
    (*Print[""];*)
    (*Print["******* For RR1Sol no.",i,"*******"];*)
    (*Print["******* ",RR1Sols[[i]]];*)

    (*Print[
    "/////////     Synthesizing second RR constraint (link7) for the \
RR1Sol /////////"];*)
    ptc7 = {RR1Sols[[i, 1]], RR1Sols[[i, 2]]};
    ptc8 = {RR1Sols[[i, 3]], RR1Sols[[i, 4]]};
    Fiveptsc7 =
     Table[rem.A1frm[[m]].Inverse[A1frm[[1]]].hom[ptc7], {m, 5}];
    Fiveptsc8 =
     Table[rem.B1frm[[n]].Inverse[B1frm[[1]]].hom[ptc8], {n, 5}];
    thetas6 =
     Table[JointAngle[{1, 0}, Fiveptsc8[[m]] - Fiveptsc7[[m]]], {m,
       5}];
    FrameAttachedtoLink6 =
     Table[Disp[Flatten[{thetas6[[j]], Fiveptsc7[[j]]}]], {j, 5}];

    AFrames2 = Append[Take[AFrames, 5], FrameAttachedtoLink6];
    Linkage = Join[SixRloop, {ptc7, ptc8}];

    A2frm = Chop[AFrames2[[Connections[[2, 1]]]], 10^-4];
    B2frm = Chop[AFrames2[[Connections[[2, 2]]]], 10^-4];
    {RR2oneSol, RR2Sols} =
     FindRRConstraint[A2frm, B2frm, Linkage, data];
    If[RR2Sols == 0,(*Print[
     "No second dep RR2Sol exists for this RR1Sol!!"];*)
     Goto [NextRR1Sol]];
    (*Print[""];*)
    (*Print["RR2 result = ",{MatrixForm[RR2Sols],Length[
    RR2Sols]}];*)
```

```
Table[
 ptc9 = {RR2Sols[[p, 1]], RR2Sols[[p, 2]]};
 ptc10 = {RR2Sols[[p, 3]], RR2Sols[[p, 4]]};
 Fiveptsc9 =
  Table[rem.A2frm[[m]].Inverse[A2frm[[1]]].hom[ptc9], {m, 5}];
 Fiveptsc10 =
  Table[rem.B2frm[[n]].Inverse[B2frm[[1]]].hom[ptc10], {n, 5}];
 thetas7 =
  Table[JointAngle[{1, 0}, Fiveptsc10[[m]] - Fiveptsc9[[m]]], {m,
    5}];

 Solutions[[
   kcount]] = {Join[
    SixRloop, {ptc7, ptc8}, {ptc9, ptc10}], {RR1Sols[[i]],
    RR2Sols[[p]]}, Connections, {thetas6, thetas7}};
 kcount++;
 , {p, Length[RR2Sols]}];

(*Print["Total linkage solutions for this RR1 constraint = ",
MatrixForm[Take[Solutions,kcount-1]]];*)



Label [NextRR1Sol];
, {i, Length[RR1Sols]}];
```

```
    Solutions = Cases[Solutions, Except[0]];
    If[Solutions == {},
     (*Print["No solution found for this connection!!"];*)
     Solutions = 0; Goto [BadLinkage]];
    (*Print[""];*)
    (*Print["Final Solutions = ",MatrixForm[Solutions]];*)
    (*Print[""];*)
    (*Table[


    ,{i,Length[Solutions]}];
    *)

    (*Print[LinkageDisplay[dataTemp,linkage,Connections,RangeXY]];*)
    Label [BadLinkage];
    Solutions];

EightBarLinkageGeneratorv2[data_, SixRloop_, AFrames_] :=
  Block[{rem, AllDoubleRRConnections, Solutions,
     NoofLinkagesForOneType, SolTemp, SolutionsIndep, SolutionsDep},


    (*Print[
    "////////////////////   Synthesis Start   ///////////////////////"];*)


    rem = {{1, 0, 0}, {0, 1, 0}};
    AllDoubleRRConnections = {{{1, 3}, {1, 4}}, {{1, 3}, {1, 5}}, {{1,
        3}, {2, 4}}, {{1, 3}, {2, 5}}, {{1, 3}, {3, 5}}, {{1, 4}, {1,
        5}}, {{1, 4}, {2, 4}}, {{1, 4}, {2, 5}}, {{1, 4}, {3, 5}}, {{1,
      5}, {2, 4}}, {{1, 5}, {2, 5}}, {{1, 5}, {3, 5}}, {{2, 4}, {2,
     5}}, {{2, 4}, {3, 5}}, {{2, 5}, {3, 5}}, {{1, 4}, {1, 4}}, {{2,
      5}, {2, 5}}, {{1, 3}, {4, 6}}, {{1, 3}, {5, 6}}, {{1, 4}, {2,
     6}}, {{1, 4}, {3, 6}}, {{1, 4}, {5, 6}}, {{1, 5}, {2, 6}}, {{1,
      5}, {3, 6}}, {{1, 5}, {4, 6}}, {{2, 4}, {1, 6}}, {{2, 4}, {5,
     6}}, {{2, 5}, {1, 6}}, {{2, 5}, {3, 6}}, {{2, 5}, {4, 6}}, {{3,
      5}, {1, 6}}, {{3, 5}, {2, 6}}};
   Solutions = SolutionsIndep = SolutionsDep = {};
   NoofLinkagesForOneType = {};



   Table[
    (*Print["************************** Type no.",i,
    "****************************************"];
    Print["Independent Connections = ",AllDoubleRRConnections[[i]]];
    Print[
    "**********************************************************************
];*)
    SolTemp =
     IndependentConstraints[data, SixRloop, AFrames,
      AllDoubleRRConnections[[i]]];
    NoofLinkagesForOneType =
     Join[NoofLinkagesForOneType, {Length[SolTemp]}];
    (*Print["^^^^ NoofLinkagesForOneType = ",NoofLinkagesForOneType];*)

    If[Length[SolTemp] != 0,
     SolutionsIndep = Join[SolutionsIndep, SolTemp]];
    (*Print["^^^^ SolutionsIndep = ",MatrixForm[SolutionsIndep]];*)
    (*If[Length[SolTemp]\[NotEqual]0,
    Table[
    Print[LinkageDisplay[dataTemp,SolTemp[[j,1]],SolTemp[[j,3]],
    RangeXY]];
    ,{j,Length[SolTemp]}];
    ];*)
    (*Print[
    "************************************************************************
";
    Print[""];*)
    , {i, 1, 17}];



   (*Print[""];
   Print[""];*)


   Table[
    (*Print["*************************** Type no.",i,
    "    ********************************************"];
    Print["Dependent Connection = ",AllDoubleRRConnections[[i]]];
    Print[
    "************************************************************************
];*)
    SolTemp =
     DependentConstraints[data, SixRloop, AFrames,
        AllDoubleRRConnections[[i]]];
      (*Print["^^^^ SolTemp = ",SolTemp];*)
      NoofLinkagesForOneType =
       Join[NoofLinkagesForOneType, {Length[SolTemp]}];
      (*Print["^^^^ NoofLinkagesForOneType = ",NoofLinkagesForOneType];*)

      If[Length[SolTemp] != 0,
       SolutionsDep = Join[SolutionsDep, SolTemp];];
      (*Print["^^^^ SolutionsDep = ",MatrixForm[SolutionsDep]];*)
      (*If[Length[SolTemp]\[NotEqual]0,
      Table[
      Print[LinkageDisplay[dataTemp,SolTemp[[j,1]],SolTemp[[j,3]],
      RangeXY]];
      ,{j,Length[SolTemp]}];
      ];*)
      (*Print[
      "*****************************************************************
];
      Print[""];*)
      , {i, 18, 32}];




    Solutions = Join[SolutionsIndep, SolutionsDep];
    If[Solutions == {}, Solutions = 0(*NoofLinkagesForOneType=0*)];
    (*Print["Solutions len = ",Length[Solutions]];*)
```

```
  (*Print[
  "///////////////////  Synthesis End  //////////////////////"];*)
  {Solutions, NoofLinkagesForOneType}];

SynthesizingAllThePossibleLinkagesForEachLoop[data_, SixRloop_,
  Elbow3Rchain1_, Elbow3Rchain2_] :=
 Block[{rem, position, theta1, theta2, theta3, psi1, psi2, psi3,
   ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, A1, A2, A4, A5, c1Frame ,
   c6Frame , ptc3intFrame, ptc4intFrame, tFramewrtc3Frame,
   tFramewrtc4Frame, c3framestoc1frame, c4framestoc6frame, A1Frames,
   A2Frames, A3Frames, A4Frames, A5Frames, AFrames, Fiveptsc3,
   Fiveptsc4, LinkageSolutionsX, NoofLinkagesForOneType,
   LinkageSolutionsX2},

  rem = {{1, 0, 0}, {0, 1, 0}};

  position = Disp[#] & /@ data;

  {ptc1, ptc2, ptc3, ptc4, ptc5, ptc6} = SixRloop;
  A1 = LinkLength[ptc1, ptc2];
  A2 = LinkLength[ptc2, ptc3];
  A4 = LinkLength[ptc4, ptc5];
  A5 = LinkLength[ptc5, ptc6];

  c1Frame = Disp[Flatten[{0, ptc1}]]  ;
  c6Frame = Disp[Flatten[{0, ptc6}]];
  ptc3intFrame = rem.Inverse[position[[1]]].hom[ptc3];
```

```
ptc4intFrame = rem.Inverse[position[[1]]].hom[ptc4];
tFramewrtc3Frame = Disp[Flatten[{0, -ptc3intFrame}]];
tFramewrtc4Frame = Disp[Flatten[{0, -ptc4intFrame}]];
(*c3c4Frame=Disp[Flatten[{JointAngle[{1,0},ptc4-ptc3],ptc3}]];
tFrameinc3c4Frame=Inverse[c3c4Frame]. position[[1]];*)
c3framestoc1frame =
 Inverse[c1Frame].position[[#]].Inverse[tFramewrtc3Frame] & /@
  Range[5];
c4framestoc6frame =
 Inverse[c6Frame].position[[#]].Inverse[tFramewrtc4Frame] & /@
  Range[5];
{theta1, theta2, theta3} =
 N[RRRInverseKinematics[c3framestoc1frame , A1, A2, Elbow3Rchain1]];
{psi1, psi2, psi3} =
 N[RRRInverseKinematics[c4framestoc6frame , A5, A4, Elbow3Rchain2]];
Print["{theta1,theta2,theta3} = ", {theta1, theta2, theta3}/Degree];
Print["{psi1,psi2,psi3} = ", {psi1, psi2, psi3}/Degree];

A1Frames = c1Frame.Zmat[theta1[[#]]] & /@ Range[5];
A2Frames =
 c1Frame.Zmat[theta1[[#]]].Xmat[A1].Zmat[theta2[[#]]] & /@ Range[5];
A5Frames = c6Frame.Zmat[psi1[[#]]] & /@ Range[5];
A4Frames =
 c6Frame.Zmat[psi1[[#]]].Xmat[A5].Zmat[psi2[[#]]] & /@ Range[5];
Fiveptsc3 = rem.position[[#]].hom[ptc3intFrame] & /@ Range[5];
Fiveptsc4 = rem.position[[#]].hom[ptc4intFrame] & /@ Range[5];
A3Frames =
 Disp[Flatten[{JointAngle[{1, 0}, Fiveptsc4[[#]] − Fiveptsc3[[#]]],
       Fiveptsc3[[#]]}]] & /@ Range[5];
AFrames = {A1Frames, A2Frames, A3Frames, A4Frames, A5Frames, 0};

{LinkageSolutionsX, NoofLinkagesForOneType} =
 EightBarLinkageGeneratorv2[data, SixRloop, AFrames];
(*Print["Total no of linkages synthesized = ",Length[
LinkageSolutionsX]];*)
(*Print[
"Total no of linkages synthesized via nooflinofonetypes = ",Total[
NoofLinkagesForOneType]];*)

If[Length[LinkageSolutionsX] == 0,
 (*Print["WOW!!"];*)
 (*LinkageSolutionsX2=0*)
 ,
 LinkageSolutionsX =
  Flatten[{#, {data, theta1, theta2, theta3, psi1, psi2}}, 1] & /@
   LinkageSolutionsX
 (*Print["first  solution = ",LinkageSolutionsX[[1]]*)
 ];

(*Table[
If[Length[LinkageSolutionsX[[i]]]\[Equal]  {},Print["wow"]];
(*If[LinkageSolutionsX[[i]]\[NotEqual] 0,
LinkageSolutionsX[[i]]=Join[LinkageSolutionsX[[i]],{data,theta1}];
];*)
```

```
  ,{i,Length[LinkageSolutionsX]}];*)

  {LinkageSolutionsX, NoofLinkagesForOneType}];

  TranslatingMyAnglestoBriansAnglesandCheck5[theta1_, theta2_, theta3_,
 psi1_, psi2_, linkage_, thetas6and7_, RRconnections_, FTLA_, Subs_,
 J_] := Block[{thetas1fromX, thetas2fromX, thetas3fromX,
  thetas4fromX, thetas5fromX, thetas6fromX, thetas7fromX,
  thetas8fromX, ftla, requiredAngleList, currtemp, NextIteration,
  j8t1, j1t2, j2t3, j3t4, j4t5, j5t8, lnkList, lnkList2,
  SubsFixedAngles, lnkListRRConstraints, c1, c2, c3, c4, c5, c6, c7,
  c8, c9, c10, MyAnglesFive, BriansAnglesFive,
  OffsetListtoConvertMyAngles, th1, th2, th3, th4, th5, th6, th7,
  th8, JlistFiveTps, BriansAnglesFiveConfigs, lnkListReverse,
  lnkListRRConstraintsReverse, j2t1, j1t8, j3t2, j5t4, j4t3, j8t5,
  NextLink, SynDefectCheck},

 (*Print[
 "////////////////////  Five Tps check and Angle translation Start
\
//////////////////////////"];*)
 (*Print["Linkage = ",linkage];*)
 (*Print["RRconnections = ",RRconnections];*)
 {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10} = linkage;
 thetas8fromX = Table[N[JointAngle[{1, 0}, c1 - c6]], {i, 5}];
 (*Print["thetas8 = ",thetas8fromX];*)
```

```
  SubsFixedAngles = Subs[[2]];
  (*Print["SubsFixedAngles = ",SubsFixedAngles];*)

  {thetas1fromX, thetas2fromX, thetas3fromX, thetas4fromX,
    thetas5fromX, thetas6fromX, thetas7fromX} =
   Chop[Mod[{theta1, theta2 + theta1,
      theta3 + theta1 + theta2, (psi2 + psi1 + \[Pi]), (psi1 + \[Pi]),
       thetas6and7[[1]], thetas6and7[[2]]}, 2 \[Pi]]];

  MyAnglesFive = {thetas1fromX, thetas2fromX, thetas3fromX,
    thetas4fromX, thetas5fromX, thetas6fromX, thetas7fromX,
    thetas8fromX};
  (*Print[
  "MyAngles = \
{thetas1fromX,thetas2fromX,thetas3fromX,thetas4fromX,thetas5fromX,\
thetas6fromX,thetas7fromX,thetas8fromX} = ",MatrixForm[MyAnglesFive/
  Degree]];*)



  ftla = Flatten[FTLA, 1];
  requiredAngleList = ConstantArray[0, Length[ftla]];
  (*Flatten ftla and tabulate all the angles in all the loops in a \
list*)
  Table[
   requiredAngleList[[i]] = {ftla[[i, 1]], ftla[[i, 2]], ftla[[i, 4]]}
   , {i, Length[ftla]}];
  (*Print["requiredAngleList RAW = ",MatrixForm[
  requiredAngleList]];*)

  (*Remove duplicate angles in the list*)
  Table[
   currtemp = requiredAngleList[[i, 3]];
   If[currtemp == 0, Goto [NextIteration]];
   Table[
    If[requiredAngleList[[j, 3]] == currtemp,
      requiredAngleList[[j]] = {0, 0, 0}];
    (*requiredAngleList=Cases[requiredAngleList,Except[0]];*)
    , {j, i + 1, Length[requiredAngleList]}];
   Label[NextIteration];
   , {i, Length[requiredAngleList] - 1}];
  requiredAngleList = Cases[requiredAngleList, Except[{0, 0, 0}]];
  (*Print["requiredAngleList after removing duplicates = ",MatrixForm[
  requiredAngleList]];*)

  (*Make a list of links in terms of Brians joint notations to find \
Brians angles that correspond to my angle list {thetas1 ... thetas8}
 A new list of reverses are added to ensure that links mentioned in \
reverse are detected*)
  lnkListRRConstraints = {{Symbol[
      StringJoin[{"j", ToString[RRconnections[[1, 1]]], "t6"}]],
     Symbol[StringJoin[{"j6t",
        ToString[RRconnections[[1, 2]]]}]]}, {Symbol[
      StringJoin[{"j", ToString[RRconnections[[2, 1]]], "t7"}]],
     Symbol[StringJoin[{"j7t", ToString[RRconnections[[2, 2]]]}]]}};
  lnkListRRConstraintsReverse = {{Symbol[
      StringJoin[{"j", ToString[RRconnections[[1, 2]]], "t6"}]],
     Symbol[StringJoin[{"j6t",
        ToString[RRconnections[[1, 1]]]}]]}, {Symbol[
      StringJoin[{"j", ToString[RRconnections[[2, 2]]], "t7"}]],
     Symbol[StringJoin[{"j7t", ToString[RRconnections[[2, 1]]]}]]}};
  (*Print["lnkListRRConstraints = ",lnkListRRConstraints];*)
  lnkList =
   lnkList2 =
    Join[{{j8t1, j1t2}, {j1t2, j2t3}, {j2t3, j3t4}, {j3t4,
       j4t5}, {j4t5, j5t8}}, lnkListRRConstraints, {{j5t8, j8t1}}];
  lnkListReverse =
   Join[{{j2t1, j1t8}, {j3t2, j2t1}, {j4t3, j3t2}, {j5t4,
      j4t3}, {j8t5, j5t4}},
    lnkListRRConstraintsReverse, {{j1t8, j8t5}}];
  (*Print[
  "Links using Brian's conventions for which angles are to be found = \
",lnkList];
  Print["Links in reverse using Brian's conventions for which angles \
are to be found = ",lnkListReverse];*)




  Table[
   Table[
    If[lnkList[[i]] == Take[requiredAngleList[[j]], {1, 2}],(*Print[
     "found"];*)lnkList2[[i]] = requiredAngleList[[j, 3]];
     Goto [NextLink]];

    If[lnkListReverse[[i]] ==
       Take[requiredAngleList[[j]], {1, 2}],(*Print["Reverse found"];*)
     lnkList2[[i]] = requiredAngleList[[j, 3]] + \[Pi];
     Goto [NextLink]];

    , {j, Length[requiredAngleList]}];

   Label[NextLink];
   , {i, Length[lnkList]}];





  (*Print["lnkList2 before subs= ",MatrixForm[lnkList2]];*)
  lnkList2 = lnkList2 /. SubsFixedAngles;
  (*Print[
  "MyAngles {thetas1fromX,..., thetas8fromX} in terms of Brians \
angles = ",MatrixForm[lnkList2]];*)

  OffsetListtoConvertMyAngles = {th1, th2, th3, th4, th5, th6, th7,
     th8} - lnkList2;
  (*Print["OffsetListtoConvertMyAngles = ",
  OffsetListtoConvertMyAngles/Degree];*)
```

```
(*Table[
Print[Cases[lnkList2[[i]],_Number]];
(*Print[Select[lnkList2[[i]],NumberQ]];*)
(*If[NumberQ[lnkList2[[i]]]\[Equal]True,Print["Number is present"],
Print["No number"]];*)

,{i,Length[MyAnglesFive]}];*)



BriansAnglesFive = Table[
  Mod[(MyAnglesFive[[i]] + OffsetListtoConvertMyAngles[[i]]),
   2 \[Pi]]
  , {i, Length[MyAnglesFive]}];
(*Print[MatrixForm[BriansAnglesFive/Degree]];*)

BriansAnglesFiveConfigs =
 Chop[Table[BriansAnglesFive [[All, i]], {i, 5}]];
(*Print[MatrixForm[BriansAnglesFiveConfigs/Degree]];*)

JlistFiveTps = Table[
  Sign[J /.
    Thread[{th2, th3, th4, th5, th6, th7} ->
      Take[BriansAnglesFive [[All, i]], {2, 7}]]]
  , {i, 5}];
(*Print["Jlist sign for the five tps are ",MatrixForm[
JlistFiveTps]];*)
SynDefectCheck = If[Length[Union[JlistFiveTps]] == 1, 0, 1];
```

```
  (*If[SynDefectCheck\[Equal]0,Print["^^^^ No Synthesis defect"],
  Print["^^^^ Synthesis defect"]];*)



  (*Print[
  "/////////////////// Five Tps check and Angle translation End \
///////////////////////"];*)
  {SynDefectCheck, BriansAnglesFiveConfigs}]

  (*Function to find the angle the isotropic vector makes with the \
horizontal. If the vector is imaginary then return i*)
ConvtoReal[x_] := Module[{px, py, angle, End1},
   If[NumberQ[x] == False || N[Abs[Norm[x] - 1]] > 10^-3,
    angle = I; Goto End1,
    px = Re[x]; py = Im[x];
    (*Print["Norm ",Norm[x]];*)
    angle = Mod[Chop[JointAngle[{1, 0}, {px, py}], 10^-4], 2 \[Pi]]];

   Label End1;
   angle];

ConditionOpAnglesEightbarNEW[kvals_, kvecs_, opAngle_, depAngles_] :=
  Module[{k, end, theta2, theta3, theta4, theta5, theta6, theta7, i,
    len, Solution, Sol, theta22, theta23, theta24, n},
   (*Filtering solutions for complex infinity and complex nos*)
   len = 20;
   k = 1;
```

```
Solution = ConstantArray[{0, 0, 0, 0, 0, 0, 0, 0}, len];
(*Print["kvals total ",kvals];Print[""];
Print["kvecs total ",kvecs];Print[""];*)
Table[
 n = 1;
 (*Print["kvals = ",kvals[[i]],"      Norm = ",Norm[kvals[[i]]]];*)
 (*NumberQ[x]\[Equal] False||N[Abs[Norm[x]- 1]]>10^-3*)
 If[NumberQ[kvals[[i]]] == False ||
   N[Abs[Norm[kvals[[i]]] - 1]] > 10^-3, Goto [end]];
 Solution[[i, opAngle]] = N[ConvtoReal[kvals[[i]]]];



 If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
  "WARNING!! dividing by zero"];*)
  Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
 Solution[[i, depAngles[[n, 1]]]] =
  N[ConvtoReal[
     kvecs[[i, depAngles[[n, 2, 1]]]]/
      kvecs[[i, depAngles[[n, 2, 2]]]]]];
 If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
  "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
  Goto [end]];
 (*If[Im[Solution[[i,depAngles[[n,1]]]]]\[NotEqual]0,Print[
 "WARNING",Im[Solution[[i,depAngles[[n,1]]]]]];Solution[[i]]=0;
 Goto [end]];*)
 n++;
```

```
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;
```

```
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;
```

```
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;


If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
```

```
(*Print["Solution for ",i, " iteration in degrees = ",Solution[[
i]]/Degree];
Print["Solution for ",i, " iteration in radians = ",Solution[[
i]]];*)
(*Solution[[i,]]=N[ConvtoReal[kvecs[[i,12]]/kvecs[[i,10]]]];
(*If[theta4\[Equal]\[ImaginaryI],theta4\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,10]]]];
(*If[theta5\[Equal]\[ImaginaryI],theta5\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,9]]]];
(*If[theta8\[Equal]\[ImaginaryI],theta8\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,8]]]];*)
(*If[theta10\[Equal]\[ImaginaryI],theta10\[Equal]10000;Goto [
end]];*)



(*Print["{Theta2,Theta3,Theta4,Theta5,Theta6,Theta7} = ",{Mod[
2\[Pi]-theta2,2\[Pi]],Mod[theta3,2\[Pi]],Mod[2\[Pi]-theta4,
2\[Pi]],Mod[2\[Pi]-theta5,2\[Pi]],Mod[2\[Pi]-theta6,2\[Pi]],Mod[
2\[Pi]-theta7,2\[Pi]]}/Degree];*)

(*Print[" "];*)
(*
thetaRest=ConvtoReal[#]&/@kvals[[i]];Print[thetaRest];*)
```

```
    (*Solution[[k]]={Mod[2\[Pi]-theta2,2\[Pi]],Mod[theta3,2\[Pi]],Mod[
    2\[Pi]-theta4,2\[Pi]],Mod[2\[Pi]-theta5,2\[Pi]],Mod[2\[Pi]-theta6,
    2\[Pi]],Mod[2\[Pi]-theta7,2\[Pi]]};
    k++;
    *)
    Label[end];


    , {i, Length[kvals]}];


   Sol = Chop[ Cases[Solution, Except[{0, 0, 0, 0, 0, 0, 0, 0}]]];
   Sol];


ForwardKinematicsEightBar[ipAngles_, groundAngle_, Mmat_, Nmat_,
   Tmat_, opAngEigVal_, depAngles_, depAnglesTmatRatios_, FTLA_,
   Subs_, AdjMatWithJointN_] :=
  Block[{SolutionBox, FTLA2, BrianIpAngle, FixedOffsetforIpAngle, th1,
     ipAnglesOffset, subsLinklength, subsFixedAngles,
    subsAnglesinFirstPosition, StartingSolAnglesTemp,
    StartingSolAngles, opAngEigValno, depAnglesno,
    ipSubs, \[Theta]1, \[Theta]1c, kvals, kvecs, subsNM,
    SolutionBoxShortened,
    BadLinkage, \[Theta]2, \[Theta]3, \[Theta]4, \[Theta]5, \
\[Theta]6, \[Theta]7, \[Theta]8(*,MReal,
    NReal,(*\[Theta]2,\[Theta]3,\[Theta]4,\[Theta]5,\[Theta]6,\[Theta]\
7,\[Theta]8,*)S*)},


   (*Print[
   "//////////////////// Forward Kinematics Start \
//////////////////////////";*)
   (*ClearAll[\[Theta]2,\[Theta]3,\[Theta]4,\[Theta]5,\[Theta]6,\
\[Theta]7,\[Theta]8];*)
   SolutionBox =
    SolutionBoxShortened = ConstantArray[0, Length[ipAngles]];
   subsLinklength = Subs[[1]];
   subsFixedAngles = Subs[[2]];
   subsAnglesinFirstPosition = Subs[[3]];
   (*Print["subsLinklength = ",subsLinklength]*);
   (*Print["subsFixedAngles = ",subsFixedAngles];
   Print["subsAnglesinFirstPosition = ",
   subsAnglesinFirstPosition];*)

   (*Find what is the angle C1C2 or link1 angle (input angle) using \
Brian's notation*)
   BrianIpAngle = 0;
   (*Print["FTLA = ",FTLA];*)
   FTLA2 = Flatten[FTLA, 1];
   Table[
    If[StringMatchQ[ToString[FTLA2[[j, 1]]], "j8t1"] == True &&
       StringMatchQ[ToString[FTLA2[[j, 2]]], "j1t2"] == True,
      BrianIpAngle = FTLA2[[j, 4]]];
    , {j, Length[FTLA2]}];
   (*Print["Brian Angle = ",BrianIpAngle];*)
   FixedOffsetforIpAngle = BrianIpAngle - th1;
   FixedOffsetforIpAngle = FixedOffsetforIpAngle /. subsFixedAngles;
```

```
    (*Print["FixedOffsetforIpAngle = ",FixedOffsetforIpAngle/
    Degree];*)

    (*Since ipangle = th1+fixedangle,therefore th1 = ipangle-
    fixedangle.
    Thus the ipAngles have to be subtracted by this fixedangle to get \
the correct ipangles that is compatible with Brian's code input*)
    (*Print["ipAngles = ",ipAngles];*)
    ipAnglesOffset =
     Table[Mod[ipAngles[[j]] - FixedOffsetforIpAngle, 2 \[Pi]], {j,
       Length[ipAngles]}];
    (*Print["ipAnglesOffset = ",ipAnglesOffset/Degree];*)

    (*Compiling a starting angle list for Post Processing*)
    StartingSolAnglesTemp = {\[Theta]2, \[Theta]3, \[Theta]4, \
\[Theta]5, \[Theta]6, \[Theta]7} /. subsAnglesinFirstPosition;
    StartingSolAngles =
     Table[ConvtoReal[StartingSolAnglesTemp[[i]]], {i,
       Length[StartingSolAnglesTemp]}];
    (*Print["Starting Angles = ",StartingSolAngles/Degree];*)
    (*StartingSolAngles=Join[{ipAnglesOffset[[1]]},
    StartingSolAngles];*)
    (*Print["Starting Angles = ",StartingSolAngles/Degree];*)

    (*Creating two variables to guide the eigensystem store the eigen \
values and the eigen vectors in appropriate locations in the list {1,
    2,3,4,5,6,7,8} *)
```

```
   opAngEigValno =
    ToExpression[
     StringCases[ToString[opAngEigVal], RegularExpression["\\d"]]];
   depAnglesno =
    Table[{ToExpression[
        StringCases[ToString[depAngles[[i]]],
         RegularExpression["\\d"]]], depAnglesTmatRatios[[i]]}, {i, 5}];
   (*Print["opAngEigValno = ",opAngEigValno];
   Print["depAnglesno = ",depAnglesno];*)

   (*Substituting the ground angle in the two matricres Mmat and Nmat*)
\

   (*MReal=Mmat/.{\[Theta]8\[Rule]
   Exp[\[ImaginaryI]  groundAngle],\[Theta]8c\[Rule]
   Exp[-\[ImaginaryI] groundAngle]};
   NReal=Nmat/.{\[Theta]8\[Rule]
   Exp[\[ImaginaryI]  groundAngle],\[Theta]8c\[Rule]
   Exp[-\[ImaginaryI] groundAngle]};*)
   (*Print["ipAngles  = ",ipAngles/Degree];*)

   (*Finding all possible solutions and capturing them in the \
Solution Box for every input angle.
   The first solution should include the starting angle solution that \
is the list of angles of the links in the starting configuration in \
which it is synthesized*)
   (*Print["Mmat ",MatrixForm[Mmat]];
   Print["Nmat ",MatrixForm[Nmat]];*)
   Table[
    ipSubs = {\[Theta]1 -> Exp[I ipAnglesOffset[[i]]], \[Theta]1c ->
       Exp[-I ipAnglesOffset[[i]]]};
    (*Print[MatrixForm[Nmat]];
    Print[MatrixForm[Mmat]];*)
    {kvals, kvecs} = Chop[Eigensystem[{-Nmat, Mmat} /. ipSubs], 10^-3];
    (*Print["kvals = ",kvals];*)
    (*Print["kvecs = ",kvecs];*)

    SolutionBox[[i]] =
     ConditionOpAnglesEightbarNEW[kvals, kvecs, opAngEigValno,
      depAnglesno];
    SolutionBoxShortened[[i]] = Take[#, {2, 7}] & /@ SolutionBox[[i]];
    (*Print[""];
    Print["Input angle for this solution = ",ipAnglesOffset[[i]]/
    Degree];
    Print["Solution no ",i," = ",MatrixForm[SolutionBoxShortened[[i]]/
    Degree]];*)
    If[SolutionBox[[i]] == {}, SolutionBox = 0;
     SolutionBoxShortened = 0; Goto [BadLinkage](*Print[
     "WARNING!!!!"]*)];
    (*Print["Solution no ",i," = ",SolutionBox[[i]]];*)

    , {i, Length[ipAngles]}];
```

```
    Label[BadLinkage];
    (*Print[
    "///////////////////   Forward Kinematics End  \
///////////////////////"];*)

    {BrianIpAngle, FixedOffsetforIpAngle, ipAnglesOffset,
     StartingSolAngles, SolutionBox, SolutionBoxShortened}];

    (*A general module for sorting the solutions to the forward \
kinematics into trajectories*)
SortingA[(*dim_,*)FKeqns_, input_, FKsolns_] :=
 Module[{x, y, yV, F, J, br, bran, Its, tol,(*log,*)i, j, k, ycur,
    ToBeAdded, Added, nmatch, NotAdded, ToBeJoined, branches,
    NotJoined},

   Off[ReplaceAll::reps];
   (*dim- includes linkage's dimensions as substitutions*)
   (*FKeqns- the fwd kin eqns written symbolically*)
   (*input- the input variable written as substitutions*)
   (*FKsolns- fwd kin solutions indexed by position, solution*)
   x = input;(*input values indexed by position*)
   y = FKsolns;(*output values indexed by position*)
   yV = FKsolns[[1, 1, All, 1]];(*a vector of the output symbols*)
   F = FKeqns(*/.dim*);(*the fwd kin eqns with the input and outputs \
left symbolic*)
   J = D[F, {yV}];(*Jacobian of F with the input and outputs left \
symbolic*)
```

```
br = {{x[[1]], #}} & /@ y[[1]];(*the active branches:
indices are branch, position, (1-input,2-output) *)
bran = {};(*the completed branches: same indices*)
Its = 8;(*Newton iterations*)
tol = 10^-2;(*tolerance when comparing numbers*)
log = {};
For[i = 2, i <= Length[x], i++,(*i is the current position*)
 ycur = br[[All, -1, 2]];
 Do[ycur = (yV /. x[[i]] /. #) -
      Inverse[J /. x[[i]] /. #].(F /. x[[i]] /. #) & /@ ycur;
  ycur = Thread[yV -> #] & /@ ycur, {Its}];
 (*ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@
 ycur[[All, All,2]];*)
 (*Print["ip, ycur = ",{x[[i]],ycur}];*)
 (*Print["input angle ",x[[i]]," FK solns from Dixon ",y[[i,All,
 All,2]]," FK solns from Newton ",ycur[[All, All,2]]];*)
 (*ToBeAdded=Position[y[[i,All,All,2]],x_/;(Norm[x-#]<tol),1,
 Heads\[Rule]False]&/@ycur[[All, All,2]];*)
 ToBeAdded =
  Position[Map[Exp[I*#] &, y[[i, All, All, 2]], {2}],
     x_ /; (Norm[x - #] < tol), 1, Heads -> False] & /@
   Map[Exp[I*#] &, ycur[[All, All, 2]], {2}];
 (*Print[ToBeAdded];*)
 (*ToBeAdded=Position[Round[Map[Exp[I*#]&,y[[i,All,All,2]],{2}],
 tol],Round[#,tol]]&/@Map[Exp[I*#]&,ycur[[All, All,2]],{2}];*)
 (*ToBeAdded=Position[Round[Map[Exp[I*#]^2&,y[[i,All,All,2]],1],
 tol],Round[#,tol]]&/@Map[Exp[I*#]^2&,ycur[[All, All,2]],1];*)
```

```
ToBeAdded = ToBeAdded[[All, All, 1]];
Added = {};
For[j = Length[br], j >= 1, j = j - 1,(*j is the current branch*)
 nmatch = Length[ToBeAdded[[j]]];
 Which[nmatch == 1,
  AppendTo[br[[j]], {x[[i]], y[[i, ToBeAdded[[j, 1]]]]}];
  AppendTo[Added, ToBeAdded[[j, 1]]],
  nmatch == 0,
  AppendTo[bran, br[[j]]];
  br = Delete[br, j];
  AppendTo[log,
   "Path ended at " ~~ x[[i]] ~~ " where Det[J] = " ~~
    ToString[Det[J /. x[[i]] /. ycur[[j]]]] ~~ "."],
  nmatch > 1,
  Do[br = Insert[br, br[[j]], j], {nmatch - 1}];
  Do[AppendTo[
    br[[j - 1 + k]], {x[[i]], y[[i, ToBeAdded[[j, k]]]]}], {k,
    nmatch}];
  Added = Join[Added, ToBeAdded[[j]]];
  AppendTo[log,
   "Paths may be splitting at " ~~ x[[i]] ~~ " where Det[J] = " ~~
    ToString[Det[J /. x[[i]] /. ycur[[j]]]]]
  ]];
NotAdded = Complement[Range[Length[y[[i]]]], Added];
If[Length[NotAdded] > 0,
 AppendTo[log,
  "At " ~~ x[[i]] ~~ ", there was " ~~ ToString[Length[NotAdded]] ~~
```

```
       " new branch(es) added."]];
    Do[AppendTo[br, {{x[[i]], y[[i, k]]}}], {k, NotAdded}];
    ];
   bran = Map[Flatten, bran, {2}];
   br = Map[Flatten, br, {2}];
   (*End of main sorting*)
   ToBeJoined = Position[bran[[All, 1]], #] & /@ br[[All, -1]];
   ToBeJoined = ToBeJoined[[All, All, 1]];
   branches = {};
   For[j = 1, j <= Length[br], j++,
    nmatch = Length[ToBeJoined[[j]]];
    Which[nmatch == 0,
     AppendTo[branches, br[[j]]],
     nmatch > 0,
     Do[AppendTo[branches, Join[Most[br[[j]]], bran[[k]]]], {k,
       ToBeJoined[[j]]}]
     ]];
   NotJoined = Complement[Range[Length[bran]], Flatten[ToBeJoined]];
   Do[AppendTo[branches, bran[[k]]], {k, NotJoined}];
   branches]

RemoveDuplicateBranches[list_] :=
  Block[{i, j, intersection, temp, listjshort, SortlistSecondtoLast,
    unionijLen, list2},
   list2 = list;

   Table[
(*Print[];
Print["######## i = ",i];*)



Table[
 (*Print["{list i, list j} = ",{Length[list2[[i]]],Length[list2[[
 j]]]}];*)
 unionijLen = Length[Union[list2[[i]], list2[[j]]]];
 (*Print["length of Union of i and j = ",unionijLen];*)

 If[unionijLen < Length[list2[[i]]] + Length[list2[[j]]],
  (*Print["j branch is dependent"];*)
  listjshort = list2[[j]];
  intersection = Intersection[list2[[i]], list2[[j]]];
  Table[
   temp = DeleteCases[listjshort , intersection[[j]]];
   listjshort = temp, {j, Length[intersection]}];
  (*Print["new length of list j = ",Length[listjshort]];*)
  list2[[j]] = listjshort;
  (*,
  Print["j branch is independent"]*)
  ];

 (*Print[];*)
 , {j, i + 1, Length[list2]}];

(*Print["before sorting list2 has branches len = ",Length/@list2];*)
```

```
   SortlistSecondtoLast =
    Sort[list2[[i + 1 ;; Length[list2]]], Length[#1] > Length[#2] &];
   list2 = Join[list2[[1 ;; i]], SortlistSecondtoLast];
   (*Print["Final length of branches sorted = ",Length/@list2];*)
   , {i, Length[list2] - 1}];

  list2 = Cases[list2, Except[{}]];
  list2]

SortingTrajectories[ipAnglesOffset_, SolutionBoxShortened_,
   FiveBrianTpsConfigAngles_, FTLA_, Subs_, groundAngle_,
   FiveTpsLocs_, rangeOfMotionforLink1_] :=
  Block[{ftla, SubsNM, loop1x, loop1y, loop2x, loop2y, loop3x, loop3y,
     F, th1, th2, th3, th4, th5, th6, th7, th8, ipAnglesOffsetSubs,
    SolutionBoxShortenedSubs, BranchesRAW, SortedBranchesRAW, kunion,
    NextListMember, SortedBranchesEXTRACT, thetasAngTraj, ColorList,
    TpsVisualize, BranchesFull, SortedTrajectory, Problem, BranchSols,
     pNorm, PUNorm, rad, groundAngleN, rangeIpOnCylinder, tubeRange},
   (*Print[
   "////////////////////   Sorting Start   ///////////////////////"];*)

   ftla = FTLA;
   SubsNM = Flatten[{Subs[[1]], Subs[[2]]}];

   (*loop equations*)
   loop1x =
 Total[Table[{ftla[[1, i, 3]] Cos[ftla[[1, i, 4]]]}, {i,
     Length[ftla[[1]]]}]] /. SubsNM;
loop1y =
 Total[Table[{ftla[[1, i, 3]] Sin[ftla[[1, i, 4]]]}, {i,
     Length[ftla[[1]]]}]] /. SubsNM;
loop2x =
 Total[Table[{ftla[[2, i, 3]] Cos[ftla[[2, i, 4]]]}, {i,
     Length[ftla[[2]]]}]] /. SubsNM;
loop2y =
 Total[Table[{ftla[[2, i, 3]] Sin[ftla[[2, i, 4]]]}, {i,
     Length[ftla[[2]]]}]] /. SubsNM;
loop3x =
 Total[Table[{ftla[[3, i, 3]] Cos[ftla[[3, i, 4]]]}, {i,
     Length[ftla[[3]]]}]] /. SubsNM;
loop3y =
 Total[Table[{ftla[[3, i, 3]] Sin[ftla[[3, i, 4]]]}, {i,
     Length[ftla[[3]]]}]] /. SubsNM;
F = {loop1x, loop1y, loop2x, loop2y, loop3x, loop3y} /.
  th8 -> groundAngle;
(*Print[F];*)




(*ipangleswithOffset as subs*)
ipAnglesOffsetSubs = th1 -> # & /@ ipAnglesOffset;
(*FKsols as subs*)
```

```
SolutionBoxShortenedSubs =
 ConstantArray[0, Length[SolutionBoxShortened]];
Table[
 SolutionBoxShortenedSubs[[i]] =
   Thread[{th2, th3, th4, th5, th6, th7} -> #] & /@
    SolutionBoxShortened[[i]];
 , {i, Length[SolutionBoxShortened]}];




(*Calling Marks sorting function and Sort the branches based on \
size*)
BranchesRAW =
 SortingA[Flatten[F], ipAnglesOffsetSubs, SolutionBoxShortenedSubs];
If[Length[BranchesRAW] > 16,(*Print[
 "No of branches found are greater than 16"];*)
 SortedTrajectory = 0; Goto [Problem]];
SortedBranchesRAW = Sort[BranchesRAW, Length[#1] > Length[#2] &];
(*Print["Length of ipanglesoffset = ",Length[ipAnglesOffset]];
Print["Lengths of the branches before removing duplicates = ",
Length/@SortedBranchesRAW];*)




(*Remove duplicates from branches*)
SortedBranchesRAW = RemoveDuplicateBranches[SortedBranchesRAW];
```

```
   (*Print["Lengths of the branches found after removing dupls = ",
   Length/@SortedBranchesRAW];*)




   (*Extracting angles from branches subs list SortedBranchesRAW and \
storing them in SortedBranchesEXTRACT*)
   SortedBranchesEXTRACT = ConstantArray[0, Length[SortedBranchesRAW]];
   groundAngleN = N[groundAngle];
   Table[SortedBranchesEXTRACT[[
       i]] = ({th1, th2, th3, th4, th5, th6, th7, th8} /.
         SortedBranchesRAW[[i]]) /. th8 -> groundAngleN;
     , {i, Length[SortedBranchesRAW]}];




   (*Plotting all the branches with the plots {theta2,...,
   theta7} vs ipAngle and also plotting the tps as 5 points for all \
angle {theta2,...,theta7}*)
   rad = (3/8)*rangeOfMotionforLink1/Degree;
   thetasAngTraj =
    ConstantArray[{0, 0, 0, 0, 0, 0}, Length[SortedBranchesEXTRACT]];
   Table[
    Table[
      thetasAngTraj[[i, j]] =
    MapThread[{#1, #2} &, {SortedBranchesEXTRACT[[i, All, 1]],
       SortedBranchesEXTRACT[[i, All, j + 1]]}]
   , {j, 6}];
 , {i, Length[SortedBranchesEXTRACT]}];

TpsVisualize =
 Table[
  MapThread[{#1, #2} &, {FiveBrianTpsConfigAngles[[All, 1]],
     FiveBrianTpsConfigAngles[[All, j]]}]/Degree
  , {j, 2, 7}];

ColorList = {Black, Blue, Green, Magenta, Gray, Brown, Purple,
  Orange, Yellow, Cyan, Pink, Lighter[Green], Lighter[Cyan],
  Lighter[Purple], Lighter[Yellow], Lighter[Gray], Lighter[Brown]};

(*Main display of different trajectories uncomment later*)
(*Print[
Table[
Graphics3D[{
Inset[Style[i+1,20],{10,rad,rad}],
CapForm[None],Blue,Opacity[0.1],Tube[{{(ipAnglesOffset[[1]]/
Degree),0,0},{Last[ipAnglesOffset]/Degree,0,0}},rad],Opacity[
1],(*Line[{{(ipAnglesOffset[[1]]/Degree),0,0},{(Last[
ipAnglesOffset]/Degree),0,0}}],*)
Thickness[0.005],
{ColorList[[#]],Line[{MakeCylindrical[thetasAngTraj[[#,i]]/Degree,
rad]}]}&/@ Range[1,Length[SortedBranchesEXTRACT]],
    PointSize[0.015],Darker[Red],Point[MakeCylindrical[TpsVisualize[[
    i]],rad]]

    },
    AspectRatio\[Rule] Automatic,PlotRange\[Rule]{{(ipAnglesOffset[[
    1]]/Degree)-5,Last[ipAnglesOffset]/Degree+5},{-rad,rad},{-rad,
    rad}},Axes\[Rule]True,AxesLabel\[Rule]{X,Y,Z},
    SphericalRegion\[Rule] True,Boxed\[Rule]False,ImageSize\[Rule]300(*,
    ViewVertical\[Rule]{0.5,-3,1.5}*)]
    ,{i,1,6}]];*)



    (*Finding the branches that go from start to finish*)
    BranchesFull = {};
    Table[
     If[Length[SortedBranchesEXTRACT[[i]]] == Length[ipAnglesOffset],
        BranchesFull = Append[BranchesFull, SortedBranchesEXTRACT[[i]]]];
     , {i, Length[SortedBranchesEXTRACT]}];
    If[Length[BranchesFull] == 0,(*Print["No full branches found"];*)
     SortedTrajectory = 0; Goto [Problem]];



    (* Now check if the five tps lie on the full branches(usually one) \
found *)
    BranchSols = {};
```

```
Table[
 (*Print["BranchFULL ",i];*)
 pNorm =
  Table[NormofDiffBetAngVectors[FiveBrianTpsConfigAngles[[j]],
    BranchesFull[[i, FiveTpsLocs[[j]]]]], {j, 5}];
 PUNorm = Norm[pNorm];
 (*Print["pNorm = ",pNorm];
 Print["PUNorm = ",PUNorm];*)
 If[PUNorm < taskTolerance,
  BranchSols = Append[BranchSols, BranchesFull[[i]]]];
 , {i, 1, Length[BranchesFull]}];
(*Print["No of BranchFull that have the five tps on it = ",Length[
BranchSols]];*)
If[Length[BranchSols] == 0, SortedTrajectory = 0; Goto [Problem]];
SortedTrajectory = BranchSols[[1]];
(*Print["Sorted Traj ",MatrixForm[SortedTrajectory/Degree]];*)



(*
(*Main display of different trajectories uncomment later*)
If[Abs[ipAnglesOffset[[1]]-Last[ipAnglesOffset]]/Degree>
rangeOfMotionforLink1/Degree,rangeIpOnCylinder={0,360};rad=135;
tubeRange={{0,0,0},{360,0,0}},rangeIpOnCylinder={(ipAnglesOffset[[
1]]/Degree),Last[ipAnglesOffset]/Degree};
tubeRange={{(ipAnglesOffset[[1]]/Degree),0,0},{Last[
ipAnglesOffset]/Degree,0,0}}];
Print[
Table[
Graphics3D[{
Inset[Style[i+1,20],{10,rad,rad}],
CapForm[None],Blue,Opacity[0.1],Tube[tubeRange,rad],Opacity[1],
(*CapForm[None],Blue,Opacity[0.1],Tube[{{(ipAnglesOffset[[1]]/
Degree),0,0},{Last[ipAnglesOffset]/Degree,0,0}},rad],*)(*Opacity[
1],*)(*Line[{{(ipAnglesOffset[[1]]/Degree),0,0},{(Last[
ipAnglesOffset]/Degree),0,0}}],*)
Thickness[0.005],
{ColorList[[#]],Line[{MakeCylindrical[thetasAngTraj[[#,i]]/Degree,
rad]}]}&/@ Range[1,Length[SortedBranchesEXTRACT]],
PointSize[0.015],Darker[Red],Point[MakeCylindrical[TpsVisualize[[
i]],rad]]

},
AspectRatio\[Rule] Automatic,
PlotRange\[Rule]{rangeIpOnCylinder,{-rad,rad},{-rad,
rad}},(*{{(ipAnglesOffset[[1]]/Degree)-5,Last[ipAnglesOffset]/
Degree+5},{-rad,rad},{-rad,rad}}*)Axes\[Rule]True,
AxesLabel\[Rule]{X,Y,Z},SphericalRegion\[Rule] True,
Boxed\[Rule]False,ImageSize\[Rule]300(*,ViewVertical\[Rule]{0.5,-3,
1.5}*)]
,{i,1,6}]];



*)
```

```
   Label [Problem];
   (*Print[
   "///////////////////   Sorting End   //////////////////////"];*)
   SortedTrajectory];

   DisplayTP[data_, sc_, color_, color2_, colordot_] :=
 Module[{position, orig, ex, ey, frame, rem},
  rem = {{1, 0, 0}, {0, 1, 0}};
  position = Table[Disp[data[[i]]], {i, 5}];
  orig = Table[rem.position[[i]].{0, 0, 1}, {i, 5}];
  ex = Table[rem.position[[i]].{sc, 0, 1}, {i, 5}];
  ey = Table[rem.position[[i]].{0, sc, 1}, {i, 5}];
  frame =
   Table[{Thickness[0.005], color, Line[{orig[[i]], ey[[i]]}],
     color2, Line[{orig[[i]], ex[[i]]}], colordot, PointSize[0.007],
     Point[orig[[i]]]}, {i, 5}];
  {frame}]

(*DisplayTP2[position_,sc_, \
color_,color2_,colordot_]:=Module[{orig,ex,ey,frame,npos,rem},
npos−5;
rem = {{1,0,0},{0,1,0}};
orig=Table[rem.position[[i]].{0,0,1}, {i,npos}];
ex=Table[rem.position[[i]].{sc,0,1},{i,npos}];
ey=Table[rem.position[[i]].{0,sc,1},{i,npos}];
frame=Table[{Thickness[0.0007*sc],color,Line[{orig[[i]],ey[[i]]}],\
color2 ,  Line[{orig[[i]],ex[[i]]}],colordot,PointSize[0.001*sc],Point[\
orig[[i]]]},{i,npos}];
{frame}]*)

(*FindScaleforTPdisplay[data_,defaultScaler_]:=Module[{LDist,npos,\
maxDist,Scaler},
npos=5;
LDist=Table[LinkLength[{data[[1,2]],data[[1,3]]},{data[[i,2]],data[[i,\
3]]}],{i,2,npos}];
maxDist = Max[LDist];
Scaler=maxDist/5;
If[Scaler\[LessEqual]defaultScaler,Scaler=defaultScaler];
Scaler]*)

ClippingRange[data_, scale_] :=
 Module[{MinX, MaxX, MinY, MaxY, xdiff, ydiff, CorrectionFactor,
   maxdiff, RangeX, offset},
  MinX = Min[Table[data[[i, 2]], {i, 5}]];
  MaxX = Max[Table[data[[i, 2]], {i, 5}]];
  MinY = Min[Table[data[[i, 3]], {i, 5}]];
  MaxY = Max[Table[data[[i, 3]], {i, 5}]];

  xdiff = Abs[MaxX - MinX];
  ydiff = Abs[MaxY - MinY];
  maxdiff = Max[Abs[{xdiff, ydiff}]];

  CorrectionFactor = 0.5*Abs[xdiff - ydiff];
  If[xdiff > ydiff,
   MaxY += CorrectionFactor; MinY -= CorrectionFactor;,
   MaxX += CorrectionFactor; MinX -= CorrectionFactor;
    ];

  offset = maxdiff*(scale - 1);
  RangeX = {{MinX - offset, MaxX + offset}, {MinY - offset,
     MaxY + offset}};
  RangeX]

LinkageDisplay[data_, linkage_, linkConnections_, RangeXY_] :=
 Module[{ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, ptc7, ptc8, ptc9, ptc10,
   DisplayLinkage, link1, link2, link3, link4, FirstTernaryLink,
   SecondTernaryLink, ThirdTernaryLink, FourthTernaryLink,
   TpOriginalDisplay},

  {ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, ptc7, ptc8, ptc9, ptc10} =
   linkage;
  (*Print[linkConnections];*)
  TpOriginalDisplay =
   DisplayTP[data, 4, Darker[Green], Darker[Red], Pink];
  link1 = linkConnections[[1, 1]];
  link2 = linkConnections[[1, 2]];
  FirstTernaryLink = {linkage[[link1]], linkage[[7]],
    linkage[[link1 + 1]]};
  SecondTernaryLink = {linkage[[link2]], linkage[[8]],
    linkage[[link2 + 1]]};
```

```
link3 = linkConnections[[2, 1]];
link4 = linkConnections[[2, 2]];
ThirdTernaryLink = {linkage[[link3]], linkage[[9]],
  linkage[[link3 + 1]]};
FourthTernaryLink = {linkage[[link4]], linkage[[10]],
  linkage[[link4 + 1]]};

DisplayLinkage = Graphics[{
   Thickness[0.007],
   Lighter[Blue], Line[{ptc1, ptc2}], Darker[Blue],
   Line[{ptc2, ptc3}], PointSize[0.015], Point[{ptc2}],
   Lighter[Yellow], Line[{ptc5, ptc6}], Darker[Yellow],
   Line[{ptc4, ptc5}], Point[{ptc5}],

   Opacity[0.3],
   Darker[Gray], Polygon[FirstTernaryLink],
   Polygon[SecondTernaryLink],
   Darker[Red], Polygon[ThirdTernaryLink],
   Polygon[FourthTernaryLink],
   Opacity[1],

   Darker[Gray], Line[{linkage[[7]], linkage[[8]]}],
   PointSize[0.015], Point[{linkage[[7]], linkage[[8]]}],
   Darker[Red], Line[{linkage[[9]], linkage[[10]]}],
   PointSize[0.015], Point[{linkage[[9]], linkage[[10]]}],
      Black, Line[{ptc1, ptc6}], PointSize[0.015], Point[{ptc1, ptc6}],
      Darker[Green], Line[{ptc3, ptc4}], Point[{ptc3, ptc4}],
      TpOriginalDisplay



      }, Axes -> True, AspectRatio -> Automatic, PlotRange -> RangeXY,
     ImageSize -> 500];

   DisplayLinkage]

FindExpressionForVertices[ModAdjMatrixSym_, AdjMatWithJointN_,
    FTLA_] :=
   Block[{klist1, klist2, kcount, j8t1, j5t8, compTempVal, locs,
     Verticesatlocs, VerticesNos, VerticesOrder, FinalVerticesSol,
     klistPrev},

    FinalVerticesSol = 0;
    klistPrev = 0;
    klist1 = klist2 = ConstantArray[0, 100];
    kcount = 1;

    (*Print["FTLA = ",MatrixForm[FTLA]];
    Print[""];*)



    (*Find expression of all the joints encountered in the FTLA,
    klist1 is a list of all the joint with repetitions
```

```
klist2 gives expression for each joint corr to klist1 *)
Table[
 Table[
  klist1[[kcount]] = FTLA[[i, j, 2]];
  klist2[[kcount]] =
   klistPrev +
    FTLA[[i, j, 3]]*{Cos[FTLA[[i, j, 4]]], Sin[FTLA[[i, j, 4]]]};
  klistPrev = klist2[[kcount]];
  (*Print["klist2 curr = ",klist2[[kcount]]];*)
  kcount++;
  , {j, Length[FTLA[[i]]]}];
 klistPrev = 0;
 , {i, 3}];

klist1 = Cases[klist1, Except[0]];
klist2 = Cases[klist2, Except[0]];
(*Print[""];
Print["klist1 = ",klist1];
Print["klist2 = ",klist2];
Print["len = ",Length[klist2]]*);



(*Removing the joints or points j8t1 and j5t8 which are the
ground pivots as we already know their locations *)
Table[
 If[klist1[[i]] == j8t1, klist1[[i]] = 0; klist2[[i]] = 0];
 If[klist1[[i]] == j5t8, klist1[[i]] = 0; klist2[[i]] = 0];
```

```
  , {i, Length[klist1]}];

klist1 = Cases[klist1, Except[0]];
klist2 = Cases[klist2, Except[0]];




(*Removing repetitions form klist1 and klist2*)
Table[
 compTempVal = klist1[[i]];
 Table[
  If[klist1[[j]] == compTempVal, klist1[[j]] = 0; klist2[[j]] = 0];
  , {j, i + 1, Length[klist1]}];
 , {i, Length[klist1] - 1}];

klist1 = Cases[klist1, Except[0]];
klist2 = Cases[klist2, Except[0]];

(*Print["klist1 = ",klist1];
Print["klist2 = ",klist2];
Print[""];*)




(*Finding the locations for each joint using string handling
on its name and extracting the coordinates of the joint in the \
Adjacency matrix*)
   locs = Table[
     ToExpression[
      StringCases[ToString[klist1[[i]]],
       RegularExpression["\\d"]]], {i, 8}];
   (*Print["locs = ",locs];
   Print[""];*)

   (*Finding the name of the joints in my Adjacency matrix using the
   coordinates*)
   Verticesatlocs =
    Table[ModAdjMatrixSym[[locs[[i, 1]], locs[[i, 2]]]], {i, 8}];
   (*Print["Verticesatlocs=",Verticesatlocs];
   Print[""];*)

   (*Finding the nos of the vertices in my convention*)
   VerticesNos = Table[
     ToExpression[
      StringJoin[
       StringCases[ToString[Verticesatlocs[[i]]],
        RegularExpression["\\d"]]]], {i, 8}];
   (*Print["VerticesNos = ",VerticesNos];
   Print["VerticesNos Sorted = ",Sort[VerticesNos]];
   Print[""];*)

   (*Performing ordering on klist2 to get the expression for each joint
   in ascending order eg. {c2,c3,c4,c5,c7,c8,c9,c10}*)
```

```
    VerticesOrder = Ordering[VerticesNos];
    (*Print["Performing ordering we get = ",VerticesOrder];*)

    FinalVerticesSol = klist2[[VerticesOrder]];
    (*Print["FinalVerticesSol = ",FinalVerticesSol];*)

    klist1 = klist1[[VerticesOrder]];
    klist2 = klist2[[VerticesOrder]];
    (*Print["klist1 = ",klist1];
    Print["klist2 = ",klist2];
    Print[""];*)

    FinalVerticesSol];

MakePolygon[list_] := Block[{len, drawPolygon},
  len = Length[list];
  If[len == 3,
   drawPolygon = {(*Darker[Red],Line[{list[[1]],list[[2]]}],
      Line[{list[[2]],list[[3]]}],Line[{list[[3]],list[[1]]}],*)
      Opacity[0.4], EdgeForm[Directive[{Thin, Opacity[0.5]}]],
      Polygon[{list[[1]], list[[2]], list[[3]]}], Opacity[1]};
   ];

  If[len == 4,
   drawPolygon = {(*Darker[Red],Line[{list[[1]],list[[2]]}],
      Line[{list[[2]],list[[3]]}],Line[{list[[3]],list[[4]]}],
      Line[{list[[4]],list[[1]]}],*)Opacity[0.4],
        EdgeForm[Directive[{Thin, Opacity[0.5]}]],
        Polygon[{list[[1]], list[[2]], list[[3]], list[[4]]}],
        Opacity[1]};
     ];
   (*Print["drawPolygon = ",drawPolygon];*)
   drawPolygon]

LinkageAnimation[dataOrig_, dataTemp_, FTLA_, BrianIpAngle_,
   FixedOffsetforIpAngle_, ipAnglesOffset_, SortedSolutions_,
   linkageWithConnections_, ModAdjMatrixSym_, AdjMatWithJointN_, Subs_,
    J_, FiveTpsLocs_] :=
  Block[{ptc1, ptsc2, ptsc3, ptsc4, ptsc5, ptc6, ptsc7, ptsc8, ptsc9,
    ptsc10, DisplayAnimation, link1, link2, link3, link4,
    FirstTernaryLink, SecondTernaryLink, ThirdTernaryLink,
    FourthTernaryLink, TpOriginalDisplay, len, subsLinklength,
    subsFixedAngles, subsAnglesinFirstPosition, TpCurrentDisplay,
    tempVertices, klist1, klist2, kcount, j8t1, j5t8, compTempVal,
    locs, Verticesatlocs, VerticesNos, VerticesOrder, FinalVerticesSol,
     FinalVerticesSolwithSubs, th1, th2, th3, th4, th5, th6, th7, th8,
    LinkSelectList, RRConsist, ternary1, ternary2, ternay3, ternary4,
    quarternary1, quarternary2, polygonsToBeAnimated,
    polygonsCompressed, polygonAnimations, positionOrig, positionTemp,
    FramesonC3C4, posTempinC3C4frame, xaxis, yaxis, origin, fivePtsSet,
     lengthDixonDet, maxDist, frameXYlength, rem, RangeXY, AllPoints,
    xMin, xMax, yMin, yMax, xOffset, yOffset},

   (*Print[
   "//////////////////////   Animation Start   ////////////////////////"];*)
   rem = {{1, 0, 0}, {0, 1, 0}};
   RangeXY = ClippingRange[dataOrig, FilterScaleFactor];
   len = Length[ipAnglesOffset];
   subsLinklength = Subs[[1]];
   subsFixedAngles = Subs[[2]];
   subsAnglesinFirstPosition = Subs[[3]];
   ptsc2 =
    ptsc3 = ptsc4 =
      ptsc5 = ptsc7 =
        ptsc8 = ptsc9 =
          ptsc10 =
           LinkSelectList =
            ternary1 =
             ternary2 =
              ternay3 =
               ternary4 =
                quarternary1 = quarternary2 = ConstantArray[0, len];



   maxDist =
    Max[LinkLength[{dataOrig[[1, 2]],
         dataOrig[[1, 3]]}, {dataOrig[[#, 2]],
         dataOrig[[#, 3]]}] & /@ {2, 3, 4, 5}];
   frameXYlength = If[maxDist/5 > 5, maxDist/5, 5];
   TpOriginalDisplay =
    DisplayTP[dataOrig, frameXYlength, Darker[Green], Darker[Red],
     Gray];
   TpCurrentDisplay =
    DisplayTP[dataTemp, frameXYlength, Darker[Gray], Darker[Gray],
     Gray];

   ptc1 = linkageWithConnections[[1, 1]];
   ptc6 = linkageWithConnections[[1, 6]];
   (*Print["{ptc1,ptc6} = ",{ptc1,ptc6}];*)
   tempVertices = ConstantArray[0, 10];

   (*Print["dataOrig = ",dataOrig];
   Print["dataTemp = ",dataTemp];
   Print["FTLA = ",MatrixForm[FTLA]];
   Print["BrianIpAngle = ",BrianIpAngle];
   Print["FixedOffsetforIpAngle = ",FixedOffsetforIpAngle];
   Print["Linkage = ",linkageWithConnections];
   Print["ModAdjMatrixSym= ",MatrixForm[ModAdjMatrixSym]];
   Print["AdjMatWithJointN= ",MatrixForm[AdjMatWithJointN]];*)

   FinalVerticesSol =
    FindExpressionForVertices[ModAdjMatrixSym, AdjMatWithJointN, FTLA];
   (*Print["***** FinalVerticesSol in main animation function ",
   FinalVerticesSol];*)

   FinalVerticesSolwithSubs =
    FinalVerticesSol /. subsLinklength /. subsFixedAngles;
   (*Print[
   "***** FinalVerticesSol in main animation function with subs ",
   FinalVerticesSolwithSubs];
   *)

   Table[
    ptsc2[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[1]] /.
        Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
          SortedSolutions[[i]]]);
    ptsc3[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[2]] /.
        Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
          SortedSolutions[[i]]]);
    ptsc4[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[3]] /.
        Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
          SortedSolutions[[i]]]);
    ptsc5[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[4]] /.
        Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
          SortedSolutions[[i]]]);
    ptsc7[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[5]] /.
        Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
          SortedSolutions[[i]]]);
    ptsc8[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[6]] /.
         Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
           SortedSolutions[[i]]]);
    ptsc9[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[7]] /.
         Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
           SortedSolutions[[i]]]);
    ptsc10[[i]] =
     ptc6 + (FinalVerticesSolwithSubs[[8]] /.
         Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
           SortedSolutions[[i]]]);
    (*{ptsc2[[i]],ptsc3[[i]],ptsc4[[i]],ptsc5[[i]],ptsc7[[i]],ptsc8[[
    i]],ptsc9[[i]],ptsc10[[i]]}=ptc6+
    FinalVerticesSolwithSubs/.Thread[{th1,th2,th3,th4,th5,th6,th7,
    th8}\[Rule] SortedSolutions[[i]]];*)
    , {i, len}];
   (*Print["ptsc2 = ",ptsc2];*)



   fivePtsSet = Table[
     {ptc1, ptsc2[[FiveTpsLocs[[i]]]], ptsc3[[FiveTpsLocs[[i]]]],
      ptsc4[[FiveTpsLocs[[i]]]], ptsc5[[FiveTpsLocs[[i]]]], ptc6,
      ptsc7[[FiveTpsLocs[[i]]]], ptsc8[[FiveTpsLocs[[i]]]],
      ptsc9[[FiveTpsLocs[[i]]]], ptsc10[[FiveTpsLocs[[i]]]]}
     , {i, 5}];
   (*Print["FivePtsSet = {c1,....,c10}",fivePtsSet];*)
```

```
lengthDixonDet = Table[
  {{LinkLength[ptc1, ptsc2[[i]]]},
   {LinkLength[ptsc2[[i]], ptsc3[[i]]]},
   {LinkLength[ptsc3[[i]], ptsc4[[i]]]},
   {LinkLength[ptsc4[[i]], ptsc5[[i]]]},
   {LinkLength[ptsc5[[i]], ptc6]},
   {LinkLength[ptsc7[[i]], ptsc8[[i]]]},
   {LinkLength[ptsc9[[i]], ptsc10[[i]]]}
   }
  , {i, len}];
(*Print["lengthDixonDet = ",MatrixForm[lengthDixonDet]];*)


positionOrig = Table[Disp[dataOrig[[i]]], {i, 5}];
positionTemp = Table[Disp[dataTemp[[i]]], {i, 5}];


FramesonC3C4 =
 Chop[Table[
   Disp[Flatten[{JointAngle[{1, 0}, ptsc4[[i]] - ptsc3[[i]]],
      ptsc3[[i]]}]], {i, len}]];
(*Print["FramesonC3C4 = ",FramesonC3C4[[len]]];*)
posTempinC3C4frame =
 Chop[Inverse[FramesonC3C4[[1]]].positionTemp[[1]]];
(*Print["posTempinC3C4frame  = ",posTempinC3C4frame ];*)
xaxis =
 Table[rem.FramesonC3C4[[i]].posTempinC3C4frame .{frameXYlength, 0,
    1}, {i, len}];
yaxis =
 Table[rem.FramesonC3C4[[i]].posTempinC3C4frame .{0, frameXYlength,
    1}, {i, len}];
origin =
 Table[rem.FramesonC3C4[[i]].posTempinC3C4frame .{0, 0, 1}, {i,
   len}];

(*


(* CHECKS *)
Print["#### CHECK \
#############################################################################################\
####################################"];
Print["Test for ptsc2"];
Print[Table[Chop[LinkLength[ptc1,ptsc2[[i]]]-LinkLength[ptc1,ptsc2[[
1]]]],{i,len}]];
Print["Test for ptsc3"];
Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc3[[i]]]-LinkLength[
ptsc2[[1]],ptsc3[[1]]]],{i,len}]];
Print["Test for ptsc4"];
Print[Table[Chop[LinkLength[ptsc3[[i]],ptsc4[[i]]]-LinkLength[
ptsc3[[1]],ptsc4[[1]]]],{i,len}]];
Print["Test for ptsc5"];
Print[Table[Chop[LinkLength[ptc6,ptsc5[[i]]]-LinkLength[ptc6,ptsc5[[
1]]]],{i,len}]];
Print["Test2 for ptsc5"];
Print[Table[Chop[LinkLength[ptsc4[[i]],ptsc5[[i]]]-LinkLength[
ptsc4[[1]],ptsc5[[1]]]],{i,len}]];
Print["Test for link C7C8 Green link"];
Print[Table[Chop[LinkLength[ptsc7[[i]],ptsc8[[i]]]-LinkLength[
ptsc7[[1]],ptsc8[[1]]]],{i,len}]];
Print["Test for link C9C10 Blue link"];
Print[Table[Chop[LinkLength[ptsc9[[i]],ptsc10[[i]]]-LinkLength[
ptsc9[[1]],ptsc10[[1]]]],{i,len}]];
Print["Test for link C1C7 link"];
Print[Table[Chop[LinkLength[ptc1,ptsc7[[i]]]-LinkLength[ptc1,ptsc7[[
1]]]],{i,len}]];
Print["Test for link C2C7 link"];
Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc7[[i]]]-LinkLength[
ptsc2[[1]],ptsc7[[1]]]],{i,len}]];
Print["Test for link C1C9 link"];
Print[Table[Chop[LinkLength[ptc1,ptsc9[[i]]]-LinkLength[ptc1,ptsc9[[
1]]]],{i,len}]];
Print["Test for link C2C9 link"];
Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc9[[i]]]-LinkLength[
ptsc2[[1]],ptsc9[[1]]]],{i,len}]];
Print["####   CHECK   \
################################################################\
#######################"];

*)
```

```
RRConsist = Flatten[linkageWithConnections[[2]]];
(*Print["RRConnectionList = ",RRConsist];*)

Table[LinkSelectList[[i]] =
  {{ptc1, ptsc2[[i]]}, {ptsc2[[i]], ptsc3[[i]]}, {ptsc3[[i]],
    ptsc4[[i]]}, {ptsc4[[i]], ptsc5[[i]]}, {ptsc5[[i]],
    ptc6}, {ptsc7[[i]], ptsc8[[i]]}}, {i, len}];

polygonsToBeAnimated = Table[
  {Flatten[{LinkSelectList[[i, RRConsist[[1]]]], {ptsc7[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[2]]]], {ptsc8[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[3]]]], {ptsc9[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[4]]]], {ptsc10[[i]]}}, 1]}
  , {i, len}];

(*Print["Finding all four ternary links = ",polygonsToBeAnimated[[
1]]];
Print[""];*)




(*


polygonsCompressed=ConstantArray[{0,0,0,0},len];
Table[
polygonsCompressed[[i,1]]=Flatten[polygonsToBeAnimated[[i,1]],1];
polygonsCompressed[[i,2]]=Flatten[polygonsToBeAnimated[[i,2]],1];
```

```
If[RRConsist[[3]]\[Equal]RRConsist[[1]],
polygonsCompressed[[i,1]]=Union[polygonsToBeAnimated[[i,1]],Flatten[
polygonsToBeAnimated[[i,3]],1]],
If[RRConsist[[3]]\[Equal]RRConsist[[2]],
polygonsCompressed[[i,2]]=Union[polygonsToBeAnimated[[i,2]],Flatten[
polygonsToBeAnimated[[i,3]],1]],
polygonsCompressed[[i,3]]=Flatten[polygonsToBeAnimated[[i,3]],1]]];

If[RRConsist[[4]]\[Equal]RRConsist[[1]],
polygonsCompressed[[i,1]]=Union[polygonsToBeAnimated[[i,1]],Flatten[
polygonsToBeAnimated[[i,4]],1]],
If[RRConsist[[4]]\[Equal]RRConsist[[2]],
polygonsCompressed[[i,2]]=Union[polygonsToBeAnimated[[i,2]],Flatten[
polygonsToBeAnimated[[i,4]],1]],
polygonsCompressed[[i,4]]=Flatten[polygonsToBeAnimated[[i,4]],1]]];

polygonsCompressed[[i]]=Cases[polygonsCompressed[[i]],Except[0]];

,{i,len}];

(*Print["polygonsCompressed = ",MatrixForm[polygonsCompressed]];*)
Print["polygonsCompressed one sol = ",polygonsCompressed[[1]]];




(*polygonAnimations=Table[
Flatten[Table[MakePolygon[polygonsCompressed[[i,j]]]
,{j,Length[polygonsCompressed[[i]]]}]];
(*Print["drawPolygon = ",MakePolygon[polygonsCompressed[[i,
1]]]]];*)
,{i,len}];

Print[polygonAnimations[[1]]];*)



(*Super Cheap Programming*)
If[Length[polygonsCompressed[[1]]]\[Equal]3,
DisplayAnimation=
Table[
            Graphics[{
(*jointcurve,*)
TpCurrentDisplay,TpOriginalDisplay,

Thickness[0.006],
Darker[Gray],Line[{ptc1,ptc6}],
Darker[Red],Line[{ptc1,ptsc2[[i]],ptsc3[[i]],ptsc4[[i]],ptsc5[[i]],
ptc6}],
PointSize[0.015],Point[{ptsc2[[i]],ptsc3[[i]],ptsc4[[i]],ptsc5[[
i]]}],



Green,Line[{ptsc7[[i]],ptsc8[[i]]}],Darker[Green],Point[{ptsc7[[i]],
ptsc8[[i]]}],
Blue , Line [{ ptsc9 [[ i ]] , ptsc10 [[ i ]] }] , Darker [ Purple ] , Point [{ ptsc9 [[
i ]] , ptsc10 [[ i ]] }] ,


MakePolygon [ polygonsCompressed [[ i , 1 ]]] , MakePolygon [
polygonsCompressed [[ i , 2 ]]] , MakePolygon [ polygonsCompressed [[ i , 3 ]]] ,


Darker [ Gray ] , Point [{ ptc1 , ptc6 }] , Line [{ yaxis [[ i ]] , origin [[ i ]] , xaxis [[
i ]] }]
},

Axes \[ Rule ] True , AspectRatio \[ Rule ] Automatic , ImageSize \[ Rule ]  500 ,
PlotRange \[ Rule]{{ −40,20},{ −10,50}}
]
,{ i , len }]; ,

DisplayAnimation=
Table [
            Graphics [{
(∗jointcurve ,∗)
TpCurrentDisplay , TpOriginalDisplay ,

Thickness [0.006] ,
Darker [ Gray ] , Line [{ ptc1 , ptc6 }] ,
Darker [Red] , Line [{ ptc1 , ptsc2 [[ i ]] , ptsc3 [[ i ]] , ptsc4 [[ i ]] , ptsc5 [[ i ]] ,
ptc6 }] ,
PointSize[0.015],Point[{ptsc2[[i]],ptsc3[[i]],ptsc4[[i]],ptsc5[[
i]]}],


Green,Line[{ptsc7[[i]],ptsc8[[i]]}],Darker[Green],Point[{ptsc7[[i]],
ptsc8[[i]]}],
Blue,Line[{ptsc9[[i]],ptsc10[[i]]}],Darker[Purple],Point[{ptsc9[[
i]],ptsc10[[i]]}],



MakePolygon[polygonsCompressed[[i,1]]],MakePolygon[
polygonsCompressed[[i,2]]],MakePolygon[polygonsCompressed[[i,3]]],
MakePolygon[polygonsCompressed[[i,4]]],

Darker[Gray],Point[{ptc1,ptc6}],Line[{yaxis[[i]],origin[[i]],xaxis[[
i]]}]
},

Axes\[Rule]True,AspectRatio\[Rule]Automatic,ImageSize\[Rule] 500,
PlotRange\[Rule]{{-40,20},{-10,50}}
]
,{i,len}];];



*)
```

```
(*Calculating the Plot Range for the Animation*)
AllPoints =
 Flatten[{xaxis, origin, yaxis, {ptc1, ptc6}, ptsc2, ptsc3, ptsc4,
   ptsc5, ptsc7, ptsc8, ptsc9, ptsc10}, 1];
(*Print["AllPoints = ",AllPoints];*)
{xMin, xMax, yMin, yMax} = {Min[AllPoints[[All, 1]]],
  Max[AllPoints[[All, 1]]], Min[AllPoints[[All, 2]]],
  Max[AllPoints[[All, 2]]]};
{xOffset, yOffset} = {0.02*Abs[RangeXY[[1, 1]] - RangeXY[[1, 2]]],
  0.02*Abs[RangeXY[[2, 1]] - RangeXY[[2, 2]]]};



DisplayAnimation =
 Table[
            Graphics[{
    (*Inset[Grid[{{Style["Product of J list",Blue,12]},
    {(J[[1]]*J[[2]]*J[[3]])/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style["J list",Blue,
    12]},{J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style[
    "Sign of J list",Blue,12]},{Sign[J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]]},{Style[
    "RR connections",Blue,12]},{RRConsist},
    {Style["data Temp",Blue,12]},{dataTemp}
    },Frame\[Rule] All,ItemSize\[Rule]20,Background\[Rule]{Lighter[
    Yellow],None,Lighter[Yellow],None,Lighter[Yellow],
    None}],{RangeXY[[1,1]]+15,RangeXY[[2,2]]-18}],*)

    (*Inset[Grid[{{Style["Product of J list",Blue,12]},
    {(J[[1]]*J[[2]]*J[[3]])/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style["J list",Blue,
    12]},{J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style[
    "Sign of J list",Blue,12]},{Sign[J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]]},{Style[
    "RR connections",Blue,12]},{RRConsist},
    {Style["data Temp",Blue,12]},{dataTemp}
    },Frame\[Rule] All,ItemSize\[Rule]20,Background\[Rule]{Lighter[
    Yellow],None,Lighter[Yellow],None,Lighter[Yellow],
    None}],{-30,-10}],*)



    (*jointcurve,*)
    TpCurrentDisplay, TpOriginalDisplay,

    Thickness[0.006],
    Darker[Gray], Line[{ptc1, ptc6}],

    Lighter[Green], MakePolygon[polygonsToBeAnimated[[i, 1]]],
    Lighter[Green], MakePolygon[polygonsToBeAnimated[[i, 2]]],
    Lighter[Blue], MakePolygon[polygonsToBeAnimated[[i, 3]]],
    Lighter[Blue], MakePolygon[polygonsToBeAnimated[[i, 4]]],
        Thickness[0.004],
        Green, Line[{ptsc7[[i]], ptsc8[[i]]}], Darker[Green],
        Point[{ptsc7[[i]], ptsc8[[i]]}],
        Blue, Line[{ptsc9[[i]], ptsc10[[i]]}], Darker[Purple],
        Point[{ptsc9[[i]], ptsc10[[i]]}],

        Black,
        Line[{ptc1, ptsc2[[i]], ptsc3[[i]], ptsc4[[i]], ptsc5[[i]],
          ptc6}],
        PointSize[0.011],
        Point[{ptsc2[[i]], ptsc3[[i]], ptsc4[[i]], ptsc5[[i]],
          ptsc7[[i]], ptsc8[[i]], ptsc9[[i]], ptsc10[[i]]}],

        Darker[Gray], Point[{ptc1, ptc6}],
        Line[{yaxis[[i]], origin[[i]], xaxis[[i]]}],
        Polygon[{ptsc3[[i]], ptsc4[[i]], origin[[i]]}]
        },

       Axes -> True, AspectRatio -> Automatic, ImageSize -> 400,
       PlotRange -> {{xMin - xOffset, xMax + xOffset}, {yMin - yOffset,
          yMax + yOffset}}(*{{-35,18},{-5,30}}*)
       ]
      , {i, len}];

    (*Print["Linkage Pic = ",LinkageDisplay[dataTemp,
    linkageWithConnections[[1]],linkageWithConnections[[2]],RangeXY]];*)
```

```
  Print[ListAnimate[DisplayAnimation]];
  (*Label[ProbFound];
  {Prob,M}*)
  (*Print[
  "///////////////////   Animation End   //////////////////////"];*)

  ]

PlayEightBarLinkageAnimation[FinalSol_] := Block[{K},
  K = LinkageAnimation[FinalSol[[3]], FinalSol[[4]], FinalSol[[5]],
     FinalSol[[6]], FinalSol[[7]], FinalSol[[8]],
     FinalSol[[9]], {FinalSol[[1]], FinalSol[[2]]}, FinalSol[[10]],
     FinalSol[[11]], FinalSol[[12]], FinalSol[[13]], FinalSol[[14]]];
  (*Print["No of images in the animation = ",Length[K]];*)
  Print[ListAnimate[K]];
  ]

  AnalysisOfEachSynthesizedLinkage2[SynthesizedLinkageSol_] :=
 Block[{AdjMatrix, ModAdjMatrix, ModAdjMatrixSym, LinkageToAnalyze,
   MmatReal, NmatReal, Tmat, opAngle, depAngles, depAnglesTmatRatios,
   FTLA, Subs, AdjMatWithJointN, J, inputAngles, FiveTpsLocs,
   FiveBrianTpsConfigAngles, SynDefectCheck, BrianIpAngle,
   FixedOffsetforIpAngle, ipAnglesOffset, StartingSolAngles,
   Solutions, SolutionBoxShortened, groundAngle, ThisSolutionFailed,
   SortedSolutions, TrackTopology, SaveSolution,
   rangeOfMotionforLink1},
SaveSolution = 0;
groundAngle =
 JointAngle[{1, 0},
  SynthesizedLinkageSol[[1, 1]] - SynthesizedLinkageSol[[1, 6]]];


{AdjMatrix, ModAdjMatrix, ModAdjMatrixSym} =
 FindModifiedAdjacencyMatrix3[SynthesizedLinkageSol[[1]],
  SynthesizedLinkageSol[[3]]];
LinkageToAnalyze = {ModAdjMatrix, 8, 1};


{MmatReal, NmatReal, Tmat, opAngle, depAngles, depAnglesTmatRatios,
  FTLA, Subs, AdjMatWithJointN, J, TrackTopology} =
 Chop[DixonDeterminantForOneLinkage[LinkageToAnalyze], 10^-7];


{inputAngles, FiveTpsLocs, rangeOfMotionforLink1} =
 MakeDivisionsinTps2[SynthesizedLinkageSol[[6]], IncrementAngle];


{SynDefectCheck, FiveBrianTpsConfigAngles} =
 TranslatingMyAnglestoBriansAnglesandCheck5[
  SynthesizedLinkageSol[[6]], SynthesizedLinkageSol[[7]],
  SynthesizedLinkageSol[[8]], SynthesizedLinkageSol[[9]],
  SynthesizedLinkageSol[[10]], SynthesizedLinkageSol[[1]],
  SynthesizedLinkageSol[[4]], SynthesizedLinkageSol[[3]], FTLA,
  Subs, J];
 (*If[SynDefectCheck\[Equal]1,Goto [ThisSolutionFailed]];*)


 {BrianIpAngle, FixedOffsetforIpAngle, ipAnglesOffset,
   StartingSolAngles, Solutions, SolutionBoxShortened} =
  ForwardKinematicsEightBar[inputAngles, groundAngle, MmatReal,
   NmatReal, Tmat, opAngle, depAngles, depAnglesTmatRatios, FTLA,
   Subs, AdjMatWithJointN];
 If[SolutionBoxShortened == 0, Goto [ThisSolutionFailed]];
 (*Print[" ipAnglesOffset = ",ipAnglesOffset/Degree];*)


 (*SortedSolutions=SortingTrajectories[ipAnglesOffset,
 SolutionBoxShortened,FiveBrianTpsConfigAngles,FTLA,Subs,groundAngle,
 FiveTpsLocs];*)
 Quiet[Check[
   SortedSolutions =
     SortingTrajectories[ipAnglesOffset, SolutionBoxShortened,
      FiveBrianTpsConfigAngles, FTLA, Subs, groundAngle, FiveTpsLocs,
       rangeOfMotionforLink1];
   , Print[
    "Ill conditioned or Singular Matrix has been encountered"], \
{Inverse::luc, Inverse::sing}], {Inverse::luc, Inverse::sing}];
 If[SortedSolutions == 0, Goto [ThisSolutionFailed]];
 (*Quiet[Check[Inverse[N[HilbertMatrix[20]]],
 "Badly conditioned Matrix has been encountered",Inverse::luc],
Inverse::luc];*)
Off[FrontEndObject::notavail];
(*LinkageAnimation[data,SynthesizedLinkageSol[[5]],FTLA,
BrianIpAngle,FixedOffsetforIpAngle,ipAnglesOffset,
SortedSolutions,{SynthesizedLinkageSol[[1]],SynthesizedLinkageSol[[
3]]},ModAdjMatrixSym,AdjMatWithJointN(*{{-53.2`,20.6`},{-21.85`,
51.95`}}*),Subs,J,FiveTpsLocs];*)


SaveSolution = {SynthesizedLinkageSol[[1]],
   SynthesizedLinkageSol[[3]], data, SynthesizedLinkageSol[[5]],
  FTLA, BrianIpAngle, FixedOffsetforIpAngle, ipAnglesOffset,
   SortedSolutions, ModAdjMatrixSym, AdjMatWithJointN, Subs, J,
   FiveTpsLocs, TrackTopology, SynDefectCheck};
noOfDefectFreeLinkages++;

Label[ThisSolutionFailed];
Evaluate[progress++];
SaveSolution]

EightbarGeneratorMainLoop[data_, SixRloop_, TolTheta5_, TolXfive_,
 TolYfive_, TolXC1_, TolYC1_, TolXC6_, TolYC6_, noOfLoops_,
 IncrementAngle_, taskTolerance_, FilterScaleFactor_,
 Elbow3Rchain1_, Elbow3Rchain2_] :=
Block[{ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, a1, a2, a3, a4, a5, a8,
  dataTemp, positionTemp, position, groundAngle, A1, A2, A3, A4, A5,
   A8, TolTheta, TolX, TolY, RangeXY, Maxlen, Minlen, noOfSols,
 TimeTaken, n, FinalLinkageSolution, T, RandomizeTP,
 TopologiesUnique, TrackTopology, LinkageSolutionsX,
 NextSolutionAsThisOneFailed, Kanimate, FTLA, BrianIpAngle,
 FixedOffsetforIpAngle, ipAnglesOffset, SortedSolutions,
 ModAdjMatrixSym, AdjMatWithJointN, Subs, J, FiveTpsLocs,
 SolutionBoxShortened, FiveBrianTpsConfigAngles, MmatReal,
 NmatReal, Tmat, opAngle, depAngles, depAnglesTmatRatios,
 AdjMatrix, ModAdjMatrix, theta1, theta2, theta3, psi1, psi2, psi3,
  LinkageToAnalyze, inputAngles, NoofLinkagesForOneConType,
 StartingSolAngles, Solutions, A1Frames, A2Frames, A3Frames,
 A4Frames, A5Frames, AFrames, c1Frame, c6Frame, ptc3intFrame,
 ptc4intFrame, tFramewrtc3Frame, tFramewrtc4Frame, c3c4Frame,
 tFrameinc3c4Frame, c3framestoc1frame, c4framestoc6frame,
 Fiveptsc3, Fiveptsc4, FinalLinkageSolutionSorted , check,
 ratiomaxmin, FinalLinkageSols, Scaler, rem, dataArray,
 timeRandomizedata, timeSynLinkagesForAllLoops, Ksol,
 SynthesizedLinkageSols, timeAnalyzeAllLinkages, GoodSols,
 GoodSolutions, SixRloopArray, timeRandomizeSixRloop},



(*Randomize task positions and store them in dataArray*)
Print[];
Print["//// Generating all the randomized task positions /////"];
{timeRandomizedata, dataArray} =
 AbsoluteTiming[
  N[GenerateRandomizedData[data, noOfLoops, TolTheta5, TolXfive,
      TolYfive]]];
    Print["timeRandomizedata = ", timeRandomizedata/60, " min"];
    Print["dataArray len = ", Length[dataArray]];

    (*Synthesize all the linkages possible for each randomized task \
position*)
    Print[]; Print[];
    Print["//// Synthesizing all the possible linkages for each of the \
randomized task positions /////"];
    {timeSynLinkagesForAllLoops, Ksol} = AbsoluteTiming[
      ParallelTable[
       SynthesizingAllThePossibleLinkagesForEachLoop[dataArray[[i]],
        SixRloop, Elbow3Rchain1, Elbow3Rchain2], {i, Length[dataArray]}]
      ];




    (*
    (*Randomizing base pivots c1 and c6*)
    Print[];
    Print["//// Generating all the randomized 6R loops /////"];
    {timeRandomizeSixRloop,SixRloopArray}=AbsoluteTiming[N[
    GenerateRandomizedSixRloop[SixRloop,noOfLoops,TolXC1,TolYC1,TolXC6,
    TolYC6]]];
    Print["timeRandomizeSixRloop = ",timeRandomizeSixRloop/60," min"];
    Print["SixRloopArray len = ",Length[SixRloopArray]];
    (*Synthesize all the linkages possible for each randomized \
SixRloop*)
    Print[]; Print[];
    Print["//// Synthesizing all the possible linkages for each of the \
randomized 6R loop /////"];
    {timeSynLinkagesForAllLoops,Ksol}=AbsoluteTiming[
    ParallelTable[SynthesizingAllThePossibleLinkagesForEachLoop[data,
    SixRloopArray[[i]],Elbow3Rchain1,Elbow3Rchain2],{i,Length[
    SixRloopArray]}]
    ];*)



    NoofLinkagesForOneConType = Ksol[[All, 2]];
    SynthesizedLinkageSols = Ksol[[All, 1]];
    SynthesizedLinkageSols = Cases[SynthesizedLinkageSols, Except[0]];
    Print["____ timeSynLinkagesForAllLoops = ",
     timeSynLinkagesForAllLoops/60, " min"];
    (*Print[
    "Length of solutions after removing the zero solution data = ",
    Length[SynthesizedLinkageSols]];*)
    (*Print[
    "Length of solutions after removing the zero solution data 2 = ",
    Length[dataArray]];*)
    SynthesizedLinkageSols = Flatten[SynthesizedLinkageSols, 1];
    Print["____ Length of total no of linkages synthesized from all \
the randomized tps = ", Length[SynthesizedLinkageSols]];
```

```
   (*Calling Analysis Function*)
   Print[]; Print[];
   Print[
    "//// Analyzing all the synthesized linkages for defects and \
animating the good linkages /////"];
   Print[{ProgressIndicator[
      Dynamic[progress], {0, Length[SynthesizedLinkageSols]}],
     Dynamic[progress]}];
   {timeAnalyzeAllLinkages, GoodSols} =
    AbsoluteTiming[
     ParallelTable[
      AnalysisOfEachSynthesizedLinkage2[SynthesizedLinkageSols[[i]]](*;
      Evaluate[progress++];*)
      , {i,(*40,42*)(*17,17*)Length[SynthesizedLinkageSols]}]];



   (*Display Statistics on No. of Solutions and Time taken*)
   GoodSolutions = Cases[GoodSols, Except[0]];
   Print["____ timeAnalyzeAllLinkages = ", timeAnalyzeAllLinkages/60,
    " min"];
   Print["____ No of Good Solutions found = ", Length[GoodSolutions]];
   Print["____ Total time taken to run the program = ", \
(timeRandomizedata + timeSynLinkagesForAllLoops +
        timeAnalyzeAllLinkages)/60, " min"];
    (*Print[
    "Total time taken to run the program = ",(timeRandomizeSixRloop+
    timeSynLinkagesForAllLoops+timeAnalyzeAllLinkages)/60," min"];*)



    GoodSolutions];

    (*Sofa bed linkage*)
(*data ={{0 Degree,-4,6.1},{-49 Degree,-5.4,18.5},{-53 \
Degree,-14,24},{-42 Degree,-20,22.4},{0 Degree,-28.6,13.6}};
ptC1={-6,0}; ptC6={2,0};
ptC2={8.2,8};ptC5={11.5,15.3};
ptC3={-8,6.1};ptC4={-4,6.1};
Tol\[Theta]m={0,5,10,5,0} ;
TolXm={2,2,2,2,2};
TolYm={2,2,2,2,2};
noOfLoops =1;*)

(*Straight line motion linkage*)
data = {{0.0 Degree, 0, 0}, {0.0 Degree, 15, 0}, {0.0 Degree, 50,
     0}, {0.0 Degree, 85, 0}, {0.0 Degree, 100, 0}};
{ptC1, ptC2, ptC3, ptC4, ptC5,
    ptC6} = {{40.0, -150.0}, {99.15, -112.56}, {0.0, 0.0}, {30.0,
     0.0}, {116.99, -82.65}, {0.0, -150.0}};
Tol\[Theta]m = {15, 15, 15, 15, 15};
```

```
TolXm = {2, 2, 2, 2, 2};
TolYm = {2, 2, 2, 2, 2};
noOfLoops = 1;
(*16,17 has issues. Talk to Brian about the bug you found*)

FilterScaleFactor =
  2.5;(*Now used only for setting the x and y offset in Plot Range of \
animation and filtering the defect free solutions*)
IncrementAngle = 0.5 Degree;
taskTolerance = .01;



   fiveTpsDisplay = DisplayTP[data, 20, Darker[Red], Darker[Green], Gray];

sixRloopDisplay = {Thickness[0.007],
   Opacity[0.9], Gray,
   Line[{ptC1, {103.7, -120.97}, {15, 0}, {45, 0}, {109.6, -101.13},
     ptC6}],
   Line[{ptC1, {108.96, -137.93}, {50, 0}, {80, 0}, {114.73, -114.86},
      ptC6}],
   Line[{ptC1, {109.97, -147.9}, {85, 0}, {115, 0}, {116.18, -119.99},
      ptC6}],
   Line[{ptC1, {110, -149.7}, {100, 0}, {130, 0}, {115.97, -119.17},
     ptC6}],
   Opacity[1],
   Thickness[0.007], Black,
   Line[{ptC1, ptC2, ptC3, ptC4, ptC5, ptC6}],
   Line[{ptC3, {data[[1, 2]], data[[1, 3]]}, ptC4}], PointSize[0.02],
   Black, Point[{ptC2, ptC3, ptC4, ptC5}], Red, Point[{ptC1, ptC6}]}];

Graphics[{sixRloopDisplay, fiveTpsDisplay}, Axes -> True,
 PlotRange -> Automatic, Background -> White]

\!\(\*
GraphicsBox[{{
{GrayLevel[0.5], Thickness[0.007], Opacity[0.9],
      LineBox[{{40., -150.}, {103.7, -120.97}, {15, 0}, {45, 0}, {
        109.6, -101.13}, {0., -150.}}],
      LineBox[{{40., -150.}, {108.96, -137.93}, {50, 0}, {80, 0}, {
        114.73, -114.86}, {0., -150.}}],
      LineBox[{{40., -150.}, {109.97, -147.9}, {85, 0}, {115, 0}, {
        116.18, -119.99}, {0., -150.}}],
      LineBox[{{40., -150.}, {110, -149.7}, {100, 0}, {130, 0}, {
        115.97, -119.17}, {0., -150.}}]},
{GrayLevel[0], Thickness[0.007], Opacity[1],
      LineBox[{{40., -150.}, {99.15, -112.56}, {0., 0.}, {30., 0.}, {
       116.99, -82.65}, {0., -150.}}],
      LineBox[{{0., 0.}, {0, 0}, {30., 0.}}],
{GrayLevel[0], PointSize[0.02],
       PointBox[{{99.15, -112.56}, {0., 0.}, {30., 0.}, {
        116.99, -82.65}}],
{RGBColor[1, 0, 0], PointBox[{{40., -150.}, {0., -150.}}]}}}}, {
{RGBColor[
NCache[
Rational[2, 3], 0.6666666666666666], 0, 0], Thickness[0.005],
      LineBox[{{0., 0.}, {0., 20.}}],
{RGBColor[0,
NCache[
Rational[2, 3], 0.6666666666666666], 0],
       LineBox[{{0., 0.}, {20., 0.}}]},
{GrayLevel[0.5], PointSize[0.007], PointBox[{0., 0.}]}},
{RGBColor[
NCache[
Rational[2, 3], 0.6666666666666666], 0, 0], Thickness[0.005],
      LineBox[{{15., 0.}, {15., 20.}}],
{RGBColor[0,
NCache[
Rational[2, 3], 0.6666666666666666], 0],
       LineBox[{{15., 0.}, {35., 0.}}]},
{GrayLevel[0.5], PointSize[0.007], PointBox[{15., 0.}]}},
{RGBColor[
NCache[
Rational[2, 3], 0.6666666666666666], 0, 0], Thickness[0.005],
      LineBox[{{50., 0.}, {50., 20.}}],
{RGBColor[0,
NCache[
Rational[2, 3], 0.6666666666666666], 0],
       LineBox[{{50., 0.}, {70., 0.}}]},
{GrayLevel[0.5], PointSize[0.007], PointBox[{50., 0.}]}},
{RGBColor[
NCache[
Rational[2, 3], 0.6666666666666666], 0, 0], Thickness[0.005],
      LineBox[{{85., 0.}, {85., 20.}}],
{RGBColor[0,
NCache[
Rational[2, 3], 0.6666666666666666], 0],
       LineBox[{{85., 0.}, {105., 0.}}]},
{GrayLevel[0.5], PointSize[0.007], PointBox[{85., 0.}]}},
{RGBColor[
NCache[
Rational[2, 3], 0.6666666666666666], 0, 0], Thickness[0.005],
      LineBox[{{100., 0.}, {100., 20.}}],
{RGBColor[0,
NCache[
Rational[2, 3], 0.6666666666666666], 0],
       LineBox[{{100., 0.}, {120., 0.}}]},
{GrayLevel[0.5], PointSize[0.007], PointBox[{100., 0.}]}}}},
Axes->True,
Background->GrayLevel[1],
PlotRange->Automatic]\)

progress = 0; SetSharedVariable[progress];
noOfDefectFreeLinkages = 0; SetSharedVariable[noOfDefectFreeLinkages];
SixRloop = {ptC1, ptC2, ptC3, ptC4, ptC5, ptC6};
elbow1 = JointAngle[ptC1 - ptC2, ptC3 - ptC2]; elbow2 =
 JointAngle[ptC6 - ptC5, ptC4 - ptC5];
Elbow3Rchain1 = If[elbow1 > \[Pi], 1, -1]; Elbow3Rchain2 =
 If[elbow2 > \[Pi], 1, -1];
```

```
Dynamic[noOfDefectFreeLinkages]

Sols = EightbarGeneratorMainLoop[data, SixRloop, Tol\[Theta]m, TolXm,
   TolYm, TolXC1, TolYC1, TolXC6, TolYC6, noOfLoops, IncrementAngle,
   taskTolerance, FilterScaleFactor, Elbow3Rchain1, Elbow3Rchain2];

   MaxLengthBoxRatio = 1;
MinLengthBoxRatio = 0.02;
RangeXYtemp = ClippingRange[data, FilterScaleFactor];
Maxlen = MaxLengthBoxRatio*
   Abs[RangeXYtemp[[1, 1]] - RangeXYtemp[[1, 2]]];
Minlen = MinLengthBoxRatio*Maxlen;
{RangeXYtemp, Maxlen, Minlen};

FilterEightbarLinkageSolutions[Sols_, Maxlen_, Minlen_] :=
  Block[{FinalSols, check, ratiomaxmin},
   FinalSols = ConstantArray[0, Length[Sols]];
   Table[
    (*Print[];*)
    {check, ratiomaxmin} =
     LinkageLinkLenBoundCheck[Sols[[i, 1]], Sols[[i, 2]], Maxlen,
      Minlen];
    (*Print["check = ",check];
    Print["ration max/min = ",ratiomaxmin];*)
    If[check == 1, FinalSols[[i]] = Join[Sols[[i]], {ratiomaxmin}]];
    , {i, Length[Sols]}];
   FinalSols = Cases[FinalSols, Except[0]];
   FinalSols];


SaveSolution[Sol_] :=
 Block[{OldSolutions, CurrentAddress, EarlierSols, CurrentSols},
  (*SetDirectory[
  "C:\\Users\\Jarvis3\\Dropbox\\Latest Research\\New \
Eightbars\\SavedSolutions"];*)
  (*filename=ToString[fileName];*)
  CurrentAddress =
   StringJoin[ToString[$HomeDirectory],
    "\\Dropbox\\Latest Research\\New Eightbars\\SavedSolutions"];
  (*Speak["Searching for the current directory"];*)
  SetDirectory[CurrentAddress];
  Print["Current Directory = ", Directory[]];
  (*Speak[
  "The Save Solutions folder has been found in the Dropbox"];*)

  EarlierSols =<< StraightlineWithoutOverlapEightbar;
  Print["Total no of Solutions Stored in the temp file is = ",
   Length[EarlierSols]];
  (*Speak[StringJoin[
  "Total number of Solutions currently Stored in temp file are ",
  ToString[Length[EarlierSols]],", ,"]];*)
  (*Speak[ToString[Length[EarlierSols]]];*)
  If[EarlierSols == {}, {Sol} >> StraightlineWithoutOverlapEightbar,
   AppendTo[EarlierSols, Sol] >> StraightlineWithoutOverlapEightbar];
  Print["Current Solution has been added"];
  CurrentSols =<< StraightlineWithoutOverlapEightbar;
  Print["Total no of Solution Currently Stored are ",
   Length[CurrentSols]];
  (*Speak["The new Solution has been saved"];*)
  ]

  SaveAnimation[FinalSol_, videoFormat_] :=
 Block[{K, currentDateTime, fileName, Kanimation, CurrentAddress},
  (*SetDirectory[
  "C:\\Users\\Jarvis3\\Dropbox\\Latest Research\\New \
Eightbars\\Videos"];*)
  (*CurrentAddress=StringJoin[ToString[$HomeDirectory],
  "\\Dropbox\\Latest Research\\New Eightbars\\Videos"];
  SetDirectory[CurrentAddress];*)
  Print["Current Directory = ", Directory[]];

  K = LinkageAnimation[FinalSol[[3]], FinalSol[[4]], FinalSol[[5]],
    FinalSol[[6]], FinalSol[[7]], FinalSol[[8]],
    FinalSol[[9]], {FinalSol[[1]], FinalSol[[2]]}, FinalSol[[10]],
    FinalSol[[11]], FinalSol[[12]], FinalSol[[13]], FinalSol[[14]]];
  Kanimation = ListAnimate[K];

  currentDateTime =
   StringReplace[ToString[DateString[]], Whitespace -> " "];
  currentDateTime = StringReplace[currentDateTime, ":" -> " "];
  (*fileName=StringJoin["Animation ",currentDateTime,".mov"];*)
  fileName =
   StringJoin["EightbarAnimation ", currentDateTime, ".",
    ToString[videoFormat]];
  Print["File name = ", fileName];
  (*Print["Animation Length = ",Length[Kanimation]];*)
  Print["Original no. of images = ", Length[K]];
  Export[fileName, Kanimation];
  Speak["The Annimation for the selected linkage has been saved"];
  Speak[" to the Videos folder in DropBox"];
  ]
```

C Mathematica code for Eight-bar Linkage Design obtained from a 4R Serial Chain

```
rem = {{1, 0, 0}, {0, 1, 0}};

npos = 5;

LinkLength[a_, b_] :=
  If[Norm[a] == 0 && Norm[b] == 0, 0, N[Sqrt[Dot[b - a, b - a]]]];

JointAngle[b_, c_] :=
  If[Norm[b] == 0 || Norm[c] == 0, 0,
   Mod[ArcTan[Dot[b, c], Det[{b, c}]], 2 \[Pi]]];

Zmat[\[Theta]_] := {{Cos[\[Theta]], -Sin[\[Theta]],
    0}, {Sin[\[Theta]], Cos[\[Theta]], 0}, {0, 0, 1}};

Xmat[a_] := {{1, 0, a}, {0, 1, 0}, {0, 0, 1}};

(*rem = {{1,0,0},{0,1,0}};*)

hom[v_] := {v[[1]], v[[2]], 1};

Disp[x_] := {{Cos[x[[1]]], -Sin[x[[1]]], x[[2]]}, {Sin[x[[1]]],
    Cos[x[[1]]], x[[3]]}, {0, 0, 1}};

RRRInverseKinematics[hframestogframe_, a1_, a2_, sgn_] :=
 Module[{npos, hFramesOrigIngFrame,
   s2, \[Theta]1, \[Theta]2, \[Theta]3, \[Psi], \[Alpha]},
  npos = Length[hframestogframe];
  hFramesOrigIngFrame =
   Table[{hframestogframe[[i, 1, 3]], hframestogframe[[i, 2, 3]]}, {i,
      npos}];
  s2 = Table[Norm[hFramesOrigIngFrame[[i]]]^2 , {i, npos}];
  \[Theta]2 =
   Table[ sgn*ArcCos[(s2[[i]] - a1^2 - a2^2)/(2 a1 a2)], {i, npos}];
  \[Psi] =
   Table[ArcTan[a1 + a2 Cos[\[Theta]2[[i]]],
     a2 Sin[\[Theta]2[[i]]]], {i, npos}];
  \[Theta]1 =
   Table[ArcTan[hFramesOrigIngFrame[[i, 1]],
      hFramesOrigIngFrame[[i, 2]]] - \[Psi][[i]], {i,
      npos}]; \[Alpha] =
    Table[ArcTan[hframestogframe[[i, 1, 1]],
      hframestogframe[[i, 2, 1]]], {i, npos}];
   \[Theta]3 =
    Table[\[Alpha][[i]] - \[Theta]1[[i]] - \[Theta]2[[i]], {i, npos}];
   {\[Theta]1, \[Theta]2, \[Theta]3}] (*  \
\[Theta]1+\[Theta]2+\[Theta]3=\[Alpha] thus \[Theta]3=\[Alpha]-\
\[Theta]1-\[Theta]2  *)

(*RandomVariables[RangeT_,RangeX_,RangeY_,TaskP_]:=Module[{Xv,Yv,Tv,\
data,position,ptA,ptB,K},
Xv=RandomReal[RangeX,5];
Yv=RandomReal[RangeY,5];
Tv=RandomReal[RangeT,5]*Degree;
K=TaskP;
data=Table[{(K[[i,1]]+Tv[[i]]),K[[i,2]]+Xv[[i]],K[[i,3]]+Yv[[i]]},{i,\
5}];
position=Table[Disp[data[[i]]],{i,5}];
{position,data}];*)

RandomVariables5[AngleLimits_, xLimits_, yLimits_, data_] :=
 Module[{xTemp, yTemp, angleTemp, dataTemp, positionTemp},

  dataTemp = positionTemp = ConstantArray[0, 5];

  Table[
   angleTemp = RandomReal[AngleLimits[[i]]]*Degree;
```

```
    xTemp = RandomReal[xLimits[[i]]];
    yTemp = RandomReal[yLimits[[i]]];
    (*Print[{angleTemp,xTemp,yTemp}];*)
    dataTemp[[i]] = {(data[[i, 1]] + angleTemp), data[[i, 2]] + xTemp,
      data[[i, 3]] + yTemp};
    positionTemp[[i]] = Disp[dataTemp[[i]]];
    , {i, 5}];

  {dataTemp, positionTemp}]

DiffBetAngs[a_, b_] := Module[{VecA, VecB, diff1, diff2, finalDiff},
  VecA = {Cos[a], Sin[a]};
  VecB = {Cos[b], Sin[b]};
  diff1 = Mod[Abs[JointAngle[VecA, VecB]], 6.28];
  diff2 = Mod[Abs[JointAngle[VecB, VecA]], 6.28];
  If[diff1 <= diff2, finalDiff = diff1, finalDiff = diff2];
  N[Abs[finalDiff]]]

(*Difference between angle vectors*)
NormofDiffBetAngVectors[a_, b_] :=
  Module[{VecA, VecB, diff1, diff2, finalDiff},
    VecA = Table[{Cos[a[[i]]], Sin[a[[i]]]}, {i, Length[a]}];
    VecB = Table[{Cos[b[[i]]], Sin[b[[i]]]}, {i, Length[b]}];

    diff1 = Table[
      If[Mod[JointAngle[VecA[[i]], VecB[[i]]], 2 \[Pi]] <
        Mod[JointAngle[VecB[[i]], VecA[[i]]], 2 \[Pi]],
        Mod[JointAngle[VecA[[i]], VecB[[i]]], 2 \[Pi]],
        Mod[JointAngle[VecB[[i]], VecA[[i]]], 2 \[Pi]]]
       , {i, Length[a]}];
    (*Print[diff1];*)

    N[Norm[diff1]]];

MakeCylindrical[list_, rad_] := Block[{newList, list2},
   list2 = list Degree;
   newList = Table[
     {list[[i, 1]], rad*Cos[list2[[i, 2]]], rad*Sin[list2[[i, 2]]]}
     , {i, Length[list]}];
   (*newList=Table[
   Table[
   {list[[i,j,1]],rad*Cos[list2[[i,j,2]]],rad*Sin[list2[[i,j,2]]]}
   ,{j,Length[list[[i]]]}]
   ,{i,Length[list]}];*)
   newList]

MakeDivisionsinTps2[angSequence_, inc_] :=
 Module[{theta1, theta2, theta3, theta4, theta5, Thetas, minidivs,
    ipAngles, angstemp, FiveTpsLocs, miniRangeAntiClockwise,
    miniRangeClockwise, rangeOfMotionforLink1},

   Thetas = {theta1, theta2, theta3, theta4, theta5} = angSequence;
   (*Print["five theta1 = ",Thetas/Degree];*)
   ipAngles = 0; rangeOfMotionforLink1 = 0;
```

```
FiveTpsLocs = {1, 0, 0, 0, 0};

Table[
 miniRangeAntiClockwise =
  N[JointAngle[{Cos[Thetas[[i]]],
      Sin[Thetas[[i]]]}, {Cos[Thetas[[i + 1]]],
      Sin[Thetas[[i + 1]]]}]];
 (*Print["miniRange = ",N[miniRange/Degree]];*)
 miniRangeClockwise =
  N[2 \[Pi] -
     JointAngle[{Cos[Thetas[[i]]],
       Sin[Thetas[[i]]]}, {Cos[Thetas[[i + 1]]],
       Sin[Thetas[[i + 1]]]}]];
 (*Print[
 "{miniRangeAntiClockwise,miniRangeClockwise} = \
",{miniRangeAntiClockwise,miniRangeClockwise}/Degree];*)

 If[miniRangeAntiClockwise <= miniRangeClockwise,
  rangeOfMotionforLink1 =
   rangeOfMotionforLink1 + miniRangeAntiClockwise;
  minidivs = Floor[miniRangeAntiClockwise/inc];
  (*Print["minidivs = ",N[minidivs]];*)
  angstemp =
   Table[Mod[Thetas[[i]] + j*inc, 2 \[Pi]], {j, 0, minidivs}],

  rangeOfMotionforLink1 = rangeOfMotionforLink1 + miniRangeClockwise;
  minidivs = Floor[miniRangeClockwise/inc];
```

```
    (*Print["minidivs = ",N[minidivs]];*)
    angstemp =
     Table[Mod[Thetas[[i]] - j*inc, 2 \[Pi]], {j, 0, minidivs}]];

   FiveTpsLocs[[i + 1]] = FiveTpsLocs[[i]] + minidivs + 1;
   If[Length[ipAngles] == 0, ipAngles = angstemp,
    ipAngles = Flatten[Append[ipAngles, angstemp]]];
   , {i, 4}];

  (*Print["FiveTpsLocs = ",FiveTpsLocs];*)
  ipAngles = N[Flatten[Append[ipAngles, theta5]]];
  (*Print["rangeOfMotionforLink1 in deg = ",rangeOfMotionforLink1/
  Degree];*)
  {ipAngles, FiveTpsLocs, rangeOfMotionforLink1}]

LinkageLinkLenBoundCheck4R[linkage_, RRcons_, MaxLen_, MinLen_] :=
 Block[{lenList, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, linkList,
   linkList2, linkLengthArray, maxlen, minlen, linkageCheck, p1, p5,
   p1loc, p5loc, pt1, pt5},
  (*Print["linkage = ",linkage];
  Print["RRcons = ",RRcons];*)



  linkageCheck = 1;
  lenList = {};
  {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10} = linkage;
```

```
p1 = Flatten[Position[RRcons, 1], 1];
p1loc = 4 + 2*(p1[[1]] - 1) + p1[[2]];
p5 = Flatten[Position[RRcons, 5], 1];
p5loc = 4 + 2*(p5[[1]] - 1) + p5[[2]];
(*Print["{p1,p1loc,p5,p5loc} = ",{p1,p1loc,p5,p5loc}];*)
pt1 = linkage[[p1loc]];
pt5 = linkage[[p5loc]];

linkList = {{c1, pt1}, {c1, c2}, {c2, c3}, {c3, c4}, {c4, pt5}, {c5,
    c6}, {c7, c8}, {c9, c10}};
(*Print[];Print[];Print["linkList = ",linkList];*)

linkList2 = Join[
  {{linkList[[RRcons[[1, 1]], 1]],
    c5}, {linkList[[RRcons[[1, 1]], 2]], c5},
   {linkList[[RRcons[[1, 2]], 1]],
    c6}, {linkList[[RRcons[[1, 2]], 2]], c6},
   {linkList[[RRcons[[2, 1]], 1]],
    c7}, {linkList[[RRcons[[2, 1]], 2]],
    c7}, {linkList[[RRcons[[2, 2]], 1]],
    c8}, {linkList[[RRcons[[2, 2]], 2]], c8},
   {linkList[[RRcons[[3, 1]], 1]],
    c9}, {linkList[[RRcons[[3, 1]], 2]],
    c9}, {linkList[[RRcons[[3, 2]], 1]],
    c10}, {linkList[[RRcons[[3, 2]], 2]], c10}}, {{c5, c6}, {c7,
    c8}, {c9, c10}, {c1, c2}, {c2, c3}, {c3, c4}, {c4, pt5}, {c1,
    pt1}}];
  (*Print[];Print[];Print[{linkList2,Length[linkList2]}];*)

  linkLengthArray = ConstantArray[0, Length[linkList2]];
  Table[linkLengthArray[[i]] =
    LinkLength[linkList2[[i, 1]], linkList2[[i, 2]]]];
   If[linkLengthArray[[i]] <= 10^-3, linkLengthArray[[i]] = 0], {i,
    Length[linkList2]}];
  linkLengthArray = Cases[linkLengthArray, Except[0]];
  (*Print[MatrixForm[linkLengthArray]];*)
  {maxlen, minlen} = {Max[linkLengthArray], Min[linkLengthArray]};
  (*Print["{Maxlen,Minlen} allowed = ",{MaxLen,MinLen}];
  Print["{maxlen,minlen} = ",{maxlen,minlen}];*)

  If[maxlen > MaxLen || minlen < MinLen, linkageCheck = 0;
   (*,Print["Linkage comply with min/max length requirement"];*)];

  {linkageCheck, maxlen/minlen}
  ]

GenRndTaskPosforLoops[data_, TolTheta_, TolX_, TolY_, noOfLoops_] :=
 Block[{dataTemp, positionTemp, position},
  position = Disp[#] & /@ data;
  dataTemp = positionTemp = ConstantArray[0, noOfLoops - 1];
  Table[
   {dataTemp[[i]], positionTemp[[i]]} =
    RandomVariables5[TolTheta, TolX, TolY, data]
   , {i, 1, noOfLoops - 1}];
```

```
dataTemp = Join[{data}, dataTemp];
positionTemp = Join[{position}, positionTemp];
Print["Length of dataTemp and positionTemp = ", {Length[dataTemp],
  Length[positionTemp]}];
{dataTemp, positionTemp}]

FindChainAngles[noOfJointsinChain_, sideLen_, ptC1_, ptsC3_] :=
Block[{noOfLinks, noLinksOneSide, Halflink, constTerm, c1c3Distances,
   AnglesVal, Eq1, Eq2, IntAng, th1, kconst, Theta1s, InteriorAngs,
  NotReachable, fiveCrankAngles, fivec1c3Angles},

c1c3Distances = LinkLength[ptC1, #] & /@ ptsC3;
(*Print["c1c3Distances = ",c1c3Distances];*)
noOfLinks = noOfJointsinChain - 1;
Table[If[
   c1c3Distances[[i]] >
    noOfLinks*sideLen, {Theta1s, InteriorAngs} = {0, 0};(*Print[
   "Increase the length of the chain link"];*)Goto [NotReachable]];
  , {i, 5}];

(*fivec1c3Angles=Table[JointAngle[{1,0},ptsC3[[i]]-ptC1],{i,5}];*)
(*Print[fivec1c3Angles/Degree];*)
noLinksOneSide = Floor[noOfLinks/2];
Halflink = Mod[noOfLinks, 2];
constTerm = N[Halflink*sideLen/2];
(*Print[
"{no of joints, no of links, no of links on one side, half link \
present ,constTerm} = ",{noOfJointsinChain ,noOfLinks ,noLinksOneSide ,
  Halflink ,constTerm}];*)

  AnglesVal = Prepend[ConstantArray[0, noLinksOneSide - 1], th1];
  kconst = 0;
  Table[
   (*Print[" i = ",i];*)
   AnglesVal[[
     i]] = (If[Mod[noOfJointsinChain, 2] == 0, 2, 1] +
       kconst)*(\[Pi]/2 - IntAng/2);
   kconst = kconst + 2;
   , {i, noLinksOneSide, 2, -1}];
  (*Print[" AnglesVal = ",AnglesVal];*)

  Eq1 = sideLen*Total[Cos[#] & /@ AnglesVal] + constTerm;
  (*Print["Eq1 = ",Eq1];*)
  Eq2 = 2 (th1) + (noOfJointsinChain - 2) (IntAng - \[Pi]);
  (*Print["Eq2 = ",Eq2];*)

  Theta1s = InteriorAngs = ConstantArray[0, 5];
  Table[
   {Theta1s[[i]], InteriorAngs[[i]]} = {th1, IntAng} /.
      FindRoot[{Eq1 == c1c3Distances[[i]]/2(*(40/2)*),
        Eq2 == 0}, {{th1, 45 Degree}, {IntAng, 135 Degree}}];
   , {i, 5}];

  (*fiveCrankAngles=Table[Mod[fivec1c3Angles[[i]]-Theta1s[[i]],
  2\[Pi]],{i,5}];*)
  (*{Theta1s,InteriorAngs}={th1,IntAng}/.FindRoot[{Eq1\[Equal](40/2),
  Eq2\[Equal]0},{{th1,45 Degree},{IntAng,135 Degree}}];*)

  Label [NotReachable];
  {Theta1s, InteriorAngs}]

FindAllPointsforChain[noOfJointsinChain_, sideLen_, ptC1_, ptsC3_,
  fiveThetas1_, fiveIntAngs_, elbowPos_] :=
 Block[{noOfIntPts, intPtsxy, noOfPtsonOneside, fiveGndAngles,
   c1c3Distances, Thetas, noLinksOneSide, noOfLinks, len, temp,
   fiveSubsChains, fiveSubsChainsRotated,
   fiveSubsChainsRotatedShiftedtoC1, FinalFiveChains},

  c1c3Distances = LinkLength[ptC1, #] & /@ ptsC3;
  (*Print["fivec1c3 dists = ",c1c3Distances];*)
  fiveGndAngles = JointAngle[{1, 0}, # - ptC1] & /@ ptsC3;
  (*Print["fiveGndAngles = ",fiveGndAngles/Degree];*)



  noOfLinks = noOfJointsinChain - 1;
  noLinksOneSide = Floor[noOfLinks/2];
  intPtsxy =
   ConstantArray[ConstantArray[{0, 0}, 5], noOfJointsinChain - 2];
  Thetas = ConstantArray[{0, 0, 0, 0, 0}, noLinksOneSide];
  Thetas[[1]] = fiveThetas1;
```

```
Table[
 Thetas[[i]] = Mod[fiveIntAngs - (\[Pi] - Thetas[[i - 1]]), 2 \[Pi]];
 , {i, 2, Length[Thetas]}];



len = Length[intPtsxy];
temp = ConstantArray[{0, 0}, 5];
Table[
 intPtsxy[[i]] =
  Table[temp[[j]] +
    sideLen*{Cos[Thetas[[i, j]]], Sin[Thetas[[i, j]]]}, {j, 5}];
 If[i < (len - i + 1),
  intPtsxy[[len - i + 1]] = Table[
    {c1c3Distances[[j]], 0} + {-temp[[j, 1]], temp[[j, 2]]} +
     sideLen*{-Cos[Thetas[[i, j]]], Sin[Thetas[[i, j]]]}
    , {j, 5}]];
 temp = intPtsxy[[i]];
 , {i, Ceiling[len/2]}];




If[elbowPos == -1,
 intPtsxy[[All, All, 2]] = -intPtsxy[[All, All, 2]]];
fiveSubsChains = Transpose[intPtsxy];
fiveSubsChainsRotated = Table[
  {{1, 0, 0}, {0, 1, 0}}.Zmat[fiveGndAngles[[i]]].hom[#] & /@
   fiveSubsChains[[i]]
```

```
    , {i, 5}] ;
  fiveSubsChainsRotatedShiftedtoC1 =
   Table[(# + ptC1) & /@ fiveSubsChainsRotated[[i]], {i,
     Length[fiveSubsChainsRotated]}];
  FinalFiveChains = Table[
    Join[{ptC1}, fiveSubsChainsRotatedShiftedtoC1[[i]], {ptsC3[[i]]}]
    , {i, 5}];

  FinalFiveChains]


  FillAFrames[FinChains_, data_] :=
 Block[{AFrames, FinChains2, len, tempAngle, linkAngles5},
  FinChains2 = FinChains;
  Table[FinChains2[[i]] =
    Append[FinChains2[[i]], {data[[i, 2]], data[[i, 3]]}], {i, 5}];
  (*Print[FinChains2];*)
  len = 2*Length[FinChains2[[1]]] - 2;
  (*Print[len];*)
  AFrames = ConstantArray[0, len];
  AFrames[[1]] =
   Table[Disp[{0, FinChains2[[1, 1, 1]], FinChains2[[1, 1, 2]]}], {i,
     5}];
  linkAngles5 = ConstantArray[0, 4];

  Table[
   (*Print[i];*)
   linkAngles5[[i - 1]] =
    Table[JointAngle[{1, 0},
      FinChains2[[k, i]] - FinChains2[[k, i - 1]]], {k, 5}];
   AFrames[[i]] =
    Table[Disp[
      Flatten[{JointAngle[{1, 0},
         FinChains2[[j, i]] - FinChains2[[j, i - 1]]],
        FinChains2[[j, i - 1]]}]], {j, 5}];
   , {i, 2, Length[FinChains2[[1]]]}];
  (*Print[linkAngles5/Degree];*)

  {linkAngles5, AFrames}]
(*AFrames2=FillAFrames[FinChains,data];*)

(*Check the equivalence between two lists, straight or inverted*)
CheckEquivalence[A_, B_] :=
 If[Chop[Norm[A - B]] <= 10^-3 || Chop[Norm[Reverse[A] - B]] <= 10^-3,
   1, 0]

(*Combining only two lists at a time*)
CombineTwoListsAtaTime[A_, B_] := Block[{sol},
  sol = {};
  (*Print["Total no of solutions possible are = ",Length[A]*Length[
  B]];*)
  Table[
   Table[
    sol = Append[sol, {A[[i]], B[[j]]}];
    , {j, Length[B]}]
   , {i, Length[A]}];
  (*Print["sol = ",sol];*)
  sol]

(*Combining any no of lists of lists of different sizes and depths*)
CombiGenerator[list_] :=
 Block[{depth, noOfSols, sol, solfin, soltemp, list2, OnelistisZero},
  depth = Length[list];
  list2 = Cases[list, Except[0]];
  If[Length[list2] < depth, solfin = 0; Goto [OnelistisZero]];
  noOfSols = 1;
  Table[noOfSols = noOfSols*Length[list[[i]]], {i, depth}];
  (*Print["depth = ",depth];*)
  (*Print["list = ",list];*)
  (*Print["noOfSols possible = ",noOfSols];*)

  sol = list[[1]];
  Table[
   sol = CombineTwoListsAtaTime[sol, list[[i + 1]]];
   (*Print["i = ",i];
   Print["sol current ",sol];*)
   , {i, depth - 1}];
  (*Print["sol before post processing = ",MatrixForm[sol]];*)

  (*post processing*)
  solfin = ConstantArray[0, noOfSols];
  Table[
    soltemp = Flatten[sol[[i]]];
    solfin[[i]] =
     Table[Table[soltemp[[9*(j - 1) + k]], {k, 9}], {j, depth}];
    (*Print["soltemp = ",soltemp];
    Print["solfin = ",solfin[[i]]];*)
    , {i, noOfSols}];
  (*Print["sol after post processing = ",solfin];*)
  Label [OnelistisZero];
  solfin]

(*Retrieve location of a specific RRlinkpair from a list of link \
pairs like FirstRRpairs*)
locationInList[a_, list_] := Block[{loc},
  loc = 0;
  Table[
   If[Norm[Abs[a - list[[i]]]] < 10^-4, loc = i]
   , {i, Length[list]}];
  loc]

(*Fing new Aframe given the RRlink solution with the five angles made \
by the link, RRlinkpairList and the location of AFrames to be filled*)

FindNewAFrame[AFrames_, RRlinkSol_, RRlinkpair_,
  AFrameLoctobeFilled_] :=
 Block[{rem, fiveptsA, fiveptsB, fiveJointAngles, AFrames2, n},

  rem = {{1, 0, 0}, {0, 1, 0}};
  AFrames2 = AFrames;
  fiveptsA =
   Table[rem.AFrames2[[RRlinkpair[[1]]]][[n]].Inverse[
      AFrames2[[RRlinkpair[[1]]]][[1]]].{RRlinkSol[[1]],
      RRlinkSol[[2]], 1}, {n, 5}];
  fiveptsB =
   Table[rem.AFrames2[[RRlinkpair[[2]]]][[n]].Inverse[
      AFrames2[[RRlinkpair[[2]]]][[1]]].{RRlinkSol[[3]],
      RRlinkSol[[4]], 1}, {n, 5}];
  fiveJointAngles = RRlinkSol[[5 ;; 9]];
  (*Print["ptsA = ",fiveptsA];
  Print["ptsB = ",fiveptsB];
  Print["fiveJointAngles = ",fiveJointAngles/Degree];*)
  AFrames2[[AFrameLoctobeFilled]] =
   Table[Disp[{fiveJointAngles[[n]], fiveptsA[[n, 1]],
      fiveptsA[[n, 2]]}], {n, 5}];
  (*Print["curr A frame = ",MatrixForm[AFrames2[[
  AFrameLoctobeFilled]]]];*)

  AFrames2]

(*Synthesis of RR constraint link*)
(*Remember data is a global variable*)
RRConstraintSyn[A_, B_, preLinks_] :=
 Module[{ASdisp, BSdisp, W, x, y, G, u, v, R, ConstraintEqn,
   DesignEQN, numsol1, RawSolution, RealSolution, OneSolution,
   EndFunction, CheckNextSolution, FilteredSolution, ptS1, ptS2, rem,
   Userlnk, fiveptsA, fiveptsB, fiveLinkAngles, preLinks2, maxDist,
   RealSolTemp},

  maxDist =
   Max[LinkLength[{data[[1, 2]], data[[1, 3]]}, {data[[#, 2]],
        data[[#, 3]]}] & /@ {2, 3, 4, 5}];
  (*Print["maxDist = ",maxDist];*)

  OneSolution = 0;
  FilteredSolution = 0;
  rem = {{1, 0, 0}, {0, 1, 0}};

  ptS1 = rem.A[[1]].{0, 0, 1};
  ptS2 = rem.B[[1]].{0, 0, 1};
  Userlnk = Flatten[{ptS1, ptS2}];
  (*Print["Userlnk = ",Userlnk];
  Print["prelinks = ",MatrixForm[preLinks]];*)
  preLinks2 = Append[preLinks, Userlnk];
  (*Print["prelinks after appending = ",MatrixForm[preLinks2]];*)



  ASdisp = Table[Chop[A[[i]].Inverse[A[[1]]]], {i, 5}];
  BSdisp = Table[Chop[B[[i]].Inverse[B[[1]]]], {i, 5}];
  G = {u, v, 1};
  W = {x, y, 1};
  ConstraintEqn =
   Table[Expand[
       Dot[ASdisp[[i]].G - BSdisp[[i]].W,
         ASdisp[[i]].G - BSdisp[[i]].W]] - R^2, {i, 5}];
  DesignEQN =
   Table[Chop[ConstraintEqn[[i + 1]] - ConstraintEqn[[1]]], {i,
     5 - 1}];
  numsol1 = NSolve[DesignEQN == {0, 0, 0, 0}, {u, v, x, y}];
  RawSolution = Sort[{u, v, x, y} /. numsol1];
  (*Print["RawSol Solution = ",MatrixForm[RawSolution]];*)



  (*Removing the imaginary solutions*)
  RealSolution =
   Table[Table[
      If[Im[RawSolution[[i, j]]] != 0, 0, RawSolution[[i, j]]], {j,
       4}], {i, Length[RawSolution]}];
  RealSolution = Chop[Cases[RealSolution, Except[{0, 0, 0, 0}]]];



  (*Removing the solutions that are not numeric*)
  Table[
   If[NumberQ[RealSolution[[i, 1]]] == False ||
       NumberQ[RealSolution[[i, 2]]] == False ||
       NumberQ[RealSolution[[i, 3]]] == False ||
       NumberQ[RealSolution[[i, 4]]] == False,
      (*kkk++;
      Print[RealSolution[[i]]];
      Print[maxDist*10^2];Print[];*)
      RealSolution[[i]] = {0, 0, 0, 0}];
    , {i, Length[RealSolution]}];

  RealSolTemp = RealSolution;




  (*Removing the solutions that are large*)
  Table[
    (*Print["Current Sol = ",RealSolution[[i]]];
    Print["sum of Abs of 4 vals = ",(Abs[RealSolution[[i,1]]]+Abs[
    RealSolution[[i,2]]]+Abs[RealSolution[[i,3]]]+Abs[RealSolution[[i,
    4]]])];*)
    If[(Abs[RealSolution[[i, 1]]] + Abs[RealSolution[[i, 2]]] +
          Abs[RealSolution[[i, 3]]] + Abs[RealSolution[[i, 4]]]) >
       maxDist*10,
      (*kkk++;
      Print[RealSolution[[i]]];
      Print[maxDist*10^2];Print[];*)
      RealSolution[[i]] = {0, 0, 0, 0}];
    , {i, Length[RealSolution]}];
  RealSolution = Chop[Cases[RealSolution, Except[{0, 0, 0, 0}]]];
  (*Print["Real Solutions ",MatrixForm[RealSolution]];*)
  If[Length[RealSolution] == {} || 0, OneSolution = 0;
```

```
   FilteredSolution = 0; Goto [EndFunction];];
  (*Print["Real Solutions ",MatrixForm[RealSolution]];*)




  (*Removing the solutions that are already present as part of the \
user defined chain or known solutions*)
  Table[
   Table[
    If[Chop[Norm[RealSolution[[j]] - preLinks2[[i]]]] <= 10^-3 ||
       Chop[Norm[Reverse[RealSolution[[j]]] - preLinks2[[i]]]] <= 10^-3
      , RealSolution[[j]] = {0, 0, 0, 0}; Goto [CheckNextSolution]];
    , {i, Length[preLinks2]}];

   Label [CheckNextSolution];
   , {j, Length[RealSolution]}];
  FilteredSolution = Cases[RealSolution, Except[{0, 0, 0, 0}]];
  (*Print["Filtered Solution = ", MatrixForm[FilteredSolution]];
  Print[];*)
  (*Print[];Print[];*)
  If[Length[FilteredSolution] == 0, OneSolution = 0;
   FilteredSolution = 0; Goto [EndFunction]];




  (*Capturing the angle for each of the synthesized RR constraints*)
  Table[
   fiveptsA =
    rem.#.{FilteredSolution[[i, 1]], FilteredSolution[[i, 2]], 1} & /@
      ASdisp;
   fiveptsB =
    rem.#.{FilteredSolution[[i, 3]], FilteredSolution[[i, 4]], 1} & /@
      BSdisp;
   fiveLinkAngles =
    Table[JointAngle[{1, 0}, fiveptsB[[j]] - fiveptsA[[j]]], {j, 5}];
   (*Print[fiveLinkAngles/Degree];*)
   FilteredSolution[[i]] = Join[FilteredSolution[[i]], fiveLinkAngles];
   , {i, Length[FilteredSolution]}];



  OneSolution = FilteredSolution[[1]];
  Label[EndFunction];

  (*Print["FilteredSolution = ",MatrixForm[FilteredSolution]];*)
  {RealSolTemp, FilteredSolution}]

updatepreLinks[prelinks2_, currRRpair_, currLinkNo_, currRRSol_] :=
  Block[{prelinks3, upPreLinks, zeroCd, c1, c2, c3, c4, RRcoords2,
   newpreLinks},

  prelinks3 = {0, 0};
  (*Print["prelinks2 first part = ",MatrixForm[prelinks2[[1]]]];
  Print["prelinks2 second part = ",MatrixForm[prelinks2[[2]]]];*)
  (*Print["currRRpair = ",currRRpair];
  Print["currLinkNo = ",currLinkNo];*)

  RRcoords2 = prelinks2[[1]];
  RRcoords2 =
   Append[RRcoords2, {{currRRSol[[1]],
      currRRSol[[2]]}, {currRRSol[[3]], currRRSol[[4]]}}];
  prelinks3[[1]] = RRcoords2;
  (*Print["updated RRcoords2 = ",prelinks3[[1]]];*)

  newpreLinks = {Flatten[{RRcoords2[[currLinkNo, 1]],
      RRcoords2[[currRRpair[[1]], 1]]}]};
  newpreLinks =
   Append[newpreLinks,
    Flatten[{RRcoords2[[currLinkNo, 1]],
      RRcoords2[[currRRpair[[1]], 2]]}]];
  newpreLinks =
   Append[newpreLinks,
    Flatten[{RRcoords2[[currLinkNo, 2]],
      RRcoords2[[currRRpair[[2]], 1]]}]];
  newpreLinks =
   Append[newpreLinks,
    Flatten[{RRcoords2[[currLinkNo, 2]],
      RRcoords2[[currRRpair[[2]], 2]]}]];
  prelinks3[[2]] = Join[prelinks2[[2]], newpreLinks];
  (*Print["up prelinks first part = ",MatrixForm[prelinks3[[1]]]];
  Print["up prelinks second part = ",MatrixForm[prelinks3[[2]]]];*)
```

```
  prelinks3]

SynthesisForDependentPairs2[AFrames_, jointList_, preLinks_,
  RRpairList_, FirstRRpairs_] :=
 Block[{noOfIndepConpairs, firstSols, soltemp, solSpecial,
   locNewAframes, rem, fiveptsA, fiveptsB, fiveJointAngles, AFrames2,
   AFramesforCurrSol, AFrames3, preLinks2, finSol, ThisOneFailed,
   AfrmPointer, n, solution, soltemp2, linkptobeCon, AfrmLoc,
   AfrmLocIndep, ktemp, soltemp3, AFramesTemp, RRlinksols, j,
   solcurrtemp, soltemp4, c1, c2, c3, c4, zeroCd, RRcoords,
   linkageTemp, anglesTemp},

  finSol = 0; zeroCd = {0, 0};
  (*{c1,c2,c3,c4}={{preLinks[[1,1]],preLinks[[1,2]]},{preLinks[[1,3]],
  preLinks[[1,4]]},{preLinks[[2,3]],preLinks[[2,4]]},{preLinks[[3,3]],
  preLinks[[3,4]]}};*)
  (*RRcoords={{zeroCd,c1},{c1,c2},{c2,c3},{c3,c4},{c4,zeroCd}};*)
  RRcoords =
   Flatten[{{{zeroCd, jointList[[1]]}},
     Table[{jointList[[k]], jointList[[k + 1]]}, {k,
       Length[jointList] - 1}], {{Last[jointList], zeroCd}}}, 1];
  (*Print[RRcoords];*)
  preLinks2 = {RRcoords, preLinks};
  (*Print["preLinks2 = ",preLinks2];*)



  (*Print["FirstRRpairs = ",FirstRRpairs];*)
```

```
firstSols =
 Table[RRConstraintSyn[AFrames[[FirstRRpairs[[i, 1]]]],
    AFrames[[FirstRRpairs[[i, 2]]]], preLinks][[2]], {i,
   Length[FirstRRpairs]}];
AFrames2 = AFrames;
(*Print["currRRsol = ",RRpairList];*)
noOfIndepConpairs =
 Length[RRpairList] -
  Length[Complement[Union[RRpairList, FirstRRpairs], FirstRRpairs]];
(*Print["noOfIndepConpairs = ",noOfIndepConpairs];*)
linkptobeCon = noOfIndepConpairs + 1;
AfrmLocIndep = Length[AFrames] - Count[AFrames, 0];
AfrmLoc = Length[AFrames] - Count[AFrames, 0] + linkptobeCon;
(*Print["{linkptobeCon,AfrmLoc} = ",{linkptobeCon,AfrmLoc}];*)
soltemp = ConstantArray[0, noOfIndepConpairs];



(*Find all the solutions upto the no of noOfIndepConpairs*)
(*Print[];*)
Table[
 soltemp[[i]] =
   firstSols[[locationInList[RRpairList[[i]], FirstRRpairs]]];
 (*Print["length of sol no ",i, " = ",Length[soltemp[[i]]]];*)
 , {i, noOfIndepConpairs}];
soltemp2 = CombiGenerator[soltemp];
(*Print[soltemp2[[1]]];*)
If[soltemp2 == 0, finSol = 0; Goto [ThisOneFailed]];
```

```
  Table[soltemp2[[i]] =
    Join[{{linkptobeCon, AfrmLoc, RRpairList, AFrames,
       preLinks2}}, {soltemp2[[i]]}], {i, Length[soltemp2]}];
  (*Table[Print[MatrixForm[soltemp2[[kk]]]],{kk,1,1(*Length[
  soltemp2]*)}];*)
  (*Print[];Print[];*)




  (*Update the AFrames and preLinks for each of the solutions from \
soltemp2 owing to the independent link pairs*)
  Table[
   (*Print[];
   Print["******************************************************************
];
   Print["Sol no = ",i];
   Print["RRpairlist = ",soltemp2[[i,1,3]]];
   Print["Sol is = ",MatrixForm[soltemp2[[i,2]]]];*)
   Table[
     (*Print[{soltemp2[[i,2]]}];*)
     (*Print[{soltemp2[[i,2,j]],currRRsol[[j]],AfrmLocIndep+j}];*)
     soltemp2[[i, 1, 4]] =
      FindNewAFrame[soltemp2[[i, 1, 4]], soltemp2[[i, 2, j]],
       RRpairList[[j]], AfrmLocIndep + j];
     soltemp2[[i, 1, 5]] =
      updatepreLinks[soltemp2[[i, 1, 5]], RRpairList[[j]], 5 + j,
       soltemp2[[i, 2, j]]];
     (*Print[soltemp2[[i,1,4]]];*)
     , {j, noOfIndepConpairs}];
  , {i,(*1,1*)Length[soltemp2]}];




(*Now synthesize the dependent RRlinks using the updated AFrames*)
soltemp3 = soltemp2;
j = 1;
Table[
 (*Print["^^^^^",i];*)

 While[j <= Length[soltemp3],
  {linkptobeCon, AfrmLoc, AFramesTemp} = {soltemp3[[j, 1, 1]],
     soltemp3[[j, 1, 2]], soltemp3[[j, 1, 4]]};
  (*Print["sol no = ",j];
  Print["prelinks 2part = ",MatrixForm[soltemp3[[j,1,5]][[2]]]];
  Print["prelinks 1part = ",MatrixForm[soltemp3[[j,1,5]][[1]]]];*)
  RRlinksols =
   RRConstraintSyn[AFramesTemp[[RRpairList[[linkptobeCon, 1]]]],
     AFramesTemp[[RRpairList[[linkptobeCon, 2]]]],
     preLinks(*soltemp3[[j,1,5]][[2]]*)][[2]];
  (*Print["last pair solutions = ",MatrixForm[RRlinksols]];*)
  soltemp3[[j, 1, 1]]++; soltemp3[[j, 1, 2]]++;
  solcurrtemp = Table[soltemp3[[j]], {k, Length[RRlinksols]}];
  (*Print["len solcurrtemp = ",Length[RRlinksols]];*)
  Table[
     solcurrtemp[[k, 2]] =
      Append[solcurrtemp[[k, 2]], RRlinksols[[k]]];
     solcurrtemp[[k, 1, 4]] =
      FindNewAFrame[solcurrtemp[[k, 1, 4]], RRlinksols[[k]],
       RRpairList[[linkptobeCon]], AfrmLoc];
     (*solcurrtemp[[k,1,4]]=FindNewAFrame[solcurrtemp[[k,1,4]],
     RRlinksols[[k]],RRpairList[[linkptobeCon]],AfrmLoc];*)
     , {k, Length[solcurrtemp]}];

    soltemp3[[j]] = solcurrtemp;
    j++;
    ];

  soltemp3 = Flatten[soltemp3, 1];
  j = 1;
  , {i, Length[RRpairList] - noOfIndepConpairs}];

(*Print["len soltemp3 = ",Length[soltemp3]];*)
(*soltemp4=Table[{RRpairList,jointList,soltemp3[[i,2]]},{i,Length[
soltemp3]}];
finSol=soltemp4;*)
soltemp4 = ConstantArray[0, Length[soltemp3]];
Table[
  linkageTemp =
   Join[jointList,
    Flatten[Table[{{soltemp3[[i, 2, n, 1]],
        soltemp3[[i, 2, n, 2]]}, {soltemp3[[i, 2, n, 3]],
        soltemp3[[i, 2, n, 4]]}}, {n, 3}], 1]];
   anglesTemp = Table[Take[soltemp3[[i, 2, k]], -5], {k, 3}];
   soltemp4[[i]] = {RRpairList, linkageTemp, anglesTemp};
   , {i, Length[soltemp3]}];
  finSol = soltemp4;



  Label [ThisOneFailed];
  finSol]

GeneralSynthesis[FirstRRpairs_, IndependentConstraints_,
  DependentConstraints_, AFrames_, jointList_, openorclose_] :=
 Block[{lenIndep, lenDep, rem, lenJointList, givenLinkSols,
   noOfConstraintsReq, dof, noOfBars, noOfLinks, currRRsol, preLinks,
   synBoxIndep, synBoxDep, firstSols, soltemp, NextAllsolsIndep,
   NextAllsolsDep, noOfIndepConpairs, solSpecial, synBoxIndepFinal,
   synBoxDepFinal, FinalSynSolutions, synsoltemp, linkageTemp,
   anglesTemp, FirstRRpairsSynSols},

  (*Print[
  "################################### SYNTHESIS FUNCTION  \
#######################################"];*)
  (*
  Input:
  FirstRRpairs are the link pairs (linkpairs) that could be selected \
from the user defined chain, to apply an RR constraint.
  Allsols is a list of all the RRsols,
  that is all the different ways in which (three in case of 4R open \
chain) linkpairs can be selected to apply the three RR constraints.
  AFrames is a list of five coordinate frames that capture the \
location of each link of the linkage. For the user defined 4R chain,
  five links are defined.
  Therefore five entries out of 8 will be filled up before hand.
  jointList is list of coordinates of the user defined chain when the \
chain is in the first position. openorclose is an indicator,
  0 means its an open chain and 1 means its a closed chain.

  The responsibility of the synthesis function is to find the \
physical locations of the RR constraints that will be added using the \
Allsols list.
  *)

  lenIndep = Length[IndependentConstraints];
  lenDep = Length[DependentConstraints];
  synBoxIndep = ConstantArray[0, lenIndep];
  synBoxDep = ConstantArray[0, lenDep];
  synBoxIndepFinal = {};
  synBoxDepFinal = {};
  rem = {{1, 0, 0}, {0, 1, 0}};



  (*Print[
 "==========================================================================
];
```

```
  Print["(I) Initial Understanding"];
  Print["==============================================================================
];*)
  (*No of constraints required info and
  Create a list of user defined links.
  Each link is represented by 4 nos (start coord and end coord)*)
  lenJointList = Length[jointList];
  noOfLinks = If[openorclose == 0, lenJointList + 1, lenJointList];
  dof = If[openorclose == 0, lenJointList,
    3 (lenJointList - 1) - 2 lenJointList];
  noOfBars = noOfLinks + dof - 1;
  noOfConstraintsReq = dof - 1;
  givenLinkSols =
   Table[Flatten[{jointList[[i]], jointList[[i + 1]]}], {i,
     lenJointList - 1}];
  If[openorclose == 1,
   givenLinkSols =
    Append[givenLinkSols, Flatten[{jointList[[1]], Last[jointList]}]]];
  (*Print[
 "==============================================================================
];*)
  (*
  Print["User has specified ",If[openorclose\[Equal]0,"an Open chain",
  "a Closed chain"]];
  Print["No of joints (R) specified by the user = ",lenJointList];
  Print["No of links specified by the user = ",noOfLinks];
  Print["dof of the user defined open or closed loop = ",dof];
  Print["Linkage possible = ",noOfBars," bar"];
  Print["No of constraints required to make this a single dof \
linkage(should be equal to dof-1) = ",noOfConstraintsReq];
  (*Print["Verifies by length of each RR conns sol = ",Length[
  Allsols[[1]]]];*)
  Print["User defined links (prelinks) = ",givenLinkSols];
  *)
  preLinks = givenLinkSols;
  (*Print[
  "=================================================================
];
  Print[];Print[];*)




  (*Print[
  "=================================================================
];
  Print["(II) Finding the RRsolutions for the FirstRRpairs as they \
will be required for each RRpairListSol. This is to avoid redundant \
calculations."];
  Print["=============================================================
];
  Print["FirstRRpairs  = ",FirstRRpairs];*)
  FirstRRpairsSynSols = Table[
    RRConstraintSyn[AFrames[[FirstRRpairs[[i, 1]]]],
     AFrames[[FirstRRpairs[[i, 2]]]], preLinks], {i,
    Length[FirstRRpairs]}];

  (*Print[];
  Print["FirstRRpairsSynSols raw = ",Table[MatrixForm[
  FirstRRpairsSynSols[[i,1]]],{i,Length[FirstRRpairs]}]];
  Print["FirstRRpairsSynSols filtered = ",Table[MatrixForm[
  FirstRRpairsSynSols[[i,2]]],{i,Length[FirstRRpairs]}]];
  Print[];*)

  firstSols = FirstRRpairsSynSols[[All, 2]];
  (*firstSols=(*Parallel*)Table[
  RRConstraintSyn[AFrames[[FirstRRpairs[[i,1]]]],AFrames[[
  FirstRRpairs[[i,2]]]],preLinks][[2]],{i,Length[FirstRRpairs]}];*)
  Print["Length of RRsolutions for the FirstRRpairs"];
  Print[Table[Length[firstSols[[i]]], {i, Length[FirstRRpairs]}]];
  (*Print[
 "==============================================================================
];
  Print[];Print[];*)




  (*Print[
 "==============================================================================
];
  Print["(III) Finding the Linkage Solutions for the Independent \
Constraints."];
```

```
  Print["==========================================================================
];*)
  Table[
   (*Print[
   "//////////////////////////////////////////////////////////////////////////
];
   Print["Independent constraints no = ",i, "    with RRpairListSol ",
   IndependentConstraints[[i]]  ];
   Print["/////////////////////////////////////////////////////////////////////
];*)
   currRRsol = IndependentConstraints[[i]];
   soltemp = ConstantArray[0, Length[currRRsol]];

   Table[
    (*Print["RRpair = ",currRRsol[[j]]];*)
    soltemp[[j]] =
     firstSols[[locationInList[currRRsol[[j]], FirstRRpairs]]];
    If[soltemp[[j]] == 0,(*Print[
     "xxxxxx Bad RRlinkpair encountered, go to next RRlinkpairList \
xxxxxx "];*)Goto [NextAllsolsIndep]];
    (*Print["RRsolutions for this RRpair is = ",MatrixForm[soltemp[[
    j]]]];*)
    , {j,(*1,1*)Length[currRRsol]}];

   (*Print["soltemp = ",MatrixForm[soltemp]];*)
   synBoxIndep[[i]] = {currRRsol, CombiGenerator[soltemp]};
   (*Print[
   "Length of the solutions produces by various combinations = ",
   Length[synBoxIndep[[i,2]]]];*)

   (*Post Processing the indep solutions*)
   Table[
    synsoltemp = synBoxIndep[[i, 2, k]];
    (*synBoxIndepFinal=Join[synBoxIndepFinal,{{currRRsol,jointList,
    synsoltemp}}];*)
    linkageTemp =
     Join[jointList,
      Flatten[Table[{{synsoltemp[[n, 1]],
          synsoltemp[[n, 2]]}, {synsoltemp[[n, 3]],
          synsoltemp[[n, 4]]}}, {n, 3}], 1]];
    anglesTemp = Table[Take[synsoltemp[[m]], -5], {m, 3}];
    synBoxIndepFinal =
     Join[synBoxIndepFinal, {{currRRsol, linkageTemp, anglesTemp}}];
    , {k, Length[synBoxIndep[[i, 2]]]}];

   Label [NextAllsolsIndep];
   (*Print[
   "/////////////////////////////////////////////////////////////////////////
];
   Print[];*)
   , {i, Length[IndependentConstraints]}];
```

```
  (*Print[
  "===========================================================================
];
  Print["(IV) Finding the Linkage Solutions for the Dependent \
Constraints."];
  Print["=====================================================================
];*)
  Table[
   (*Print[
   "///////////////////////////////////////////////////////////////////////////
];
   Print["Dependent constraints no = ",i, "    with RRpairListSol ",
   DependentConstraints[[i]]  ];
   Print["///////////////////////////////////////////////////////////////////
];*)
   currRRsol = DependentConstraints[[i]];
   synBoxDep[[i]] =
    SynthesisForDependentPairs2[AFrames, jointList, preLinks,
     currRRsol, FirstRRpairs];
   (*Print["synBoxDep = ",MatrixForm[synBoxDep[[i]]]];*)
   (*Print["Sol length = ",Length[synBoxDep[[i]]]];*)
   If[synBoxDep[[i]] == 0,(*Print[
    "xxxxxx Bad RRlinkpairlist encountered, go to next RRlinkpairList \
xxxxxx "];*)Goto [NextAllsolsDep]];

   (*Post Processing the indep solutions*)
```

```
    Table[

     synBoxDepFinal = Join[synBoxDepFinal, {synBoxDep[[i, k]]}];
     (*Print[MatrixForm[synBoxIndep[[i,2,k]]]];*)
     , {k, Length[synBoxDep[[i]]]}];

    Label [NextAllsolsDep];
    (*Print[
    "////////////////////////////////////////////////////////////////////////////
];
    Print[];*)
    , {i, Length[DependentConstraints]}];
   (*Print[]; Print[];*)




   (*Print[
   "=========================================================================
];
   Print["(V) Final Solutions from Synthesis due to both Indep and Dep \
Constraints."];
   Print["=====================================================================
];
   Print["Length of solutions from Independent Constraints = ",Length[
   synBoxIndepFinal]];
   Print["Length of solutions from Dependent Constraints = ",Length[
   synBoxDepFinal]];*)
```

```
  FinalSynSolutions = Flatten[{synBoxIndepFinal, synBoxDepFinal}, 1];
  (*Print[" Total Length = ",Length[FinalSynSolutions]];
  Print["==============================================================================
];*)


  (*Print[
  "############################## END OF SYNTHESIS FUNCTION \
####################################"];*)
  FinalSynSolutions]

  TranslatingMyAnglestoBriansAnglesandCheck5[thetas2_, thetas3_,
  thetas4_, thetas5_, thetas6_, thetas7_, thetas8_, linkage_,
  RRconnections_, FTLA_, Subs_, J_] :=
 Block[{thetas1fromX, thetas2fromX, thetas3fromX, thetas4fromX,
   thetas5fromX, thetas6fromX, thetas7fromX, thetas8fromX, ftla,
   requiredAngleList, currtemp, NextIteration, j8t1, j1t2, j2t3, j3t4,
    j4t5, j5t8, lnkList, lnkList2, SubsFixedAngles,
   lnkListRRConstraints, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10,
   MyAnglesFive, BriansAnglesFive, OffsetListtoConvertMyAngles, th1,
   th2, th3, th4, th5, th6, th7, th8, JlistFiveTps,
   BriansAnglesFiveConfigs, lnkListReverse,
   lnkListRRConstraintsReverse, j2t1, j1t8, j3t2, j5t4, j4t3, j8t5,
   NextLink, thetas1, ptSecGndPivotloc, ptSecGndPivot,
   MyAnglesPosition1, BrianAnglesPosition1Subs,
   BrianAnglesPosition1, \[Theta]1, \[Theta]2, \[Theta]3, \[Theta]4, \
\[Theta]5, \[Theta]6, \[Theta]7, \[Theta]8, AnglesDiffMineandBrian},
  (*Print[];
  Print["////////////////////  Five Tps check and Angle translation \
Start  //////////////////////////"];*)
  (*Print[];
  Print["Linkage = ",linkage];
  Print["RRconnections = ",RRconnections];*)
  {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10} = linkage;
  SubsFixedAngles = Subs[[2]];
  BrianAnglesPosition1Subs = Subs[[3]];
  (*Print["BrianAnglesPosition1Subs = ",BrianAnglesPosition1Subs];*)
  (*Print["SubsFixedAngles = ",SubsFixedAngles];*)

  (*Calculating the angle of the ground link*)
  ptSecGndPivotloc = locationInList[1, Flatten[RRconnections]] + 4;
  ptSecGndPivot = linkage[[ptSecGndPivotloc]];
  thetas1 = Table[JointAngle[{1, 0}, c1 - ptSecGndPivot], {i, 5}];

  MyAnglesFive = {thetas1fromX, thetas2fromX, thetas3fromX,
     thetas4fromX, thetas5fromX, thetas6fromX, thetas7fromX,
     thetas8fromX} =
    Chop[Mod[{thetas1, thetas2, thetas3, thetas4, thetas5, thetas6,
       thetas7, thetas8}, 2 \[Pi]]];
  (*Print["MyAngles = ",MatrixForm[MyAnglesFive/Degree]];*)

  MyAnglesPosition1 = MyAnglesFive[[All, 1]];
  (*Print["MyAnglesPosition1 = ",MyAnglesPosition1/Degree];*)
  BrianAnglesPosition1 =
   Mod[Arg[{\[Theta]1, \[Theta]2, \[Theta]3, \[Theta]4, \[Theta]5, \
\[Theta]6, \[Theta]7, \[Theta]8} /. BrianAnglesPosition1Subs],
     2 \[Pi]];
  (*Print["BrianAnglesPosition1 = ",BrianAnglesPosition1/Degree];*)

  AnglesDiffMineandBrian =
   Chop[Mod[MyAnglesPosition1 - BrianAnglesPosition1, 2 \[Pi]], 10^-4];
  (*Print["AnglesDiffMineandBrian = ",AnglesDiffMineandBrian/
  Degree];*)

  BriansAnglesFiveConfigs =
   Chop[Mod[MyAnglesFive - AnglesDiffMineandBrian, 2 \[Pi]], 10^-4];
  (*Print["BriansAnglesFiveConfigs = ",MatrixForm[
  BriansAnglesFiveConfigs/Degree]];*)

  (*

  ftla=Flatten[FTLA,1];
  requiredAngleList=ConstantArray[0,Length[ftla]];
  (*Flatten ftla and tabulate all the angles in all the loops in a \
list*)
  Table[
  requiredAngleList[[i]]={ftla[[i,1]],ftla[[i,2]],ftla[[i,4]]}
  ,{i,Length[ftla]}];
  Print["requiredAngleList RAW = ",MatrixForm[
  requiredAngleList/.SubsFixedAngles]];



  (*Remove duplicate angles in the list*)
  Table[
  currtemp=requiredAngleList[[i,3]];
  If[currtemp\[Equal]0,Goto [NextIteration]];
  Table[
  If[requiredAngleList[[j,3]]\[Equal] currtemp,requiredAngleList[[
  j]]={0,0,0}];
  (*requiredAngleList=Cases[requiredAngleList,Except[0]]];*)
  ,{j,i+1,Length[requiredAngleList]}];
  Label[NextIteration];
  ,{i,Length[requiredAngleList]-1}];
  requiredAngleList=Cases[requiredAngleList,Except[{0,0,0}]];
  Print["requiredAngleList after removing duplicates = ",MatrixForm[
  requiredAngleList]];



  (*Make a list of links in terms of Brians joint notations to find \
Brians angles that correspond to my angle list {thetas1 ... thetas8}
 A new list of reverses are added to ensure that links mentioned in \
reverse are detected*)
  lnkListRRConstraints={{Symbol[StringJoin[{"j",ToString[
  RRconnections[[1,1]]],"t6"}]],Symbol[StringJoin[{"j6t",ToString[
  RRconnections[[1,2]]]}]]},{Symbol[StringJoin[{"j",ToString[
  RRconnections[[2,1]]],"t7"}]],Symbol[StringJoin[{"j7t",ToString[
  RRconnections[[2,2]]]}]]},
  {Symbol[StringJoin[{"j",ToString[RRconnections[[3,1]]],"t8"}]],
  Symbol[StringJoin[{"j8t",ToString[RRconnections[[3,2]]]}]]}
  };
  lnkListRRConstraintsReverse={{Symbol[StringJoin[{"j",ToString[
  RRconnections[[1,2]]],"t6"}]],Symbol[StringJoin[{"j6t",ToString[
  RRconnections[[1,1]]]}]]},{Symbol[StringJoin[{"j",ToString[
  RRconnections[[2,2]]],"t7"}]],Symbol[StringJoin[{"j7t",ToString[
  RRconnections[[2,1]]]}]]},
  {Symbol[StringJoin[{"j",ToString[RRconnections[[3,2]]],"t8"}]],
  Symbol[StringJoin[{"j8t",ToString[RRconnections[[3,1]]]}]]}
  };
  Print["{lnkListRRConstraints, lnkListRRConstraintsReverse} = \
",{lnkListRRConstraints,lnkListRRConstraintsReverse}];
  Print[];

  lnkList=lnkList2=Join[{{j1t2,j2t3},{j2t3,j3t4},{j3t4,j4t5}},
  lnkListRRConstraints];
  lnkListReverse=Join[{{j3t2,j2t1},{j4t3,j3t2},{j5t4,j4t3}},
  lnkListRRConstraintsReverse];

  Print["Links using Brian's conventions for which angles are to be \
found = ",lnkList];
  Print["Links in reverse using Brian's conventions for which angles \
are to be found = ",lnkListReverse];
```

```
  Table[
  Table[
  If[lnkList[[i]]\[Equal] Take[requiredAngleList[[j]],{1,2}],(*Print[
  "found"];*) lnkList2[[i]]=requiredAngleList[[j,3]]; Goto [NextLink]];

  If[lnkListReverse[[i]]\[Equal] Take[requiredAngleList[[j]],{1,
  2}],(*Print["Reverse found"];*) lnkList2[[i]]=requiredAngleList[[j,
  3]]+\[Pi];   Goto [NextLink]];

  ,{j,Length[requiredAngleList]}];

  Label[NextLink];
  ,{i,Length[lnkList]}];



  Print["lnkList2 before subs= ",MatrixForm[lnkList2]];
  lnkList2=lnkList2/.SubsFixedAngles;
  Print["MyAngles {thetas1fromX,..., thetas8fromX} in terms of Brians \
angles = ",MatrixForm[lnkList2]];

  OffsetListtoConvertMyAngles={th1,th2,th3,th4,th5,th6,th7,th8}-
  lnkList2;
  Print["OffsetListtoConvertMyAngles = ",OffsetListtoConvertMyAngles/
  Degree];
  (*Table[
  Print[Cases[lnkList2[[i]],_Number]];
  (*Print[Select[lnkList2[[i]],NumberQ]];*)
  (*If[NumberQ[lnkList2[[i]]]\[Equal]True,Print["Number is present"],
  Print["No number"]];*)

  ,{i,Length[MyAnglesFive]}];*)



  BriansAnglesFive = Table[
  Mod[(MyAnglesFive[[i]]+OffsetListtoConvertMyAngles[[i]]),2\[Pi]]
  ,{i,Length[MyAnglesFive]}];
  Print[MatrixForm[BriansAnglesFive/Degree]];

  BriansAnglesFiveConfigs=Chop[Table[BriansAnglesFive [[All,i]],{i,
  5}]];
  Print["BriansAnglesFiveConfigs = ",MatrixForm[
  BriansAnglesFiveConfigs/Degree]];

  JlistFiveTps=Table[
  Sign[J/.Thread[{th2,th3,th4,th5,th6,th7}\[Rule]Take[
  BriansAnglesFive [[All,i]],{2,7}]]]
  ,{i,5}];
  (*Print["Jlist sign for the five tps are ",MatrixForm[
  JlistFiveTps]];*)

  *)

  (*Print[
  "////////////////////  Five Tps check and Angle translation End  \
  ///////////////////////"];*)
    {BrianAnglesPosition1, AnglesDiffMineandBrian,
     BriansAnglesFiveConfigs}]

  (*Function to find the angle the isotropic vector makes with the \
  horizontal. If the vector is imaginary then return i*)
  ConvtoReal[x_] := Module[{px, py, angle, End1},
     (*Print["norm = ",Norm[x]];*)
     If[NumberQ[x] == False || N[Abs[Norm[x] - 1]] > 10^-3,(*Print[
      "WARNING!!"];*)
      angle = I; Goto End1,
      (*px=Re[x];py=Im[x];
      (*Print["Norm ",Norm[x]];*)
      angle=Mod[Chop[JointAngle[{1,0},{px,py}]],2\[Pi]];*)
      angle = Mod[Chop[Arg[x]], 2 \[Pi]]
      ];

     Label End1;
     angle];

  ConditionOpAnglesEightbarNEW[kvals_, kvecs_, opAngle_, depAngles_] :=
    Module[{k, end, theta2, theta3, theta4, theta5, theta6, theta7, i,
      len, Solution, Sol, theta22, theta23, theta24, n},
     (*Filtering solutions for complex infinity and complex nos*)
     len = 20;
     k = 1;
```

```
Solution = ConstantArray[{0, 0, 0, 0, 0, 0, 0, 0}, len];
(*Print["kvals total ",kvals];Print[""];
Print["kvecs total ",kvecs];Print[""];*)
Table[
 n = 1;
 (*Print["kvals = ",kvals[[i]],"     Norm = ",Norm[kvals[[i]]]];*)
 (*NumberQ[x]\[Equal] False||N[Abs[Norm[x]- 1]]>10^-3*)
 If[NumberQ[kvals[[i]]] == False ||
   N[Abs[Norm[kvals[[i]]] - 1]] > 10^-3, Goto [end]];
 Solution[[i, opAngle]] = N[ConvtoReal[kvals[[i]]]];



 If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
  "WARNING!! dividing by zero"];*)
  Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
 Solution[[i, depAngles[[n, 1]]]] =
  N[ConvtoReal[
     kvecs[[i, depAngles[[n, 2, 1]]]]/
      kvecs[[i, depAngles[[n, 2, 2]]]]]];
 If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
  "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
  Goto [end]];
 (*If[Im[Solution[[i,depAngles[[n,1]]]]]\[NotEqual]0,Print[
 "WARNING",Im[Solution[[i,depAngles[[n,1]]]]]];Solution[[i]]=0;
 Goto [end]];*)
 n++;
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;
```

```
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*)Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;
```

```
If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*) Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];
n++;


If[kvecs[[i, depAngles[[n, 2, 2]]]] == 0,(*Print[
 "WARNING!! dividing by zero"];*)
 Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0}; Goto [end]];
Solution[[i, depAngles[[n, 1]]]] =
 N[ConvtoReal[
    kvecs[[i, depAngles[[n, 2, 1]]]]/
     kvecs[[i, depAngles[[n, 2, 2]]]]]]];
If[Im[Solution[[i, depAngles[[n, 1]]]]] == {1},(*Print[
 "Problem found!!"];*) Solution[[i]] = {0, 0, 0, 0, 0, 0, 0, 0};
 Goto [end]];



(*Print["Solution for ",i, " iteration in degrees = ",Solution[[
i]]/Degree];
Print["Solution for ",i, " iteration in radians = ",Solution[[
i]]];*)
(*Solution[[i,]]=N[ConvtoReal[kvecs[[i,12]]/kvecs[[i,10]]]]];
(*If[theta4\[Equal]\[ImaginaryI],theta4\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,10]]]]];
(*If[theta5\[Equal]\[ImaginaryI],theta5\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,9]]]]];
(*If[theta8\[Equal]\[ImaginaryI],theta8\[Equal]10000;Goto [end]];*)

Solution[[i,]]=N[ConvtoReal[kvecs[[i,11]]/kvecs[[i,8]]]]];*)
(*If[theta10\[Equal]\[ImaginaryI],theta10\[Equal]10000;Goto [
end]];*)



(*Print["{Theta2,Theta3,Theta4,Theta5,Theta6,Theta7} = ",{Mod[
2\[Pi]-theta2,2\[Pi]],Mod[theta3,2\[Pi]],Mod[2\[Pi]-theta4,
2\[Pi]],Mod[2\[Pi]-theta5,2\[Pi]],Mod[2\[Pi]-theta6,2\[Pi]],Mod[
2\[Pi]-theta7,2\[Pi]]}/Degree];*)

(*Print[" "];*)
(*
thetaRest=ConvtoReal[#]&/@kvals[[i]]; Print[thetaRest];*)

(*Solution[[k]]={Mod[2\[Pi]-theta2,2\[Pi]],Mod[theta3,2\[Pi]],Mod[
    2\[Pi]-theta4 ,2\[Pi]] ,Mod[2\[Pi]-theta5 ,2\[Pi]] ,Mod[2\[Pi]-theta6 ,
    2\[Pi]] ,Mod[2\[Pi]-theta7 ,2\[Pi]]};
    k++;
    *)
    Label[end];


    , {i, Length[kvals]}];

   Sol = Chop[ Cases[Solution, Except[{0, 0, 0, 0, 0, 0, 0, 0}]]];
   Sol];

ForwardKinematicsEightBar[ipAngles_, Mmat_, Nmat_, Tmat_,
   opAngEigVal_, depAngles_, depAnglesTmatRatios_, FTLA_, Subs_,
   AnglesDiffMineandBrian_, BrianAnglesPosition1_] :=
  Block[{SolutionBox, FTLA2, BrianIpAngle, FixedOffsetforIpAngle, th1,
     ipAnglesOffset ,(*subsLinklength ,subsFixedAngles ,
    subsAnglesinFirstPosition ,*)StartingSolAnglesTemp ,
    StartingSolAngles , opAngEigValno , depAnglesno ,
    ipSubs , \[Theta]2 , \[Theta]2c , kvals , kvecs , subsNM,
    SolutionBoxShortened , BadLinkage(*,MReal,
    NReal,(*\[Theta]2,\[Theta]3,\[Theta]4,\[Theta]5,\[Theta]6,\[Theta]\
7,\[Theta]8,*)S*)},

   (*Print[
   "//////////////////// Forward Kinematics Start \
////////////////////////"];*)
```

```
    SolutionBox =
      SolutionBoxShortened = ConstantArray[0, Length[ipAngles]];
    (*Print["opAngEigVal,depAngles = ",{opAngEigVal,depAngles}];*)
    (*ClearAll[\[Theta]2,\[Theta]3,\[Theta]4,\[Theta]5,\[Theta]6,\
\[Theta]7,\[Theta]8];
    subsLinklength=Subs[[1]];
    subsFixedAngles=Subs[[2]];
    subsAnglesinFirstPosition=Subs[[3]];*)
    (*Print["subsLinklength = ",subsLinklength]*);
    (*Print["subsFixedAngles = ",subsFixedAngles];
    Print["subsAnglesinFirstPosition = ",
    subsAnglesinFirstPosition];*)



    (*
    (*Find what is the angle C1C2 or link1 angle (input angle) using \
Brian's notation*)
    BrianIpAngle=0;
    (*Print["FTLA = ",FTLA];*)
    FTLA2=FlattenFTLA,1];
    Table[
    If[StringMatchQ[ToString[FTLA2[[j,1]]],"j8t1"]\[Equal]True&&
    StringMatchQ[ToString[FTLA2[[j,2]]],"j1t2"]\[Equal]True,
    BrianIpAngle=FTLA2[[j,4]]];
    ,{j,Length[FTLA2]}];
    (*Print["Brian Angle = ",BrianIpAngle];*)
    FixedOffsetforIpAngle=BrianIpAngle-th1;
    FixedOffsetforIpAngle=FixedOffsetforIpAngle/.subsFixedAngles;
    (*Print["FixedOffsetforIpAngle = ",FixedOffsetforIpAngle/
    Degree];*)
    *)

    FixedOffsetforIpAngle = AnglesDiffMineandBrian[[2]];
    (*Print["FixedOffsetforIpAngle = ",FixedOffsetforIpAngle/
    Degree];*)



    (*Since ipangle = th1+fixedangle,therefore th1 = ipangle-
    fixedangle.
    Thus the ipAngles have to be subtracted by this fixedangle to get \
the correct ipangles that is compatible with Brian's code input*)
    (*Print["ipAngles = ",ipAngles];*)
    ipAnglesOffset =
     Table[Mod[ipAngles[[j]] - FixedOffsetforIpAngle, 2 \[Pi]], {j,
       Length[ipAngles]}];
    (*Print["ipAnglesOffset = ",ipAnglesOffset/Degree];*)

    (*Compiling a starting angle list for Post Processing*)
    (*StartingSolAnglesTemp={\[Theta]2,\[Theta]3,\[Theta]4,\[Theta]5,\
\[Theta]6,\[Theta]7}/.subsAnglesinFirstPosition;
    StartingSolAngles=Table[ConvtoReal[StartingSolAnglesTemp[[i]]],{i,
    Length[StartingSolAnglesTemp]}];*)
    StartingSolAngles = BrianAnglesPosition1[[3 ;; 8]];
    (*Print["@@@@ StartingSolAngles = ",StartingSolAngles/Degree];*)
```

```
(*Print["Starting Angles = ",StartingSolAngles/Degree];*)
(*StartingSolAngles=Join[{ipAnglesOffset[[1]]},
StartingSolAngles];*)
(*Print["Starting Angles = ",StartingSolAngles/Degree];*)

(*Creating two variables to guide the eigensystem store the eigen \
values and the eigen vectors in appropriate locations in the list {1,
2,3,4,5,6,7,8} *)
opAngEigValno =
 ToExpression[
  StringCases[ToString[opAngEigVal], RegularExpression["\\d"]]];
depAnglesno =
 Table[{ToExpression[
     StringCases[ToString[depAngles[[i]]],
      RegularExpression["\\d"]]], depAnglesTmatRatios[[i]]}, {i, 5}];
(*Print["opAngEigValno = ",opAngEigValno];
Print["depAnglesno = ",depAnglesno];*)

(*Substituting the ground angle in the two matricres Mmat and Nmat*)
\

(*MReal=Mmat/.{\[Theta]8\[Rule]
Exp[\[ImaginaryI]  groundAngle],\[Theta]8c\[Rule]
Exp[-\[ImaginaryI] groundAngle]};
NReal=Nmat/.{\[Theta]8\[Rule]
Exp[\[ImaginaryI]  groundAngle],\[Theta]8c\[Rule]
Exp[-\[ImaginaryI] groundAngle]};*)
```

```
   (*Print["ipAngles  = ",ipAngles/Degree];*)




   (*Finding all possible solutions and capturing them in the \
Solution Box for every input angle.
   The first solution should include the starting angle solution that \
is the list of angles of the links in the starting configuration in \
which it is synthesized*)
   (*Print["Mmat = ",MatrixForm[Mmat]];
   Print["Nmat = ",MatrixForm[-Nmat]];*)
   Table[
    ipSubs = {\[Theta]2 -> Exp[I ipAnglesOffset[[i]]], \[Theta]2c ->
       Exp[-I ipAnglesOffset[[i]]]};
    {kvals, kvecs} = Chop[Eigensystem[{-Nmat, Mmat} /. ipSubs], 10^-3];
    (*Print["kvals = ",kvals];
    Print["kvecs = ",kvecs];*)

    SolutionBox[[i]] =
     ConditionOpAnglesEightbarNEW[kvals, kvecs, opAngEigValno,
      depAnglesno];
    (*Print["SolutionBox[[i] = ",SolutionBox[[i]]];*)
    SolutionBoxShortened[[i]] = Take[#, {3, 8}] & /@ SolutionBox[[i]];
    (*Print[""];
    Print["Input angle for this solution = ",ipAnglesOffset[[i]]/
    Degree];
    Print[" Solution no ",i," = ",MatrixForm[SolutionBoxShortened[[i]]/
    Degree]];*)
    If[SolutionBox[[i]] == {}, SolutionBox = 0;
     SolutionBoxShortened = 0;(*Print["FK failed!!"];*)
     Goto [BadLinkage](*Print["WARNING!!!!"]*)];
    (*Print[" Solution no ",i," = ",SolutionBox[[i]]];*)

    , {i, Length[ipAngles]}];



   Label[BadLinkage];


   (*Print[
   "////////////////////   Forward Kinematics End  \
///////////////////////"];*)

   {ipAnglesOffset, StartingSolAngles, SolutionBox,
    SolutionBoxShortened}];


    (*A general module for sorting the solutions to the forward \
kinematics into trajectories*)
SortingA[(*dim_,*)FKeqns_, input_, FKsolns_] :=
 Module[{x, y, yV, F, J, br, bran, Its, tol,(*log,*)i, j, k, ycur,
   ToBeAdded, Added, nmatch, NotAdded, ToBeJoined, branches,
   NotJoined, SingularMatrixFound},
  (*dim- includes linkage's dimensions as substitutions*)
  (*FKeqns- the fwd kin eqns written symbolically*)
  (*input- the input variable written as substitutions*)
  (*FKsolns- fwd kin solutions indexed by position , solution*)
  x = input;(*input values indexed by position*)
  y = FKsolns;(*output values indexed by position*)
  yV = FKsolns[[1, 1, All, 1]];(*a vector of the output symbols*)
  F = FKeqns(*/.dim*);(*the fwd kin eqns with the input and outputs \
left symbolic*)
  J = D[F, {yV}];(*Jacobian of F with the input and outputs left \
symbolic*)
  br = {{x[[1]], #}} & /@ y[[1]];(*the active branches:
  indices are branch, position, (1-input,2-output) *)
  bran = {};(*the completed branches: same indices*)
  Its = 4;(*Newton iterations*)
  tol = 10^-2;(*tolerance when comparing numbers*)
  log = {};
  For[i = 2, i <= Length[x], i++,(*i is the current position*)
   ycur = br[[All, -1, 2]];
   If[Abs[Det[J /. x[[i]] /. #]] < 10^-10,(*Print[
      "Singular Matrix found in SortingA function Problem!! = ",Det[
      J/.x[[i]]/.#]];*)
      branches = 0; Goto [SingularMatrixFound]] & /@ ycur;
   Do[ycur = (yV /. x[[i]] /. #) -
        Inverse[J /. x[[i]] /. #].(F /. x[[i]] /. #) & /@ ycur;
     ycur = Thread[yV -> #] & /@ ycur, {Its}];
   (*ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@
   ycur[[All, All,2]];*)
   (*Print["ip, ycur = ",{x[[i]],ycur}];*)
```

```
(*Print["input angle ",x[[i]]," FK solns from Dixon ",y[[i,All,
All,2]]," FK solns from Newton ",ycur[[All, All,2]]];*)
(*ToBeAdded=Position[y[[i,All,All,2]],x_/;(Norm[x-#]<tol),1,
Heads\[Rule]False]&/@ycur[[All, All,2]];*)
ToBeAdded =
 Position[Map[Exp[I*#] &, y[[i, All, All, 2]], {2}],
    x_ /; (Norm[x - #] < tol), 1, Heads -> False] & /@
  Map[Exp[I*#] &, ycur[[All, All, 2]], {2}];
(*Print[ToBeAdded];*)
(*ToBeAdded=Position[Round[Map[Exp[I*#]&,y[[i,All,All,2]],{2}],
tol],Round[#,tol]]&/@Map[Exp[I*#]&,ycur[[All, All,2]],{2}];*)
(*ToBeAdded=Position[Round[Map[Exp[I*#]^2&,y[[i,All,All,2]],1],
tol],Round[#,tol]]&/@Map[Exp[I*#]^2&,ycur[[All, All,2]],1];*)
ToBeAdded = ToBeAdded[[All, All, 1]];
Added = {};
For[j = Length[br], j >= 1, j = j - 1,(*j is the current branch*)
 nmatch = Length[ToBeAdded[[j]]];
 Which[nmatch == 1,
  AppendTo[br[[j]], {x[[i]], y[[i, ToBeAdded[[j, 1]]]]}];
  AppendTo[Added, ToBeAdded[[j, 1]]],
  nmatch == 0,
  AppendTo[bran, br[[j]]];
  br = Delete[br, j];
  AppendTo[log,
   "Path ended at " ~~ x[[i]] ~~ " where Det[J] = " ~~
    ToString[Det[J /. x[[i]] /. ycur[[j]]]] ~~ "."],
  nmatch > 1,
    Do[br = Insert[br, br[[j]], j], {nmatch - 1}];
    Do[AppendTo[
       br[[j - 1 + k]], {x[[i]], y[[i, ToBeAdded[[j, k]]]]}], {k,
       nmatch}];
    Added = Join[Added, ToBeAdded[[j]]];
    AppendTo[log,
     "Paths may be splitting at " ~~ x[[i]] ~~ " where Det[J] = " ~~
       ToString[Det[J /. x[[i]] /. ycur[[j]]]]]
    ]];
  NotAdded = Complement[Range[Length[y[[i]]]], Added];
  If[Length[NotAdded] > 0,
   AppendTo[log,
    "At " ~~ x[[i]] ~~ ", there was " ~~ ToString[Length[NotAdded]] ~~
      " new branch(es) added."]];
  Do[AppendTo[br, {{x[[i]], y[[i, k]]}}], {k, NotAdded}];
  ];
bran = Map[Flatten, bran, {2}];
br = Map[Flatten, br, {2}];
(*End of main sorting*)
ToBeJoined = Position[bran[[All, 1]], #] & /@ br[[All, -1]];
ToBeJoined = ToBeJoined[[All, All, 1]];
branches = {};
For[j = 1, j <= Length[br], j++,
 nmatch = Length[ToBeJoined[[j]]];
 Which[nmatch == 0,
  AppendTo[branches, br[[j]]],
  nmatch > 0,
    Do[AppendTo[branches, Join[Most[br[[j]]], bran[[k]]]], {k,
      ToBeJoined[[j]]}]
    ]];
  NotJoined = Complement[Range[Length[bran]], Flatten[ToBeJoined]];
  Do[AppendTo[branches, bran[[k]]], {k, NotJoined}];

  Label [SingularMatrixFound];
  branches]

RemoveDuplicateBranches[list_] :=
 Block[{i, j, intersection, temp, listjshort, SortlistSecondtoLast,
   unionijLen, list2},
  list2 = list;

  Table[
   (*Print[];
   Print["######## i = ",i];*)



   Table[
    (*Print["{list i, list j} = ",{Length[list2[[i]]],Length[list2[[
    j]]]}];*)
    unionijLen = Length[Union[list2[[i]], list2[[j]]]];
    (*Print["length of Union of i and j = ",unionijLen];*)

    If[unionijLen < Length[list2[[i]]] + Length[list2[[j]]],
     (*Print["j branch is dependent"];*)
```

```
      listjshort = list2[[j]];
      intersection = Intersection[list2[[i]], list2[[j]]];
      Table[
       temp = DeleteCases[listjshort, intersection[[j]]];
       listjshort = temp, {j, Length[intersection]}];
      (*Print["new length of list j = ",Length[listjshort]];*)
      list2[[j]] = listjshort;
      (*,
      Print["j branch is independent"]*)
      ];

     (*Print[];*)
     , {j, i + 1, Length[list2]}];

    (*Print["before sorting list2 has branches len = ",Length/@list2];*)

    SortlistSecondtoLast =
     Sort[list2[[i + 1 ;; Length[list2]]], Length[#1] > Length[#2] &];
    list2 = Join[list2[[1 ;; i]], SortlistSecondtoLast];
    (*Print["Final length of branches sorted = ",Length/@list2];*)
    , {i, Length[list2] - 1}];

   list2 = Cases[list2, Except[{}]];
   list2]

SortingTrajectories[ipAnglesOffset_, SolutionBoxShortened_,
   FiveBrianTpsConfigAngles_, FTLA_, Subs_, BrianAnglesPosition1_,
    FiveTpsLocs_, rangeOfMotionforLink1_] :=
   Block[{ftla, SubsNM, loop1x, loop1y, loop2x, loop2y, loop3x, loop3y,
      F, th1, th2, th3, th4, th5, th6, th7, th8, ipAnglesOffsetSubs,
     SolutionBoxShortenedSubs, BranchesRAW, SortedBranchesRAW, kunion,
     NextListMember, SortedBranchesEXTRACT, thetasAngTraj, ColorList,
     TpsVisualize, BranchesFull, SortedTrajectory, Problem, BranchSols,
      pNorm, PUNorm, rad, groundAngleN, rangeIpOnCylinder, tubeRange},
    (*Print[
    "///////////////////   Sorting Start   //////////////////////"];*)

    (*Print["FiveBrianTpsConfigAngles = ",FiveBrianTpsConfigAngles/
    Degree];
    Print["BrianAnglesPosition1 = ",BrianAnglesPosition1/Degree];
    *)
    ftla = FTLA;
    SubsNM = Flatten[{Subs[[1]], Subs[[2]]}];
    (*loop equations*)
    loop1x =
     Total[Table[{ftla[[1, i, 3]] Cos[ftla[[1, i, 4]]]}, {i,
         Length[ftla[[1]]]}]] /. SubsNM;
    loop1y =
     Total[Table[{ftla[[1, i, 3]] Sin[ftla[[1, i, 4]]]}, {i,
         Length[ftla[[1]]]}]] /. SubsNM;
    loop2x =
     Total[Table[{ftla[[2, i, 3]] Cos[ftla[[2, i, 4]]]}, {i,
         Length[ftla[[2]]]}]] /. SubsNM;
    loop2y =
     Total[Table[{ftla[[2, i, 3]] Sin[ftla[[2, i, 4]]]}, {i,
         Length[ftla[[2]]]}]] /. SubsNM;
    loop3x =
     Total[Table[{ftla[[3, i, 3]] Cos[ftla[[3, i, 4]]]}, {i,
         Length[ftla[[3]]]}]] /. SubsNM;
    loop3y =
     Total[Table[{ftla[[3, i, 3]] Sin[ftla[[3, i, 4]]]}, {i,
         Length[ftla[[3]]]}]] /. SubsNM;
    F = {loop1x, loop1y, loop2x, loop2y, loop3x, loop3y} /.
      th1 -> BrianAnglesPosition1[[1]];
    (*Print[MatrixForm[F]];*)
    (*ipangleswithOffset as subs*)
    ipAnglesOffsetSubs = th2 -> # & /@ ipAnglesOffset;



    (*FKsols as subs*)
    SolutionBoxShortenedSubs =
     ConstantArray[0, Length[SolutionBoxShortened]];
    Table[
     SolutionBoxShortenedSubs[[i]] =
       Thread[{th3, th4, th5, th6, th7, th8} -> #] & /@
        SolutionBoxShortened[[i]];
     , {i, Length[SolutionBoxShortened]}];
    (*Print["len of SolutionBoxShortenedSubs = ",Length[
    SolutionBoxShortenedSubs]];*)
```

```
  (*Calling Marks sorting function and Sort the branches based on \
size*)
  BranchesRAW =
   SortingA[Flatten[F], ipAnglesOffsetSubs, SolutionBoxShortenedSubs];
  If[Length[BranchesRAW] > 16,(*Print[
   "No of branches found are greater than 16"];*)
   SortedTrajectory = 0; Goto [Problem]];
  If[BranchesRAW == 0,(*Print[
   "SortingA called by SortingTrajectories function produced a \
singular matrix"];*)
   SortedTrajectory = 0; Goto [Problem]];
  SortedBranchesRAW = Sort[BranchesRAW, Length[#1] > Length[#2] &];
  (*Print["Length of ipanglesoffset = ",Length[ipAnglesOffset]];*)
  (*Print["Lengths of the branches before removing duplicates = ",
  Length/@SortedBranchesRAW];*)



  (*Remove duplicates from branches*)
  SortedBranchesRAW = RemoveDuplicateBranches[SortedBranchesRAW];
  (*Print["Lengths of the branches found after removing dupls = ",
  Length/@SortedBranchesRAW];*)



  (*Extracting angles from branches subs list SortedBranchesRAW and \
storing them in SortedBranchesEXTRACT*)
  SortedBranchesEXTRACT = ConstantArray[0, Length[SortedBranchesRAW]];
  groundAngleN = N[ BrianAnglesPosition1[[1]]];
```

```
   Table[SortedBranchesEXTRACT[[
        i]] = ({th1, th2, th3, th4, th5, th6, th7, th8} /.
          SortedBranchesRAW[[i]]) /. th1 -> groundAngleN;
     , {i, Length[SortedBranchesRAW]}];
   (*Print["len of SortedBranchesEXTRACT[[1]] = ",Length[
   SortedBranchesEXTRACT[[1]]]];*)



   (*Plotting all the branches with the plots {theta2,...,
   theta7} vs ipAngle and also plotting the tps as 5 points for all \
angle {theta2,...,theta7}*)
   rad = (3/8)*(rangeOfMotionforLink1/Degree);
   (*Print["rad = ",rad];*)
   thetasAngTraj =
    ConstantArray[{0, 0, 0, 0, 0, 0}, Length[SortedBranchesEXTRACT]];
   Table[
    Table[
      thetasAngTraj[[i, j]] =
       MapThread[{#1, #2} &, {SortedBranchesEXTRACT[[i, All, 2]],
         SortedBranchesEXTRACT[[i, All, j + 2]]}]
      , {j, 6}];
    , {i, Length[SortedBranchesEXTRACT]}];
   (*Print["thetasAngTraj = ",thetasAngTraj[[1]]];*)

   TpsVisualize =
    Table[
     MapThread[{#1, #2} &, {FiveBrianTpsConfigAngles[[2]],
        FiveBrianTpsConfigAngles[[j + 2]]}]/Degree
     , {j, 6}];
   (*Print[" TpsVisualize = ",TpsVisualize];*)
   (*TpsVisualize=
   Table[
   MapThread[{#1,#2}&,{FiveBrianTpsConfigAngles[[All,2]],
   FiveBrianTpsConfigAngles[[All,j+2]]}]/Degree
   ,{j,6}];
   Print[" TpsVisualize = ",TpsVisualize];*)
   ColorList = {Black, Blue, Green, Magenta, Gray, Brown, Purple,
     Orange, Yellow, Cyan, Pink, Lighter[Green], Lighter[Cyan],
     Lighter[Purple], Lighter[Yellow], Lighter[Gray], Lighter[Brown]};




   (*Main display of different trajectories uncomment later*)
   If[Abs[ipAnglesOffset[[1]] - Last[ipAnglesOffset]]/Degree >
     rangeOfMotionforLink1/Degree, rangeIpOnCylinder = {0, 360};
    rad = 135; tubeRange = {{0, 0, 0}, {360, 0, 0}},
    rangeIpOnCylinder = {(ipAnglesOffset[[1]]/Degree),
      Last[ipAnglesOffset]/Degree};
    tubeRange = {{(ipAnglesOffset[[1]]/Degree), 0,
       0}, {Last[ipAnglesOffset]/Degree, 0, 0}}];
```

```
(*Print[
Table[
Graphics3D[{
Inset[Style[StringJoin["\[Theta]",ToString[i+2]],20],{20,rad,rad}],
CapForm[None],Brown,Opacity[0.1],Tube[{{(ipAnglesOffset[[1]]/
Degree),0,0},{Last[ipAnglesOffset]/Degree,0,0}},rad],Opacity[
1],(*Line[{{(ipAnglesOffset[[1]]/Degree),0,0},{(Last[
ipAnglesOffset]/Degree),0,0}}],*)
Thickness[0.005],
{ColorList[[#]],Line[{MakeCylindrical[thetasAngTraj[[#,i]]/Degree,
rad]}]}&/@ Range[1,Length[SortedBranchesEXTRACT]],
PointSize[0.02],Darker[Red],Point[MakeCylindrical[TpsVisualize[[
i]],rad]]

},
AspectRatio\[Rule] Automatic,PlotRange\[Rule]{{(ipAnglesOffset[[
1]]/Degree),Last[ipAnglesOffset]/Degree},{-rad,rad},{-rad,rad}},
Axes\[Rule]True,AxesLabel\[Rule]{X,Y,Z},SphericalRegion\[Rule]
True,Boxed\[Rule]False,ImageSize\[Rule]300(*,
ViewVertical\[Rule]{0.5,-3,1.5}*)]
,{i,1,6}]];*)




(*Finding the branches that go from start to finish*)
BranchesFull = {};
```

```
    Table[
     If[Length[SortedBranchesEXTRACT[[i]]] == Length[ipAnglesOffset],
       BranchesFull = Append[BranchesFull, SortedBranchesEXTRACT[[i]]]];
     , {i, Length[SortedBranchesEXTRACT]}];
    If[Length[BranchesFull] == 0,(*Print["No full branches found"];*)
     SortedTrajectory = 0; Goto [Problem]];





    (* Now check if the five tps lie on the full branches(usually one) \
found *)
    BranchSols = {};
    Table[
     (*Print["BranchFULL ",i];*)
     pNorm =
      Table[NormofDiffBetAngVectors[FiveBrianTpsConfigAngles[[All, j]],
         BranchesFull[[i, FiveTpsLocs[[j]]]]], {j, 5}];
     PUNorm = Norm[pNorm];
     (*Print["pNorm = ",pNorm];
     Print["PUNorm = ",PUNorm];*)
     If[PUNorm < taskTolerance,
      BranchSols = Append[BranchSols, BranchesFull[[i]]](*;Print[
      "{pNorm, PUNorm} = ",{pNorm,PUNorm}]*)];
     , {i, 1, Length[BranchesFull]}];
    (*Print["No of BranchFull that have the five tps on it = ",Length[
BranchSols]];*)
If[Length[BranchSols] == 0, SortedTrajectory = 0; Goto [Problem]];
SortedTrajectory = BranchSols[[1]];

(*Print["Sorted Traj ",MatrixForm[SortedTrajectory/Degree]];*)





(*Main display of different trajectories uncomment later*)
(*Print["ipAnglesOffset = ",ipAnglesOffset/Degree];*)
(*Print[
Table[
Graphics3D[{
Inset[Style[StringJoin["\[Theta]",ToString[i+2]],20],{20,rad,rad}],
CapForm[None],Brown,Opacity[0.1],Tube[{{(ipAnglesOffset[[1]]/
Degree),0,0},{Last[ipAnglesOffset]/Degree,0,0}},rad],Opacity[
1],(*Line[{{(ipAnglesOffset[[1]]/Degree),0,0},{(Last[
ipAnglesOffset]/Degree),0,0}}],*)
Thickness[0.005],
{ColorList[[#]],Line[{MakeCylindrical[thetasAngTraj[[#,i]]/Degree,
rad]}]}&/@ Range[1,Length[SortedBranchesEXTRACT]],
PointSize[0.02],Darker[Red],Point[MakeCylindrical[TpsVisualize[[
i]],rad]]

},
  AspectRatio\[Rule] Automatic,PlotRange\[Rule]{{(ipAnglesOffset[[
  1]]/Degree),Last[ipAnglesOffset]/Degree},{-rad,rad},{-rad,rad}},
  Axes\[Rule]True,AxesLabel\[Rule]{X,Y,Z},SphericalRegion\[Rule]
  True,Boxed\[Rule]False,ImageSize\[Rule]300(*,
  ViewVertical\[Rule]{0.5,-3,1.5}*)]
  ,{i,1,6}]];*)




  Label [Problem];
  (*Print[
  "////////////////////  Sorting End  //////////////////////"];*)
  SortedTrajectory];

  DisplayTP[data_, sc_, color_, color2_, colordot_] :=
Module[{position, orig, ex, ey, frame, rem},
 rem = {{1, 0, 0}, {0, 1, 0}};
 position = Table[Disp[data[[i]]], {i, 5}];
 orig = Table[rem.position[[i]].{0, 0, 1}, {i, 5}];
 ex = Table[rem.position[[i]].{sc, 0, 1}, {i, 5}];
 ey = Table[rem.position[[i]].{0, sc, 1}, {i, 5}];
 frame =
  Table[{Thickness[0.005], color, Line[{orig[[i]], ey[[i]]}],
    color2, Line[{orig[[i]], ex[[i]]}], colordot, PointSize[0.007],
    Point[orig[[i]]]}, {i, 5}];
 {frame}]
```

```
DisplayTP2[position_, sc_, color_, color2_, colordot_] :=
 Module[{orig, ex, ey, frame, rem},
  rem = {{1, 0, 0}, {0, 1, 0}};
  orig = Table[rem.position[[i]].{0, 0, 1}, {i, 5}];
  ex = Table[rem.position[[i]].{sc, 0, 1}, {i, 5}];
  ey = Table[rem.position[[i]].{0, sc, 1}, {i, 5}];
  frame =
   Table[{Thickness[0.0007*sc], color, Line[{orig[[i]], ey[[i]]}],
     color2, Line[{orig[[i]], ex[[i]]}], colordot,
     PointSize[0.001*sc], Point[orig[[i]]]}, {i, 5}];
  {frame}]

FindScaleforTPdisplay[data_, defaultScaler_] :=
 Module[{LDist, npos, maxDist, Scaler},
  npos = 5;
  LDist =
   Table[LinkLength[{data[[1, 2]], data[[1, 3]]}, {data[[i, 2]],
      data[[i, 3]]}], {i, 2, npos}];
  maxDist = Max[LDist];
  Scaler = maxDist/5;
  If[Scaler <= defaultScaler, Scaler = defaultScaler];
  Scaler]

ClippingRange[data_, scale_] :=
 Module[{MinX, MaxX, MinY, MaxY, xdiff, ydiff, CorrectionFactor,
   maxdiff, RangeX, offset},
  MinX = Min[Table[data[[i, 2]], {i, 5}]];
  MaxX = Max[Table[data[[i, 2]], {i, 5}]];
  MinY = Min[Table[data[[i, 3]], {i, 5}]];
  MaxY = Max[Table[data[[i, 3]], {i, 5}]];

  xdiff = Abs[MaxX - MinX];
  ydiff = Abs[MaxY - MinY];
  maxdiff = Max[Abs[{xdiff, ydiff}]];

  CorrectionFactor = 0.5*Abs[xdiff - ydiff];
  If[xdiff > ydiff,
   MaxY += CorrectionFactor; MinY -= CorrectionFactor;,
   MaxX += CorrectionFactor; MinX -= CorrectionFactor;
   ];

  offset = maxdiff*(scale - 1);
  RangeX = {{MinX - offset, MaxX + offset}, {MinY - offset,
     MaxY + offset}};
  RangeX]

LinkageDisplay[data_, linkage_, linkConnections_, RangeXY_] :=
 Module[{ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, ptc7, ptc8, ptc9, ptc10,
   DisplayLinkage, link1, link2, link3, link4, FirstTernaryLink,
   SecondTernaryLink, ThirdTernaryLink, FourthTernaryLink,
   TpOriginalDisplay},

  {ptc1, ptc2, ptc3, ptc4, ptc5, ptc6, ptc7, ptc8, ptc9, ptc10} =
   linkage;
  (*Print[linkConnections];*)
  TpOriginalDisplay =
   DisplayTP[data, 4, Darker[Green], Darker[Red], Pink];
  link1 = linkConnections[[1, 1]];
  link2 = linkConnections[[1, 2]];
  FirstTernaryLink = {linkage[[link1]], linkage[[7]],
    linkage[[link1 + 1]]};
  SecondTernaryLink = {linkage[[link2]], linkage[[8]],
    linkage[[link2 + 1]]};

  link3 = linkConnections[[2, 1]];
  link4 = linkConnections[[2, 2]];
  ThirdTernaryLink = {linkage[[link3]], linkage[[9]],
    linkage[[link3 + 1]]};
  FourthTernaryLink = {linkage[[link4]], linkage[[10]],
    linkage[[link4 + 1]]};

  DisplayLinkage = Graphics[{
      Thickness[0.007],
      Lighter[Blue], Line[{ptC1, ptC2}], Darker[Blue],
      Line[{ptC2, ptC3}], PointSize[0.015], Point[{ptC2}],
      Lighter[Yellow], Line[{ptC5, ptC6}], Darker[Yellow],
      Line[{ptC4, ptC5}], Point[{ptC5}],

      Opacity[0.3],
      Darker[Gray], Polygon[FirstTernaryLink],
        Polygon[SecondTernaryLink],
        Darker[Red], Polygon[ThirdTernaryLink],
        Polygon[FourthTernaryLink],
        Opacity[1],

        Darker[Gray], Line[{linkage[[7]], linkage[[8]]}],
        PointSize[0.015], Point[{linkage[[7]], linkage[[8]]}],
        Darker[Red], Line[{linkage[[9]], linkage[[10]]}],
        PointSize[0.015], Point[{linkage[[9]], linkage[[10]]}],

        Black, Line[{ptC1, ptC6}], PointSize[0.015], Point[{ptC1, ptC6}],
        Darker[Green], Line[{ptC3, ptC4}], Point[{ptC3, ptC4}],
        TpOriginalDisplay



        }, Axes -> True, AspectRatio -> Automatic, PlotRange -> RangeXY,
       ImageSize -> 500];

     DisplayLinkage]

  FindExpressionForVertices[ModAdjMatrixSym_, AdjMatWithJointN_,
     FTLA_] :=
     Block[{klist1, klist2, kcount, j8t1, j5t8, compTempVal, locs,
       Verticesatlocs, VerticesNos, VerticesOrder, FinalVerticesSol,
       klistPrev},

      FinalVerticesSol = 0;
      klistPrev = 0;
      klist1 = klist2 = ConstantArray[0, 100];
      kcount = 1;

      (*Print["FTLA = ",MatrixForm[FTLA]];
      Print[""];*)


      (*Find expression of all the joints encountered in the FTLA,
      klist1 is a list of all the joint with repetitions
      klist2 gives expression for each joint corr to klist1 *)
      (*Removing the expression for calculating point c1(j1t2) as we \
  know it. Thus we start with the second joint for each of the three \
  loops in FTLA*)
      Table[
       Table[
        klist1[[kcount]] = FTLA[[i, j, 2]];
        klist2[[kcount]] =
         klistPrev +
          FTLA[[i, j, 3]]*{Cos[FTLA[[i, j, 4]]], Sin[FTLA[[i, j, 4]]]};
        klistPrev = klist2[[kcount]];
        (*Print["klist2 curr = ",klist2[[kcount]]];*)
        kcount++;
        , {j, 2, Length[FTLA[[i]]]}];
       klistPrev = 0;
       , {i, 3}];
```

```
klist1 = Cases[klist1, Except[0]];
klist2 = Cases[klist2, Except[0]];
(*Print[""];
Print["klist1 = ",klist1];
Print["klist2 = ",klist2];
Print["len = ",Length[klist2]]*)



(*
(*Removing the joints or points j8t1 and j5t8 which are the
ground pivots as we already know their locations *)
Table[
If[klist1[[i]]\[Equal]j8t1,klist1[[i]]=0;klist2[[i]]=0];
If[klist1[[i]]\[Equal]j5t8,klist1[[i]]=0;klist2[[i]]=0];
,{i,Length[klist1]}];

klist1 = Cases[klist1,Except[0]];
klist2= Cases[klist2,Except[0]];

*)



(*Removing repetitions form klist1 and klist2*)
Table[
 compTempVal = klist1[[i]];
  Table[
   If[klist1[[j]] == compTempVal, klist1[[j]] = 0; klist2[[j]] = 0];
```

```
      , {j, i + 1, Length[klist1]}];
     , {i, Length[klist1] - 1}];

   klist1 = Cases[klist1, Except[0]];
   klist2 = Cases[klist2, Except[0]];

   (*Print["klist1 = ",klist1];
   Print["klist2 = ",klist2];
   Print[""];*)



%  (*Finding the locations for each joint using string handling
   on its name and extracting the coordinates of the joint in the \
Adjacency matrix*)
   locs = Table[
     ToExpression[
      StringCases[ToString[klist1[[i]]],
       RegularExpression["\\d"]]], {i, 9}];
   (*Print["locs = ",locs];
   Print[""];*)

   (*Finding the name of the joints in my Adjacency matrix using the
   coordinates*)
   Verticesatlocs =
    Table[ModAdjMatrixSym[[locs[[i, 1]], locs[[i, 2]]]], {i, 9}];
   (*Print["Verticesatlocs=",Verticesatlocs];
Print[""];*)

(*Finding the nos of the vertices in my convention*)
VerticesNos = Table[
  ToExpression[
   StringJoin[
    StringCases[ToString[Verticesatlocs[[i]]],
     RegularExpression["\\d"]]]], {i, 9}];
(*Print["VerticesNos = ",VerticesNos];
Print[""];*)

(*Performing ordering on klist2 to get the expression for each joint
in ascending order eg. {c2,c3,c4,c5,c7,c8,c9,c10}*)
VerticesOrder = Ordering[VerticesNos];
(*Print["Performing ordering we get = ",VerticesOrder];*)



klist1 = klist1[[VerticesOrder]];
klist2 = klist2[[VerticesOrder]];
(*Print["klist1 ordered = ",klist1];
Print["klist2 ordered = ",klist2];
Print[""];
*)

FinalVerticesSol = klist2;
(*Print["FinalVerticesSol = ",FinalVerticesSol];*)
```

```
    FinalVerticesSol];

MakePolygon[list_] := Block[{len, drawPolygon},
   len = Length[list];
   If[len == 3,
    drawPolygon = {(*Darker[Red],Line[{list[[1]],list[[2]]}],
        Line[{list[[2]],list[[3]]}],Line[{list[[3]],list[[1]]}],*)
        Opacity[0.25],
        (*EdgeForm[Directive[Thick,Darker[Red]]],Polygon[{list[[1]],
        list[[2]],list[[3]]}],*)
        EdgeForm[Directive[{Thin, Opacity[0.3]}]],
        Polygon[{list[[1]], list[[2]], list[[3]]}],
        Polygon[{list[[1]], list[[2]], list[[3]]}],
        Opacity[1]};
    ];
   (*If[len\[Equal]4,
   drawPolygon={(*Darker[Red],Line[{list[[1]],list[[2]]}],Line[{list[[
   2]],list[[3]]}],Line[{list[[3]],list[[4]]}],Line[{list[[4]],list[[
   1]]}],*)
   Opacity[0.25],
   (*Red,EdgeForm[Directive[Thick,Darker[Red]]],Polygon[{list[[1]],
   list[[2]],list[[3]],list[[4]]}],*)
   Lighter[Green],(*EdgeForm[Directive[Thick,Darker[Red]]],*)
   Polygon[{list[[1]],list[[2]],list[[3]],list[[4]]}],
   Opacity[1]};
   ];*)
```

```
  (*Print["drawPolygon = ",drawPolygon];*)
  drawPolygon]

LinkageAnimation[Linkage_, RRConnectionsList_, dataOrig_, dataTemp_,
  FTLA_, FixedOffsetforIpAngle_, ipAnglesOffset_, SortedSolutions_,
  ModAdjMatrixSym_, AdjMatWithJointN_, Subs_, J_, FiveTpsLocs_,
  LinkageTopology_] :=
 Block[{ptc1, ptsc2, ptsc3, ptsc4, ptsc5, ptsc6, ptsc7, ptsc8, ptsc9,
   ptsc10, DisplayAnimation, link1, link2, link3, link4,
   FirstTernaryLink, SecondTernaryLink, ThirdTernaryLink,
   FourthTernaryLink, TpOriginalDisplay, len, subsLinklength,
   subsFixedAngles, subsAnglesinFirstPosition, TpCurrentDisplay,
   tempVertices, klist1, klist2, kcount, j8t1, j5t8, compTempVal,
   locs, Verticesatlocs, VerticesNos, VerticesOrder, FinalVerticesSol,
    FinalVerticesSolwithSubs, th1, th2, th3, th4, th5, th6, th7, th8,
   LinkSelectList, RRConsist, ternary1, ternary2, ternay3, ternary4,
   quarternary1, quarternary2, polygonsToBeAnimated,
   polygonsCompressed, polygonAnimations, positionOrig, positionTemp,
   FramesonC3C4, posTempinC3C4frame, xaxis, yaxis, origin, fivePtsSet,
    lengthDixonDet, maxDist, frameXYlength, rem, RangeXY,
   RRConnectionsListFlattened, locationForSecondPtonEndEff,
   EndEffFrames, pointsArrayTemp, posTempinEndEffFrame,
   secondGroundPivot, locationForSecondGndPivot,
   secondGroundPivotArray, Display4RchainAnimation, AllPoints, xMin,
   xMax, yMin, yMax, xOffset, yOffset},
  (*Print[
  "/////////////////////  Animation Start  ////////////////////////"];*)
  rem = {{1, 0, 0}, {0, 1, 0}};
  RangeXY = ClippingRange[dataOrig, FilterScaleFactor];
  len = Length[ipAnglesOffset];
  subsLinklength = Subs[[1]];
  subsFixedAngles = Subs[[2]];
  subsAnglesinFirstPosition = Subs[[3]];
  ptsc2 = ptsc3 = ptsc4 = ptsc5 =
      ptsc6 =
       ptsc7 = ptsc8 =
         ptsc9 = ptsc10 =
           LinkSelectList =
            ternary1 =
             ternary2 =
              ternay3 =
               ternary4 =
                quarternary1 = quarternary2 = ConstantArray[0, len];



  maxDist =
   Max[LinkLength[{dataOrig[[1, 2]],
        dataOrig[[1, 3]]}, {dataOrig[[#, 2]],
        dataOrig[[#, 3]]}] & /@ {2, 3, 4, 5}];
  frameXYlength = If[maxDist/5 > 5, maxDist/5, 5];
  TpOriginalDisplay =
   DisplayTP[dataOrig, frameXYlength, Darker[Green], Darker[Red],
     Gray];
  TpCurrentDisplay =
   DisplayTP[dataTemp, frameXYlength, Darker[Gray], Darker[Gray],
     Gray];

  ptc1 = Linkage[[1]];
  tempVertices = ConstantArray[0, 10];

  (*Print["dataOrig = ",dataOrig];
  Print["dataTemp = ",dataTemp];
  Print["FTLA = ",MatrixForm[FTLA]];
  Print["BrianIpAngle = ",BrianIpAngle];
  Print["FixedOffsetforIpAngle = ",FixedOffsetforIpAngle];
  Print["Linkage = ",linkageWithConnections];
  Print["ModAdjMatrixSym= ",MatrixForm[ModAdjMatrixSym]];
  Print["AdjMatWithJointN= ",MatrixForm[AdjMatWithJointN]];*)

  FinalVerticesSol =
   FindExpressionForVertices[ModAdjMatrixSym, AdjMatWithJointN, FTLA];
  (*Print["***** FinalVerticesSol in main animation function ",
  FinalVerticesSol];*)

  FinalVerticesSolwithSubs =
   FinalVerticesSol /. subsLinklength /. subsFixedAngles;
  (*Print[
  "***** FinalVerticesSol in main animation function with subs ",
  FinalVerticesSolwithSubs];
```

```
*)

Table[
 ptsc2[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[1]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc3[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[2]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc4[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[3]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc5[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[4]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc6[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[5]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc7[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[6]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc8[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[7]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc9[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[8]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);
 ptsc10[[i]] =
  ptc1 + (FinalVerticesSolwithSubs[[9]] /.
     Thread[{th1, th2, th3, th4, th5, th6, th7, th8} ->
       SortedSolutions[[i]]]);


 , {i, len}];
(*Print["ptsc2 = ",ptsc2];*)

pointsArrayTemp = {{1}, ptsc2, ptsc3, ptsc4, ptsc5, ptsc6, ptsc7,
  ptsc8, ptsc9, ptsc10};

(*fivePtsSet=Table[
{ptc1,ptsc2[[FiveTpsLocs[[i]]]],ptsc3[[FiveTpsLocs[[i]]]],ptsc4[[
FiveTpsLocs[[i]]]],ptsc5[[FiveTpsLocs[[i]]]],ptc6,ptsc7[[
FiveTpsLocs[[i]]]],ptsc8[[FiveTpsLocs[[i]]]],ptsc9[[FiveTpsLocs[[
i]]]],ptsc10[[FiveTpsLocs[[i]]]]}
,{i,5}];*)
(*Print["FivePtsSet = {c1,....,c10}",fivePtsSet];*)
```

```
lengthDixonDet = Table[
  {{LinkLength[ptc1, ptsc2[[i]]]},
   {LinkLength[ptsc2[[i]], ptsc3[[i]]]},
   {LinkLength[ptsc3[[i]], ptsc4[[i]]]},
   {LinkLength[ptc1, ptsc5[[i]]]},
   {LinkLength[ptsc5[[i]], ptsc6[[i]]]},
   {LinkLength[ptsc7[[i]], ptsc8[[i]]]},
   {LinkLength[ptsc9[[i]], ptsc10[[i]]]}
   }
  , {i, len}];
(*Print["lengthDixonDet = ",MatrixForm[lengthDixonDet]];*)


positionOrig = Table[Disp[dataOrig[[i]]], {i, 5}];
positionTemp = Table[Disp[dataTemp[[i]]], {i, 5}];

(*Here we observe that unlike the 6R loop case where we knew that \
the end effector is the link made by link c3,c4.
But in case of the 4R open chain we only know that the end effector \
has point c4 connected to it.
We dont know which other points are connected to the end effector \
link. That information can be found from the RRconnetionslist.
When ever an RR constraint is synthesized between some link and \
link 5, it had a common point with the end effector link 5. We need \
to find one such point.
We do that by flattening the RRconnecitonsList eg {1,5,2,3,5,6}.
```

```
  Here we observe that 5 is in the second location and hence the \
point associated with the RR constraint connected to the end effector \
link 5 is point c6.
  This was obtained by adding 4 to the location which in this case is \
2*)

  RRConnectionsListFlattened = Flatten[RRConnectionsList];
  locationForSecondPtonEndEff =
   4 + Flatten[Position[RRConnectionsListFlattened, 5]][[1]];
  (*Print["locationForSecondPtonEndEff = ",
  locationForSecondPtonEndEff];*)

  (*Print["ptsc8 = ",ptsc8];
  Print["pointsArrayTemp = ",pointsArrayTemp[[
  locationForSecondPtonEndEff]]];*)

  EndEffFrames =
   Chop[Table[
     Disp[Flatten[{JointAngle[{1, 0},
         pointsArrayTemp[[locationForSecondPtonEndEff, i]] -
          ptsc4[[i]]],
        pointsArrayTemp[[locationForSecondPtonEndEff, i]]}]], {i,
      len}]];
  (*Print["EndEffFrames = ",EndEffFrames[[len]]];*)
  posTempinEndEffFrame =
   Chop[Inverse[EndEffFrames[[1]]].positionTemp[[1]]];
  xaxis =
  Table[rem.EndEffFrames[[i]].posTempinEndEffFrame .{frameXYlength,
      0, 1}, {i, len}];
 yaxis =
  Table[rem.EndEffFrames[[i]].posTempinEndEffFrame .{0,
      frameXYlength, 1}, {i, len}];
 origin =
  Table[rem.EndEffFrames[[i]].posTempinEndEffFrame .{0, 0, 1}, {i,
     len}];




 (*In case of 6R loop the point c6 by default was the ground pivot \
as it was defined by the user.
 But thats not the case with the 4R chain.
 Whichever RR constrained is connected the the ground link link1 \
defines this ground pivot. For eg if the RRconnectionsList is {13,25,
 56}, the first synthesized RR constraint defines the ground pivot \
as point c5*)
 locationForSecondGndPivot =
  4 + Flatten[Position[RRConnectionsListFlattened, 1]][[1]];
 (*Print["locationForSecondGndPivot = ",locationForSecondGndPivot];*)

 secondGroundPivotArray =
  pointsArrayTemp[[locationForSecondGndPivot]];
 (*Print["secondGroundPivotArray = ",secondGroundPivotArray];*)
```

```
  (*Print[" origin = ",origin];*)
  (*


  (* CHECKS *)
  Print["#### CHECK \
#########################################################\
######################"];
  Print["Test for ptsc2"];
  Print[Table[Chop[LinkLength[ptc1,ptsc2[[i]]]-LinkLength[ptc1,ptsc2[[
  1]]]],{i,len}]];
  Print["Test for ptsc3"];
  Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc3[[i]]]-LinkLength[
  ptsc2[[1]],ptsc3[[1]]]],{i,len}]];
  Print["Test for ptsc4"];
  Print[Table[Chop[LinkLength[ptsc3[[i]],ptsc4[[i]]]-LinkLength[
  ptsc3[[1]],ptsc4[[1]]]],{i,len}]];
  Print["Test for ptsc5"];
  Print[Table[Chop[LinkLength[ptc6,ptsc5[[i]]]-LinkLength[ptc6,ptsc5[[
  1]]]],{i,len}]];
  Print["Test2 for ptsc5"];
  Print[Table[Chop[LinkLength[ptsc4[[i]],ptsc5[[i]]]-LinkLength[
  ptsc4[[1]],ptsc5[[1]]]],{i,len}]];
  Print["Test for link C7C8 Green link"];
  Print[Table[Chop[LinkLength[ptsc7[[i]],ptsc8[[i]]]-LinkLength[
  ptsc7[[1]],ptsc8[[1]]]],{i,len}]];
  Print["Test for link C9C10 Blue link"];
  Print[Table[Chop[LinkLength[ptsc9[[i]],ptsc10[[i]]]-LinkLength[
  ptsc9[[1]],ptsc10[[1]]]],{i,len}]];
  Print["Test for link C1C7 link"];
  Print[Table[Chop[LinkLength[ptc1,ptsc7[[i]]]−LinkLength[ptc1,ptsc7[[
  1]]]],{i,len}]];
  Print["Test for link C2C7 link"];
  Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc7[[i]]]−LinkLength[
  ptsc2[[1]],ptsc7[[1]]]],{i,len}]];
  Print["Test for link C1C9 link"];
  Print[Table[Chop[LinkLength[ptc1,ptsc9[[i]]]−LinkLength[ptc1,ptsc9[[
  1]]]],{i,len}]];
  Print["Test for link C2C9 link"];
  Print[Table[Chop[LinkLength[ptsc2[[i]],ptsc9[[i]]]−LinkLength[
  ptsc2[[1]],ptsc9[[1]]]],{i,len}]];
  Print["#### CHECK \
#################################################################\
########################"];

  *)



  Table[LinkSelectList[[i]] =
    {{ptc1, secondGroundPivotArray[[i]]}, {ptc1,
      ptsc2[[i]]}, {ptsc2[[i]], ptsc3[[i]]}, {ptsc3[[i]],
      ptsc4[[i]]}, {ptsc4[[i]], origin[[i]]}, {ptsc5[[i]],
      ptsc6[[i]]}, {ptsc7[[i]], ptsc8[[i]]}, {ptsc9[[i]],
      ptsc10[[i]]}}, {i, len}];
```

```
RRConsist = Flatten[RRConnectionsList];
(*Print["RRConnectionList = ",RRConsist];*)
polygonsToBeAnimated = Table[
  {Flatten[{LinkSelectList[[i, RRConsist[[1]]]], {ptsc5[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[2]]]], {ptsc6[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[3]]]], {ptsc7[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[4]]]], {ptsc8[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[5]]]], {ptsc9[[i]]}}, 1],
   Flatten[{LinkSelectList[[i, RRConsist[[6]]]], {ptsc10[[i]]}}, 1]}
  , {i, len}];
```

```
(*Calculating the Plot Range for the Animation*)
AllPoints =
 Flatten[{xaxis, origin, yaxis, {ptc1}, ptsc2, ptsc3, ptsc4, ptsc5,
   ptsc6, ptsc7, ptsc8, ptsc9, ptsc10, secondGroundPivotArray}, 1];
(*Print["AllPoints = ",AllPoints];*)
{xMin, xMax, yMin, yMax} = {Min[AllPoints[[All, 1]]],
  Max[AllPoints[[All, 1]]], Min[AllPoints[[All, 2]]],
  Max[AllPoints[[All, 2]]]};
{xOffset, yOffset} = {0.1*Abs[RangeXY[[1, 1]] - RangeXY[[1, 2]]],
  0.1*Abs[RangeXY[[2, 1]] - RangeXY[[2, 2]]]};
```

```
DisplayAnimation =
 Table[
            Graphics[{
    (*Inset[Grid[{{Style["Product of J list",Blue,12]},
    {(J[[1]]*J[[2]]*J[[3]])/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style["J list",Blue,
    12]},{J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style[
    "Sign of J list",Blue,12]},{Sign[J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]]},{Style[
    "RR connections",Blue,12]},{RRConsist},
    {Style["data Temp",Blue,12]},{dataTemp}
    },Frame\[Rule] All,ItemSize\[Rule]20,Background\[Rule]{Lighter[
    Yellow],None,Lighter[Yellow],None,Lighter[Yellow],
    None}],{RangeXY[[1,1]]+15,RangeXY[[2,2]]-18}],*)

    (*Inset[Grid[{{Style["Product of J list",Blue,12]},
    {(J[[1]]*J[[2]]*J[[3]])/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style["J list",Blue,
    12]},{J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]},{Style[
    "Sign of J list",Blue,12]},{Sign[J/.Thread[{th2,th3,th4,th5,th6,
    th7}\[Rule]SortedSolutions[[i]][[2;;7]]]]},{Style[
    "RR connections",Blue,12]},{RRConsist},
    {Style["data Temp",Blue,12]},{dataTemp}
    },Frame\[Rule] All,ItemSize\[Rule]20,Background\[Rule]{Lighter[
    Yellow],None,Lighter[Yellow],None,Lighter[Yellow],
    None}],{-30,-10}],*)



    (*jointcurve,*)
    TpCurrentDisplay, TpOriginalDisplay,

    Thickness[0.009], Gray,
    Line[{ptc1, secondGroundPivotArray[[i]]}],



    PointSize[0.012],
    Thickness[0.005],
    Lighter[Green], MakePolygon[polygonsToBeAnimated[[i, 1]]],
    Lighter[Green], MakePolygon[polygonsToBeAnimated[[i, 2]]],
    Lighter[Blue], MakePolygon[polygonsToBeAnimated[[i, 3]]],
    Lighter[Blue], MakePolygon[polygonsToBeAnimated[[i, 4]]],
    Lighter[Red], MakePolygon[polygonsToBeAnimated[[i, 5]]],
    Lighter[Red], MakePolygon[polygonsToBeAnimated[[i, 6]]],

    Darker[Green], Line[{ptsc5[[i]], ptsc6[[i]]}],
    Darker[Green], Point[{ptsc5[[i]], ptsc6[[i]]}],
    Darker[Blue], Line[{ptsc7[[i]], ptsc8[[i]]}],
    Darker[Blue], Point[{ptsc7[[i]], ptsc8[[i]]}],
    Darker[Red], Line[{ptsc9[[i]], ptsc10[[i]]}],
    Darker[Red], Point[{ptsc9[[i]], ptsc10[[i]]}],





    Thickness[0.005], Black,
    Line[{yaxis[[i]], origin[[i]], xaxis[[i]]}],


    Thickness[0.006],
    Black,
    Line[{ptc1, ptsc2[[i]], ptsc3[[i]], ptsc4[[i]], origin[[i]]}],
    PointSize[0.012], Point[{ptsc2[[i]], ptsc3[[i]], ptsc4[[i]]}],
    PointSize[0.018], Black,
    Point[{ptc1, secondGroundPivotArray[[i]]}]

    (*,Polygon[{ptsc3[[i]],ptsc4[[i]],origin[[i]]}]*)
    },

   Axes -> True, AspectRatio -> Automatic, ImageSize -> 500,
   Background -> White,
   PlotRange -> {{xMin - xOffset, xMax + xOffset}, {yMin - yOffset,
      yMax + yOffset}}(*{{-35,18},{-5,30}}*)
   ]
  , {i, len}];

(*Print["Linkage Pic = ",LinkageDisplay[dataTemp,
linkageWithConnections[[1]],linkageWithConnections[[2]],RangeXY]];*)

Print[ListAnimate[DisplayAnimation]]
(*Export["Eightbar.avi",ListAnimate[DisplayAnimation]];*)
(*Label[ProbFound];
{Prob,M}*)
(*Print[
"//////////////////// Animation End ////////////////////////"];*)

(*Display4RchainAnimation=Table[
          Graphics[{
TpCurrentDisplay,TpOriginalDisplay,
Thickness[0.005],Black,Line[{yaxis[[i]],origin[[i]],xaxis[[i]]}],
Thickness[0.006],Black,Line[{ptc1,ptsc2[[i]],ptsc3[[i]],ptsc4[[i]],
origin[[i]]}],
PointSize[0.012],Point[{ptsc2[[i]],ptsc3[[i]],ptsc4[[i]]}],
PointSize[0.018],Black,Point[{ptc1}]
},

Axes\[Rule]True,AspectRatio\[Rule]Automatic,ImageSize\[Rule] 600,
PlotRange\[Rule]RangeXY(*{{-35,18},{-5,30}}*)
]
,{i,len}];

{Display4RchainAnimation,DisplayAnimation}*)
```

```
  ]

  GenerateSubsList[ipList_, opList_] :=
 MapThread[
  ToExpression[
     StringJoin[ToString[#1], "\[Rule]", ToString[#2]]] &, {ipList,
    opList}]

SymmetryBreaker[Chains5_, offsetAngle_, linkLen_] :=
 Block[{n, angleVars, angleVars2, xval, xloopEqn, yloopEqn,
    anglesumEqn, fnArray, anglesStart, JacobianfnArray,
    anglesStartSubs, \[Theta]1start, xvals, fnArrayWithIp,
    anglesSol, \[Theta]1, solTemp, solTemp2, newChains,
    anglesArrayFinal, newChainsFinal, \[Theta]1startWithOffset},

  n = Length[Chains5[[1]]];
  (*Print["fiveSubsChainsComplete = ",MatrixForm[Chains5]];
  Print["no of joints = ",n];
  Print["user defined link length = ",linkLen];
  Print["offset angle in degrees = ",offsetAngle/Degree];*)

  angleVars =
    Table[Symbol[StringJoin["\[Theta]", ToString[i]]], {i, 1, n - 1}];
  (*Print["Angles considered angleVars = ",angleVars];*)
  angleVars2 = angleVars[[2 ;; Length[angleVars]]];
```

```
anglesStart = Chop[Table[
   Table[
     JointAngle[{1, 0}, Chains5[[i, j + 1]] - Chains5[[i, j]]]
     , {j, 2, n - 1}]
   , {i, 5}]];
(*Print["anglesStart = ",anglesStart/Degree];*)
anglesStartSubs = Table[
  GenerateSubsList[angleVars2, anglesStart[[i]]]
  (*MapThread[ToExpression[StringJoin[ToString[#1],"\[Rule]",
  ToString[#2]]]&,{angleVars2,anglesStart[[i]]}]*)
  , {i, 5}];
(*Print["anglesStartSubs = ",anglesStartSubs];*)
\[Theta]1start =
 Table[JointAngle[{1, 0}, Chains5[[i, 2]] - Chains5[[i, 1]]], {i,
   5}];
(*Print["\[Theta]1start = ",\[Theta]1start/Degree];*)
\[Theta]1startWithOffset = \[Theta]1start + offsetAngle;
(*Print["\[Theta]1startWithOffset = ",\[Theta]1startWithOffset/
Degree];*)
xvals = Chop[Chains5[[All, 4]][[All, 1]]];
(*Print["xvals = ",xvals];Print[];*)

xloopEqn = Total[linkLen*Cos[#] & /@ angleVars] - xval;
yloopEqn = Total[linkLen*Sin[#] & /@ angleVars] ;
anglesumEqn = Total[angleVars + \[Pi] - (n - 2) \[Pi]];
fnArray = {xloopEqn, yloopEqn};
(*Print["fnArray loop equations = ",MatrixForm[fnArray]];*)
```

```
fnArrayWithIp =
 Table[fnArray /. {\[Theta]1 -> (\[Theta]1start[[i]] + offsetAngle),
     xval -> xvals[[i]]}, {i, 5}];
(*Print["fnArrayWithIp loop equations = ",fnArrayWithIp];*)
JacobianfnArray =
 Table[D[fnArray[[i]], {angleVars2}], {i, Length[fnArray]}];
(*Print["Jacobian matrix = ",MatrixForm[JacobianfnArray]];
Print[];*)



(*anglesSol=Table[
anglesStart[[i]]-(Inverse[JacobianfnArray/.anglesStartSubs[[
i]]]).(fnArrayWithIp[[i]]/.anglesStartSubs[[i]])
,{i,5}];*)
anglesSol = ConstantArray[0, 5];
Table[
 solTemp = anglesStart[[i]];
 Do[
  solTemp2 =
   solTemp - (Inverse[
       JacobianfnArray /.
        GenerateSubsList[angleVars2, solTemp]]).(fnArrayWithIp[[
        i]] /. GenerateSubsList[angleVars2, solTemp]);
  solTemp = solTemp2;
  , {4}];
 anglesSol[[i]] = solTemp2;
 , {i, 5}];
```

```
(*Print["anglesSol = ",anglesSol/Degree];*)
anglesArrayFinal = Table[
  angleVars /. \[Theta]1 -> \[Theta]1startWithOffset[[i]] /.
   GenerateSubsList[angleVars2, anglesSol[[i]]]
  , {i, 5}];
(*Print["anglesArrayFinal = ",anglesArrayFinal/Degree];Print[];*)



newChains = ConstantArray[0, 5];
Table[
 solTemp = {0, 0};
 newChains[[i]] = Table[
   solTemp =
    solTemp +
     linkLen*{Cos[anglesArrayFinal[[i, j]]],
       Sin[anglesArrayFinal[[i, j]]]}
   (*Print["solTemp = ",solTemp];*)
   , {j, n - 2}]
 , {i, 5}];
newChainsFinal =
 Table[Join[{{0, 0}}, newChains[[i]], {Chains5[[i, n]]}], {i, 5}];
(*{linkLen*{Cos[\[Theta]1],Sin[\[Theta]1]},
Chains5[[i,1]]+linkLen*{Cos[anglesSol[[i,1]]],Sin[anglesSol[[i,
1]]]},Chains5[[i,1]]+linkLen*{Cos[anglesSol[[i,1]]],Sin[anglesSol[[
i,1]]]}+linkLen*{Cos[anglesSol[[i,2]]],Sin[anglesSol[[i,2]]]},
Chains5[[i,4]]}
,{i,5}];*)
```

```
  (*Print["newChains = ",MatrixForm[newChains]];
  Print["newChainsFinal = ",MatrixForm[newChainsFinal]];
  Print["Comparison withe the user defines open chains"];
  Print["fiveSubsChainsComplete = ",MatrixForm[Chains5]];*)
  newChainsFinal]

TransferChainsToOrigin[Chains5_] :=
 Block[{actualOrigin, transferedChains5, len, c1cnAngles, c1cnFrames,
   rem, transferedChains5Local},
  rem = {{1, 0, 0}, {0, 1, 0}};
  len = Length[Chains5[[1]]];
  (*Print["len = ",len];*)
  actualOrigin = Chains5[[1, 1]];
  (*Print["actualOrigin = ",actualOrigin];*)
  transferedChains5 =
   Table[(# - actualOrigin) & /@ Chains5[[i]], {i, 5}];
  (*Print["transferedChains5 = ",transferedChains5];*)
  c1cnAngles =
   Table[JointAngle[{1, 0}, transferedChains5[[i, len]]], {i, 5}];
  (*Print["c1cnAngles = ",c1cnAngles/Degree];*)
  c1cnFrames = Table[Disp[{c1cnAngles[[i]], 0, 0}], {i, 5}];
  (*Print["c1cnFrames = ",MatrixForm[c1cnFrames]];*)
  transferedChains5Local =
   Chop[Table[
     rem.Inverse[c1cnFrames[[i]]].hom[#] & /@
      transferedChains5[[i]], {i, 5}]];
  (*Print["transferedChains5Local = ",transferedChains5Local];*)
  transferedChains5Local]

TransferChainsBackOriginal[Chains5Orig_, transferedChains5Local_] :=
 Block[{actualOrigin, rem, len, c1cnAngles, c1cnFrames,
   transferedLocalChains5},
  rem = {{1, 0, 0}, {0, 1, 0}};
  len = Length[Chains5Orig[[1]]];
  (*Print["len = ",len];*)
  actualOrigin = Chains5Orig[[1, 1]];
  (*Print["actualOrigin = ",actualOrigin];*)

  c1cnAngles =
   Table[JointAngle[{1, 0}, Chains5Orig[[i, len]] - actualOrigin], {i,
      5}];
  (*Print["c1cnAngles = ",c1cnAngles/Degree];*)
  c1cnFrames =
   Table[Disp[Flatten[{c1cnAngles[[i]], actualOrigin}]], {i, 5}];
  (*Print["c1cnFrames = ",MatrixForm[c1cnFrames]];*)
  transferedLocalChains5 =
   Chop[Table[
     rem.c1cnFrames[[i]].hom[#] & /@ transferedChains5Local[[i]], {i,
      5}]];
  (*Print["transferedLocalChains5 = ",transferedLocalChains5];*)
  transferedLocalChains5]

  (*(*Sofa bed linkage*)
data ={{0 Degree,-4,6.1},{-49 Degree,-5.4,18.5},{-53 \
Degree,−14,24},{−42 Degree,−20,22.4},{0 Degree,−28.6,13.6}};
ptC1={−6,0}; ptC4={−8,6.1};
Tol\[Theta]m={0,5,10,5,0} ;
TolXm={2,2,2,2,2};
TolYm={2,2,2,2,2};
Elbow4Rchain = −1;
sideLength=15;
noOfLoops =1;*)

(*Example 2: Port Closure Linkage*)
data = {{5 Degree, 5.1, −51.5}, {35 Degree, 19.1, −36.02}, {75 Degree,
     21.0, −21.2}, {105 Degree, 16.5, −9.8}, {123 Degree, 7.8, 3.5}};
ptC1 = {−9.26, 1.35}; ptC4 = {3.1, −51.7};
Tol\[Theta]m = {5, 5, 5, 5, 5} ;
TolXm = {2, 2, 2, 2, 2};
TolYm = {2, 2, 2, 2, 2};
Elbow4Rchain = −1;
sideLength = 24;
noOfLoops = 1;  (*No of iterations the linkage algorithm will run\.10*)

(*Rectilinear Motion Linkage*)
(*data ={{0 Degree,0,0},{0 Degree,22,0},{0 Degree,50,0},{0 \
Degree,78,0},{0 Degree,100,0}};
ptC1={50,−90}; ptC4={−3.536,−3.536};
Tol\[Theta]m={0,0,0,0,0} ;
TolXm={2,4,6,4,2};
TolYm={0.0,0.0,0.0,0.0,0.0};
```

```
Elbow4Rchain = -1;
sideLength=45;
noOfLoops =1;*)
(* Output data statistics
Loop size 10:Synthesis sols=12179 Time Taken=88.09 sec Linkages \
Analyzed=11236 defect-free linkage=1102 Time taken=8406.58 Total time \
taken=8494.67 sec=141 min 34 sec
Loop size 100:Synthesis sols=110454 Time Taken=784.17 sec*)



IncrementAngle = 1 Degree;
taskTolerance = .01;

Tol\[Theta] = TolX = TolY = ConstantArray[0, 5];
Table[
  Tol\[Theta][[i]] = {-Tol\[Theta]m[[i]], Tol\[Theta]m[[i]]};
  TolX[[i]] = {-TolXm[[i]], TolXm[[i]]};
  TolY[[i]] = {-TolYm[[i]], TolYm[[i]]};
  , {i, 5}];

    (* ---- Synthesis ---- *)

{dataTemp, positionTemp} =
  GenRndTaskPosforLoops[data, Tol\[Theta], TolX, TolY, noOfLoops];
FinalSynSolsAllIterations = {};
SetSharedVariable[FinalSynSolsAllIterations];
Print["##################################################################\
########################"];
TimeForSynthesis = AbsoluteTiming[
   ParallelTable[

    Print["-------------------------------Loop no ", n,
     "-----------------------------------"];

    Print["dataTemp Randomized data ", Chop[N[dataTemp[[n]]], 10^-4]];
    fiveptsC4 =
     Table[rem.positionTemp[[n, i]].Inverse[positionTemp[[n, 1]]].hom[
        ptC4], {i, 5}];
    linkLenSelected = sideLength;
    {fiveTh1, fiveIntAngs} =
     FindChainAngles[4, linkLenSelected, ptC1, fiveptsC4];

    FinChains =
     Chop[FindAllPointsforChain[4, linkLenSelected, ptC1, fiveptsC4,
       fiveTh1, fiveIntAngs, Elbow4Rchain]];
    FinChainsTransfAndRot = TransferChainsToOrigin[FinChains];(*Print[
    "FinChainsTransfAndRot = ",FinChainsTransfAndRot];*)
    FinChainsTransfAndRotSymBreak =
     SymmetryBreaker[FinChainsTransfAndRot, 3.5 Degree, sideLength];
    FinChains2 =
     TransferChainsBackOriginal[FinChains,
      FinChainsTransfAndRotSymBreak];
    {linkAngles5, AFrames} =
     Chop[FillAFrames[FinChains2, dataTemp[[n]]]];
    (*Synthesizing all the possible linkage solutions for both \
independent and dependent constraints*)
    FinalSynSols =
     Quiet[Chop[
       GeneralSynthesis[FirstRRpairs, indepsAll, depsAll, AFrames,
        FinChains2[[1]], 0]]];
    Table[
     FinalSynSols[[i]] =
      Join[FinalSynSols[[i]], {linkAngles5, dataTemp[[n]]}], {i,
      Length[FinalSynSols]}];
    FinalSynSolsAllIterations =
     Join[FinalSynSolsAllIterations, FinalSynSols];
    (*FinalSynSols2 >>
    SynSols4Rchain;*) (*Storing the synthesis solutions in a txt file \
in the documents folder*)
    (*FinalSynSols=<<
    SynthesisSolutions4Rchain; *)(*Retrieving the synthesis solutions \
from the txt file in the documents folder*)

    Print[
     "**** No of linkages synthesized for the current iteration = ",
     Length[FinalSynSols]];
    (*Print["**** Total Linkages up till now synthesized = ",Length[
    FinalSynSols2]];*)

    (*][[1]];
    TimeTaken=Join[TimeTaken,{T}];*)
    (*Print["Time to complete the loop = ",If[T>60,T/60]];*)



    (*TopologiesUnique=Union[TrackTopology];
    Print["//// Unique topologies found are = ",{Length[
    TopologiesUnique],TopologiesUnique}];
    Print["//// No of solutions found for loop no. ",n," = ",noOfSols];
    Print["//// Time taken to complete this loop = ",T/60(*If[T>60,T/
    60,T]*)];
    Print[
    "-----------------------------------------------------------------\
-------------"];*)




    Label[RandomizeTP];
    (*{dataTemp,positionTemp}=RandomVariables5[Tol\[Theta],TolX,TolY,
    data];*)

    (*n++*)
    , {n, 1, noOfLoops}]

   ];

Print[];
Print["____Total no of Linkages synthesized for all iterations = ",
  Length[FinalSynSolsAllIterations]];
Print["____Time Taken to run the Synthesis for all iterations (min) = \
", TimeForSynthesis[[1]]/60];
Print["################################################################\
#######################"];

  (* ____ Analysis ____ *)

FinalSynSols = FinalSynSolsAllIterations;
trackTopology = ConstantArray[0, Length[FinalSynSolsAllIterations]];
FinalLinkageSolution = {};
SetSharedVariable[{FinalLinkageSolution}];
SetSharedVariable[countIterations];
countIterations = 0;
{ProgressIndicator[
  Dynamic[countIterations], {0, Length[FinalSynSols]}],
 Dynamic[countIterations]}

 Print["################################################################\
#######################"];
TimeForAnalysis = AbsoluteTiming[
   ParallelTable[

     (*Print["###################################    Linkage no : ",i,
     "  #####################################"];*)
     (*Print[FinalSynSols[[i,4]]];*)
     linkAngles5 = FinalSynSols[[i, 4]];
```

```
    dataTempLocal = FinalSynSols[[i, 5]];
    {thetas2, thetas3, thetas4, thetas5} = linkAngles5;
    (*Print["{thetas2,thetas3,thetas4,thetas5} = ",linkAngles5/
    Degree];*)
    {thetas6, thetas7, thetas8} = FinalSynSols[[i, 3]];
    (*Print["{thetas6,thetas7,thetas8} = ",{thetas6,thetas7,thetas8}/
    Degree];*)
    (*Print["Linkage coords {c1,..,c10} = ",LinkageSolutionsX[[i,1]]];
    Print["RR ConnectionList = ",LinkageSolutionsX[[i,3]]];
    (*Print["{theta1,theta2,theta3} = ",MatrixForm[Mod[{theta1,
    theta2,theta3},2\[Pi]]/Degree]];
    Print["{psi1,psi2,psi3} = ",MatrixForm[Mod[{psi1,psi2,psi3},
    2\[Pi]]/Degree]];*)*)

    (* Creating the modified adj matrix of the linkage to be sent to \
Brians algo next *)
    {AdjMatrix, ModAdjMatrix, ModAdjMatrixSym} =
     FindModifiedAdjacencyMatrix3[FinalSynSols[[i, 1]],
      FinalSynSols[[i, 2]]];
    LinkageToAnalyze = {ModAdjMatrix, 1, 1};
    (*Print[MatrixForm[ModAdjMatrix]];*)
    (*Print[ "LinkageToAnalyze = ",LinkageToAnalyze];*)

    (* Call Brian's algorithm and check if the output is correct *)
    {MmatReal, NmatReal, Tmat, opAngle, depAngles,
      depAnglesTmatRatios, FTLA, SubswithIP, Subs, AdjMatWithJointN,
      J, trackTopology[[i]]} =
      Quiet[Chop[DixonDeterminantForOneLinkage[LinkageToAnalyze],
         10^-7]];
     (*Print["Mmat = ",MatrixForm[MmatReal]];
     Print["Nmat = ",MatrixForm[NmatReal]];
     Print[];
     Print["Subs = ",MatrixForm[Subs]];*)
     (*Print["{opAngle,depAngles} = ",{opAngle,depAngles}];
     Print["Tmat = ",Tmat];
     Print["depAnglesTmatRatios = ",depAnglesTmatRatios];
     Print["AdjMatWithJointN = ",MatrixForm[AdjMatWithJointN]];*)
     (*Print["FTLA = ",MatrixForm[FTLA]];*)
     (*Print[];*)
     (*Print["Topology = ",TrackTopology[[i]]];*)

     (* Populate the ipAngles using the 5 ipangles and increment \
Angle specified by user.
     Also keep track of the five tps in the array using the array \
FiveTpsLocs *)
     {inputAngles, FiveTpsLocs, rangeOfMotionforLink1} =
      MakeDivisionsinTps2[linkAngles5[[1]], IncrementAngle];
     (*Print[
     "{inputAngles,FiveTpsLocs,rangeOfMotionforLink1} = \
",{inputAngles/Degree,FiveTpsLocs,rangeOfMotionforLink1/Degree}];*)
     (*Print["FiveTpsLocs = ",FiveTpsLocs];*)
     (*Print["inputAngles = ",inputAngles/Degree];
     Print["ipAngles length = ",Length[inputAngles]];*)
```

```
(* Five Tps check and angle translation *)
{BrianAnglesPosition1, AnglesDiffMineandBrian,
  FiveBrianTpsConfigAngles} =
 Quiet[TranslatingMyAnglestoBriansAnglesandCheck5[thetas2,
   thetas3, thetas4, thetas5, thetas6, thetas7, thetas8,
   FinalSynSols[[i, 2]], FinalSynSols[[i, 1]], FTLA, SubswithIP,
   J]];
(*Print["FiveBrianTpsConfigAngles = ",MatrixForm[
FiveBrianTpsConfigAngles/Degree]];
Print["BrianAnglesPosition1 = ",BrianAnglesPosition1/Degree];*)

(* Forward Kinematics:
Using Dixon Det to find all Sols (Configurations)*)
{ipAnglesOffset, StartingSolAngles, SolutionsBox,
  SolutionBoxShortened} =
 Quiet[Chop[
   ForwardKinematicsEightBar[inputAngles, MmatReal, NmatReal,
    Tmat, opAngle, depAngles, depAnglesTmatRatios, FTLA, Subs,
    AnglesDiffMineandBrian, BrianAnglesPosition1]]];
(*Print["ipAnglesOffset = ",ipAnglesOffset/Degree];*)
If[SolutionBoxShortened == 0, Goto [NextSolutionAsThisOneFailed]];
(*Print["SolutionBoxShortened = ",SolutionBoxShortened];*)

(*Sorting the Solutions using different methods*)
SortedSolutions =
 Quiet[Chop[
   SortingTrajectories[ipAnglesOffset, SolutionBoxShortened,
    FiveBrianTpsConfigAngles, FTLA, Subs, BrianAnglesPosition1,
    FiveTpsLocs, rangeOfMotionforLink1]]];
If[SortedSolutions == 0, Goto [NextSolutionAsThisOneFailed]];

(*Displaying information for the useful Linkages*)
(*Print["####################################      Linkage no : ",i,
"  ####################################"];
Print["FinalSynSols = ",FinalSynSols[[i]]];*)
(*solutionIndex=Join[solutionIndex,{i}];*)
(*Print[MatrixForm[MmatReal]];
Print[MatrixForm[NmatReal]];*)
(*Print["LinkageDefectCriteria = ",LinkageDefectCriteria];*)
(*Print[ListLinePlot[LinkageDefectCriteria,Filling\[Rule]Axis,
PlotRange\[Rule]All]];*)
(*LinkageAnimation[FinalSynSols[[i,2]],FinalSynSols[[i,1]],data,
dataTemp[[n]],FTLA,AnglesDiffMineandBrian[[2]],ipAnglesOffset,
SortedSolutions,ModAdjMatrixSym,AdjMatWithJointN,Subs,J,
FiveTpsLocs,TrackTopology[[i]]];*)

(*Save Linkage solution in the form {tripairs,
linkage coordinates, data, dataTemp, FTLA,
AnglesDiffMineandBrian,ipAnglesOffset,SortedSolutions,
ModAdjMatrixSym,AdjMatWithJointN,Subs,J,FiveTpsLocs,
TrackTopology[[i]].}*)
TempSol = {FinalSynSols[[i, 2]], FinalSynSols[[i, 1]], data,
  dataTempLocal, FTLA, AnglesDiffMineandBrian, ipAnglesOffset,
  SortedSolutions, ModAdjMatrixSym, AdjMatWithJointN, Subs, J,
          FiveTpsLocs, trackTopology[[i]]};
        FinalLinkageSolution = Join[FinalLinkageSolution, {TempSol}];
        (*noOfSols++;*)

        (*Speak[StringJoin[
        "Congratulations Kaustooobh, new solution found! Number of \
Solutions as of now are ",ToString[noOfSols]]];*)
        Label[NextSolutionAsThisOneFailed];
        countIterations = countIterations + 1;
        , {i, Length[FinalSynSols]}];

    ];

Print[];
Print["____Total no of defect free Linkages for all iterations = ",
  Length[FinalLinkageSolution]];
Print["____Time Taken to run the Analysis (min) = ",
  TimeForAnalysis[[1]]/60];
Print["____Total time taken to run the program (min) = ", \
(TimeForSynthesis[[1]] + TimeForAnalysis[[1]])/60];
Print["############################################################\
######################"];

FilterEightbarLinkageSolutions[Sols_, Maxlen_, Minlen_] :=
  Block[{FinalSols, check, ratiomaxmin},
   FinalSols = ConstantArray[0, Length[Sols]];
   Table[
      (*Print[];*)
      {check, ratiomaxmin} =
       LinkageLinkLenBoundCheck4R[Sols[[i, 1]], Sols[[i, 2]], Maxlen,
        Minlen];
      (*Print["check = ",check];*)
      (*Print["ration max/min = ",ratiomaxmin];*)
      If[check == 1, FinalSols[[i]] = Join[Sols[[i]], {ratiomaxmin}]];
      , {i, Length[Sols]}];
    FinalSols = Cases[FinalSols, Except[0]];
    FinalSols];

(*FilteredSolsSorted=ConstantArray[0,Length[FilteredSols]];
Table[
temp=LinkageLinkLenBoundCheck4R[FilteredSols[[i,1]],FilteredSols[[i,2]\
],Maxlen,Minlen];
{check,ratiomaxmin}=temp;
Print["check = ",check];
Print["ration max/min = ",ratiomaxmin]
If[check\[Equal]1,FilteredSolsSorted[[i]]=Join[FilteredSols[[i]],{\
ratiomaxmin}]];
,{i,Length[FilteredSols]}];
FilteredSolsSorted=Cases[FilteredSolsSorted,Except[0]];*)

If[
 Length[finsol] != 0,
 Table[
   LinkageAnimation[finsol[[linkageno, 1]], finsol[[linkageno, 2]],
    finsol[[linkageno, 3]], finsol[[linkageno, 4]],
    finsol[[linkageno, 5]], finsol[[linkageno, 6, 2]],
    finsol[[linkageno, 7]], finsol[[linkageno, 8]],
    finsol[[linkageno, 9]], finsol[[linkageno, 10]],
    finsol[[linkageno, 11]], finsol[[linkageno, 12]],
    finsol[[linkageno, 13]], finsol[[linkageno, 14]]]

  , {linkageno, 1, 1(*Length[finsol]*)}];
]
```