UNIVERSITY OF CALIFORNIA,
IRVINE

Synthesis of Slider-Crank Four- and Six-Bar Mechanisms Using Toleranced Task Specification

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mechanical and Aerospace Engineering

by

Mark Mathew Plecnik

Thesis Committee:
Professor J. Michael McCarthy, Chair
Professor David Reinkensmeyer
Professor James E. Bobrow

2013

UMI Number: 1535374

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI 1535374

Published by ProQuest LLC (2013). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.
All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

© 2013 Mark Mathew Plecnik

DEDICATION

To my supportive family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I first need to thank my advisor Prof. J. Michael McCarthy. It is his dedication to the research fields of kinematics and mechanism design that serves as my inspiration. This dedication is apparent by his 30 year list of accomplishments, the long hours he spends hammering home mathematics with his graduate students, and most recently, his avid stewardship of all Wikipedia articles related to our research. Plus, without the support provided through his funding, my graduate studies would not have happened.

I next would like to thank Prof. Bobrow and Prof. Reinkensmeyer. I am very appreciative of their interest in my research. I thank my labmates as well for all the knowledge sharing that has occurred through conversations over the past two years. A special thanks goes to Kevin Hung who contributed to the design of the screw insertion linkage presented in this thesis.

I would also like to thank the National Science Foundation for its support.

ABSTRACT OF THE THESIS

Synthesis of Slider-Crank Four- and Six-Bar
Mechanisms Using Toleranced Task Specification

By

Mark Mathew Plecnik

MASTER OF SCIENCE in Mechanical and Aerospace Engineering

University of California, Irvine, 2013

Professor J. Michael McCarthy, Chair

This thesis presents a strategy for the dimensional synthesis of mechanisms using tolerance zones specified by the designer that bound a prescribed movement. The prescribed movements discussed in this work are (1) five point synthesis for a four-bar slider-crank function generator, and (2) five position synthesis for a six-bar slider-crank motion generator. The six-bar presented is of the Watt I type with a single floating P joint. The synthesis methods presented here are approximate, however, they are based on exact position methods founded by Burmester. Approximate synthesis is accomplished by randomizing the task parameters within the tolerance zones, then formulating and solving the synthesis equations for that randomized set. The resulting mechanism solutions are evaluated for the presence of branch and circuit defects. This process is repeated for several iterations until a large amount of linkage design solutions are found, of which some percent will hopefully be defect-free. The defect-free results are termed useful linkages. This synthesis method is then applied to the kinematic design of a hydraulic shovel, a rotating weaver dwell linkage, a square pattern screw insertion linkage, and a rice seedling transplanter.

Chapter 1

Introduction

Product designers and engineers are often faced with a set of functional requirements that can be satisfied by the use of a linkage. However, it is a rare event that the ideal linkage for any design scheme is actually found. The reason being is that the complexity of the processes of finding these mechanisms can rapidly explode. For example, if a designer needs to generate a certain coupler curve from a pin-jointed linkage, he might choose from a four-bar, six-bar, or eight-bar planar design. The maximum degree of the resulting coupler curve will be 6, 18, or 54, respectively [41]. Moreover, for each design he will have 1, 2, or 16 topologies to choose from, respectively [51]. Even the simplest of these cases, the four-bar design, possesses a complexity which is disguised by a façade of four simple link lengths. However, finding the right topology is only half the battle and the focus of many texts [26]. This work focuses on dimensional synthesis.

1.1 Mechanism Design Strategies

Once a designer knows the topology of the linkage to be designed, he then needs to find the linkage dimensions such that the kinematic characteristics prescribed by the functional requirements are attained. There are several strategies available for finding these dimensions. It is often a designer's first resort to invoke graphical means. Graphical methods harken back to the foundations of kinematics [25]. As well, graphical methods can be coupled with the sketch environment of any modern day CAD program to create a powerful synthesis tool. These methods can range from an intuition-based guess and check to more procedural strategies [24].

Graphical methods become very cumbersome when more complex linkages are designed. Algebraic methods utilize mathematical formulations of a linkage's geometry. Burmester pioneered both graphical and algebraic synthesis methods for the four-bar linkage [11]. He formulated and solved quadratic equations for five position synthesis of the coupler link of a four-bar. Since then, algebraic methods for precision point synthesis have been formulated for more complex mechanisms. Examples include 7 position synthesis for a planar eight-bar [49], and 8 position synthesis for a spatial serial PRS chain [48].

The functional requirements of a design problem rarely necessitate a specific number of exact task positions. Most likely, precision will only be required for certain portions of a mechanism's trajectory, allowing for approximation over the rest of the trajectory. This increase in design freedom leads to multiple linkage candidates. However, it may very well be the tacit goal statement of any designer to find the best linkage solution possible. Therefore, it naturally follows that optimization theory would be the correct tool to accomplish this. However, the word "best" needs to first be defined through an objective function. In the past, objective functions have incorporated metrics for some combination of structural error, dimensional sensitivity, workspace size, mechanism size, manipulability, and payload capability [6] [16] [29] [33] [56]. Once an objective function has been established, the designer needs to navigate a nonlinear, nonmonotonic, design space [27]. In order to find the minima in this design space researchers have employed a wide range of optimization methods including gradient-based methods [43], least squares [4], evolutionary methods [2], and other machine learning algorithms [53].

The final strategy available to the linkage designer would be to use a pre-existing design and tweak it for the desired application. Researchers have created comprehensive design tools to serve just this purpose. Hrones and Nelson created an atlas of four-bar coupler curves [23]. Tsai created atlases of graphs of kinematic chains, planar bar linkages, and gear trains amongst others [51]. A collection of Reuleaux models can serve as an atlas as well [44].

1.2 Mechanism Design Issues

However, all of the synthesis strategies described above, including the modification of pre-existing designs, are subject to produce results that fail. Results fail for a variety of reasons. Linkage designs may possess branch defects, circuit defects, or ordering defects. Furthermore, designs may place joints in impossible locations, or contain links of impractical size, large or small. Once the kinematics are attained, the designer needs to then solve issues related to rigidity, link interference, inertial concerns, and manufacturing. However, this thesis will only deal with the kinematic issues.

The motivation behind this thesis is to overcome the likelihood of mechanism solutions that fail through the introduction of an approximate method that yields a large number of design candidates. This is accomplished through iteration of a precision position synthesis method for positions that are randomly varied within tolerance zones specified by the designer. This synthesis method begins with the premise that the majority of linkage solutions will fail due to branch defects, however, the residual population of useful linkages can then be evaluated by the designer to see if they satisfy the functional requirements. Ideally, the designer will be left with a handful of useful designs to choose from.

1.3 Literature Review

This thesis proposes a mechanism design process and applies it to the dimensional synthesis of (1) a slider-crank function generator and (2) a six-bar motion generator. Function generation for a four-bar was analytically first formulated by Freudenstein [17]. His approach was later modified by Hartenberg and Denavit [21] to apply to a slider-crank. Gupta [19] generalized Freudenstein's approach to include control of the transmission angle. Akcali and Dittrich [3] started with Freudenstein's design equations and formulated a numerical procedure that minimizes the function error. Following Akcali and Dittrich, Liu and Angeles [30] formulated the design of planar and spherical four-bar function generators as optimization problems, and extended this to spatial linkages in [31]. Alizade and Kilit [5] sought an algebraic solution to the constraint equations of a spherical four-bar linkage function evaluated on five precision values. Cervantes-Sanchez et al. [13] presented an algebraic solution to the constraint equations for a spherical four-bar function generator, and later formulated the solution for a spatial function generator in [12]. The work on function generation in this thesis uses the constraint equation of the coupler link to form the synthesis equations similar to Myszka and Murray's work with slider-crank motion generators [40].

The design of a six-bar linkage that includes an RPR chain was introduced by Bagci and Burke [7]. A strategy similar to the six-bar approach of this thesis is found in Gatti and Mundo [18] who constrain a three degree-of-freedom planar six-bar chain using cams. This thesis follows the approach of Soh and McCarthy [46], who constrain a 3R chain. Other six-bar design methods include the use of geometric constraint programming by Kinzel et al. [24] to obtain a Stephenson III linkage. Shiakolas et al. [45] employed evolutionary algorithms for the synthesis of six-bars used as dwell mechanisms with prescribed timing and transmission angle constraints. The design equations used in this thesis are based on the synthesis of RR chains introduced by Burmester theory [11], also see McCarthy and Soh [34].

The design of defect-free linkages is called solution rectification. Waldron [54] presents a graphical means of defect elimination for Burmester synthesis problems. Other early work in solution rectification was completed by Bajpai and Kramer [8]. Balli and Chand [9] provide a survey of the solution rectification literature. Bawab et al. [10] presented a procedure for eliminating branch, circuit, and order defects from six-bar linkages. Mirth and Chase showed that four different circuits can develop on a Watt six-bar [36] and that six can develop on a Stephenson six-bar [37]. Further work on solution rectification for six-bars has been completed by Watanabe and Katoh [55] and Ting et al. [50].

In order to obtain useful linkages, tolerance zones are specified around the desired motion parameters. This is an example of Mirth's quasi-positions [35], which was expanded by Holte et al. [22] to provide a synthesis methodology for mixed exactly specified and approximately specified task positions. Task positions with tolerance zones are equivalent to Holte's approximately specified positions. Mlinar and Erdman [38] consider four approximately specified positions and obtain a parameterized set of Burmester curves that they term a Burmester field. Murray and Stumph [47] solved for synthesis of four exact function values and a range for a fifth value. This range is examined to ensure no circuit defects are present and that the input link is fully rotatable.

1.4 Summary

Linkage designers have an array of synthesis tools to choose from including graphical methods, algebraic methods, optimization methods and the use of design atlases. This thesis presents an approximate mechanism synthesis process. The mechanism synthesis process requires the designer to specify tolerance zones around each of the task parameters that define a prescribed motion, let it be function or motion generation. Random tasks are generated within these tolerance zones and Burmester theory is used to solve for a finite number of linkage solutions. The resulting linkages are assessed for branch and circuit defects, and the process is repeated for several iterations. A percentage of the resulting linkage solutions will be defect-free. The designer may visually inspect these linkages to determine which is best suited for his design.

Chapter 2 gives a background on the mathematics used in linkage synthesis with a focus on function and motion generation, including examples. Chapter 3 applies the proposed mechanism synthesis process to slider-crank four-bar linkages used for function generation. The synthesis process includes a method to assess branch and circuit defects in resulting linkages. The use of an algorithm incorporating randomization of task functions within tolerance zones is applied to two in depth example designs. Chapter 4 applies the design process to Watt I slider-crank six-bars that contain a single P joint. The synthesis method constrains a serial RPR chain into a six-bar. The two loops of the resulting Watt six-bars are analyzed to ensure they are defect-free. Two example designs of RPR six-bars demonstrate the application of tolerance zones to this topology as well.

Chapter 2

Mathematical Background: Linkage Synthesis Theory

The objective of linkage synthesis is to design a machine capable of achieving a prescribed movement. Linkage synthesis methods can be categorized in a variety of manners. It is convenient to divide synthesis between type, number, and dimensional methods. Type and number synthesis methods focus on determining the mechanism topology for a prescribed movement. This includes the type of machine elements to use and the number of links and joints to include [52]. Dimensional synthesis determines the link lengths of a mechanism of a specific topology that accomplishes the prescribed movement. This thesis will focus only on dimensional synthesis.

Dimensional synthesis methods are further categorized by the requirements included in their prescribed motions. These requirements are placed on the position and orientation of one or more links of a mechanism. Prescribed motion requirements extend to all the time derivatives of position and orientation as well. Figure 2.1 illustrates the most common ways movement is specified. These are function generation, path generation, and motion generation. Synthesis for function generation involves specifying a coordinated movement between two links. Synthesis for path generation focuses only on the position of a point on the mechanism. Synthesis for motion generation includes requirements on the position and orientation of an end effector rigidly attached to a single link of the mechanism [32]. Motion generation synthesis is also called body guidance and task position synthesis. If requirements are placed on the time derivatives of a task position, these requirements are called infinitesimally separated positions [15].

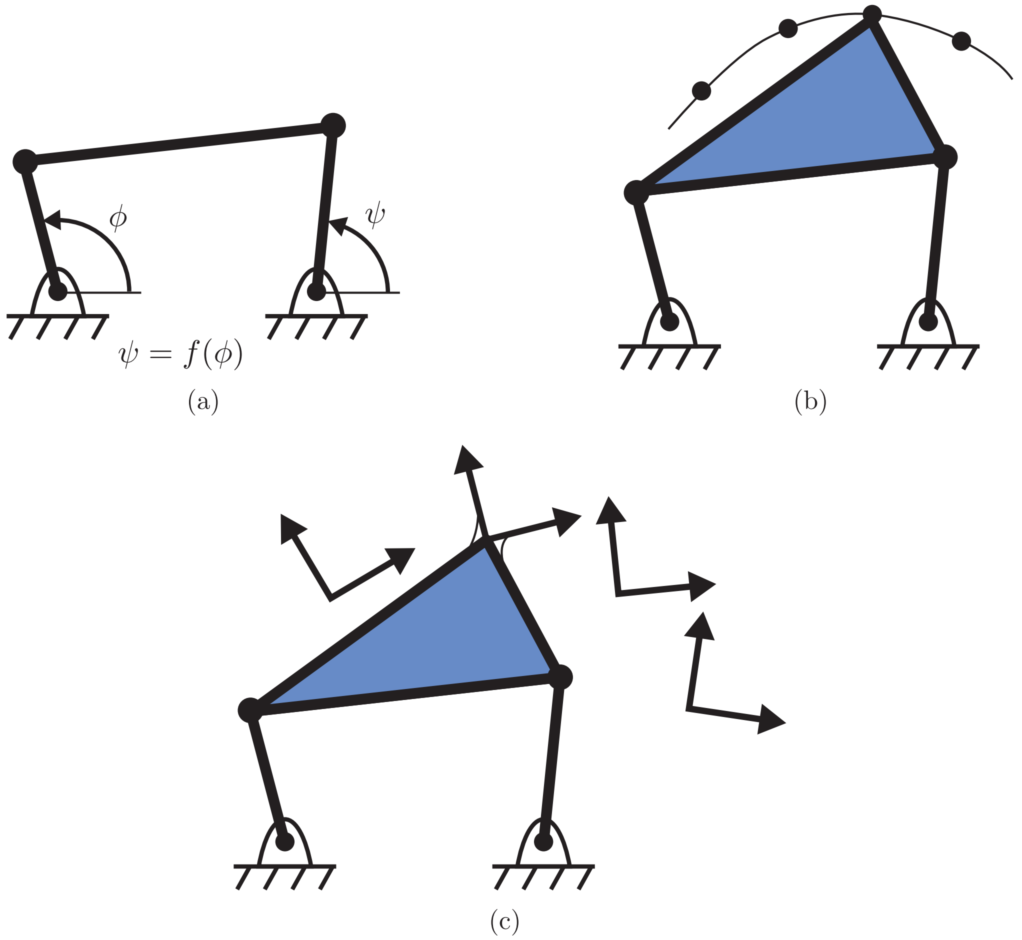


Figure 2.1: An illustration of the prescribed movements for (a) function generation, (b) path generation, and (c) motion generation.

Dimensional synthesis methods are naturally geometric in their formulation. Algebraic constraint equations for revolute and prismatic joints define circles and lines. This allows for corresponding graphical methods for many algebraic synthesis methods. For example, Figure

2.2 illustrates a graphical method for 3 task position synthesis for an RR chain where point $\mathbf{w}$ is known in the moving frame M. The objective is to find an RR chain capable of reaching task positions M_1, M_2, and M_3. In fixed frame coordinates the positions of $\mathbf{w}$ are written as $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$. Graphically, the ground pivot $\mathbf{G}$ of the RR chain is located at the intersection of the perpendicular bisectors of $\mathbf{W}_1\mathbf{W}_2$ and $\mathbf{W}_2\mathbf{W}_3$. Graphical and analytical synthesis methods for 2, 3, 4, and 5 position synthesis of an RR chain were introduced by Burmester [11], and are described by McCarthy as well [34].

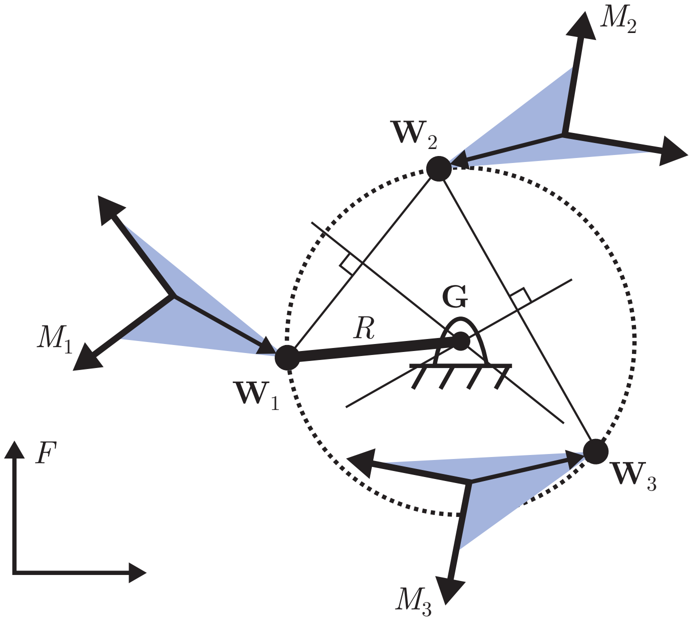


Figure 2.2: Graphical synthesis for three positions of an RR chain.

Dimensional synthesis methods can also be classified between exact and approximate methods. Synthesis methods based on the specification of a finite number of precision positions will have no error at those positions. Approximate methods range from the application of optimization theory to the use of reference books such as [23] to choose a solution design. The approximate method discussed in this thesis randomizes task positions within tolerance zones.

2.1 Function Generation

The objective of a function generator is to coordinate the motion of the input and output of a mechanism. Several different closed loop topologies can be used for function generation. As well, a function generator can have multiple inputs and outputs. For mechanisms in the plane, input and output parameters include either the rotation angle of a revolute joint or the slide of a prismatic joint. The example of a four-bar proceeds below.

For the planar 4R linkage, the angle ϕ of the input link and the angle ψ of the output link can be coordinated for a total of 5 precision points. The unknown linkage parameters to solve for are the location of the first moving pivot $\mathbf{a}_i = (a_x, a_y)$ in frame M_A, the location of the second moving pivot $\mathbf{b}_i = (b_x, b_y)$ in frame M_B, and the length of the coupler link R as shown in Figure 2.3. Note that the locations of the fixed pivots $\mathbf{O}$ and $\mathbf{C}$ are specified at (0,0) and (1,0), respectively. These parameters specify the scale and location of the linkage, attributes which are independent of function generation.

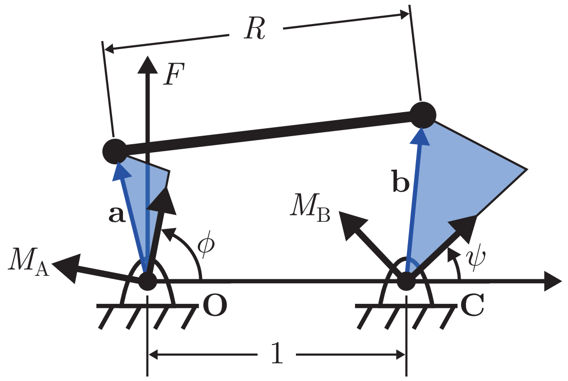


Figure 2.3: Linkage parameters for a four-bar function generator.

The desired function to generate is one that passes through the points (ϕ_i, ψ_i), for $i = 1, \ldots, 5$. The positions of the moving pivots in fixed frame coordinates are written as

$$\mathbf{A}_i = [T_i]\mathbf{a}, \tag{2.1}$$

$$\mathbf{B}_i = [S_i]\mathbf{b}, \tag{2.2}$$

where the homogeneous transformation matrices $[T_i]$ and $[S_i]$ are defined as

$$[T_i] = \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 \\ \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

$$[S_i] = \begin{bmatrix} \cos\psi_i & -\sin\psi_i & 1 \\ \sin\psi_i & \cos\psi_i & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

The geometric constraint that the points **A** and **B** must remain a fixed distance R apart is used to generate the equations,

$$(\mathbf{A}_i - \mathbf{B}_i) \cdot (\mathbf{A}_i - \mathbf{B}_i) = R^2, \qquad i = 1, \ldots, 5. \tag{2.5}$$

This system of equations in contains the 5 unknowns a_x, a_y, b_x, b_y and R in 5 quadratic equations. The solutions to these equations are finite and provide the dimensions of all the four-bar linkages that pass through the precision points (ϕ_i, ψ_i), $i = 1, \ldots, 5$.

This procedure was executed for the task function shown in Table 2.1. A single real linkage solution was found. It is shown in Figure 2.4. The red edges of the input and output links move through the desired task function points. The angles corresponding to these points are illustrated with dotted lines. The linkage's input-output function is shown in Figure 2.5.

Table 2.1: Example input-output angles for four-bar function generation.

i	ϕ_i	ψ_i
1	20°	-80°
2	30°	-50°
3	40°	-20°
4	50°	0°
5	60°	20°

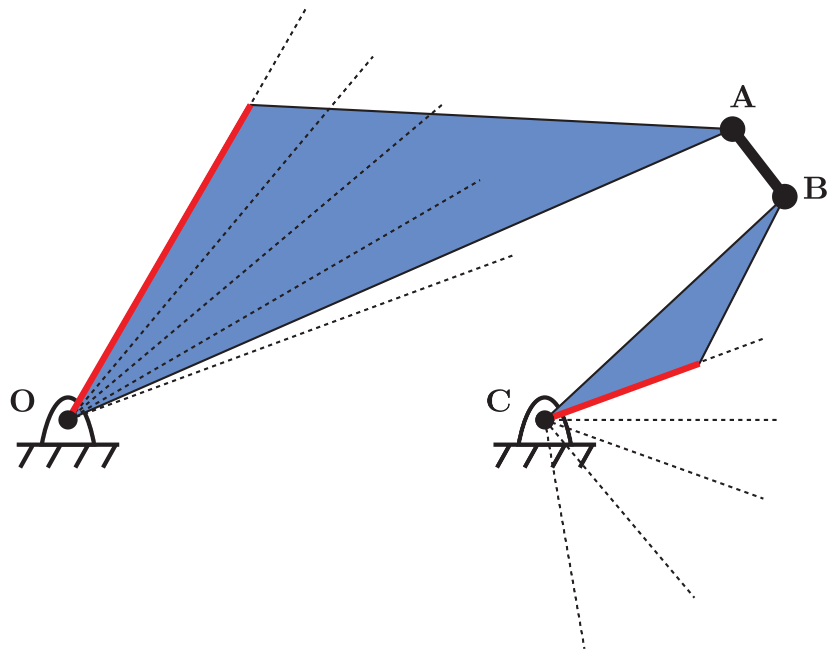


Figure 2.4: Linkage solution for the function specified in Table 2.1.

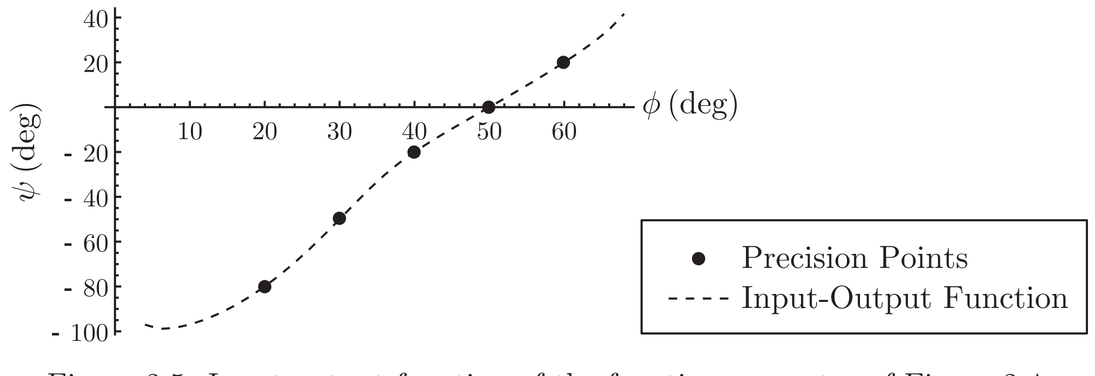


Figure 2.5: Input-output function of the function generator of Figure 2.4.

2.2 Motion Generation

Motion generation aims to guide a task frame attached to a linkage through a prescribed motion that includes both the position and orientation of that task frame. Closed form solutions for precision point synthesis exist for many mechanism topologies. Complex motion of the task frame is achieved by attaching it to a floating link, however, a linkage can only achieve a finite number of precision task positions.

A planar 4R linkage can be synthesized to achieve 5 task positions. The synthesis of a four-bar mechanism is broken down into the synthesis of RR open chains. The motion of a 2 degree of freedom RR chain cannot move a task frame in any position in 3 degree of freedom planar space. Therefore, the dimensions of an RR chain cannot be arbitrarily specified for a given set of 5 task positions.

2.2.1 Synthesis of an RR Chain

The dimensions of an RR chain are fully determined by the specification of 5 planar task positions that it must be able to pass an end effector through. Task positions are represented by homogeneous matrices $[T_i]$, $i = 1, \ldots, 5$ that transform coordinates from the task position frame to a fixed frame. The linkage parameters to be found are the location of the fixed pivot in the fixed frame, $\mathbf{G} = (u, v)$, the location of the moving pivot in the moving task frame, $\mathbf{w} = (x, y)$, and the length R of the link that connects these pivots. These parameters are shown in Figure 2.6.

These 5 parameters are related to each other by the constraint equation,

$$(\mathbf{W}_i - \mathbf{G}) \cdot (\mathbf{W}_i - \mathbf{G}) = R^2, \tag{2.6}$$

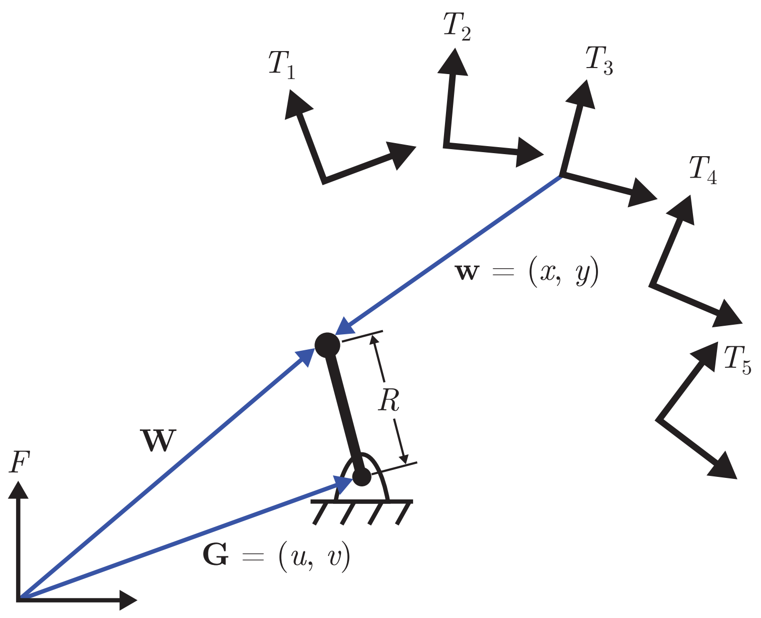


Figure 2.6: The linkage parameters u, v, x, y, and R are solved for the 5 position synthesis of the floating link of an RR crank.

where

$$\mathbf{W}_i = [T]\mathbf{w}. \tag{2.7}$$

The constraint of Equation 2.6 is that the fixed frame locations of the moving pivot $\mathbf{W}$ must lie on a circle of radius R centered at (u, v). In order to determine the linkage parameters, Equation 2.6 is written for each task position $i = 1, \ldots, 5$. This creates a determined system of 5 quadratic equations.

Both numerical and analytical solutions exist for these equations. All solutions to this system can be found using the numerical solver provided by Mathematica computational software. The analytical solution to this system can be found in [34] and is shown here for completeness.

2.2.2 Analytical Solution to Constraint Equations

The synthesis of an RR chain is the basis of all the synthesis methods presented in this thesis. The analytical procedure begins by subtracting the first equation from the following four. This cancels out the variable R and removes all the squared terms, resulting in a system of equations composed of the monomial list

$$\mathcal{M}_L = \langle u, v, 1\rangle \langle x, y, 1\rangle . \tag{2.8}$$

The system is then written as

$$a_{i1}ux + b_{i1}vx + c_{i1}x + a_{i2}uy + b_{i2}vy + c_{i2}y + a_{i3}u + b_{i3}v + c_{i3} = 0, \qquad i = 2, \dots, 5 \tag{2.9}$$

Note that each equation i is bilinear as it can be written in the form

$$\begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} a_{i1} & a_{i2} & a_{i3} \\ b_{i1} & b_{i2} & b_{i3} \\ c_{i1} & c_{i2} & c_{i3} \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = 0. \tag{2.10}$$

Multiplying Equation 2.10 gives the expression

$$\begin{bmatrix} a_{i1}u + b_{i1}v + c_{i1} & a_{i2}u + b_{i2}v + c_{i2} & a_{i3}u + b_{i3}v + c_{i3} \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = 0. \tag{2.11}$$

Equation 2.11 for $i = 2, \ldots, 5$ can be assembled into matrix form as

$$\begin{bmatrix} a_{21}u + b_{21}v + c_{21} & a_{22}u + b_{22}v + c_{22} & a_{23}u + b_{23}v + c_{23} \\ a_{31}u + b_{31}v + c_{31} & a_{32}u + b_{32}v + c_{32} & a_{33}u + b_{33}v + c_{33} \\ a_{41}u + b_{41}v + c_{41} & a_{42}u + b_{42}v + c_{42} & a_{43}u + b_{43}v + c_{43} \\ a_{51}u + b_{51}v + c_{51} & a_{52}u + b_{52}v + c_{52} & a_{53}u + b_{53}v + c_{53} \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}. \tag{2.12}$$

In order for Equation 2.12 to have a solution, the matrix on the left hand side must be of rank 2. This condition is equivalent to setting all of its 3 × 3 minors equal to zero. These minors are 4 polynomials composed of the monomials

$$\mathcal{N}_L = \langle u, v, 1 \rangle^3 . \tag{2.13}$$

By collecting coefficients in terms of the powers of v, these polynomials can be written as

$$d_{j3}v^3 + d_{j2}(u)v^2 + d_{j1}(u, u^2)v + d_{j0}(u, u^2, u^3) = 0 \qquad j = 1, \ldots, 4. \tag{2.14}$$

Assembling these equations into matrix form gives us a consistent set of homogeneous equations in terms of the unknown powers of v

$$\begin{bmatrix} d_{13} & d_{12}(u) & d_{11}(u, u^2) & d_{10}(u, u^2, u^3) \\ d_{23} & d_{22}(u) & d_{21}(u, u^2) & d_{20}(u, u^2, u^3) \\ d_{33} & d_{32}(u) & d_{31}(u, u^2) & d_{30}(u, u^2, u^3) \\ d_{43} & d_{42}(u) & d_{41}(u, u^2) & d_{40}(u, u^2, u^3) \end{bmatrix} \begin{Bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{2.15}$$

In order for a solution to exist, the determinant of the matrix in 2.15 needs to be equal to zero.

$$|D| = 0 \tag{2.16}$$

Equation 2.16 is a univariate polynomial in u. Its degree is the sum of the degrees of the columns of the matrix in (2.15), that is a degree of 6. As well, it can be shown that the 5th and 6th degree terms of this polynomial will vanish [34], resulting in a 4th degree polynomial in u. There are zero, two, or four real solutions to (2.16). Solution values of u can be substituted into (2.15) which can be solved as a linear system to find solution values of v. Values of u and v can then be substituted into (2.12) which can also be solved as a linear system to find solution values of x and y.

2.2.3 Assembling a Four-Bar Linkage

A four-bar linkage is essentially two RR chains with their floating links constrained to move as a single floating body. If both RR chains are able to reach a given set of task positions, then the resulting four-bar mechanism will also be able to reach that set of task positions, with motion now constrained to a single degree of freedom. However, whether or not that four-bar is able to smoothly move from one task position to the next is an issue that will be discussed in Section 2.4. Since there are either zero, two, or four real solutions to (2.16), and two RR chains assemble a single four-bar linkage, that means there are zero, one, or six four-bar linkages that can move through a given set of 5 task positions.

2.3 Task Specification

Section 2.2.1 discusses a procedure for finding RR chains capable of passing through a specified set of task positions. The construction of the homogeneous matrices that are used to represent these task positions is discussed in this section. A planar task position is specified by 3 coordinates with respect to a fixed frame. Those coordinates are the translation $\mathbf{d} = (d_x, d_y)$ from the origin, and the angle θ that orients the task frame with respect to the x-axis of a fixed frame.

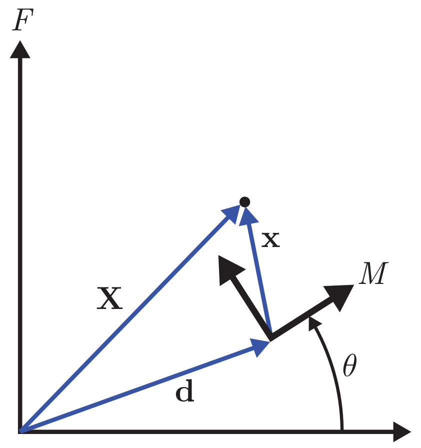


Figure 2.7: The vector **d** and angle θ define a planar displacement. The position of a point in the moving frame **x** is written in the fixed frame as **X**.

It is largely important in the fields of kinematics and robotics, amongst many others, to be able to express the coordinates of points with respect to multiple frames. For the case of a rigid planar displacement of a task position, there exists a transformation T that maps a point (x, y) in the task frame M to its position (X, Y) in the fixed frame F. The points (x, y) and (X, Y) can be represented as vectors **x** and **X**, respectively, and the transformation T can be represented by the map $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ such that

$$T: \quad \mathbf{X} = [A(\theta)]\mathbf{x} + \mathbf{d} \tag{2.17}$$

where $[A(\theta)]$ is an orthogonal matrix and **d** is the displacement vector that locates the origin of the task frame M. As well, $[A]$ in this case is limited to the orthogonal matrices of determinant $+1$ that comprise the Special Orthogonal group, $SO(2)$ [39].

The transformation T is a rigid body transformation because it preserves distance between points, thus it satisfies the property,

$$|T(\mathbf{v}) - T(\mathbf{w})|^2 = |\mathbf{v} - \mathbf{w}|^2 \qquad \forall \quad \mathbf{v}, \mathbf{w} \in \mathbb{R}^2. \tag{2.18}$$

The left and right sides of Equation 2.18 can be expanded to

$$\mathbf{v}^{\mathrm{T}}[A]^{\mathrm{T}}[A]\mathbf{v} - 2\mathbf{v}^{\mathrm{T}}[A]^{\mathrm{T}}[A]\mathbf{w} + \mathbf{w}^{\mathrm{T}}[A]^{\mathrm{T}}[A]\mathbf{w} = \mathbf{v}^{\mathrm{T}}\mathbf{v} - 2\mathbf{v}^{\mathrm{T}}\mathbf{w} + \mathbf{w}^{\mathrm{T}}\mathbf{w}. \tag{2.19}$$

Equation 2.19 is true because $[A]$ is orthogonal and $[A]^{\mathrm{T}}[A] = [I]$. Although T is a rigid body transformation, it is not a linear transformation because

$$T(\mathbf{v}+\mathbf{w}) \neq T(\mathbf{v}) + T(\mathbf{w}) \qquad \forall \quad \mathbf{v}, \mathbf{w} \in \mathbb{R}^2. \tag{2.20}$$

The inequality of (2.20) is apparent by expansion of its left and right hand sides,

$$[A]\mathbf{v} + [A]\mathbf{w} + \mathbf{d} \neq [A]\mathbf{v} + [A]\mathbf{w} + 2\mathbf{d}. \tag{2.21}$$

The only case in which (2.21) becomes an equality is for $\mathbf{d} = 0$, that is the case of pure rotation.

General planar displacements do not have a linear representations in $\mathbb{R}^2$, however, they do have a linear representations in $\mathbb{R}^3$. The rotation matrix $[A]$ and displacement vector $\mathbf{d}$ can be assembled into the matrix,

$$[T] = \begin{bmatrix} \multicolumn{2}{c}{\multirow{2}{*}{$[A(\theta)]$}} & d_x \\ & & d_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.22}$$

which is a homogeneous transformation that operates on the homogeneous coordinates $\mathbf{x} = [x, y, 1]^{\mathrm{T}}$ in frame M to return homogeneous coordinates $\mathbf{X} = [X, Y, 1]^{\mathrm{T}}$ in frame F. Note that vectors $\mathbf{x}$ and $\mathbf{X}$ have been redefined with a 1 appended to them. By operating in a higher dimensional space, linear representations of planar displacements are possible. Matrices of the form shown in Equation 2.22 belong to the Special Euclidean group, $SE(2)$ [39]. As well, a homogeneous transform operating on homogeneous coordinates will always return homogeneous coordinates.

A task position refers to the position and orientation of an end effector or tool frame. As $[T] = [T(\theta, \mathbf{d})]$, homogeneous transforms provide a compact and convenient representation of task positions.

2.4 Branch and Circuit Defects

Branch and circuit defects are common occurrences in linkage synthesis and can render a design unusable. Note that these two terms have been confused in past literature. Chase and Mirth give a thorough survey of various researchers' use of the terms branch and circuit at the time of 1993 [14]. This thesis will stay consistent with their definitions just as others have [20] [28]. A circuit consists of all the configurations a mechanism can achieve without being disassembled. A branch is all of the continuous mechanism configurations between two singular configurations on a circuit.

These ideas are illustrated in Figure 2.8 where branches and circuits of are visualized by the tracer point on the floating link of a four-bar linkage. Note that Figures 2.8(a)(b)(c) each simultaneously show the elbow up and elbow down configurations of the same linkage. The black links are the driven input for all figures. When the left link is driven (Figure 2.8(a)), the linkage can only be assembled in one of two circuits, with each circuit containing a single branch. On the other hand, when the right link is driven (Figure 2.8(b)(c)), these same circuits now contains two branches. Therefore, circuits are independent of the driven link but branches depend on the choice of driven link.

Linkage branches are partially identifiable by the forward kinematics solutions of a mechanism. In general, the forward kinematics equations for a four-bar have two solutions except at singular configurations. In the case of Figure 2.8(a), the two solution branches of the forward kinematics will coincide with the mechanism's branches. However, Figures 2.8(b)(c) demonstrate a scenario in which the forward kinematic's solution branches are divided between two circuits. Therefore, analysis of the forward kinematics equations can positively identify a branch defect, but cannot guarantee all desired mechanism configurations are contained on a single branch.

The presence of a circuit defect means a linkage needs to be disassembled and reassembled to achieve all desired mechanism configurations. The presence of a branch defect either means the linkage needs to move through a singular configuration which may cause actuation issues, or even worse, that a circuit defect exists. For the purposes of this thesis, if either a branch or circuit defect is detected, that linkage is considered unusable. The forward kinematics equations are initially used to identify branch defects, then a full simulation of the mechanisms' motion detects circuit defects.

The solution of the forward kinematics equations that give the angle ψ of the output link of the four-bar shown in Figure 2.9 in terms of the input link ϕ is as follows,

$$\psi(\phi) = \arctan\left(\frac{B}{A}\right) \pm \arccos\left(\frac{-C}{\sqrt{A^2+B^2}}\right) \tag{2.23}$$

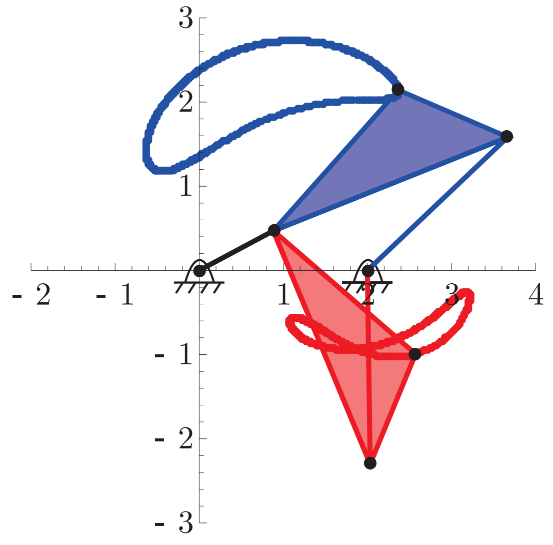
where

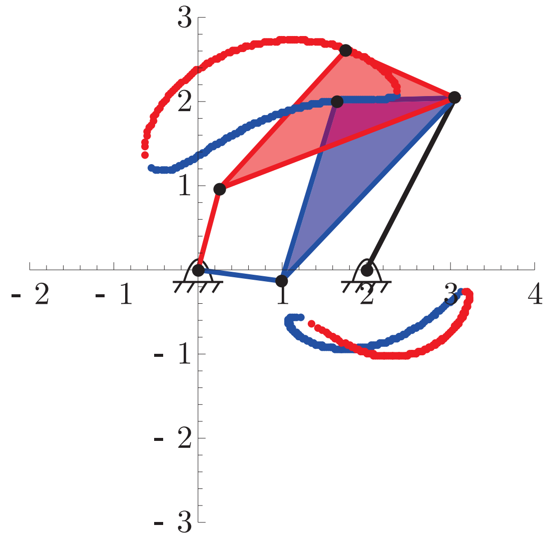
$$A = 2b(C_x - O_x) - 2ab\cos\phi$$

$$B = 2b(C_y - O_y) - 2ab\sin\phi$$

$$C = a^2 + b^2 + g^2 - h^2 - 2a\left(C_x - O_x\right)\cos\phi - 2a\left(C_y - O_y\right)\sin\phi.$$

The two solution branches are a result of the "$\pm$" in Equation 2.23. A four-bar mechanism that does not possess a branch defect will have all of its desired configurations contained entirely within the "$+$" or "$-$" branch of solutions.


(a)


(b)

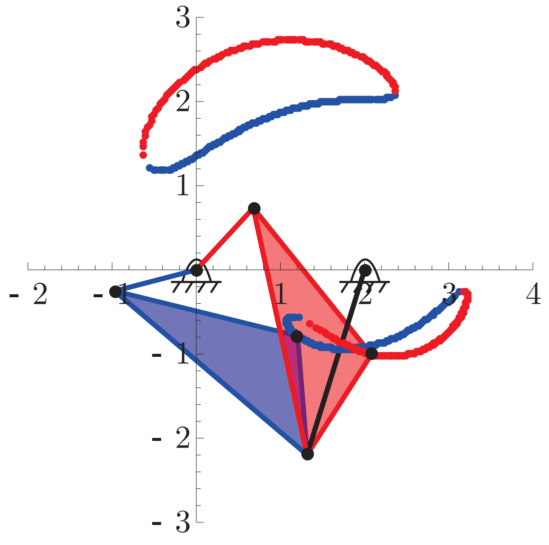

(c)

Figure 2.8: A description of various branch and circuit scenarios as the tracer point on the floating link of a four-bar. Black links are the driven input. Blue and red links represent the available configurations for a given input angle value. Illustrated in (a) is two circuits and two branches. Illustrated in (b) and (c) are two circuits and four branches.

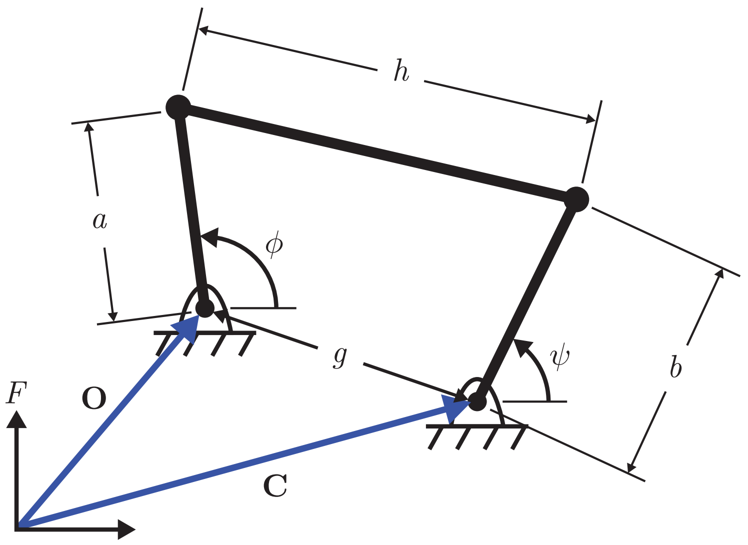


Figure 2.9: Linkage dimensions used to construct the input-output equations of a four-bar.

The results of the RR synthesis procedure described in Section 2.2.1 will often lead to defective four-bar linkages. For example, the RR synthesis procedure was executed for the task positions shown in Table 2.2. This yielded real solutions for the dimensions of four distinct RR cranks shown in Table 2.3, with parameters u, v, x, and y defined in Section 2.2.1.

Table 2.2: Task positions for a four-bar motion generator example.

i	θ_i	x_i	y_i
1	-104°	6.3	1.2
2	-65°	9.8	3
3	-50°	7.3	3.7
4	-31°	10.4	4.6
5	-5°	8.7	5.4

Table 2.3: Solutions for a four-bar motion generator example.

Solution No.	u	v	x	y
1	4.037	3.835	-3.569	-3.290
2	5.238	60.440	-1.108	-2.580
3	5.886	6.124	-2.833	-1.380
4	7.666	4.893	-2.249	0.491

A four-bar linkage is synthesized by choosing two of the four RR chain solutions from Table 2.3 and fixing their floating links relative to each other. Therefore, there are 6 possible four-bar mechanisms. However, the resulting mechanisms may or may not move smoothly between task positions due to the presence of branch or circuit defects. The forward kinematics equations need to be analyzed to ensure a condition when defects do not exist. Moreover, since the forward kinematics equations will be formulated with respect to a specific actuated joint, the analysis will change based on the choice of actuation joint. Table 2.4 presents the branching analysis results for the six mechanisms synthesized from Table 2.3. Each mechanism was analyzed twice, with respect to actuating the first ground joint and then the second.

Table 2.4: Branching analyses results for four-bar linkage solutions.

Linkage ID	Composed of Solution RR Chains		Actuated Chain	Desired Assembly Configurations Contained Per Branch
A	1	2	1	+ branch: none − branch: 1, 2, 3, 4, 5
			2	+ branch: 1, 3, 5 − branch: 2, 4
B	1	3	1	+ branch: none − branch: 1, 2, 3, 4, 5
			3	+ branch: 4, 5 − branch: 1, 2, 3
C	1	4	1	+ branch: 2, 3, 4, 5 − branch: 1
			4	+ branch: 4, 5 − branch: 1, 2, 3
D	2	3	2	+ branch: 3, 5 − branch: 1, 2, 4
			3	+ branch: 1, 2 − branch: 3, 4, 5
E	2	4	2	+ branch: 3, 5 − branch: 1, 2, 4
			4	+ branch: 1, 2, 3, 4 − branch: 5
F	3	4	3	+ branch: 2, 3, 4, 5 − branch: 1
			4	+ branch: 1, 2, 3, 4, 5 − branch: none

From Table 2.4, it is determined that the only useful four-bars linkages are mechanisms A, B, and F. Furthermore, note that these mechanisms are only useful when actuated from the proper joint. Mechanism A needs to be actuated from the ground joint of RR chain 1. Mechanism B also requires actuation from RR chain 1. Mechanism F requires actuation from RR chain 4. Of the useful four-bar linkages, mechanism A has a single link of length 10 times greater than the next largest link. Although this linkage is useful according to our established criteria, linkages composed of very long link lengths tend to have limited practical applications. Mechanisms B and F are shown in Figures 2.10 and 2.11, respectively. Both of these mechanisms move smoothly through all task positions, albeit, in a random order. The complete analysis of a four-bar should include a full simulation of the mechanism's motion to ensure a circuit defect does not exist.

2.5 Tolerance Zones

In Section 2.4, an RR synthesis procedure was executed for the task positions given in Table 2.2. The results were analyzed and two four-bar linkages were found to move smoothly through all task positions. However, it is often the case that no useful mechanisms are found for a given set of task positions. As well, if useful mechanisms are found they may fail with respect to an application-specific requirement. For example, although the two four-bar linkages above smoothly move through all task positions, they do not move through task positions in the sequential order 1, 2, 3, 4, 5. Ordering can be an example of an application-specific requirement.

In order to expand design options, this thesis utilizes an approximate synthesis method based on tolerance zones. A planar task position is defined by the specification of three precise numbers, θ, d_x, and d_y. However, it is often the case that exact position synthesis is not needed. Therefore, tolerance zones can be specified around the acceptable values of these three numbers. For five position synthesis, there are 15 parameters that can be

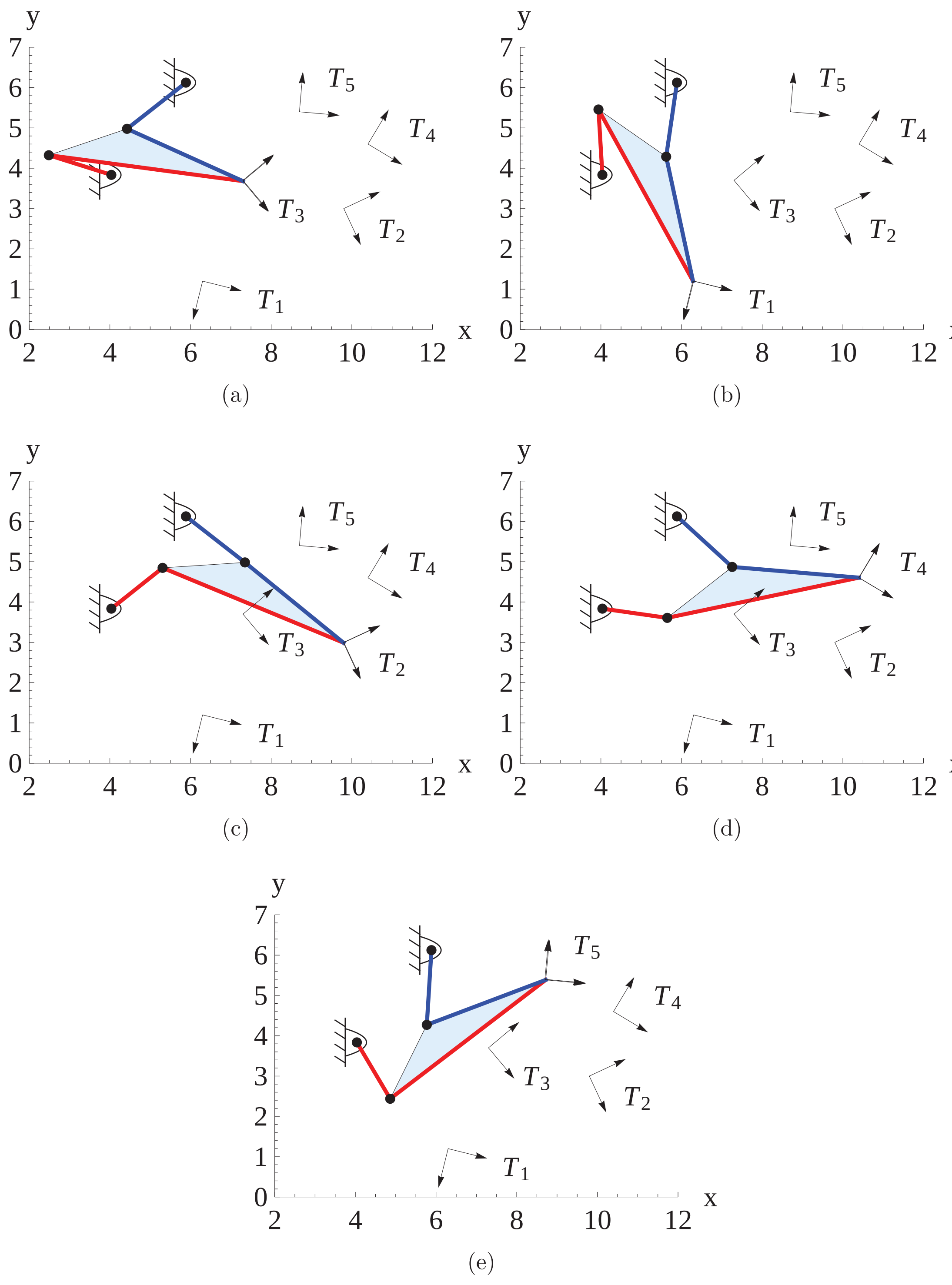


Figure 2.10: Linkage B visits all task positions in the order (a) 3, (b) 1, (c) 2, (d) 4, and (e) 5. RR chain 1 appears in red and is actuated. RR chain 3 appears in blue.

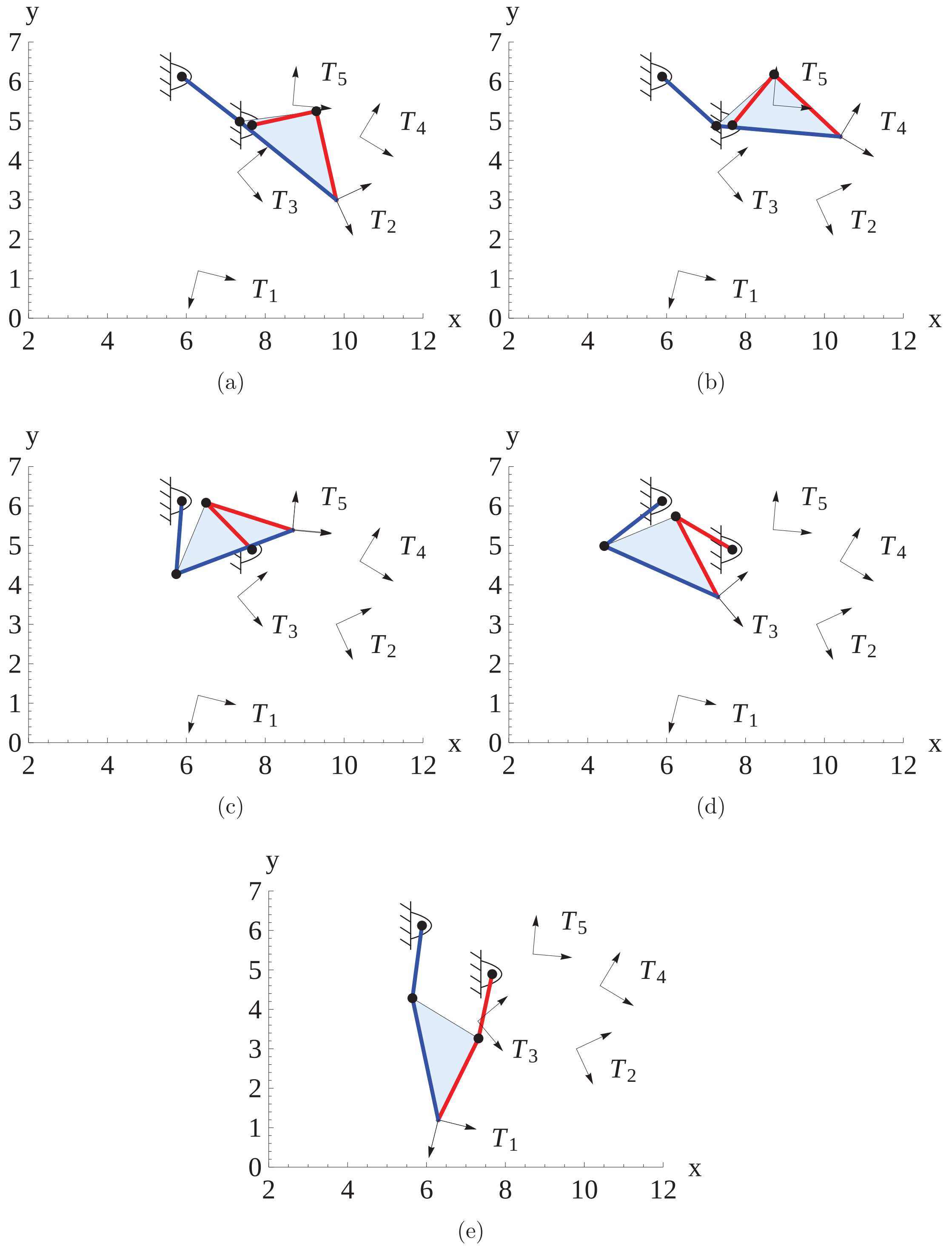


Figure 2.11: Linkage F visits all task positions in the order (a) 2, (b) 4, (c) 5, (d) 3, and (e) 1. RR chain 4 appears in red and is actuated. RR chain 3 appears in blue.

toleranced. Task positions can be randomly searched within these tolerance zones and the synthesis procedure can be executed for each new randomly generated set of task positions. Analysis of the resulting four-bar solutions can be used to generate sets of useful solutions that approximately reach the original desired task positions. The basic algorithm for this synthesis strategy is shown in Figure 2.12.

2.6 Summary

In order to find useful linkages, a given synthesis method needs to be coupled with an effective analysis method. The purpose of a dimensional synthesis method is to solve the link lengths of a mechanism of a given topology in order to fulfill some specified motion requirement. The purpose of an analysis method is to ensure the results of a synthesis method smoothly achieve that motion requirement. Circumstances in which the synthesis results do not meet the motion requirement are due to the presence of linkage defects. The analysis routines presented in this paper take into account branching and circuit defects. Another linkage defect that can occur is an ordering defect.

In the case that a synthesis method yields no useful linkages, an approximate synthesis method is proposed. This approximate method includes randomizing the motion requirement within specified tolerance zones, and then rerunning the synthesis routine for each new randomized motion requirement in order to generate a multitude of approximate linkage solutions. These solutions can be analyzed in order to find useful linkages. This strategy is illustrated in Figure 2.12.

The following chapters apply this strategy to the design of slider-crank four-bar mechanisms for function generation (Chapter 3), and the design of slider-crank Watt I six-bar mechanisms for motion generation (Chapter 4). Example applications are presented for each.

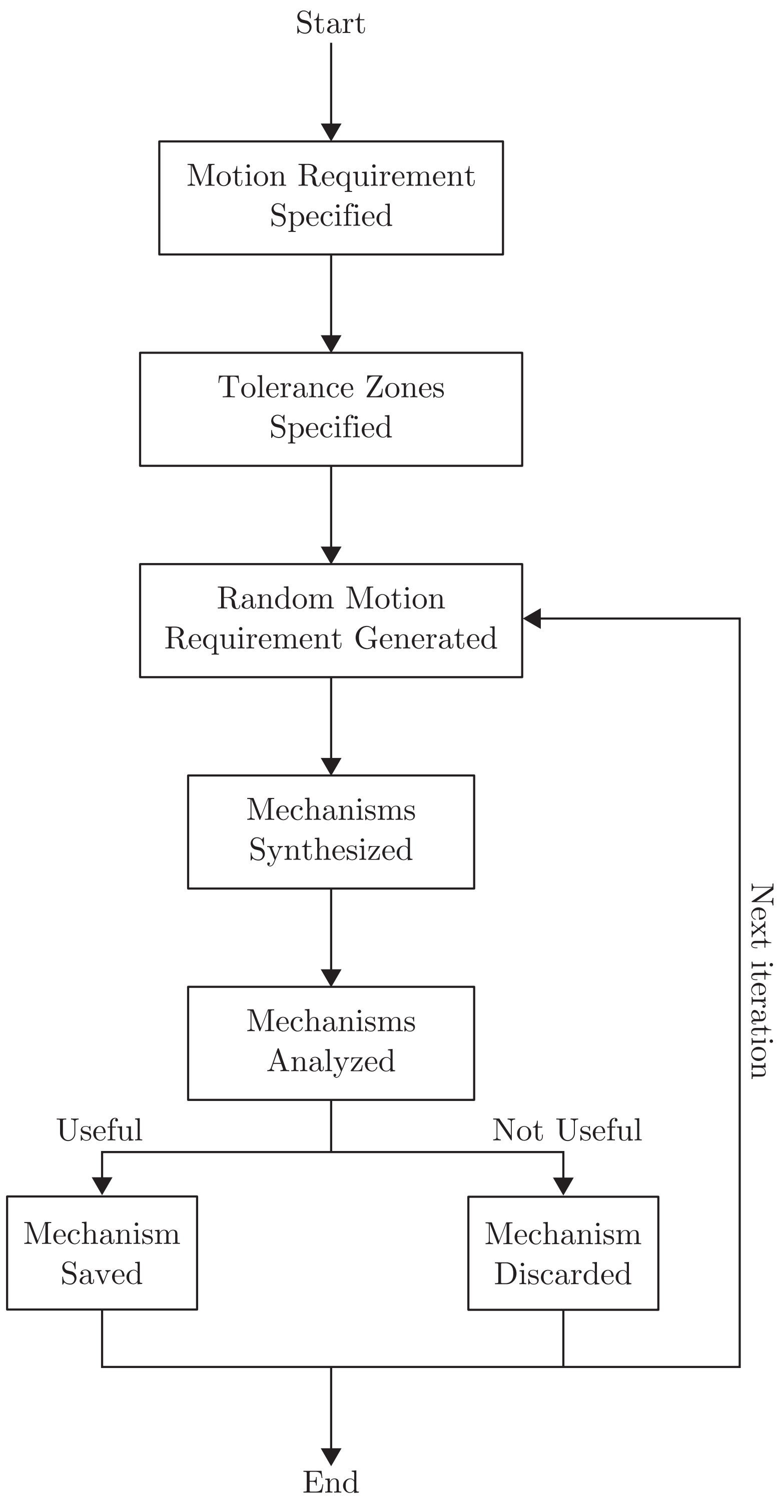


Figure 2.12: A mechanism synthesis process based on random variation of the motion requirement within tolerance zones.

Chapter 3

Function Generation for a Slider-Crank Four-Bar

This chapter presents the synthesis procedure for the coupler link of a planar slider-crank four-bar linkage in order to coordinate input by a linear actuator with the rotation of an output crank. The synthesis equations are formulated in a manner similar to the synthesis equations of a five position RR coupler link. The presence of branching and circuit defects is addressed by introducing tolerances for the input and output values of the specified task function. The proposed synthesis procedure is then executed for multiple examples. In the first example, a survey of solutions for tolerance zones of increasing size is conducted. In this example it is found that a tolerance zone of 5% of the desired full range results in a number of useful task functions and useful slider-crank function generators. To demonstrate the use of these results, two other example designs are presented: (1) a shovel actuator for a front end loader, and (2) a dwell mechanism for a rotating weaver linkage.

3.1 Introduction

The objective of the synthesis of a PRRR slider-crank function generator is to coordinate the input slide s with an output crank angle ψ. Hartenberg and Denavit [21] formulated this problem by evaluating the linkage loop equations at four precision values for the desired function, and solved the resulting non-linear equations for the dimensions of the linkage. This synthesis procedure specifies a set of five functional values $\sigma = (s_i, \psi_i)$, $i = 1, \ldots, 5$, then uses the constraint of the fixed length of the coupler link to formulate equations for the linkage dimensions.

The synthesis method will always produce at least one real solution. All solutions produced are subject to branch defects. In the case that no useful linkage solutions exist for a given σ, new task functions based on slight deviations of σ can be generated. Useful solutions may or may not result from these slight deviations. In order to discover these useful slider-crank linkages a tolerance zone is introduced around the specified precision values of the function and the values are randomly adjusted in a loop that generates a large number of slider-crank linkages. Assessment of each of the resulting linkages identifies those that operate successfully without branch defects.

3.2 Synthesis Procedure

A general slider-crank linkage is shown in Figure 3.1. The input is the slider that moves along a line with slope m and y-intercept h. In the following it can be assumed that this line is the x-axis of the fixed frame F, Figure 3.2. The goal is to find a slider-crank that achieves a set of five input-output precision values, denoted $\sigma = (s_i, \psi_i)$, $i = 1, \ldots, 5$, where s is the input slide and ψ is the output crank angle. This is called the task function, $\tau(s) = \psi$, which consists of five discrete pairs of values.

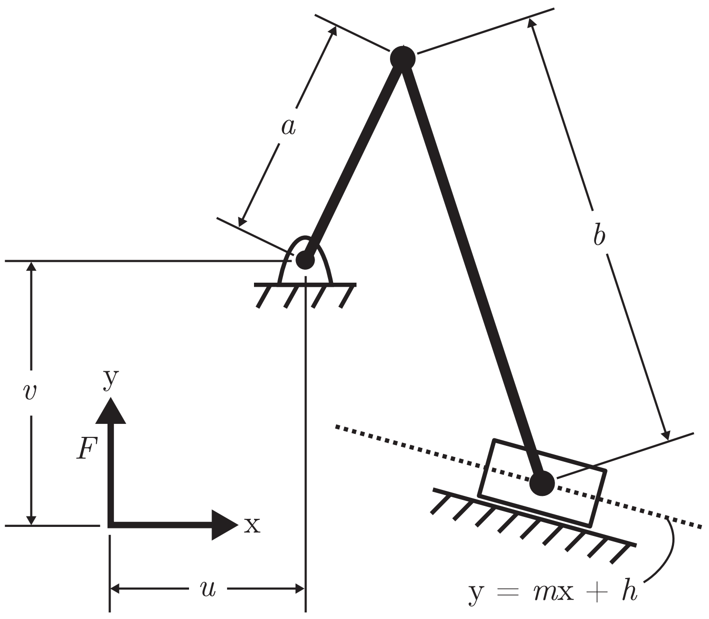


Figure 3.1: A slider-crank linkage in planar space.

The synthesis of a planar slider-crank function generator can be formulated in a manner similar to the synthesis of an RR chain that achieves five task positions. The synthesis equations are formulated using the constraint that the distance between the fixed frame positions of the revolute joints on the coupler link must be a constant value R. A moving frame M is located at an unknown point $\mathbf{G} = (u, v)$ in the fixed frame F. This will be the fixed pivot of the output crank. Let ψ be the angle of M relative to F. Notice that the five values ψ_i, $i = 1, \ldots, 5$ are the known specified precision values. The vector $\mathbf{W}_1 = (x+u, y+v)$ defines the position of the moving pivot of the output crank in F for the first specified precision value. See Figure 3.2. Vector $\mathbf{S}_i$ defines the coordinates of the pivot attached to the slider.

The requirement that the distance between the five positions of the sliding pivot $\mathbf{S}_i$ and the moving pivot $\mathbf{W}_i$ of the output crank remain constant yields the five equations,

$$(\mathbf{W}_i - \mathbf{S}_i) \cdot (\mathbf{W}_i - \mathbf{S}_i) = R^2, \quad i = 1, \ldots, 5. \tag{3.1}$$

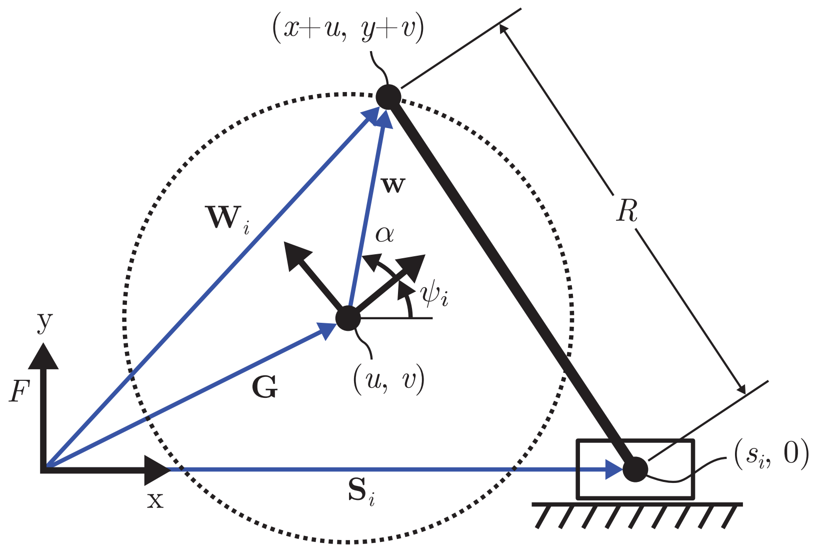


Figure 3.2: The linkage parameters u, v, x, y, and R are solved for the five point synthesis of a function generator that coordinates s and ψ.

In order to define the coordinate $\mathbf{W}_i$, $i = 2, \ldots, 5$ in terms of $\mathbf{W}_1$, note that the moving frame has five positions M_i defined by the transformations,

$$[T_i(\psi_i, \mathbf{G})] = \begin{bmatrix} \cos\psi_i & -\sin\psi_i & u \\ \sin\psi_i & \cos\psi_i & v \\ 0 & 0 & 1 \end{bmatrix}, \quad i = 1, \ldots, 5. \tag{3.2}$$

Recall that $\mathbf{G} = (u, v)$ is the unknown fixed pivot of the output crank.

Using Equation 3.2, the vectors $\mathbf{W}_i$, $i = 2, \ldots, 5$ are defined as

$$\mathbf{W}_i = [T_i][T_1]^{-1}\mathbf{W}_1 = [D_{1i}]\mathbf{W}_1. \tag{3.3}$$

The transformation $[D_{1i}]$ is the relative transformation from M_1 to M_i measured in F. It is useful to note that $\mathbf{W}_i$ are linear in the unknowns (u, v).

Now, substitute Equation 3.3 into the constraint equations Equation 3.1 to obtain

$$\begin{aligned}
(\mathbf{W}_1 - \mathbf{S}_1) \cdot (\mathbf{W}_1 - \mathbf{S}_1) &= R^2 \\
([D_{12}]\mathbf{W}_1 - \mathbf{S}_2) \cdot ([D_{12}]\mathbf{W}_1 - \mathbf{S}_2) &= R^2 \\
([D_{13}]\mathbf{W}_1 - \mathbf{S}_3) \cdot ([D_{13}]\mathbf{W}_1 - \mathbf{S}_3) &= R^2 \\
([D_{14}]\mathbf{W}_1 - \mathbf{S}_4) \cdot ([D_{14}]\mathbf{W}_1 - \mathbf{S}_4) &= R^2 \\
([D_{15}]\mathbf{W}_1 - \mathbf{S}_5) \cdot ([D_{15}]\mathbf{W}_1 - \mathbf{S}_5) &= R^2.
\end{aligned} \tag{3.4}$$

These five equations are solved to determine the five unknowns $\mathbf{W}_1 = (x+u, y+v)$, $\mathbf{G} = (u, v)$ and R.

Subtracting the first of these equations from the remaining equations cancels R^2 to obtain

$$\begin{aligned}
([D_{1i}]\mathbf{W}_i - \mathbf{S}_i) \cdot ([D_{1i}]\mathbf{W}_1 - \mathbf{S}_i) - (\mathbf{W}_1 - \mathbf{S}_1) \cdot (\mathbf{W}_1 - \mathbf{S}_1) = 0,& \\
i = 2, \dots, 5.&
\end{aligned} \tag{3.5}$$

The quadratic terms in these equations cancel so that they are bilinear in the variables (x, y, u, v), which means they can have as many as six solutions. Closer analysis shows that terms cancel in a manner such that two of these solutions are at infinity, which means in general there are at most four real solutions.

Of the four solutions obtained from the synthesis equations in (3.5), one is at infinity and defines the slider. Thus, the kinematic synthesis of slider-crank function generators yields at least one and as many three slider-crank linkages for five precision task values $\tau(s) = \psi$.

3.3 Assessment of Branching

The input-output relationship of a slider-crank linkage is double-valued, which means that a given slide s defines two values for the output crank, denoted ψ^+ or ψ^-. This means that a slider-crank that solves the design equations for a task function $\tau(s) = \psi$ will have two different output angles ψ^+ and ψ^- for each of the five specified input slide s.

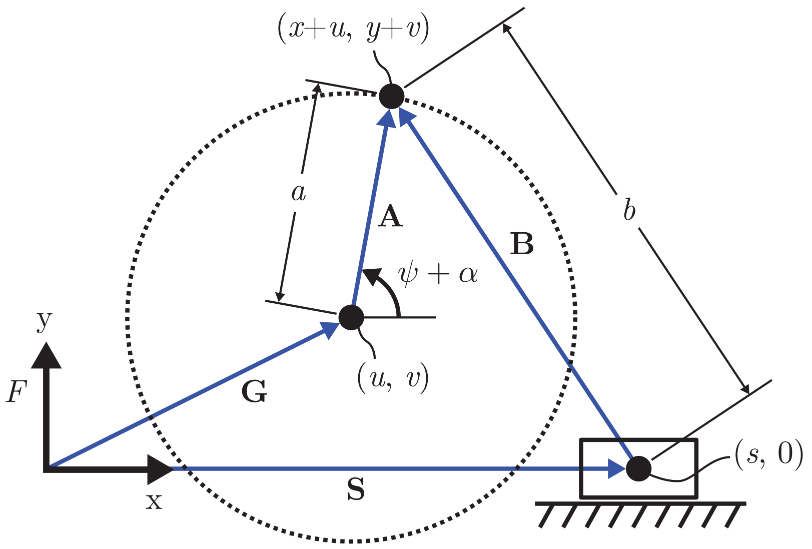


Figure 3.3: Linkage dimensions used to construct the input-output equations of a slider-crank.

The formula for the output angle ψ for a given input slide s is obtained as follows. Refer to Figure 3.3 and note that the vector loop that defines the slider-crank is given by

$$\mathbf{G} + \mathbf{A} - \mathbf{S} = \mathbf{B}, \quad \text{or} \tag{3.6}$$

$$\begin{Bmatrix} u \\ v \end{Bmatrix} + \begin{Bmatrix} a\cos(\psi+\alpha) \\ a\sin(\psi+\alpha) \end{Bmatrix} - \begin{Bmatrix} s \\ 0 \end{Bmatrix} = \begin{Bmatrix} b\cos\theta \\ b\sin\theta \end{Bmatrix} \tag{3.7}$$

The angle θ is eliminated by computing the magnitude of this equation to obtain

$$(u + a\cos(\psi + \alpha) - s)^2 + (v + a\sin(\psi + \alpha))^2 = b^2. \tag{3.8}$$

Notice that in this equation, the coordinates (x, y) are used to define the link lengths a and b. Thus, when input slide s is specified, this equation defines the offset output angle $(\psi + \alpha)$. For convenience, we note that a, b and α are determined from the values $(u,\ v,\ x,\ y)$ using the formulas

$$a = ||(x, y)|| \tag{3.9}$$

$$b = ||(x + u - s_1, y + v)|| \tag{3.10}$$

$$\alpha = \arctan\left(\frac{y}{x}\right) - \psi \tag{3.11}$$

where the double bars denote the norm of a vector.

Expand Equation 3.8 and collect the coefficients of $\cos(\psi + \alpha)$ and $\sin(\psi + \alpha)$ to obtain,

$$A(s)\cos(\psi + \alpha) + B(s)\sin(\psi + \alpha) + C(s) = 0, \tag{3.12}$$

where

$$\begin{aligned} A(s) &= 2ua - 2sa \\ B(s) &= 2va \\ C(s) &= (u - s)^2 + v^2 + a^2 - b^2. \end{aligned} \tag{3.13}$$

Solve this equation to obtain the two solutions,

$$\psi_i^+ = \arctan\left(\frac{B(s_i)}{A(s_i)}\right) + \arccos\left(\frac{-C(s_i)^2}{\sqrt{A(s_i)^2 + B(s_i)^2}}\right) - \alpha, \tag{3.14}$$

and

$$\psi_i^- = \arctan\left(\frac{B(s_i)}{A(s_i)}\right) - \arccos\left(\frac{-C(s_i)^2}{\sqrt{A(s_i)^2 + B(s_i)^2}}\right) - \alpha. \tag{3.15}$$

The condition that these solutions are real is

$$\mathbf{D} \equiv \left\{ s : \left| \frac{-C(s_i)^2}{\sqrt{A(s_i)^2 + B(s_i)^2}} \right| \leq 1 \right\} \tag{3.16}$$

Denote the values of the output crank angles as + and − to designate the two possibilities ψ_i^+ and ψ_i^- that occur in the design. For example $(+,+,+,+,+)$ represents output angles given by $(\psi_1^+, \psi_2^+, \psi_3^+, \psi_4^+, \psi_5^+)$. Notice that there are 32 possible combinations of output values corresponding to the specified inputs of which only the two cases $(+,+,+,+,+)$ and $(-,-,-,-,-)$ are useful. This suggests that for a random set of five precision values defining a task $\tau(s) = \psi$, there is a 1:15 chance that the resulting slider-crank avoids branching and is useful.

3.4 Tolerance Zones

It is conjectured that the suggested low odds of finding a non-branching solution can be overcome by synthesizing a large number of approximate linkages. This is achieved by randomly generating a large number of approximate task functions within tolerance zones. A task function $\tau(s) = \psi$ is defined by five corresponding pairs of numbers $(s_i,\ \psi_i)$ for $i = 1, \ldots, 5$. This allows for the specification of 10 distinct tolerance zones corresponding to the randomized parameters $\hat{s}_i$ and $\hat{\psi}_i$ such that

$$\hat{s}_i \in [L_{si}, H_{si}] \tag{3.17}$$

$$\hat{\psi}_i \in [L_{\psi i}, H_{\psi i}] \qquad i = 1, \ldots, 5, \tag{3.18}$$

where the L and H parameters are the lower and upper limits of each tolerance zone. Large numbers of approximate task functions are generated in order to synthesize and analyze mechanisms according to the algorithm in Figure 3.4. A survey of this algorithm proceeds. Example designs using this algorithm are presented in Sections 3.5 and 3.6.

The algorithm of Figure 3.4 was surveyed using the task function shown in Table 3.1(a). Note that applying the synthesis routine of Section 3.2 does not result in a useful linkage. Therefore, our algorithm was executed for these task positions in segments of a length of 100 iterations. For each segment, the size of tolerance zones increased. The size of the i^{th} tolerance zone for each segment was defined as a function of κ, where

$$\begin{aligned}
L_{si} &= s_i - \kappa(s_{max} - s_{min}) \\
H_{si} &= s_i + \kappa(s_{max} - s_{min}) \\
L_{\psi i} &= \psi_i - \kappa(\psi_{max} - \psi_{min}) \\
H_{\psi i} &= \psi_i + \kappa(\psi_{max} - \psi_{min}) \qquad i = 1, \ldots, 5.
\end{aligned} \tag{3.19}$$

The results of this exercise are shown in Table 3.1(b). The greatest number of useful linkages found was 51 at $\kappa = 5\%$, in which 40% of the generated task functions yielded useful designs. For all values of κ tested, useful linkages were outnumbered by defective ones.

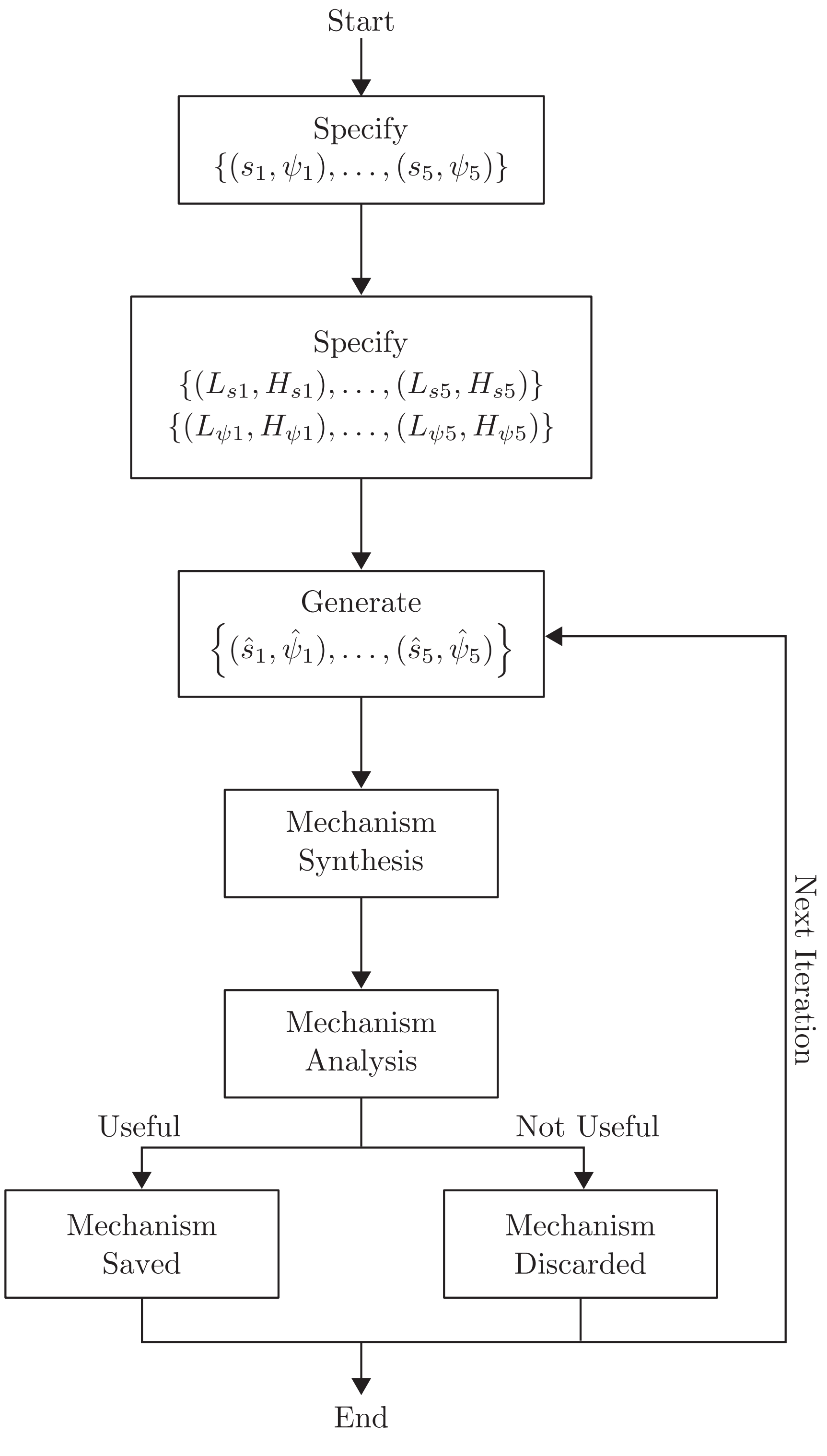

Start
Specify
$\{(s_1, \psi_1), \ldots, (s_5, \psi_5)\}$
Specify
$\{(L_{s1}, H_{s1}), \ldots, (L_{s5}, H_{s5})\}$
$\{(L_{\psi 1}, H_{\psi 1}), \ldots, (L_{\psi 5}, H_{\psi 5})\}$
Generate
$\left\{(\hat{s}_1, \hat{\psi}_1), \ldots, (\hat{s}_5, \hat{\psi}_5)\right\}$
Mechanism Synthesis
Mechanism Analysis
Useful
Not Useful
Mechanism Saved
Mechanism Discarded
Next Iteration
End

Figure 3.4: Task randomization within tolerance zones is applied to the design of slider-crank function generators.

Table 3.1: The design algorithm was executed for the task function listed in (a). The results for various sized tolerance zones are contained in (b).

(a)

i	s_i	ψ_i
1	10	20°
2	25	60°
3	45	70°
4	70	70°
5	100	50°

(b)

κ	Useful Tasks	Useful Linkages	Defective Linkages
1%	3:100	3	297
2%	18:100	20	262
3%	22:100	26	234
4%	32:100	41	207
5%	40:100	51	178
10%	33:100	35	163
50%	7:100	8	126
100%	5:100	6	126

3.5 Hydraulic Front Loader Actuation Linkage

Our synthesis procedure is further illustrated through its application to the design of a hydraulically actuated shovel for a front-end loader. A set of function points σ were chosen in order to specify the relationship between the movement of a hydraulic linear actuator and the task angle of a shovel tool. A nonlinear function $\tau(s) = \psi$ was selected as shown in Figure 3.5 in order to achieve smaller angular velocities of the shovel near its minimum and maximum angles of $-80°$ and $80°$, respectively. The functional values and corresponding tolerances for each point are shown in Table 3.2 as well as functions that resulted in useful and defective solutions. The motion objective near the shovel's minimum and maximum angles was achieved for the useful task function example as evidenced by the mechanical advantage shown in Figure 3.6.

It is significant to note that solving for the exact specified five precision points does not result in a defect free linkage. By defining tolerance zones for each point and implementing the above algorithm for 1000 loops, 21 defect free linkages were found and 1001 defective linkages were found. A defect free linkage is shown in Figure 3.7(a) and a defective linkage is shown in Figure 3.7(b). The defective linkage of Figure 3.7(b) was found to have a branch

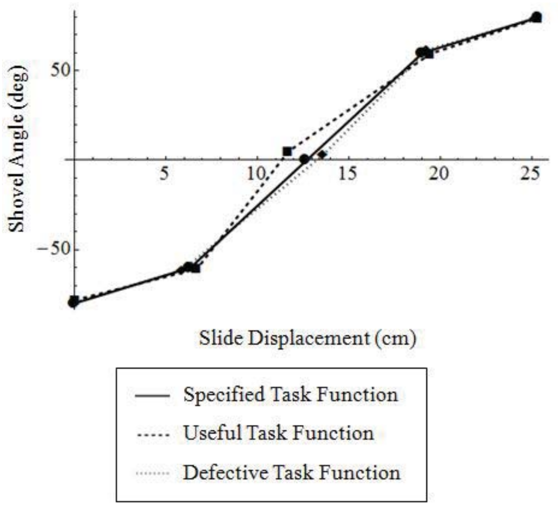


Figure 3.5: The specified shovel task function plotted alongside useful and defective examples of toleranced task functions.

Table 3.2: The specified task function, it tolerances, and tasks that resulted in useful and defective results.

Specified Task Function		Useful Task Function		Defective Task Function	
s	ψ	s	ψ	s	ψ
0 cm $+0.254$ cm	$-80°$ $+2°$	0.032 cm	$-78.17°$	0.051 cm	$-79.37°$
6.350 cm ±0.508 cm	$-60°$ $\pm2°$	6.704 cm	$-60.64°$	5.908 cm	$-61.36°$
12.700 cm ±1.270 cm	$0°$ $\pm5°$	11.710 cm	$5.00°$	13.631 cm	$3.22°$
19.050 cm ±0.508 cm	$60°$ $\pm2°$	19.465 cm	$59.47°$	19.302 cm	$61.79°$
25.400 cm -0.254 cm	$80°$ $-2°$	25.352 cm	$79.28°$	25.259 cm	$79.20°$

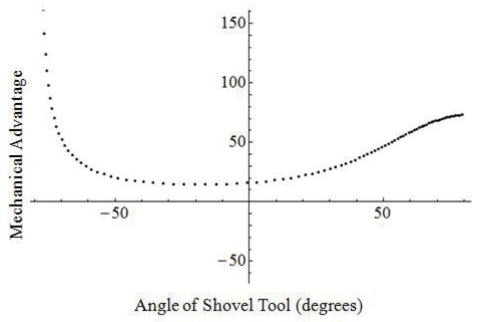
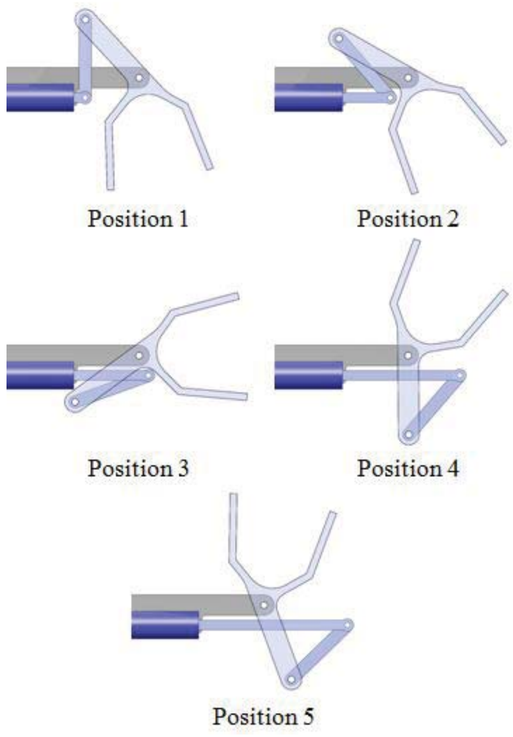
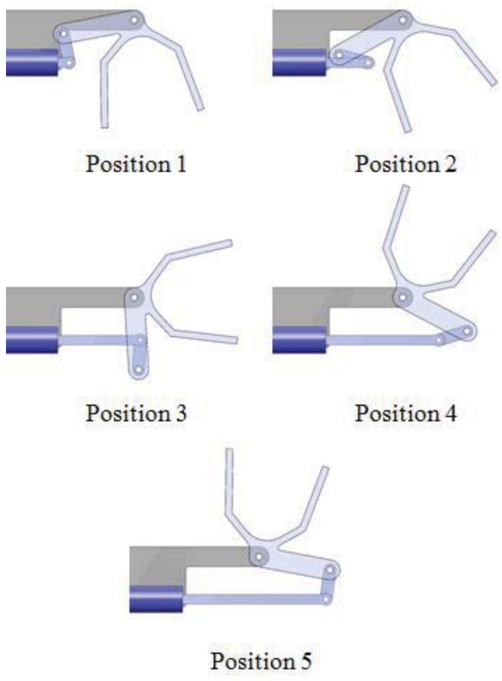


Figure 3.6: Mechanical advantage of the proposed slider-crank shovel linkage for an output force couple with a moment arm $R = 1$ cm.

defect according to the criteria specified in Section 3.3 of this paper. Simulation through solid modeling software shows the mechanism would require disassembly to move from the 2nd position to the 3rd position. To move from the 3rd position to the 4th position would require actuation on an alternate member in order to pass through a singular configuration. Therefore this linkage design is not practical for most applications.



(a) Useful linkage.



(b) Defective linkage.

Figure 3.7: (a) A defect free shovel linkage design in five task positions and (b) a defective shovel linkage design in five task positions.

3.6 Rotating Weaver Dwell Linkage

This algorithm was also used to design an actuation linkage for a rotating weaver machine. The purpose of a rotating weaver machine is to braid wire. These machines work at high speeds and are commonly used in the hydraulic hose industry. They also have applications for the weaving of various yarns. A rotating weaver machine that uses a linkage is shown in Figure 3.8. The machine consists of two radially aligned bobbin assemblies that rotate relative to each other. A linkage is attached to the rear assembly. The purpose of the linkage is to move a wire over and under the bobbins as they move passed it in a specific pattern. This pattern is subject to change for different weaving set-ups. The requirement in this case was to modify the linkage to accommodate a 3 over 3 pattern. Specifically, this pattern requires the wire to dwell longer outside the bobbin assembly, weave faster through the bobbins, dwell longer inside the bobbin assembly, then weave faster back to the outside. All of this should be accomplished through a constant rotary input.

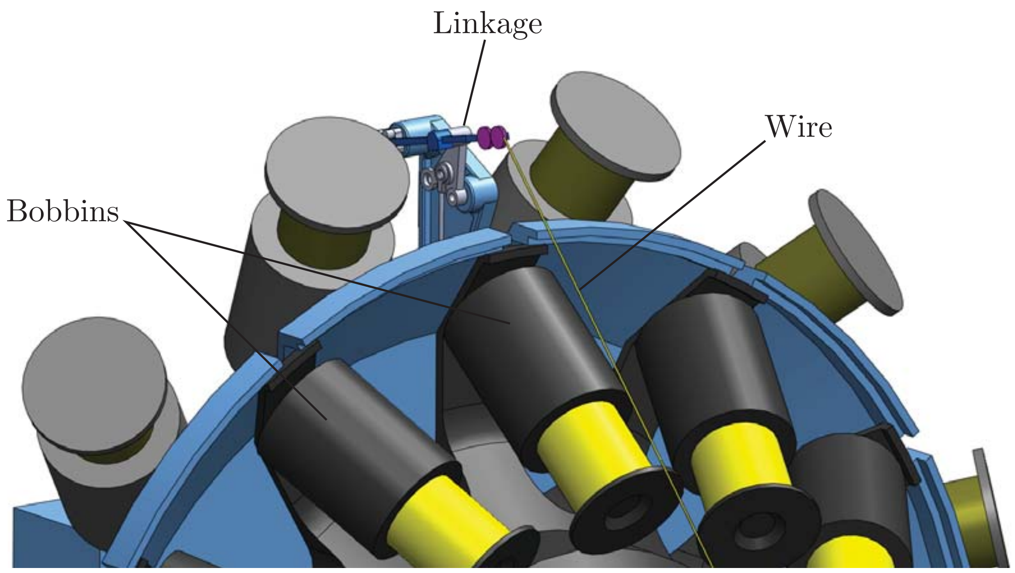


Figure 3.8: A rotating weaver. The weaver moves the yellow wire over and under rotating bobbins.

The linkage used in order to accomplish this consists of two parts: (1) a function generating linkage and (2) a path generating linkage. These two linkages are combined serially such that actuation of the function generator moves the path generator, creating a single input-output relation between the input link of the function generator and the position of the end effector. A standard weaver linkage is shown in Figure 3.9.

Note that the path of the linkage's end effector does not need to change, only its speed along that path. Therefore, modifications only need to be made to the function generation portion of the weaver linkage. The current design uses a spatial function generator. The proposed modified linkage will include a planar slider-crank function generator. In order to influence

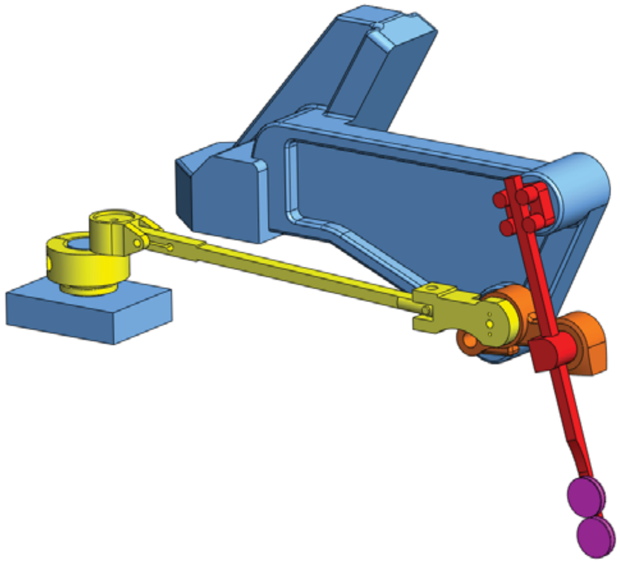


Figure 3.9: The rotating weaver linkage. Blue links are fixed with respect to all other colors. The function generator is yellow and orange. The path generator is orange and red. The end effector is pink.

the required dwells, the task function shown in Table 3.3 is specified. A sliding displacement of 3 in will be used to rock the input link of the path generator 134°. The input link of the path generator is the output link of the function generator which is the orange link in Figure 3.9.

Table 3.3: Task function and tolerances for the rotating weaver dwell linkage.

s		ψ	
0 in	±0 in	0°	±0°
0.75 in	+0.20/−0.40 in	17°	+2°/−10°
1.50 in	+0.30/−0.30 in	67°	+20°/−20°
2.25 in	+0.40/−0.20 in	117°	+10°/−2°
3.00 in	±0 in	134°	±0°

Execution of the synthesis algorithm for 500 iterations resulted in over 500 solutions. That is more than a 1:1 ratio of usable solutions to random task functions. The input-output function of the linkage chosen to be implemented in this design is shown in Figure 3.10. However, this slider-crank function generator cannot be directly integrated into the weaver linkage design because it has a sliding input. One of the design requirements for the weaver linkage is that it should be actuated by a constant rotary input. Therefore, another linkage needs to be synthesized in order to actuate the slider-crank function generator.

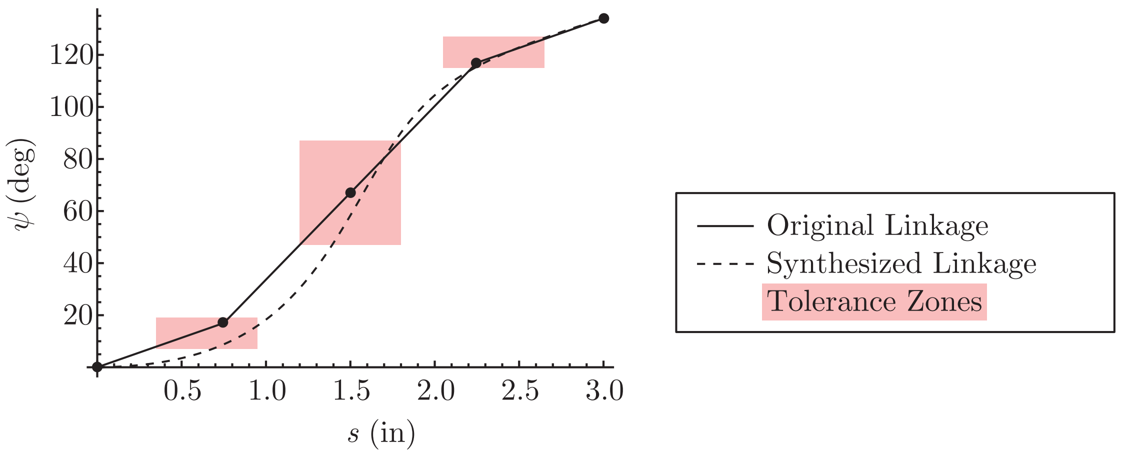


Figure 3.10: The input-output function of the synthesized linkage compared to the originally specified task function.

The strategy used to design the function generator's actuation linkage was to adapt a well known linkage solution for this problem. Instead of formulating a mathematical synthesis routine for this linkage, a simple crank-slider mechanism was chosen that is capable of oscillating a sliding link 3 in. Furthermore, this crank-slider design has a similar dwell characteristic as these slider-crank function generator, and thus the coordinated coupling of the two mechanisms compounds their dwells. When coupled, these mechanisms form a Watt II function generator. Dimensions of the design are shown in Figure 3.11. Link I of this mechanism is actuated and Link V is attached to the input link of the path generator.

A simulation was executed in which the input crank was rotated 360° at a constant rate over 9 seconds. The effect of the new function generator on the dwell and speed of the end effector of the weaver linkage is shown in Figure 3.12. If we make the assumption that the red lines at $y = \pm 5$ are taken to be the thresholds of when the end effector enters and exits a dwell, then the linkage modification increased the dwell outside the bobbin assembly by about 0.5 sec and the dwell inside the bobbin assembly by over 2 seconds. As well, this is accomplished without repositioning the input axis of the path generator nor the input axis of the entire weaver linkage.

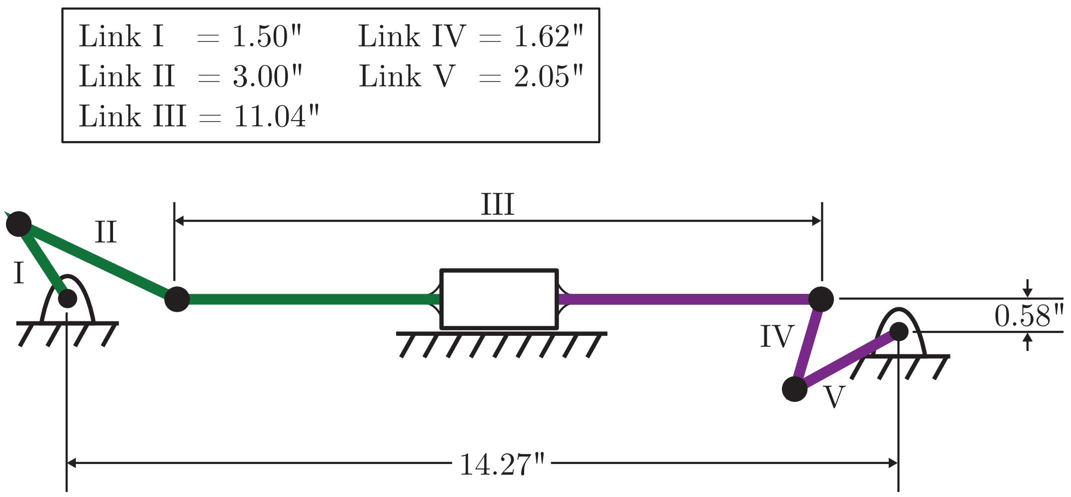


Figure 3.11: A single degree of freedom Watt II function generator. The purple linkage was designed using the synthesis algorithm of Section 3.2. The green linkage was added to actuate the purple slider-crank function generator.

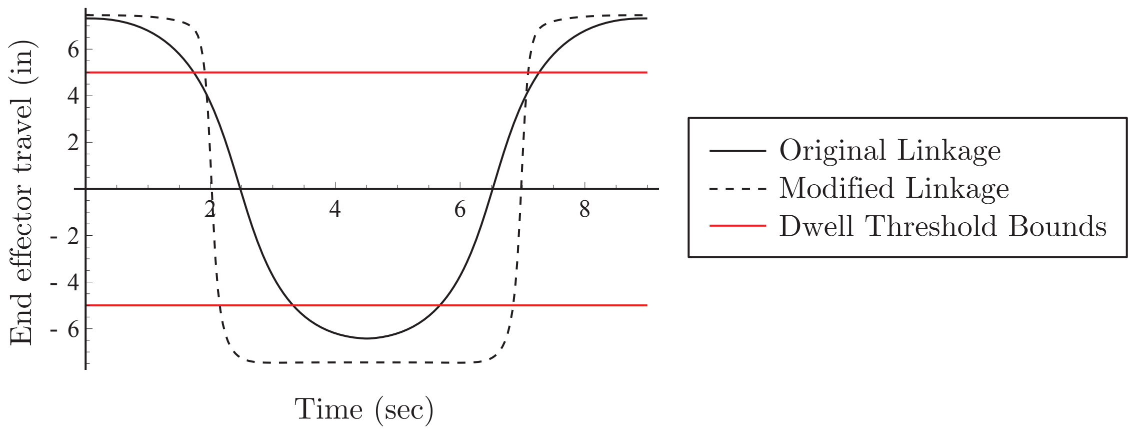


Figure 3.12: The weaving movement of the end effector through the bobbins is compared between the original and modified linkages.

A CAD model of the final linkage is shown in Figure 3.13. It is important to note that the link drawn in Figure 3.11 is kinematically different from the final linkage as it is not contained by a single plane. However, Figure 3.11 provides a good approximation of the final linkage's input crank.

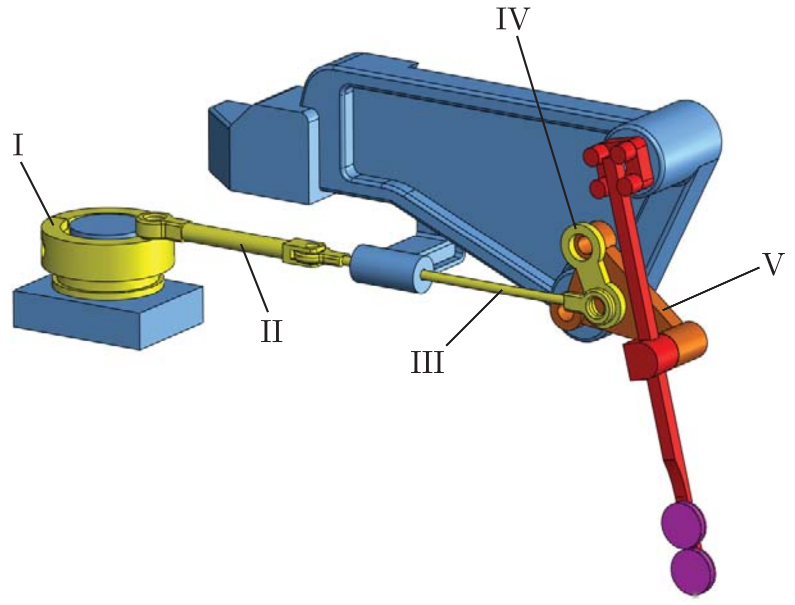


Figure 3.13: The modified weaver linkage.

3.7 Summary

This chapter presented a design process for a specific topology of a planar four-bar linkage called a PRRR slider-crank function generator. The purpose of a function generator is to coordinate an input slide with an output rotation angle. Although the process for synthesizing a slider-crank function generator for any given set of 5 task points guarantees a linkage capable of posing these positions, the results may be defective. This chapter presents a strategy for designers when confronted with defective linkages. The strategy is illustrated through two in depth examples of applying the mechanism design process to the designs of useful devices.

Chapter 4

Motion Generation for a Slider-Crank Six-Bar

This chapter discusses a synthesis technique for designing a motion generating slider-crank six-bar from a RPR serial chain. The resulting linkage will be capable of moving an end effector through five task positions. The process includes a synthesis step for each of the two kinematic loops that compose this Watt I six-bar. At each step, there is the possibility that the sublinkages returned by the synthesis results possess a branch or circuit defect. Therefore, an approximate synthesis method is accomplished by executing a finite position synthesis method for task position sets that are randomly generated within specified tolerance zones. As well, the use of randomization within tolerance zones is extended to the specified parameters of the base RPR chain. This six-bar design technique is then applied to two example designs: (1) a screw insertion linkage capable of generating a square pattern and (2) a linkage used for the transplantation of rice seedlings. The synthesis results of the first example yielded 122 defect-free linkages for 1 million iterations. The second example resulted in 50 useful linkages for 1000 iterations.

4.1 Introduction

The objective of the synthesis method described below is five task position motion generation. The synthesis procedure begins with the selection of a planar RPR serial chain. The inverse kinematics of the serial chain are solved for each task position in order to find the position and orientation of each of its links. This information is then used to find geometric constraints that reduce the mobility of the linkage to a single degree of freedom. These geometric constraints are found using procedures similar to the RR synthesis procedure discussed in Section 2.2.1, however tailored for this six-bar topology. In order to identify branch defects, an analysis procedure similar to the analysis procedure discussed in Section 2.4 is executed for the results of each synthesis step. Analysis of the resulting linkage determines if it moves the end-effector smoothly through the five task positions. The design process presented randomly selects variations of the task positions and RPR parameters in order to obtain new six-bar linkages. This dimensional synthesis algorithm yields a set of six-bar linkages that move the end-effector near the original task positions.

4.2 RPR Specification

The first step in the design procedure is to specify an RPR chain that reaches the five specified task positions

$$[T_i] = [T(\phi_i, x_i, y_i)] \quad i = 1, \ldots, 5, \tag{4.1}$$

where $[T]$ is a 3x3 planar homogeneous matrix capable of transforming coordinates to a fixed frame F from a frame displaced by $\mathbf{P} = (x, y)$ and rotated at ϕ. The specified parameters of the RPR chain are $\mathbf{O}$, p, and η as shown in Figure 4.1. Point $\mathbf{O} = (O_x, O_y)$ locates the fixed pivot, p is the length from the slider to the end effector, and η is the angular offset of the end effector frame from the **PA** frame.

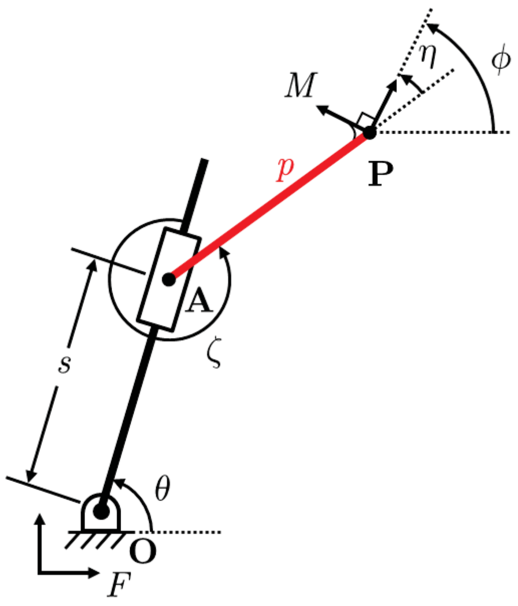


Figure 4.1: A three degree of freedom open loop RPR chain.

The position (x, y) and the orientation ϕ of the end effector is described by the following equations,

$$\begin{Bmatrix} s\cos\theta \\ s\sin\theta \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix} - \begin{Bmatrix} O_x \\ O_y \end{Bmatrix} - \begin{Bmatrix} p\cos(\theta+\zeta) \\ p\sin(\theta+\zeta) \end{Bmatrix} \tag{4.2}$$

$$\phi = \theta + \zeta + \eta \mod 2\pi, \tag{4.3}$$

where joint parameters s, θ, ζ are shown in Figure 4.1. In order to find the inverse kinematics solution for s, θ, and ζ, Equation (4.3) is substituted into Equation (4.2) and the magnitude of the resulting vector equation is computed to obtain

$$s = \pm\sqrt{(x - O_x - p\cos(\theta-\eta))^2 + (y - O_y - p\sin(\theta-\eta))^2}. \tag{4.4}$$

The angle θ is computed by dividing the y-component of (4.2) by its x-component and applying an arctan function,

$$\theta = \arctan\left(\frac{y - O_y - p\sin(\theta - \eta)}{x - O_x - p\cos(\theta - \eta)}\right) \tag{4.5}$$

The computation of the final joint parameter ζ is straightforward from Eqn. 4.3. Note that the positive and negative values of (4.4) correspond to 2 solutions of $\{s, \theta, \zeta\}$ which describe the same configuration of the RPR chain. Therefore only the positive value of s will be considered.

4.3 Synthesis of the First RR Constraint

The inverse kinematics procedure (Section 4.2) is applied for all task positions to obtain s_i and θ_i for $i = 1, \ldots, 5$. Next the slider link is connected to ground by a RR chain so that the resulting inverted slider-crank loop is capable of achieving each set of joint parameters $\{s_i, \theta_i\}$. This is accomplished by first defining the sliding point

$$\mathbf{A}_i = \begin{Bmatrix} O_x + s_i\cos\theta_i \\ O_y + s_i\sin\theta_i \end{Bmatrix} \tag{4.6}$$

which is used to create the transformations $[T(\theta_i, \mathbf{A}_i)]$ for point $\mathbf{C}$ (Fig. 4.2) from coordinates in link frame $\mathbf{CA}$ to coordinates in F. Then the relative transformations of $\mathbf{C}$ from its configuration in the first task position to the other four task positions can be defined as

$$[D_{1i}] = [T(\theta_i, \mathbf{A}_i)][T(\theta_1, \mathbf{A}_1)]^{-1} \quad i = 2, \ldots, 5. \tag{4.7}$$

Therefore the positions of $\mathbf{C}$ associated with the five task positions can be written as

$$\mathbf{C}_i = [D_{1i}]\mathbf{C}_1. \tag{4.8}$$

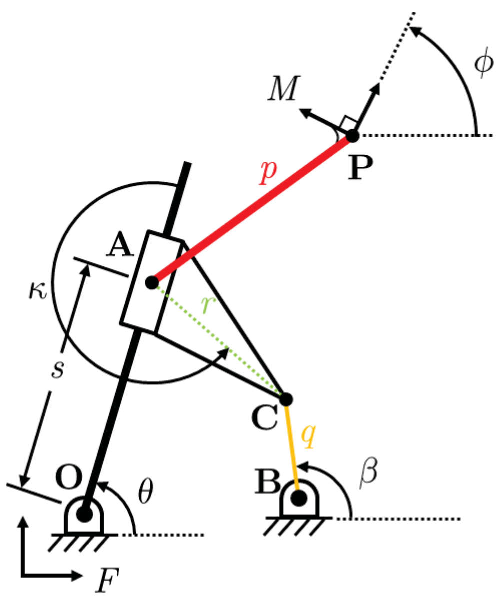


Figure 4.2: The synthesis of an RR chain constrains the linkage to two degrees of freedom.

Next, the unknown ground point **B** is defined in F as shown in Fig. 4.2. The constraint equations for an RR link that connects point **B** to **C** for all task positions is given by

$$([D_{1i}]\mathbf{C}_1 - \mathbf{B}) \cdot ([D_{1i}]\mathbf{C}_1 - \mathbf{B}) = q^2. \tag{4.9}$$

where q is the constant length of link **CB**.

These design equations can be simplified by cancelling q^2 to obtain four bilinear equations in the four unknowns B_x, B_y, C_{x1}, and C_{y1}. The solution of these design equations is the same as that shown in Section 2.2.2. One of these solutions will be the specified RP chain with fixed pivot $\mathbf{B} = \mathbf{O}$ and a moving pivot $\mathbf{C}_1$ at infinity. Therefore this design procedure will yield one or three additional real RR chains.

4.4 Assessment of the First RR Constraint

The pivots **O**, **A**, **C**, and **B** form an inverted slider-crank loop. The pivots **A** and **C** are attached to the slider and **O** and **B** are attached to ground. The vector **C** − **A** has magnitude r and the constant angle κ measured from the vector **A** − **O**.

The input-output equations for an inverted slider-crank can be used to detect branch defects. They are formulated through the vector loop equation,

$$\begin{Bmatrix} O_x \\ O_y \end{Bmatrix} + \begin{Bmatrix} s\cos\theta \\ s\sin\theta \end{Bmatrix} + \begin{Bmatrix} r\cos(\theta+\kappa) \\ r\sin(\theta+\kappa) \end{Bmatrix} = \begin{Bmatrix} B_x \\ B_y \end{Bmatrix} + \begin{Bmatrix} q\cos(\beta) \\ q\sin(\beta) \end{Bmatrix}. \tag{4.10}$$

The solutions for s are

$$s^{\pm} = \frac{-b \pm \sqrt{b^2-4c}}{2} \tag{4.11}$$

$$\begin{aligned} \text{where} \quad b =& 2(O_x-B_x)\cos\theta + 2(O_y-B_y)\sin\theta + 2r\cos\kappa \\ c =& (O_x-B_x)^2 + (O_y-B_y)^2 + r^2 - q^2 + 2r(O_x-B_x)\cos(\theta-\kappa) \\ & - 2r(O_y-B_y)\sin(\theta-\kappa). \end{aligned}$$

For a given input θ, Equation 4.11 has two solutions. The solutions to the loop equations are denoted $\{s,\beta\}^+$ and $\{s,\beta\}^-$. These are assembled into the two sets

$$\begin{aligned} \sigma^+ =& \{\{\theta_1, s_1^+, \beta_1^+\}, \ldots, \{\theta_i, s_i^+, \beta_i^+\}, \ldots, \{\theta_5, s_5^+, \beta_5^+\}\}, \\ \sigma^- =& \{\{\theta_1, s_1^-, \beta_1^-\}, \ldots, \{\theta_i, s_i^-, \beta_i^-\}, \ldots, \{\theta_5, s_5^-, \beta_5^-\}\}, \end{aligned} \tag{4.12}$$

which are two branches of the linkage. A linkage is useful if the configurations obtained for each of the task positions are on the same branch. A linkage that does not have the task positions on the same branch is not useful.

This procedure determines the values of $\{\theta_i, s_i, \beta_i\}$, $i = 1, \ldots, 5$ and compares the results to σ^+ and σ^-. If all of the configurations are on one branch or the other, the linkage is considered to be useful. This procedure can fail due to the complexity of six-bar linkage coupler curves [36], therefore a final visual confirmation is necessary.

4.5 Synthesis of the Second RR Constraint

The synthesis and analysis procedures of the first RR constraint result in 0 to 3 partial six-bar assemblies passed on to the second RR constraint procedure. The second procedure parallels the first in that 0 to 3 complete six-bar mechanisms will result from each partial six-bar, allowing for a maximum synthesis of 9 six-bar mechanisms.

The synthesis procedure for the second RR chain computes the points **D** in the link **CB** frame (Fig.4.3) and **E** in the link **PA** frame. The relative transformations from the configuration of the first task position to all other task positions that define **D** and **E** are given by

$$[G_{1i}] = [T(\beta_i, \mathbf{B})][T(\beta_1, \mathbf{B})]^{-1} \tag{4.13}$$

$$[H_{1i}] = [T(\phi_i, x_i, y_i)][T(\phi_1, x_1, y_1)]^{-1} \tag{4.14}$$

respectively, so that locations $\mathbf{D}_i$ and $\mathbf{E}_i$ in F are

$$\mathbf{D}_i = [G_{1i}]\mathbf{D}_1 \tag{4.15}$$

$$\mathbf{E}_i = [H_{1i}]\mathbf{E}_1 \quad i = 2, \ldots, 5. \tag{4.16}$$

The constraint equation for an RR link that connects points **D** and **E** is given by

$$([G_{1i}]\mathbf{D}_1 - [H_{1i}]\mathbf{E}_1) \cdot ([G_{1i}]\mathbf{D}_1 - [H_{1i}]\mathbf{E}_1) = v^2, \tag{4.17}$$

where v is the constant length of the link **ED**.

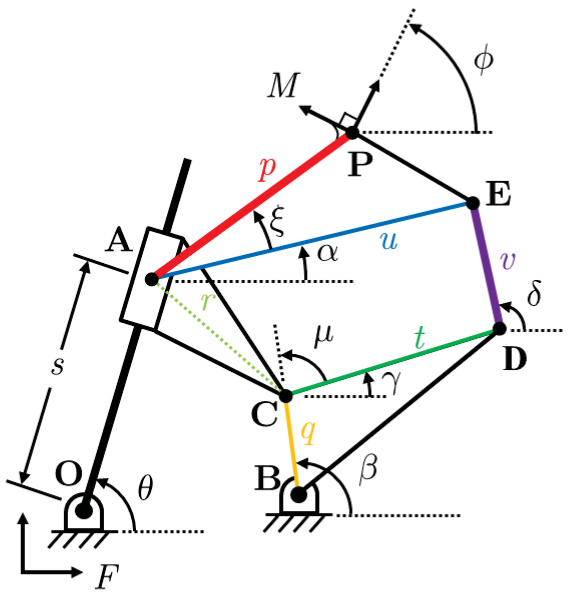


Figure 4.3: The synthesis of a second RR chain constrains the mechanism to a single degree of freedom.

Equation 4.17 has the same structure as Eqn. 4.9 forming four bilinear equations in four unknowns D_{x1}, D_{y1}, E_{x1}, and E_{y1}. As well in this case, four solutions are found, two of which may be imaginary and one that reproduces the link **CA**, that is $\mathbf{D}_1 = \mathbf{C}_1$ and $\mathbf{E}_1 = \mathbf{A}_1$. Therefore, either one or three links exist that connect link **CB** to the link **PA** for the given task positions.

4.6 Assessment of the Second RR Constraint

The moving pivot **D** is located on the link **CB**. It is at a distance t from **C** so that $\mathbf{C} - \mathbf{B}$ is at a constant angle μ from $\mathbf{D} - \mathbf{C}$. The moving pivot **E** is located on link **PA**. It is located at a distance u from **A** so that $\mathbf{P} - \mathbf{A}$ is at a constant angle ξ from $\mathbf{E} - \mathbf{A}$. Points **C**, **D**, **E**, and **A** form a 4R loop.

Using a process similar to the inverted slider-crank loop analysis presented in Section 4.3, the input-output equations of a 4R loop can be used to find branch defects. The input-output equations are formulated in order to compute α^+ and α^- for an input variable γ. First the loop equations are formed,

$$\begin{Bmatrix} \mathbf{A}_x \\ \mathbf{A}_y \end{Bmatrix} + \begin{Bmatrix} u\cos\alpha \\ u\sin\alpha \end{Bmatrix} = \begin{Bmatrix} \mathbf{C}_x \\ \mathbf{C}_y \end{Bmatrix} + \begin{Bmatrix} t\cos\gamma \\ t\sin\gamma \end{Bmatrix} + \begin{Bmatrix} v\cos\delta \\ v\sin\delta \end{Bmatrix}, \tag{4.18}$$

then are solved for α to obtain two solutions,

$$\begin{aligned} \alpha^{\pm} =& \arctan\left(\frac{B}{A}\right) \pm \arccos\left(\frac{-C}{\sqrt{A^2+B^2}}\right) \\ \text{where} \quad A =& 2u(A_x - C_x) - 2ut\cos\gamma \\ B =& 2u(A_y - C_y) - 2ut\sin\gamma \\ C =& r^2 + u^2 + t^2 - v^2 - 2v(A_x - C_x)\cos\gamma \\ & - 2t(A_y - C_y)\sin\gamma. \end{aligned} \tag{4.19}$$

For a given input γ, Equation 4.19 has two solutions. The solutions to the loop equations are denoted $\{\alpha, \delta\}^+$ and $\{\alpha, \delta\}^-$. These are assembled into the sets

$$\begin{aligned} \tau^+ =& \{\{\gamma_1, \alpha_1^+, \delta_1^+\}, \ldots, \{\gamma_i, \alpha_i^+, \delta_i^+\}, \ldots, \{\gamma_5, \alpha_5^+, \delta_5^+\}\}, \\ \tau^- =& \{\{\gamma_1, \alpha_1^-, \delta_1^-\}, \ldots, \{\gamma_i, \alpha_i^-, \delta_i^-\}, \ldots, \{\gamma_5, \alpha_5^-, \delta_5^-\}\}. \end{aligned} \tag{4.20}$$

to form the two solution branches. The values of $\{\gamma_i, \alpha_i, \delta_i\}$, $i = 1, \ldots, 5$ are determined and compared to τ^+ and τ^-. If all configurations are on a single branch, then the linkage is considered useful. Note that this condition does not guarantee the absence of circuit defects. Therefore a final visual confirmation is necessary to ensure the six-bar moves smoothly through all task positions.

4.7 Screw Insertion Linkage

This synthesis procedure begins with five task positions $[T_i]$ for $i = 1, \ldots, 5$ specified by the designer. Dimensional synthesis computes as many as nine design candidates. If the evaluation of branching for these candidate linkages fails, then the task positions are adjusted within specified tolerance zones. The adjustment process randomly selects new values $\hat{\phi}_i$, $\hat{x}_i$, and $\hat{y}_i$, $i = 1, \ldots, 5$ that lie within regions defined by the designer. The synthesis procedure is repeated for the new task positions $[\hat{T}_i]$. The design algorithm is shown in Figure 4.4.

This procedure was applied to the design of an RPR six-bar linkage that guides a screw insertion device. The requirement is for a linkage design capable of positioning a screw insertion device exactly over four locations evenly spaced on a bolt hole circle of diameter 7.18 cm. The proposed machine would stop at each location to allow the device to insert a screw. The linkage would then need to cycle out to pick up more screws and repeat the process. As well, the linkage needs to move the inserter to each bolt hole in a star pattern. Finally, the additional requirement was instated that the fixed pivots of the mechanism be outside the bolt hole circle.

An illustration of the motion requirement is shown in Figure 4.5. The specified task positions and tolerance zones for this design problem are listed in Table 4.1(a). Notice that a zero tolerance was specified on the position of the end effector for the first four tasks as precision is required. However, for the fifth task position a very large tolerance was specified. Larger tolerance zones were specified on all orientations as these were arbitrary for all positions. The dimensions of the RPR chain, $\mathbf{O}, p, \eta$, are listed in Table 4.1(b). Note that tolerances were applied to the dimensions of the RPR linkage as well.

The design algorithm of Figure 4.4 was executed for 1 million iterations which resulted in 122 useful linkage designs. The computation was performed on a 3.01 GHz AMD Phenom(tm) II X4 945 processor. The run-time for this Mathematica program was approximately 0.11

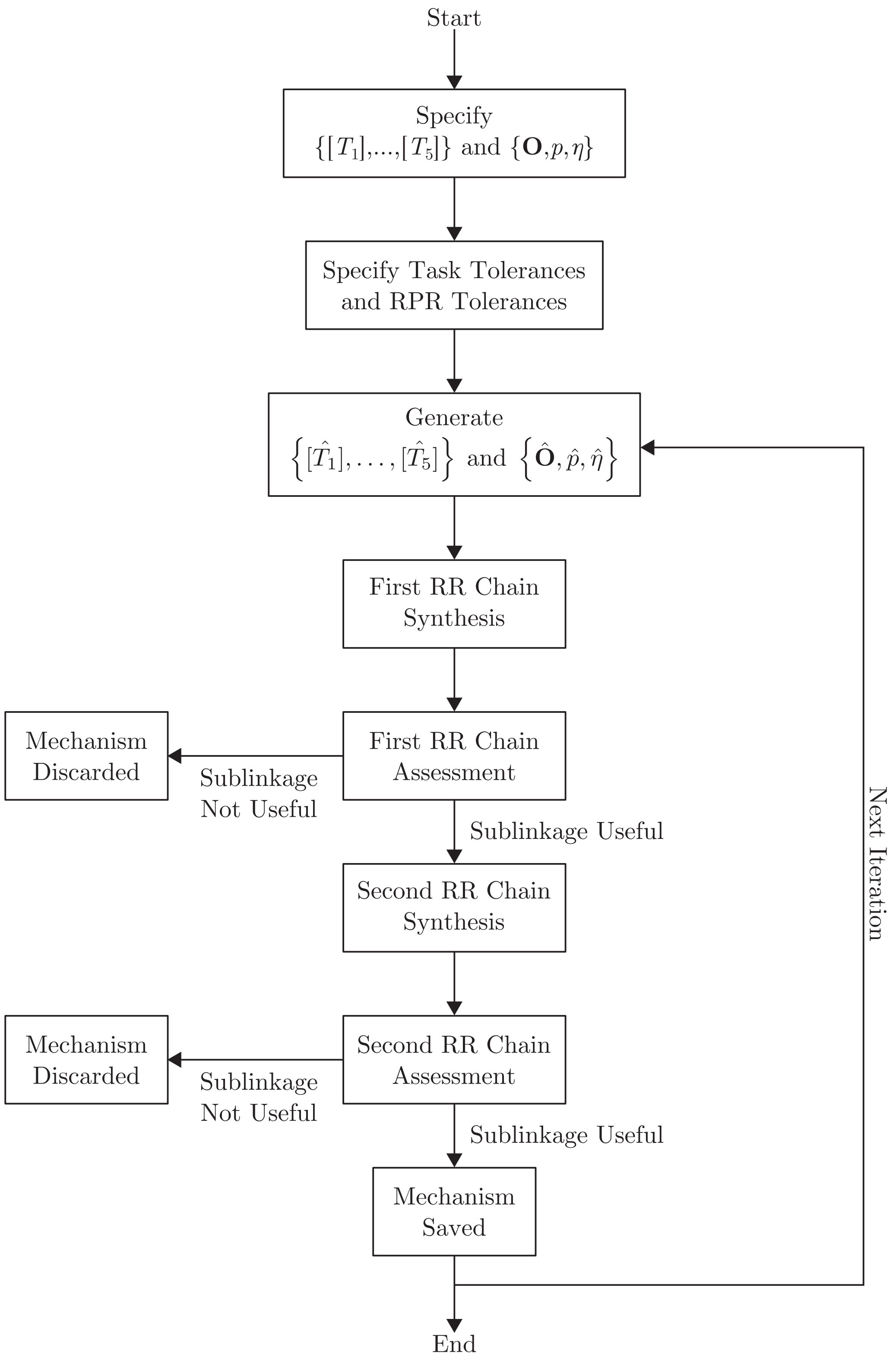

Start
Specify
$\{[T_1],...,[T_5]\}$ and $\{\mathbf{O},p,\eta\}$
Specify Task Tolerances and RPR Tolerances
Generate
$\left\{[\hat{T}_1],\ldots,[\hat{T}_5]\right\}$ and $\left\{\hat{\mathbf{O}},\hat{p},\hat{\eta}\right\}$
First RR Chain Synthesis
First RR Chain Assessment
Mechanism Discarded
Sublinkage Not Useful
Sublinkage Useful
Second RR Chain Synthesis
Second RR Chain Assessment
Mechanism Discarded
Sublinkage Not Useful
Sublinkage Useful
Mechanism Saved
Next Iteration
End

Figure 4.4: Task randomization within tolerance zones is applied to the design of six-bar linkages.

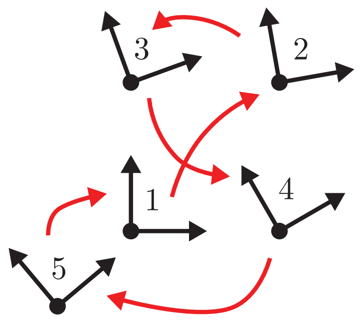


Figure 4.5: The proposed task positions for the screw insertion linkage.

Table 4.1: The task positions and tolerances for the screw insertion linkage are displayed in (a). The RPR serial chain parameters and tolerances are displayed in (b).

(a) Task Positions

Original Task Positions			Useful Task Positions		
ϕ	x (cm)	y (cm)	ϕ	x (cm)	y (cm)
0° ±10°	2.54 ±0	2.54 ±0	6.48°	2.54	2.54
10° ±10°	7.62 ±0	7.62 ±0	5.72°	7.62	7.62
20° ±10°	2.54 ±0	7.62 ±0	16.89°	2.54	7.62
30° ±10°	7.62 ±0	2.54 ±0	35.61°	7.62	2.54
40° ±10°	0 ±25.40	0 ±25.40	40.03°	7.53	-18.52

(b) RPR Parameters

	Original Parameters	Useful Parameters
O_x	0 ±25.4 cm	-20.62 cm
O_y	0 ±25.4 cm	10.39 cm
p	12.70 ±12.70 cm	18.25 cm
η	0 ±180°	18.33°

secs/iteration, or 30 hours of computation. The useful designs were visually inspected for star pattern ordering, acceptable fixed pivot location, and compactness. The linkage chosen from this visual inspection is shown in Figure 4.6. It meets all the motion requirements described above.

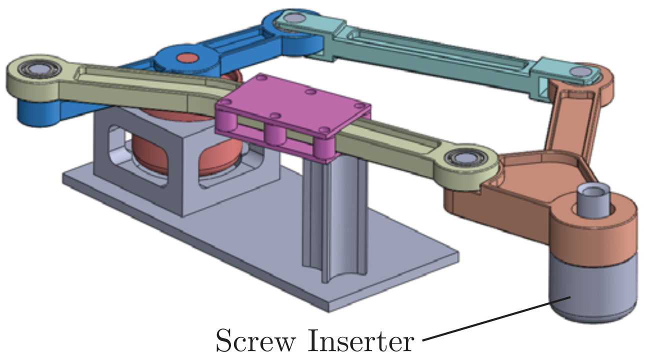
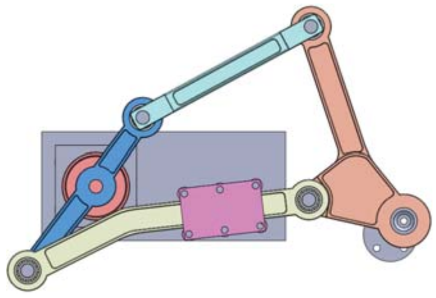
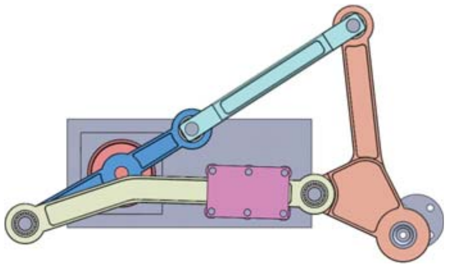


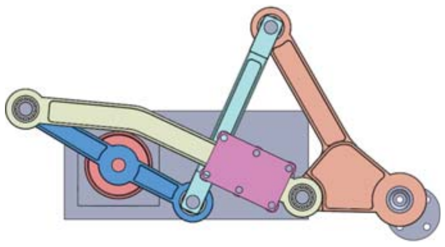
Figure 4.6: The square pattern screw insertion linkage.



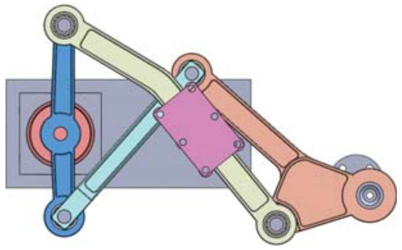
(a) Position 1.



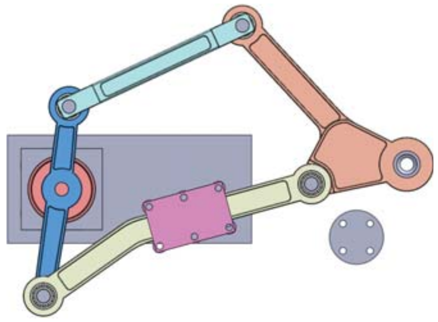
(b) Position 2.



(c) Position 3.



(d) Position 4.



(e) Position 5 (Cycle Out).

Figure 4.7: The square pattern screw insertion linkage in five task positions.

4.8 Rice Transplanter Linkage

The presented algorithm was used to design a linkage for rice transplantation. Rice transplantation is one of two major ways to establish the crop, the other being direct seeding. Rice transplantation is able to produce greater yields and serve as a form of weed control. However, it is much more labor intensive than direct seeding. It is widely practiced in Asia where often seedlings are directly hand transplanted. Hand transplanting rice onto a single hectare of land requires 30-40 person days [1]. Attempts to reduce this labor requirement through mechanization first began in Japan in the 1960s. Since then several rice transplanters have been developed, most of which require an internal combustion engine and are used mainly in industrialized Asian countries. These machines are far out of the price range for almost all small-hold Asian farmers. Therefore, they still practice labor intensive hand transplantation. An example of a mechanized rice transplanter that uses a four-bar linkage is the VST Shakti Yanji Rice Transplanter (Figure 4.8). This machine uses a four-bar in order to move rice seedlings off a holding tray and into the soil.

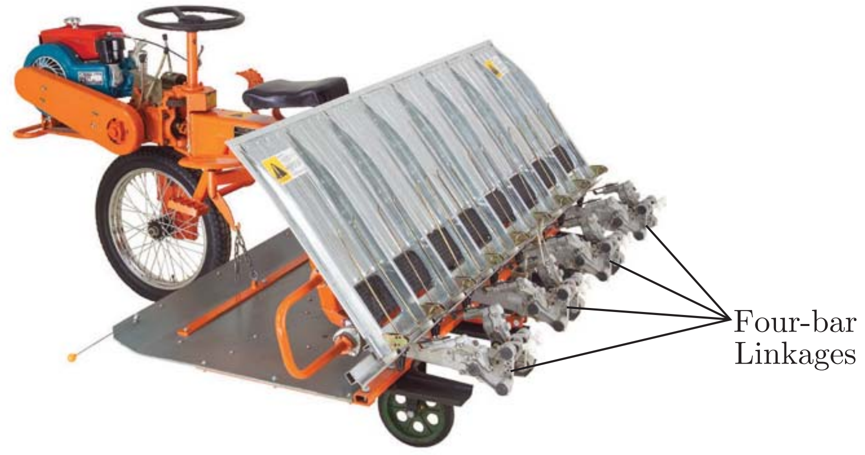


Figure 4.8: The VST Shakti Yanji Rice Transplanter.

The algorithm described in this chapter was employed to create a six-bar linkage to move seedlings into the soil. The task positions were specified so that the gripper is moved over a holding tray, where it will grab the seedlings, then it will move around the holding tray and poke into the earth, where it will plant the seedlings. The seedlings shall be gripped from 30.48 cm high, and be planted 2.54 cm into the soil. As well, they need to be oriented perpendicular to the soil before entering it. The seedlings are to be aligned lengthwise with the x-axis of the task frames shown in Figure 4.9.

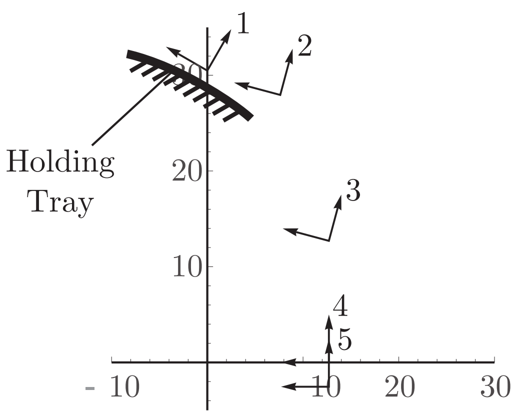


Figure 4.9: The proposed task positions for the rice seedling transplanter.

The tasks of Figure 4.9 are listed in Table 4.2 along with their corresponding tolerance zones. These tasks and tolerances were chosen after the algorithm was run several times with other task positions and tolerances through a trial and error procedure. With each successive execution of the algorithm, the results were visually inspected and a new set of tasks and tolerances was specified until the results contained a sufficient number of linkages that were useful, of practical size, and without pivots in troublesome locations i.e. underground. The ultimate tasks and tolerances used for this design exercise are those shown in Table 4.2. These tasks and tolerances were run for 1000 iterations, yielding 12 useful linkages. From inspection of these results, the linkage shown in Figure 4.10 was chosen for the design.

The chosen linkage design requires a rocker input in order to move through its motion. However, it is convenient for machines to transmit power from a rotary input of constant speed. Therefore, an actuation linkage was designed to serially attach to the motion generator of Figure 4.10. The objective of this function generator is to rock the input link of the motion generator through 220°. This is the range required by the input link of the motion generator. A Watt II mechanism of the same topology used in Section 4.7 was used here as well. This

Table 4.2: The task positions and tolerances are displayed in (a). The RPR serial chain parameters and tolerances are displayed in (b).

(a) Task Positions

Original Task Positions			Useful Task Positions		
ϕ	x (cm)	y (cm)	ϕ	x (cm)	y (cm)
60° ±1°	0 ±2.54	30.48 ±2.54	59.82°	-0.66	29.20
75° ±20°	7.62 ±5.08	27.94 +2.54/−7.62	61.29°	10.26	21.78
75° ±8°	12.70 ±2.54	12.70 +5.08/−10.16	75.13°	11.65	15.19
90° ±2°	12.70 ±0.508	0 ±1.27	89.25°	12.90	0.92
90° ±2°	12.70 ±0.508	-2.54 ±1.27	88.90°	12.81	-3.24

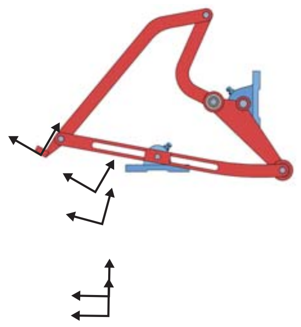
(b) RPR Parameters

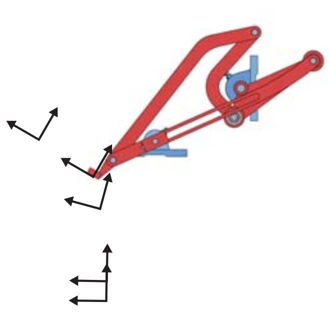
	Original Parameters	Useful Parameters
O_x	0 +63.50 cm	22.77 cm
O_y	0 +38.10 cm	28.73 cm
p	0 +25.40 cm	5.17 cm
η	0 ±180°	23.00°

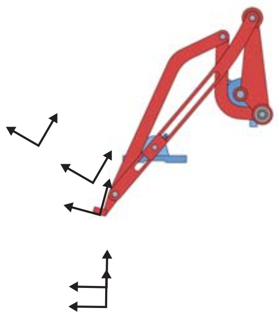
Watt II mechanism appears in yellow in Figure 4.11(a).

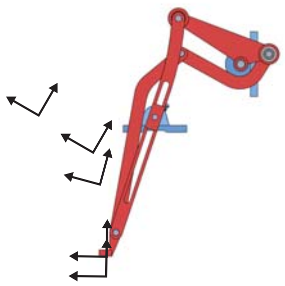
The six-bar rice transplanting linkage was compared to a four-bar rice transplanting linkage. The four-bar (Figure 4.11(b)) used for this comparison was designed previously. As well, it is similar to the design used by VST Shakti and other manufacturers. The six-bar and four-bar were compared in terms of the gripper's height throughout their motion. The motion of each linkage was simulated over a 10 second planting stroke. From Figure 4.12 it is shown that the gripper of the six-bar spends less time in the dirt, providing for a quicker planting action. This may be advantageous because if the gripper dwells too long in the dirt, then forces may develop in the x-direction of the F as the entire rice transplanting assembly will continue to move across the field.

Beside gripper height, the six-bar linkage also offers the gripper the ability to reach out and over the holding tray to grab seedlings. On the other hand, the four-bar mechanism requires seedlings to be angled out over the edge of the holding tray where they can be grabbed. This feature is facilitated by the fact that the six-bar's gripper traces an open curve as shown in


(a) Position 1.


(b) Position 2.


(c) Position 3.


(d) Position 4.

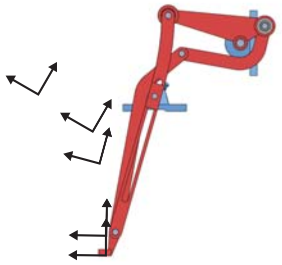
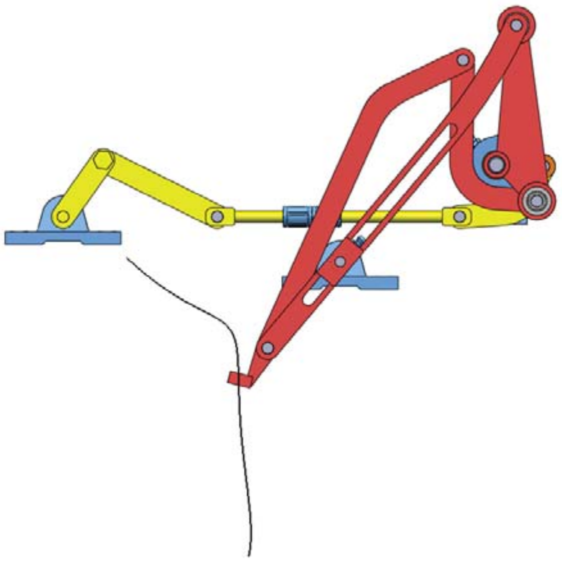

(e) Position 5.

Figure 4.10: The rice seedling transplanting linkage in five task positions.

Figure 4.11(a). The open curve has two points of zero velocity where the gripper reverses direction. These zero velocity points are enabled by the use of a rocker as the input link on the six-bar. One of these zero velocity points occur exactly when the gripper needs to grab seedlings, which may be useful. On the other hand, the gripper curve traced by the four-bar design is a closed curve (Figure 4.11(b)). It approximates a cusp on the bottom of its motion but has no zero velocity point near the seedling pick-up point. Please note that the six-bar coupler curve's classification as open or closed is a bit obfuscated. It can be regarded as an open curve of length l with no cusps, or a closed curve of length $2l$ with 2 cusps that traces over itself. Either way, the zero velocity points remain in the same locations.

As well, the six-bar and four-bar designs can be contrasted in terms of the size of their links, and the locations and number of pivots. The four-bar has a clear advantage with regards to the size and number of parts as it is just less complex. However, the six-bar's fixed and


(a)

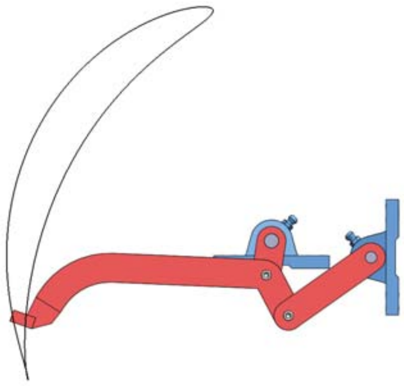

(b)

Figure 4.11: The trace point on the gripper for the (a) six-bar and (b) four-bar designs.

moving pivots are located much higher off the ground than the four-bar. This may be useful as the mechanism's bearings will be further away from mud and debris.

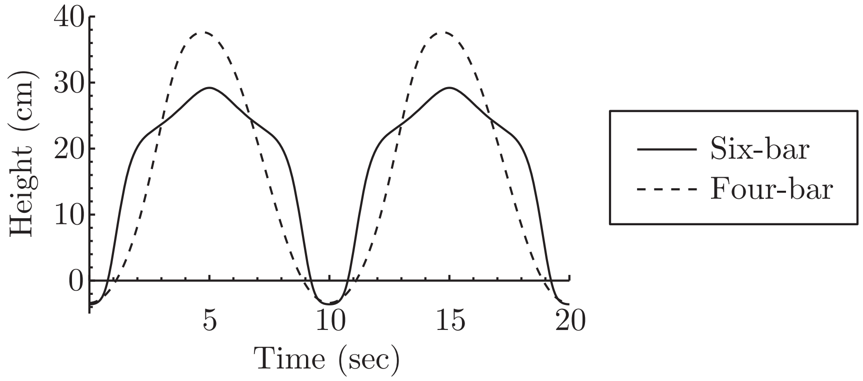


Figure 4.12: A comparison of the height of the gripper throughout the planting motion for six-bar and four-bar designs.

4.9 Summary

This chapter described a 5 task position synthesis method for Watt I slider-crank six-bar mechanisms from constrained RPR chains. The method begins by specifying a three degree of freedom serial RPR chain which is constrained to a one degree of freedom mechanism through the addition of two RR constraints. The synthesis process is broken into two main stages for synthesis of each of the RR constraints. Linkages with branch and circuit defects may result at either stage. An analysis procedure is installed such that mechanisms that possess these defects are removed. The result is only linkages that smoothly travel through all specified task positions. In order to expand on useful results, this synthesis method was coupled with an algorithm that iterates random variations of input parameters, that is the task and RPR parameters. However, randomization is constrained to operate within specified tolerance zones.

The application of the mechanism design process was demonstrated through two example problems. The first example was for the novel design of a screw insertion linkage. The second example demonstrated an attempt to improve on four-bar rice seedling transplanting linkages through the implementation of a more complex linkage. Both examples presented linkages capable of geometrically accomplishing the desired tasks.

Chapter 5

Conclusion

In this thesis, a design strategy was presented that uses Burmester theory to design RR cranks for task specifications that are randomized within tolerance zones. The design strategy has been used to define planar four-bar and six-bar linkages.

For each type of linkage, we explore the ability to find useful linkages through a variety of examples. The motivation behind synthesis for several sets of positions is that some percentage of the resulting linkages will be defect-free. For three of the four examples presented above, the ratio of useful designs to the number of iterations run was less than or equal to 1:40. The rotary weaver dwell linkage had a ratio of useful linkages to iterations that was greater than 1:1.

This paper places Burmester theory into a design algorithm that is capable of addressing linkage defects. The high frequency of these defects suggests that the probability of finding a useful design for a given motion requirement using Burmester theory alone is low. The contribution of this work shows an increase in the practicality of Burmester theory as evidenced by the examples above. In particular, the rotary weaver motion requirement was originally proposed by a company that had exhausted the linkage design strategies at their disposal. Our design strategy of combining traditional Burmester theory with toleranced task specification was successful.

5.1 Future Work

The outcome of this thesis is a set of algorithms for approaching planar linkage design that yield useful linkages. This strategy can be used for any linkage synthesis problem, including the design of useful spherical and spatial linkages. The author has shown positive results for spatial 5-SS mechanisms [42]. This approach can be applied to infinitesimally separated position synthesis as well.

Another route for future research is the addition of more structure to the randomized search process. For example, instead of specifying 10 tolerance zones for the x, y displacement parameters of 5 task positions, these parameters can be constrained to lie on specific locations of a curve. Tolerance zones can then be set on the curve parameters instead. Curves can be constructed through Bezier interpolation or other means.

Bibliography

[1] Manual rice transplanting. Fact sheet, International Rice Research Institute, September 2003.

[2] S. K. Acharyya and M. Mandal. Performance of EAs for four-bar linkage synthesis. *Mechanism and Machine Theory*, 44(9):1784–1794, September 2009.

[3] I. D. Akcali and G. Dittrich. Function generation by Galerkin's method. *Mechanism and Machine Theory*, 24(1):39–43, 1989.

[4] R. Akhras and J. Angeles. Unconstrained nonlinear least-square optimization of planar linkages for rigid-body guidance. *Mechanism and Machine Theory*, 25(1):97–118, 1990.

[5] R. Alizade and O. Kilit. Analytical synthesis of function generating spherical four-bar mechanism for the five precision points. *Mechanism and Machine Theory*, 40(7):863–878, July 2005.

[6] O. Altuzarra, A. Hernandez, O. Salgado, and J. Angeles. Multiobjective optimum design of a symmetric parallel Schönflies-motion generator. *Journal of Mechanical Design*, 131(3):031002, March 2009.

[7] C. Bagci and D. Burke. Optimum synthesis of coupler curve and uniform rotary motion driven multiloop mechanisms generating complex output motions. *Journal of Mechanical Design*, 115(4):967–977, December 1993.

[8] A. Bajpai and S. Kramer. Detection and elimination of mechanism defects in the selective precision synthesis of planar mechanisms. *Mechanism and Machine Theory*, 20(6):521–534, 1985.

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

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

[11] L. Burmester. *Lehrbuch der Kinematik.* Verlag Von Arthur Felix, Leipzig, Germany, 1886.

[12] J. J. Cervantes-Sánchez, L. Gracia, E. Alba-Ruiz, and J. M. Rico-Martínez. Synthesis of a special RPSPR spatial linkage function generator for six precision points. *Mechanism and Machine Theory*, 46(2):83–96, February 2011.

[13] J. J. Cervantes-Sánchez, H. I. Medellín-Castillo, J. M. Rico-Martínez, and E. J. González-Galván. Some improvements on the exact kinematic synthesis of spherical 4R function generators. *Mechanism and Machine Theory*, 44(1):103–121, January 2009.

[14] T. R. Chase and J. A. Mirth. Circuits and branches of single-degree-of-freedom planar linkages. *Journal of Mechanical Design*, 115(2):223–230, June 1993.

[15] P. Chen and B. Roth. Design equations for the finitely and infinitesimally separated position synthesis of binary links and combined link chains. *Journal of Engineering for Industry*, 91(1):209–219, February 1969.

[16] P. V. Chowdary, G. S. Kumar, and P. Ramu. A reliability based robust multi-objective optimal synthesis of linkage mechanisms considering tolerances. In *15th National Conference on Machines and Mechanisms*, Chennai, India, November 2011.

[17] F. Freudenstein. Approximate synthesis of four-bar linkages. *Trans. ASME*, 77:853–861, 1955.

[18] G. Gatti and D. Mundo. Optimal synthesis of six-bar cammed-linkages for exact rigid-body guidance. *Mechanism and Machine Theory*, 42(9):1069–1081, September 2007.

[19] K. C. Gupta. A general theory for synthesizing crank-type four-bar function generators with transmission angle control. *Journal of Applied Mechanics*, 45(2):415–421, June 1978.

[20] K. C. Gupta and A. S. Beloiu. Branch and circuit defect elimination in spherical four-bar linkages. *Mechanism and Machine Theory*, 33(5):491–504, July 1998.

[21] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. McGraw-Hill Co., New York, NY, 1964.

[22] J. E. Holte, T. R. Chase, and A. G. Erdman. Mixed exact-approximate position synthesis of planar mechanisms. *Journal of Mechanical Design*, 122(3):278–286, 2000.

[23] J. A. Hrones and G. I. Nelson. *Analysis of the Four-bar Linkage*. Massachusetts Institute of Technology and Wiley, New York, NY, 1951.

[24] E. C. Kinzel, J. P. Schmiedeler, and G. R. Pennock. Function generation with finitely separated precision points using geometric constraint programming. *Journal of Mechanical Design*, 129(11):1185–1190, November 2007.

[25] T. Koetsier. From kinematically generated curves to instantaneous invariants: Episodes in the history of instantaneous planar kinematics. *Mechanism and Machine Theory*, 21(6):489–498, 1986.

[26] X. Kong and C. M. Gosselin. *Type Synthesis of Parallel Mechanisms*. Springer, 2007.

[27] S. Kota and S. Chiou. Use of orthogonal arrays in mechanism synthesis. *Mechanism and Machine Theory*, 28(6):777–794, November 1993.

[28] P. Larochelle. Circuit and branch rectification of the spatial 4C mechanism. In *Proceedings of DETC*, Baltimore, MD, September 2000. ASME.

[29] M. D. Lio. Robust design of linkages–synthesis by solving non-linear optimization problems. *Mechanism and Machine Theory*, 32(8):921–932, November 1997.

[30] Z. Liu and J. Angeles. Least-square optimization of planar and spherical four-bar function generator under mobility constraints. *Journal of Mechanical Design*, 114(4):569–573, December 1992.

[31] Z. Liu and J. Angeles. Optimization of planar, spherical and spatial function generators using input-output curve planning. *Journal of Mechanical Design*, 116(3):915–919, September 1994.

[32] A. K. Mallik, A. Ghosh, and G. Dittrich. *Kinematic Analysis and Synthesis of Mechanisms*. CRC Press, 1994.

[33] F. T. S. Marín and A. P. González. Global optimization in path synthesis based on design space reduction. *Mechanism and Machine Theory*, 38(6):579–594, June 2003.

[34] J. M. McCarthy and G. S. Soh. *Geometric Design of Linkages*. Springer, Second edition, 2010.

[35] J. A. Mirth. Four-bar linkage synthesis for two precision positions combined with N quasi-positions. In *Proceedings from the 1995 ASME Design Engineering Technical Conferences*, Boston, MA, September 1995. ASME.

[36] J. A. Mirth and T. R. Chase. Circuit analysis of Watt chain six-bar mechanisms. *J. Mechanical Design*, 115(2):214–222, June 1993.

[37] J. A. Mirth and T. R. Chase. Circuit rectification for four precision position synthesis of Stephenson six-bar linkages. *Journal of Mechanical Design*, 117(4):644–646, December 1995.

[38] J. R. Mlinar and A. G. Erdman. An introduction to Burmester field theory. *Journal of Mechanical Design*, 122(1):25–30, 2000.

[39] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[40] D. H. Myszka and A. P. Murray. Pole arrangements that introduce prismatic joints into the design space of four- and five-position rigid-body synthesis. *Mechanism and Machine Theory*, 45(9):1314–1325, September 2010.

[41] R. L. Norton. *Design of Machinery*. McGraw Hill, Fifth edition, 2012.

[42] M. M. Plecnik and J. M. McCarthy. Design of a 5-SS spatial steering linkage. In *Proceedings of the ASME 2012 IDETC/CIE*, Chicago, IL, August 2012. ASME.

[43] R. Sancibrian, F. Viadero, P. Garcia, and A. Fernández. Gradient-based optimization of path synthesis problems in planar mechanisms. *Mechanism and Machine Theory*, 39(8):839–856, August 2004.

[44] J. Schröder. *Catalog of Reuleaux Models.* Darmstadt: Polytechnisches Arbeits-Institut, 1899.

[45] P. S. Shiakolas, D. Koladiya, and J. Kebrle. On the optimum synthesis of six-bar linkages using differential evolution and the geometric centroid of precision positions technique. *Mechanism and Machine Theory*, 40(3):319–335, March 2005.

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

[47] H. E. Stumph and A. P. Murray. Defect-free slider-crank function generation for 4.5 precision points, September 2000.

[48] H. J. Su, C. W. Wampler, and J. M. McCarthy. Geometric design of cylindric PRS serial chains. *Journal of Mechanical Design*, 126(2):269–277, March 2004.

[49] T. Subbian and J. D. R. Flugrad. Six and seven position triad synthesis using continuation methods. *Journal of Mechanical Design*, 116(2):660–665, June 1994.

[50] K. Ting, C. Xue, J. Wang, and K. R. Currie. Stretch rotation and complete mobility identification of Watt six-bar chains. *Mechanism and Machine Theory*, 44(10):1877–1886, October 2009.

[51] L. W. Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function.* CRC Press, 2001.

[52] J. J. Uicker Jr., G. R. Pennock, and J. Shigley. *Theory of Machines and Mechanisms.* Oxford, 2011.

[53] A. Vasiliu and B. Yannou. Dimensional synthesis of planar mechanisms using neural networks: application to path generator linkages. *Mechanism and Machine Theory*, 36(2):299–310, February 2001.

[54] K. J. Waldron. Graphical solution of the branch and order problems of linkage synthesis for multiply separated positions. *Journal of Engineering for Industry*, 99(3):591–597, August 1977.

[55] K. Watanabe and H. Katoh. Identification of motion domains of planar six-link mechanisms of the Stephenson-type. *Mechanism and Machine Theory*, 39(10):1081–1099, October 2004.

[56] X. Zhang and C. A. Nelson. Multiple-criteria kinematic optimization for the design of spherical serial mechanisms using genetic algorithms. *Journal of Mechanical Design*, 133(1):011005, January 2011.

Appendices

A Mathematica code for slider-crank four-bar function generators

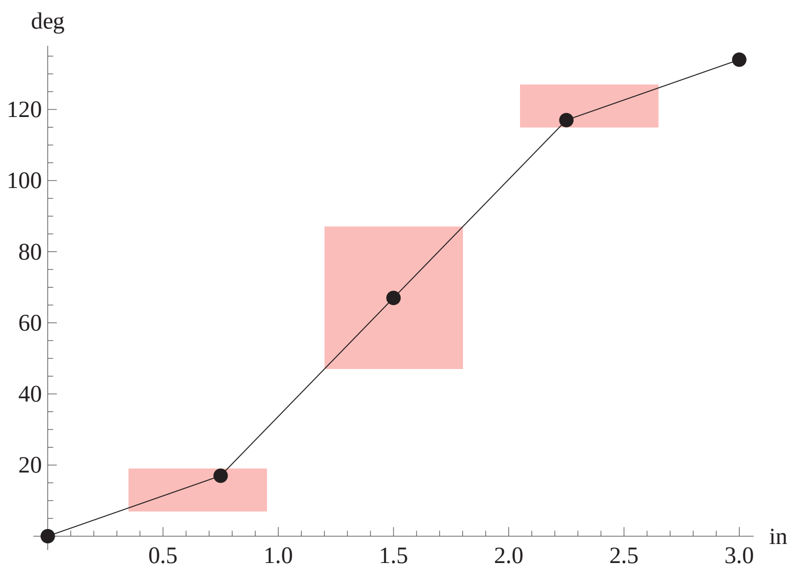
Functions

```
VectLength[B_, A_: {0, 0}] := Sqrt[(B[[1]] - A[[1]]) ^ 2 + (B[[2]] - A[[2]]) ^ 2]
VectAngle[BA_, DC_: {1, 0}] := ArcTan[BA[[1]], BA[[2]]] - ArcTan[DC[[1]], DC[[2]]]
Randomize[Points_, Tols_, Tolψ_] := Module[{sv, ψv, RandomizedPoints},
  sv = RandomReal[Tols, Length[Points]];
  ψv = RandomReal[Tolψ, Length[Points]];
  RandomizedPoints = Table[
    {Points[[i, 1]] + sv[[i]], (Points[[i, 2]] + ψv[[i]]) * Degree}, {i, Length[Points]}];
  NormalizedPoints = Table[Points[[i]] - Points[[1]], {i, Length[Points]}];
  RandomizedPoints]
RandomizeByPoint[Points_, TolsByPoint_, TolψByPoint_] :=
 Module[{sv, ψv, RandomizedPoints},
  sv = Table[RandomReal[TolsByPoint[[i]]], {i, Length[Points]}];
  ψv = Table[RandomReal[TolψByPoint[[i]]], {i, Length[Points]}];
  RandomizedPoints = Table[
    {Points[[i, 1]] + sv[[i]], (Points[[i, 2]] + ψv[[i]]) * Degree}, {i, Length[Points]}];
  NormalizedPoints = Table[Points[[i]] - Points[[1]], {i, Length[Points]}];
  RandomizedPoints]
```

```
Synthesis[Points_] :=
 Module[{T, u, v, x, y, W1, W3D, W, S, R, DotProds, ConstraintEqns, sol, n, ptO, ptA, ptB},
  (*Create transformations from moving to fixed frame for each task position*)
  T = Table[{{Cos[Points[[i, 2]]], -Sin[Points[[i, 2]]], u},
     {Sin[Points[[i, 2]]], Cos[Points[[i, 2]]], v}, {0, 0, 1}}, {i, Length[Points]}];
  (*Define variables for location of moving pivot in first task position*)
  W1 = {x + u, y + v, 1};
  (*Define 5 locations of the moving pivot*)
  W3D = Table[T[[i]].Inverse[T[[1]]].W1, {i, Length[Points]}];
  (*Transform locations onto z=0 plane*)
  W = W3D.{{1, 0, 0}, {0, 1, 0}, {0, 0, 0}};
  (*Define vectors for the slider locations*)
  S = Table[{Points[[i, 1]], 0, 0}, {i, Length[Points]}];
  (*Create set of constraint equations*)
  DotProds = Table[(W[[i]] - S[[i]]).(W[[i]] - S[[i]]) - R^2, {i, Length[Points]}];
  ConstraintEqns = Table[DotProds[[i]] - DotProds[[1]], {i, 2, Length[Points]}];
  sol = NSolve[ConstraintEqns == 0, {x, y, u, v}];
  sol = {x, y, u, v} /. sol;
  (*Remove imaginary and very large solutions*)
  n = Length[sol];
  While[n > 0,
   sol = If[Im[sol[[n]]] == {0, 0, 0, 0} && Norm[sol[[n]]] < 10^5, sol, Drop[sol, {n}]];
   n = n - 1];
  (*Organize solutions as points.  Use the following syntax: for ptA[[a]][[b,c]],
  a=associated solution, b=associated position, c=x or y coord*)
  ptO = Table[Table[{sol[[i, 3]], sol[[i, 4]]}, {j, Length[Points]}], {i, Length[sol]}];
  ptA = Table[Table[{W[[j, 1]], W[[j, 2]]} /. {x → sol[[i, 1]], y → sol[[i, 2]],
       u → sol[[i, 3]], v → sol[[i, 4]]}, {j, Length[Points]}], {i, Length[sol]}];
  ptB = Table[Table[{Points[[j, 1]], 0}, {j, Length[Points]}], {i, Length[sol]}];
  {ptO, ptA, ptB}]
Analysis[ptO_, ptA_, ptB_, RandomizedPoints_] :=
 Module[{a, b, Acoef, Bcoef, Ccoef, ψ, Test, ψSign, ψp},
  (*Acoef=(2*u-2*s)*Sqrt[(x-u)^2+(y-v)^2];
  Bcoef=2*v*Sqrt[(x-u)^2+(y-v)^2];
  Ccoef=2*(u^2+v^2-x*u-y*v+x*s-u*s);*)
  a = Table[Norm[ptA[[i]] - ptO[[i]]], {i, Length[ptA]}];
  b = Table[Norm[ptA[[i]] - ptB[[i]]], {i, Length[ptA]}];
  Acoef = Table[(2 * ptO[[i, 1]] - 2 * ptB[[i, 1]]) * a[[i]], {i, Length[ptA]}];
  Bcoef = Table[2 * ptO[[i, 2]] * a[[i]], {i, Length[ptA]}];
  Ccoef = Table[(ptO[[i, 1]] - ptB[[i, 1]])^2 +
     ptO[[i, 2]]^2 + a[[i]]^2 - b[[i]]^2, {i, Length[ptA]}];
  ψ = Table[{ArcTan[Acoef[[i]], Bcoef[[i]]] + ArcCos[
        -Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], ArcTan[Acoef[[i]], Bcoef[[i]]] -
       ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]]}, {i, 5}];
  Normalizedψ = Round[Table[ψ[[i]] - ψ[[1]], {i, 5}], .001];
  NormalizedRandomizedPoints =
   Round[Table[RandomizedPoints[[i]] - RandomizedPoints[[1]], {i, 5}], .001];
  Which[Normalizedψ[[All, 1]] == NormalizedRandomizedPoints[[All, 2]],
   Test = 1; ψSign = "pos"; ψp = ψ[[All, 1]],
   Normalizedψ[[All, 2]] == NormalizedRandomizedPoints[[All, 2]],
   Test = 1; ψSign = "neg"; ψp = ψ[[All, 2]],
    Normalizedψ[[All, 1]] ≠ NormalizedRandomizedPoints[[All, 2]] &&
     Normalizedψ[[All, 2]] ≠ NormalizedRandomizedPoints[[All, 2]], Test = 0; ψSign = 0];
   (*Test=If[Normalizedψ[[All,1]]==NormalizedRandomizedPoints[[All,2]]||
       Normalizedψ[[All,2]]==NormalizedRandomizedPoints[[All,2]],1,0];*)
   {Test, ψSign, ψp}]
GraphMechanismAndFunction[GorB_, first_, last_] := Module[{},
   q = first;
   While[q ≤ last,
    Which[GorB == "G", Solution = GoodSolutions[[q]],
     GorB == "B", Solution = BadSolutions[[q]]];
    Which[GorB == "G", SolutionFunction =
      GoodSolutionsFunctions[[q]].{{1, 0}, {0, 180 / Pi}}, GorB == "B",
     SolutionFunction = BadSolutionsFunctions[[q]].{{1, 0}, {0, 180 / Pi}}];
    Mechanism = Graphics[{Polygon[{Solution[[1, 1]], Solution[[1, 1]] + {-2, -2},
          Solution[[1, 1]] + {2, -2}}], Disk[Solution[[2, 1]]], Disk[Solution[[2, 2]]],
       Disk[Solution[[2, 3]]], Disk[Solution[[2, 4]]], Disk[Solution[[2, 5]]],
       Rectangle[Solution[[3, 1]] + {-2, -1}, Solution[[3, 1]] + {2, 1}],
       Rectangle[Solution[[3, 2]] + {-2, -1}, Solution[[3, 2]] + {2, 1}],
       Rectangle[Solution[[3, 3]] + {-2, -1}, Solution[[3, 3]] + {2, 1}],
       Rectangle[Solution[[3, 4]] + {-2, -1}, Solution[[3, 4]] + {2, 1}],
       Rectangle[Solution[[3, 5]] + {-2, -1}, Solution[[3, 5]] + {2, 1}],
       Line[{Solution[[All, 1]], Solution[[All, 2]], Solution[[All, 3]], Solution[[
          All, 4]], Solution[[All, 5]]}]}, Axes → True, AxesStyle → Directive[14]];
    FunctionPoints = Graphics[{Line[SolutionFunction], Dashed, Line[NormalizedPoints]},
      Axes → True, AxesLabel → {Δs, Δψ}, AxesStyle → Directive[14]
      (*,PlotLabel→"Dotted- Original Function,
Solid- Toleranced Function"*)];
    Print[Style["Solution ", Bold, 18], Style[q, Bold, 18]];
    Print[Mechanism]; Print[FunctionPoints]; q++]]
AnimateSliderCrankFunctionGenerator[Soln_] :=
 Module[{BxStart, BxEnd, end, StepSize, Bx, Ox, Oy, AO, BA, α, acoef,
    bcoef, ccoef1, ccoef2, BxRange1, BxRange2, Acoef, Bcoef, Ccoef, θAO, Ax,
    Ay, AxPos, AyPos, Xrange, Yrange, Xpadding, Ypadding, Xmin, Xmax, Ymin,
    Ymax, ScalingRange, GPSize, PSize, SSize, TPSize, MechLines, GroundPivotO,
    PivotA, SliderB, TaskPosMobileTrans, TaskPosMobile, TaskPosTrans,
    TaskPosColors, TaskPos, MechLinesPos, PivotAPos, SliderBPos, Animation},
   (*Constants*)
   Ox = Soln[[1, 1, 1]]; Oy = Soln[[1, 1, 2]];
   AO = VectLength[Soln[[2, 1]], Soln[[1, 1]]];
   BA = VectLength[Soln[[3, 1]], Soln[[2, 1]]];
   α = Soln[[6, 1]] - Soln[[5, 1]];
   (*Input variable and range*)
   acoef = 1;
   bcoef = -2 * VectLength[{Ox, Oy}] * Cos[VectAngle[{Ox, Oy}]];
   ccoef1 = VectLength[{Ox, Oy}] ^ 2 - (AO + BA) ^ 2;
   ccoef2 = VectLength[{Ox, Oy}] ^ 2 - (AO - BA) ^ 2;
   BxRange1 = {(-bcoef + Sqrt[bcoef ^ 2 - 4 * acoef * ccoef1]) / (2 * acoef),
     (-bcoef + Sqrt[bcoef ^ 2 - 4 * acoef * ccoef2]) / (2 * acoef)};
   BxRange2 = {(-bcoef - Sqrt[bcoef ^ 2 - 4 * acoef * ccoef1]) / (2 * acoef),
     (-bcoef - Sqrt[bcoef ^ 2 - 4 * acoef * ccoef2]) / (2 * acoef)};
   Which[Im[BxRange1] ≠ {0, 0}, BxStart = Min[BxRange1[[1]], BxRange2[[1]]];
 BxEnd = Max[BxRange1[[1]], BxRange2[[1]]],
 Table[Min[BxRange1] ≤ Soln[[4, i]] ≤ Max[BxRange1], {i, 5}] == ConstantArray[True, 5],
 BxStart = Min[BxRange1]; BxEnd = Max[BxRange1],
 Table[Min[BxRange2] ≤ Soln[[4, i]] ≤ Max[BxRange2], {i, 5}] == ConstantArray[True, 5],
 BxStart = Min[BxRange2]; BxEnd = Max[BxRange2], True,
 Print["All positions (s,ψ) don't seem to be within the same input range of s."];
 Goto[end]];
BxStart = Min[Soln[[4]]];
BxEnd = Max[Soln[[4]]];
StepSize = 100;
Bx = Array[BxStart + (BxEnd - BxStart) / (StepSize - 1) * (# - 1) &, StepSize];
Bx[[1]] = Bx[[1]] + .0001; Bx[[Length[Bx]]] = Bx[[Length[Bx]]] - .0001;
Bx = Join[Bx, Reverse[Bx]];
(*Variables*)
Acoef = Table[-2 * (Bx[[i]] - Ox) * AO, {i, Length[Bx]}];
Bcoef = Table[2 * Oy * AO, {i, Length[Bx]}];
Ccoef = Table[(Bx[[i]] - Ox)^2 + Oy^2 + AO^2 - BA^2, {i, Length[Bx]}];
Which[Soln[[7]] == "pos", θAO = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] +
     ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[Bx]}],
 Soln[[7]] == "neg", θAO = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] -
     ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[Bx]}]];
Ax = Table[Ox + AO * Cos[θAO[[i]]], {i, Length[Bx]}];
Ay = Table[Oy + AO * Sin[θAO[[i]]], {i, Length[Bx]}];
AxPos = Table[Ox + AO * Cos[Soln[[6, i]]], {i, 5}];
AyPos = Table[Oy + AO * Sin[Soln[[6, i]]], {i, 5}];
(*Bounds and scaling*)
Xrange = Max[Ox, Ax, Bx] - Min[Ox, Ax, Bx];
Yrange = Max[Oy, Ay, 0] - Min[Oy, Ay, 0];
Which[Xrange ≥ Yrange, Xpadding = .05;
 Ypadding = (Xrange + 2 * Xpadding * Xrange - Yrange) / (2 * Yrange),
 Yrange > Xrange, Ypadding = .05; Xpadding =
  (Yrange + 2 * Ypadding * Yrange - Xrange) / (2 * Xrange)];
Xmin = Min[Ox, Ax, Bx] - Xpadding * Xrange;
Xmax = Max[Ox, Ax, Bx] + Xpadding * Xrange;
Ymin = Min[Oy, Ay, 0] - Ypadding * Yrange;
Ymax = Max[Oy, Ay, 0] + Ypadding * Yrange;
Which[Xrange ≥ Yrange, ScalingRange = Xrange, Yrange > Xrange, ScalingRange = Yrange];
GPSize = .04 * ScalingRange;
PSize = .01 * ScalingRange;
SSize = .03 * ScalingRange;
TPSize = .1 * ScalingRange;
(*Graphics*)
MechLines = Table[Line[{{Ox, Oy}, {Ax[[i]], Ay[[i]]}, {Bx[[i]], 0}}], {i, Length[Bx]}];
GroundPivotO = Polygon[
  {{Ox, Oy}, {Ox, Oy} + {-.6 * GPSize, -GPSize}, {Ox, Oy} + {.6 * GPSize, -GPSize}}];
PivotA = Table[Disk[{Ax[[i]], Ay[[i]]}, PSize], {i, Length[Bx]}];
SliderB = Table[Rectangle[{Bx[[i]], 0} + {-SSize, -.5 * SSize},
   {Bx[[i]], 0} + {SSize, .5 * SSize}], {i, Length[Bx]}];
TaskPosMobileTrans = Table[{{Cos[θAO[[i]] - α], -Sin[θAO[[i]] - α], Ox},
   {Sin[θAO[[i]] - α], Cos[θAO[[i]] - α], Oy}}, {i, Length[Bx]}];
TaskPosMobile = Table[Line[{TaskPosMobileTrans[[i]].{TPSize, 0, 1},
      TaskPosMobileTrans[[i]].{0, 0, 1},
      TaskPosMobileTrans[[i]].{0, TPSize, 1}}], {i, Length[Bx]}];
  TaskPosTrans = Table[{{Cos[Soln[[5, i]]], -Sin[Soln[[5, i]]], Ox},
     {Sin[Soln[[5, i]]], Cos[Soln[[5, i]]], Oy}}, {i, 5}];
  TaskPosColors = {Lighter[Purple, .5], Lighter[Blue, .5],
    Lighter[Green, .5], Lighter[Orange, .5], Lighter[Red, .5]};
  TaskPos = Table[{TaskPosColors[[i]], Line[{TaskPosTrans[[i]].{TPSize, 0, 1},
       TaskPosTrans[[i]].{0, 0, 1}, TaskPosTrans[[i]].{0, TPSize, 1}}]}, {i, 5}];
  MechLinesPos = Table[Line[{{Ox, Oy}, {AxPos[[i]], AyPos[[i]]}, {Soln[[4, i]], 0}}],
    {i, 5}];
  PivotAPos = Table[Disk[{AxPos[[i]], AyPos[[i]]}, PSize], {i, 5}];
  SliderBPos = Table[Rectangle[{Soln[[4, i]], 0} + {-SSize, -.5 * SSize},
     {Soln[[4, i]], 0} + {SSize, .5 * SSize}], {i, 5}];
  Animation = ListAnimate[Table[Graphics[{Gray, MechLinesPos, PivotAPos,
       SliderBPos, Black, MechLines[[i]], GroundPivotO, PivotA[[i]], SliderB[[i]],
       Thick, LightRed, TaskPos, Red, TaskPosMobile[[i]]}, Axes → True,
      PlotRange → {{Xmin, Xmax}, {Ymin, Ymax}}, ImageSize → 300], {i, Length[Bx]}]];
  Label[end];
  Animation]
BatchAnimateSliderCrankFunctionGenerator[Soln_, first_, last_] :=
 Module[{n, SliderCrankFunctionGeneratorAnimations},
  n = first;
  While[n ≤ last, Print[Style[{"Solution No", n}, Bold, 18]];
   SliderCrankFunctionGeneratorAnimations =
    AnimateSliderCrankFunctionGenerator[Soln[[n]]];
   Print[SliderCrankFunctionGeneratorAnimations]; n++]]
```

Evaluation

```
(*Specify 5 input/output points as {s,ψ}*)
Points = {{0, 0}, {.75, 17}, {1.5, 67}, {2.25, 117}, {3, 134}};
(*Specify tolerances for s and ψ*)
Tols = {-1, 1};
Tolψ = {-2.5, 2.5};
TolsByPoint = {{0, 0}, {-.4, .2}, {-.3, .3}, {-.2, .4}, {0, 0}};
TolψByPoint = {{0, 0}, {-10, 2}, {-20, 20}, {-2, 10}, {0, 0}};
(*Specify number of loops to run*)
Loops = 500;
ToleranceBoxes = Table[
   Rectangle[{Points[[i, 1]] + TolsByPoint[[i, 1]], Points[[i, 2]] + TolψByPoint[[i, 1]]},
    {Points[[i, 1]] + TolsByPoint[[i, 2]], Points[[i, 2]] + TolψByPoint[[i, 2]]}], {i, 5}];
function = Graphics[
  {Pink, (*EdgeForm[Thin],*)Opacity[.5], ToleranceBoxes,
   Opacity[1], Black, PointSize[.02], Point[Points],
   Line[Points]},
  Axes → True, AspectRatio → .7, ImageSize → 400,
  (*AxesLabel→{Style[s,FontFamily→"CMUSerif"],Style[ψ,FontFamily→"CMUSerif"]},
  AxesStyle→Directive[12,FontFamily→"CMUSerif"]*)AxesLabel → {in, deg}, AxesStyle → 12
 ]
```

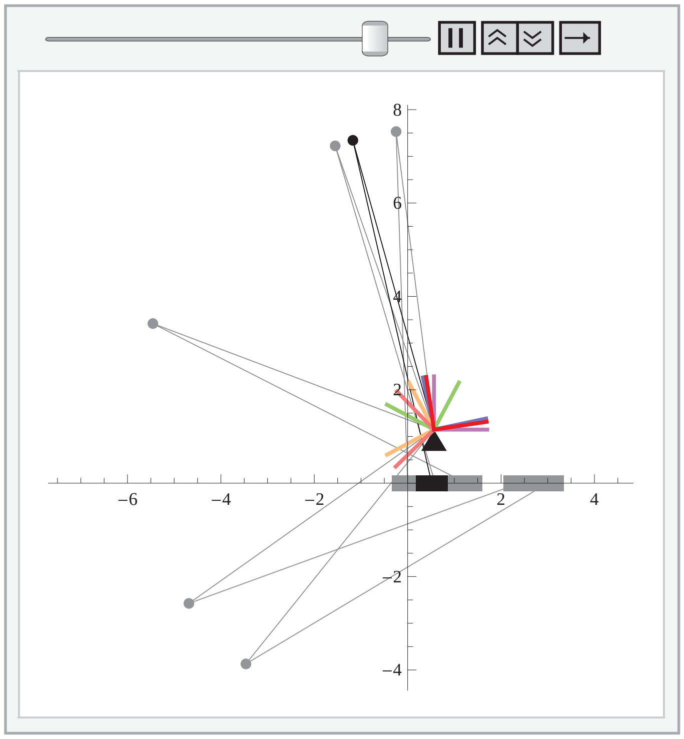


```
(*Press shift-enter to execute program*)
GoodSolutions = {};
BadSolutions = {};
GoodSolutionsFunctions = {};
BadSolutionsFunctions = {};
p = 0;
n = 1;
While[n ≤ Loops,
  (*RandomizedPoints=Randomize[Points,Tols,Tolψ];*)
  RandomizedPoints = RandomizeByPoint[Points, TolsByPoint, TolψByPoint];
  {ptO, ptA, ptB} = Synthesis[RandomizedPoints];
  m = 1;
  before = Length[GoodSolutions];
  While[m ≤ Length[ptA],
   {Test, ψSign, ψp} = Analysis[ptO[[m]], ptA[[m]], ptB[[m]], RandomizedPoints];
   If[Test == 1, GoodSolutions = Join[GoodSolutions, {{ptO[[m]], ptA[[m]], ptB[[m]],
         RandomizedPoints[[All, 1]], RandomizedPoints[[All, 2]], ψp, ψSign}}],
    BadSolutions = Join[BadSolutions, {{ptO[[m]], ptA[[m]], ptB[[m]],
         RandomizedPoints[[All, 1]], RandomizedPoints[[All, 2]], ψp, ψSign}}]];
   If[Test == 1, GoodSolutionsFunctions = Join[GoodSolutionsFunctions,
      {NormalizedRandomizedPoints}], BadSolutionsFunctions =
     Join[BadSolutionsFunctions, {NormalizedRandomizedPoints}]];
   (*Print[MatrixForm[{ptO[[m]],ptA[[m]],ptB[[m]]}]];
   Print[MatrixForm[Normalizedψ]];
   Print[MatrixForm[NormalizedRandomizedPoints]];*)
   m++];
  after = Length[GoodSolutions];
  If[after > before, p = p + 1];
  n++];
Print["Good Functions = ", p]
Print["Good Solutions = ", Length[GoodSolutions]]
Print["Bad Solutions = ", Length[BadSolutions]]
```

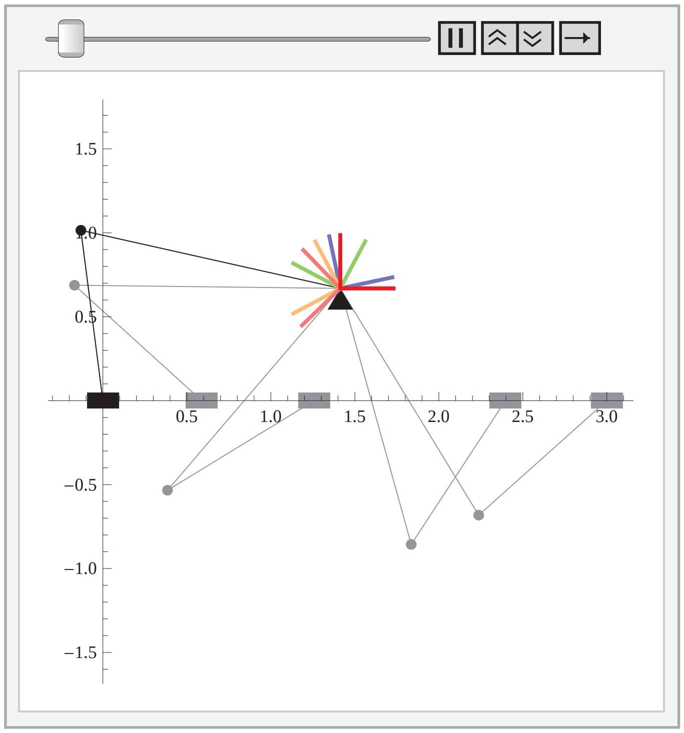
```
Good Functions = 357

Good Solutions = 598

Bad Solutions = 746
```

`BatchAnimateSliderCrankFunctionGenerator[GoodSolutions, 1, 3]`

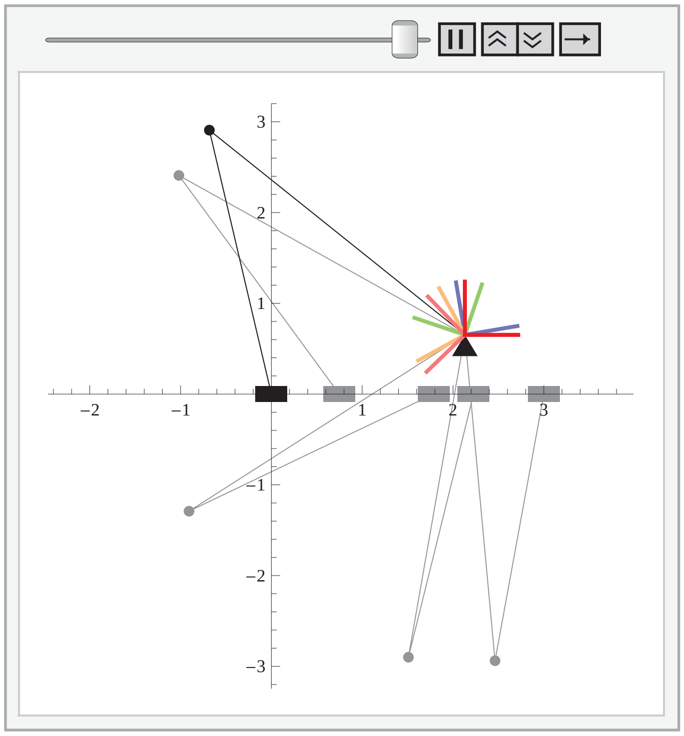
{Solution No, 1}



{Solution No, 2}



```
{Solution No, 3}
```



B Mathematica code for slider-crank six-bar motion generators

Functions

```
VectLength[B_, A_] := Sqrt[(B[[1]] - A[[1]])^2 + (B[[2]] - A[[2]])^2]
VectAngle[BA_, DC_: {1, 0}] := ArcTan[BA[[1]], BA[[2]]] - ArcTan[DC[[1]], DC[[2]]]
NormalizeAngles[θ_, θmin_, θmax_] := Module[{θp},
  θp = Table[θ[[i]], {i, Length[θ]}];
  θp = Table[Which[θmin ≤ θp[[i]] < θmax, θp[[i]], θp[[i]] > θmax,
     While[θp[[i]] > θmax, θp[[i]] = θp[[i]] - 2 * Pi]; θp[[i]], θp[[i]] ≤ θmin,
     While[θp[[i]] ≤ θmin, θp[[i]] = θp[[i]] + 2 * Pi]; θp[[i]]], {i, Length[θ]}];
  θp = Round[Table[θp[[i]], {i, Length[θ]}], .0001];
  θp]
Randomize[TaskP_, Tolϕ_, Tolx_, Toly_] := Module[{TaskPRand},
  TaskPRand = Table[{TaskP[[i, 1]] * Degree + RandomReal[Tolϕ * Degree], TaskP[[i, 2]] +
      RandomReal[Tolx], TaskP[[i, 3]] + RandomReal[Toly]}, {i, Length[TaskP]}];
  TaskPRand]
RandomizeAllParams[TaskP_, TolϕByPoint_, TolxByPoint_,
  TolyByPoint_, Ox_, Oy_, TA_, θδ1_, TolOx_, TolOy_, TolTA_, Tolθδ1_] :=
 Module[{TaskPRand, OxRand, OyRand, TARand, θδ1Rand},
  TaskPRand = Table[{(TaskP[[i, 1]] + RandomReal[TolϕByPoint[[i]]]) * Degree,
     TaskP[[i, 2]] + RandomReal[TolxByPoint[[i]]],
     TaskP[[i, 3]] + RandomReal[TolyByPoint[[i]]]}, {i, Length[TaskP]}];
  OxRand = Ox + RandomReal[TolOx];
  OyRand = Oy + RandomReal[TolOy];
  TARand = TA + RandomReal[TolTA];
  θδ1Rand = θδ1 + RandomReal[Tolθδ1];
  {TaskPRand, OxRand, OyRand, TARand, θδ1Rand}]
BezSmooth[TaskP_, BezParams_] := Module[{n, BezFunc, TaskPBez},
  n = Length[TaskP] - 1;
  BezFunc = Sum[n! / (i! * (n - i)!) * t^i * (1 - t)^(n - i) * TaskP[[i + 1]], {i, 0, n}];
  TaskPBez = Table[BezFunc, {t, BezParams}];
  {BezFunc, TaskPBez}]
BezierTaskGraph[TaskP_, BezParams_] :=
 Module[{TScale, Triad, TaskTriads, BezFunc, TaskPBez, TaskTriadsBez},
  TScale = 2;
  Triad = {{1, 0}, {0, 0}, {0, 1}} * TScale;
  TaskTriads = Table[RotationMatrix[TaskP[[i, 1]] * Degree].Triad[[j]] +
     TaskP[[i, 2 ;; 3]], {i, Length[TaskP]}, {j, Length[Triad]}];
  {BezFunc, TaskPBez} = BezSmooth[TaskP, BezParams];
  TaskTriadsBez = Table[RotationMatrix[TaskPBez[[i, 1]] * Degree].Triad[[j]] +
     TaskPBez[[i, 2 ;; 3]], {i, Length[TaskPBez]}, {j, Length[Triad]}];
```

```
Graphics[
  {Line[Triad],
   Thickness[.005], Line[TaskTriads],
   Red, Line[TaskTriadsBez],
   Dotted, Thickness[.003], Line[Table[BezFunc[[2 ;; 3]], {t, 0, 1, .01}]]},
  Axes → True, PlotRange → {{-10, 30}, {-5, 35}}]]
InverseKinematics[Ox_, Oy_, TA_, θδ1_, TaskP_] := Module[{AO, θ1, ptO, ptA, θAO},
  AO = Table[Sqrt[(TaskP[[i, 2]] - Ox - TA * Cos[TaskP[[i, 1]] - θδ1 * Degree])^2 +
      (TaskP[[i, 3]] - Oy - TA * Sin[TaskP[[i, 1]] - θδ1 * Degree])^2], {i, Length[TaskP]}];
  θ1 = Table[N[ArcTan[TaskP[[i, 2]] - Ox - TA * Cos[TaskP[[i, 1]] - θδ1 * Degree],
      TaskP[[i, 3]] - Oy - TA * Sin[TaskP[[i, 1]] - θδ1 * Degree]]], {i, Length[TaskP]}];
  ptO = Table[{Ox, Oy}, {i, Length[TaskP]}];
  ptA = Table[
    N[{Ox + AO[[i]] * Cos[θ1[[i]]], Oy + AO[[i]] * Sin[θ1[[i]]]}], {i, Length[TaskP]}];
  θAO = θ1;
  {ptO, ptA, θAO}]
FirstRRConstraint[ptA_, θAO_] :=
 Module[{T, Bx, By, Cx, Cy, W1, W3D, W, G, R, ConstraintEqns,
   ConstraintEqnsSimplified, sol1, n1, , n2, ptB, ptC, θCB, SolnsFound1},
  T = Table[{{Cos[θAO[[i]]], -Sin[θAO[[i]]], ptA[[i, 1]]},
     {Sin[θAO[[i]]], Cos[θAO[[i]]], ptA[[i, 2]]}, {0, 0, 1}}, {i, Length[ptA]}];
  W1 = {Cx, Cy, 1};
  W3D = Table[T[[i]].Inverse[T[[1]]].W1, {i, Length[ptA]}];
  W = W3D.{{1, 0, 0}, {0, 1, 0}, {0, 0, 0}};
  G = {Bx, By, 0};
  ConstraintEqns = Table[(W[[i]] - G).(W[[i]] - G) - R^2, {i, Length[ptA]}];
  ConstraintEqnsSimplified =
   Table[ConstraintEqns[[i]] - ConstraintEqns[[1]] == 0, {i, 2, Length[ptA]}];
  Quiet[Check[sol1 = NSolve[Chop[Expand[ConstraintEqnsSimplified]], {Bx, By, Cx, Cy}];,
    Print["NSolve experienced a failure during the 1st RR
       constraint and some solutions may have been lossed for
       the set of task positions: ", TaskPRand], NSolve::sfail]];
  sol1 = {Bx, By, Cx, Cy} /. sol1;
  SolnsFound1 = Length[sol1];
  n1 = Length[sol1];
  While[n1 > 0,
   sol1 = If[Im[sol1[[n1]]] == {0, 0, 0, 0} && Norm[sol1[[n1]], Infinity] < 10^10,
     sol1, Drop[sol1, {n1}]]; n1 = n1 - 1];
  ptB = Table[Table[{G[[1]], G[[2]]} /. {Bx → sol1[[i, 1]], By → sol1[[i, 2]]},
     {j, Length[ptA]}], {i, Length[sol1]}];
  ptC = Table[Table[{W[[j, 1]], W[[j, 2]]} /. {Bx → sol1[[i, 1]], By → sol1[[i, 2]],
      Cx → sol1[[i, 3]], Cy → sol1[[i, 4]]}, {j, Length[ptA]}], {i, Length[sol1]}];
  θCB = Table[Table[VectAngle[ptC[[i, j]] - ptB[[i, j]]], {j, Length[ptA]}],
    {i, Length[sol1]}];
  (*θ3=Table[Table[ArcTan[ptC[[i,j,1]]-ptB[[i,j,1]],ptC[[i,j,2]]-ptB[[i,j,2]]],
     {j,Length[ptA]}],{i,Length[sol1]}];*)
  (*This check shows whether the lengths BC and CA are exceptionally
   long (not infinity) and will lead to a badly conditioned matrix
   that Mathematica cannot invert.  These solutions are eliminated.*)
  n2 = Length[sol1];
  While[n2 > 0,
    Quiet[Check[Inverse[{{Cos[θCB[[n2, 1]]], -Sin[θCB[[n2, 1]]], ptB[[n2, 1, 1]]},
        {Sin[θCB[[n2, 1]]], Cos[θCB[[n2, 1]]], ptB[[n2, 1, 2]]}, {0, 0, 1}}],
      ptB = Drop[ptB, {n2}];
      ptC = Drop[ptC, {n2}];
      θCB = Drop[θCB, {n2}];,
      Inverse::luc]];
    n2 = n2 - 1];
   {ptB, ptC, θCB, SolnsFound1}]
FirstRRConstraintAnalysis[ptO_, ptA_, ptB_, ptC_, θAO_, θCB_] :=
 Module[{CA, CB, θδ2, acoef, bcoef, ccoef, AOpos, AOneg, AO,
    AOmod, Acoef, Bcoef, Ccoef, θAOpos, θAOneg, θAOmod, Test, AOSign},
   CA = VectLength[ptC[[1]], ptA[[1]]];
   CB = VectLength[ptC[[1]], ptB[[1]]];
   θδ2 = VectAngle[ptC[[1]] - ptA[[1]], ptA[[1]] - ptO[[1]]];
   AO = Table[VectLength[ptA[[i]], ptO[[i]]], {i, Length[ptO]}];
   acoef = Table[1, {i, Length[ptO]}];
   bcoef = Table[2 * (ptO[[i, 1]] - ptB[[i, 1]]) * Cos[θAO[[i]]] +
      2 * (ptO[[i, 2]] - ptB[[i, 2]]) * Sin[θAO[[i]]] + 2 * CA * Cos[θδ2], {i, Length[ptO]}];
   ccoef = Table[(ptO[[i, 1]] - ptB[[i, 1]]) ^ 2 + (ptO[[i, 2]] - ptB[[i, 2]]) ^ 2 +
      CA ^ 2 - CB ^ 2 + 2 * (ptO[[i, 1]] - ptB[[i, 1]]) * CA * Cos[θAO[[i]] + θδ2] +
      2 * (ptO[[i, 2]] - ptB[[i, 2]]) * CA * Sin[θAO[[i]] + θδ2], {i, Length[ptO]}];
   AOpos = Table[(-bcoef[[i]] + Sqrt[bcoef[[i]] ^ 2 - 4 * acoef[[i]] * ccoef[[i]]]) /
      (2 * acoef[[i]]), {i, Length[ptO]}];
   AOneg = Table[(-bcoef[[i]] - Sqrt[bcoef[[i]] ^ 2 - 4 * acoef[[i]] * ccoef[[i]]]) /
      (2 * acoef[[i]]), {i, Length[ptO]}];
   AOmod = Round[AO, .0001];
   AOpos = Round[AOpos, .0001];
   AOneg = Round[AOneg, .0001];
   Which[AOmod == AOpos, Test = 1; AOSign = "pos", AOmod == AOneg, Test = 1;
    AOSign = "neg", AOmod ≠ AOpos && AOmod ≠ AOneg, Test = 0; AOSign = 0];
   {AOmod, AOpos, AOneg, (*θAOmod,θAOpos,θAOneg,*)Test, AOSign}]
SecondRRConstraint[ptB_, θCB_, TaskP_, ptA_, ptC_] :=
 Module[{T1, T2, G1, W1, Dx, Dy, Ex, Ey, G3D, W3D, G, W, R, ConstraintEqns,
    ConstraintEqnsSimplified, sol2, ptD, ptE, θEA, θDC, n1, n2, SolnsFound2},
   T1 = Table[{{Cos[θCB[[i]]], -Sin[θCB[[i]]], ptB[[i, 1]]},
      {Sin[θCB[[i]]], Cos[θCB[[i]]], ptB[[i, 2]]}, {0, 0, 1}}, {i, Length[ptB]}];
   T2 = Table[{{Cos[TaskP[[i, 1]]], -Sin[TaskP[[i, 1]]], TaskP[[i, 2]]},
      {Sin[TaskP[[i, 1]]], Cos[TaskP[[i, 1]]], TaskP[[i, 3]]},
      {0, 0, 1}}, {i, Length[ptB]}];
   G1 = {Dx, Dy, 1};
   W1 = {Ex, Ey, 1};
   G3D = Table[T1[[i]].Inverse[T1[[1]]].G1, {i, Length[ptB]}];
   W3D = Table[T2[[i]].Inverse[T2[[1]]].W1, {i, Length[ptB]}];
   G = G3D.{{1, 0, 0}, {0, 1, 0}, {0, 0, 0}};
   W = W3D.{{1, 0, 0}, {0, 1, 0}, {0, 0, 0}};
   ConstraintEqns = Table[(W[[i]] - G[[i]]).(W[[i]] - G[[i]]) - R ^ 2, {i, Length[ptB]}];
   ConstraintEqnsSimplified =
    Table[ConstraintEqns[[i]] - ConstraintEqns[[1]] == 0, {i, 2, Length[ptB]}];
   Quiet[Check[sol2 = NSolve[Chop[Expand[ConstraintEqnsSimplified]], {Dx, Dy, Ex, Ey}];,
     Print["NSolve experienced a failure during the 2nd RR
        constraint and some solutions may have been lossed for
        the set of task positions: ", TaskP], NSolve::sfail]];
   sol2 = {Dx, Dy, Ex, Ey} /. sol2;
   SolnsFound2 = Length[sol2];
   n1 = Length[sol2];
   While[n1 > 0,
    sol2 = If[Im[sol2[[n1]]] == {0, 0, 0, 0} && Norm[sol2[[n1]], Infinity] < 10^10,
      sol2, Drop[sol2, {n1}]]; n1 = n1 - 1];
   ptD = Table[Table[{G[[j, 1]], G[[j, 2]]} /. {Dx → sol2[[i, 1]], Dy → sol2[[i, 2]],
        Ex → sol2[[i, 3]], Ey → sol2[[i, 4]]}, {j, Length[ptB]}], {i, Length[sol2]}];
   ptE = Table[Table[{W[[j, 1]], W[[j, 2]]} /. {Dx → sol2[[i, 1]], Dy → sol2[[i, 2]],
        Ex → sol2[[i, 3]], Ey → sol2[[i, 4]]}, {j, Length[ptB]}], {i, Length[sol2]}];
   n2 = Length[sol2];
   While[n2 > 0,
    If[Round[ptD[[n2]], .0001] ≠ Round[ptC, .0001] &&
      Round[ptE[[n2]], .0001] ≠ Round[ptA, .0001], ptD = ptD;
     ptE = ptE, ptD = Drop[ptD, {n2}]; ptE = Drop[ptE, {n2}]];
    n2 = n2 - 1];
   θEA =
    Table[Table[VectAngle[ptE[[i, j]] - ptA[[j]]], {j, Length[ptB]}], {i, Length[ptE]}];
   θDC = Table[Table[VectAngle[ptD[[i, j]] - ptC[[j]]], {j, Length[ptB]}],
      {i, Length[ptD]}];
   {ptD, ptE, θEA, θDC, SolnsFound2}]
SecondRRConstraintAnalysis[ptA_, ptC_, ptD_, ptE_, θEA_, θDC_] :=
 Module[{EA, DC, ED, A1coef, B1coef, C1coef, θEApos, θEAneg,
    θEAmod, A2coef, B2coef, C2coef, θDCpos, θDCneg, θDCmod, Test, θEASign},
   EA = VectLength[ptE[[1]], ptA[[1]]];
   DC = VectLength[ptD[[1]], ptC[[1]]];
   ED = VectLength[ptE[[1]], ptD[[1]]];
   A1coef =
    Table[2 * (ptA[[i, 1]] - ptC[[i, 1]]) * EA - 2 * EA * DC * Cos[θDC[[i]]], {i, Length[ptA]}];
   B1coef = Table[2 * (ptA[[i, 2]] - ptC[[i, 2]]) * EA - 2 * EA * DC * Sin[θDC[[i]]],
      {i, Length[ptA]}];
   C1coef = Table[(ptA[[i, 1]] - ptC[[i, 1]])^2 + (ptA[[i, 2]] - ptC[[i, 2]])^2 +
      EA^2 + DC^2 - ED^2 - 2 * (ptA[[i, 1]] - ptC[[i, 1]]) * DC * Cos[θDC[[i]]] -
      2 * (ptA[[i, 2]] - ptC[[i, 2]]) * DC * Sin[θDC[[i]]], {i, Length[ptA]}];
   θEApos = Table[ArcTan[A1coef[[i]], B1coef[[i]]] +
      ArcCos[-C1coef[[i]] / Sqrt[A1coef[[i]]^2 + B1coef[[i]]^2]], {i, Length[ptA]}];
   θEAneg = Table[ArcTan[A1coef[[i]], B1coef[[i]]] -
      ArcCos[-C1coef[[i]] / Sqrt[A1coef[[i]]^2 + B1coef[[i]]^2]], {i, Length[ptA]}];
   If[Im[θEApos] == {0, 0, 0, 0, 0} && Im[θEAneg] == {0, 0, 0, 0, 0},
    θEAmod = NormalizeAngles[θEA, 0, 2 * Pi];
    θEApos = NormalizeAngles[θEApos, 0, 2 * Pi];
    θEAneg = NormalizeAngles[θEAneg, 0, 2 * Pi];
    Which[θEAmod == θEApos, Test = 1; θEASign = "pos", θEAmod == θEAneg, Test = 1;
      θEASign = "neg", θEAmod ≠ θEApos && θEAmod ≠ θEAneg, Test = 0; θEASign = 0],
    Test = 0];
   {θEAmod, θEApos, θEAneg, (*θDCmod,θDCpos,θDCneg,*)Test, θEASign}]
GraphMech[Soln_] := Module[{MechLines, TriagBCD, TriagATaskPE,
    GroundPivotO, GroundPivotB, Pivots, Sliders, TaskPositions},
   MechLines = Table[Line[{Soln[[1, i]], Soln[[2, i]], Soln[[7, i]],
      Soln[[6, i]], Soln[[5, i]], Soln[[3, i]], Soln[[4, i]],
      Soln[[2, i]], Soln[[6, i]], Soln[[5, i]], Soln[[4, i]]}], {i, 5}];
   TriagBCD = Table[Polygon[{Soln[[3, i]], Soln[[4, i]], Soln[[5, i]]}], {i, 5}];
   TriagATaskPE = Table[Polygon[{Soln[[2, i]], Soln[[7, i]], Soln[[6, i]]}], {i, 5}];
   GroundPivotO = Polygon[{Soln[[1, 1]], Soln[[1, 1]] + {-2, -3}, Soln[[1, 1]] + {2, -3}}];
   GroundPivotB = Polygon[{Soln[[3, 1]], Soln[[3, 1]] + {-2, -3}, Soln[[3, 1]] + {2, -3}}];
   Pivots = Table[Table[Disk[Soln[[i, j]], 1], {j, 5}], {i, 4, 6}];
   Sliders =
    Table[Polygon[Transpose[{{1, 0, 0}, {0, 1, 0}}.{{Cos[Soln[[9, i]]], -Sin[Soln[[9, i]]],
          Soln[[2, i, 1]]}, {Sin[Soln[[9, i]]], Cos[Soln[[9, i]]], Soln[[2, i, 2]]},
         {0, 0, 1}}.{{2, 2, -2, -2}, {1, -1, -1, 1}, {1, 1, 1, 1}}]], {i, 5}];
   TaskPositions = Table[Line[Transpose[{{1, 0, 0}, {0, 1, 0}}.
        {{Cos[Soln[[8, i]]], -Sin[Soln[[8, i]]], Soln[[7, i, 1]]},
         {Sin[Soln[[8, i]]], Cos[Soln[[8, i]]], Soln[[7, i, 2]]}, {0, 0, 1}}.
        {{3, 0, 0}, {0, 0, 3}, {1, 1, 1}}]], {i, 5}];
   Graphics[{Gray, TriagATaskPE, TriagBCD, Black, MechLines, GroundPivotO,
     Pivots, GroundPivotB, Sliders, Thick, TaskPositions}, Axes → True]]
FindRanges[θAO_, θEA_] := Module[{Starts, Ends, n, end},
   Starts = {};
   Ends = {};
   If[Im[θEA[[Length[θAO]]]] ≠ 0 && Im[θEA[[1]]] == 0,
    Starts = Append[Starts, θAO[[1]]]];
   If[Im[θEA[[1]]] == 0 && Im[θEA[[2]]] ≠ 0,
    Ends = Append[Ends, θAO[[1]]]];
   n = 2;
   While[n ≤ Length[θAO] - 1,
    If[Im[θEA[[n - 1]]] ≠ 0 && Im[θEA[[n]]] == 0,
     Starts = Append[Starts, θAO[[n]]]];
    If[Im[θEA[[n]]] == 0 && Im[θEA[[n + 1]]] ≠ 0,
     Ends = Append[Ends, θAO[[n]]]];
    n++];
   If[Im[θEA[[n - 1]]] ≠ 0 && Im[θEA[[n]]] == 0,
    Starts = Append[Starts, θAO[[n]]]];
   If[Im[θEA[[n]]] == 0 && Im[θEA[[1]]] ≠ 0,
    Ends = Append[Ends, θAO[[n]]]];
   If[Length[Starts] == 0, Starts = {0}; Ends = {2 * Pi}; Goto[end]];
   If[Im[θEA[[n]]] == 0 && Im[θEA[[1]]] == 0,
    Ends = Drop[Append[Ends, Ends[[1]]], 1]];
   Label[end];
    {Starts, Ends}]
GraphMechBatch[set_, first_, last_] := Module[{n, RPRSixBarGraphs},
   n = first;
   While[n ≤ last, RPRSixBarGraphs = GraphMech[set[[n]]];
    Print["Solution No. ", n]; Print[RPRSixBarGraphs]; n++]]
FindRange[Starts_, Ends_, θAOSoln_] :=
 Module[{n, Starter, Ender, θAOTP, θAOStart, θAOEnd, end},
   If[Starts == {0} && Ends == {2 * Pi}, θAOStart = 0; θAOEnd = 2 * Pi; Goto[end]];
   n = 1;
   While[n ≤ Length[Starts],
     {Starter, Ender} = NormalizeAngles[{Starts[[n]], Ends[[n]]}, 0, 2 * Pi];
    If[Starter < Ender,
     θAOTP = NormalizeAngles[θAOSoln, 0, 2 * Pi];
     If[Table[Starter ≤ θAOTP[[i]] ≤ Ender, {i, 5}] == ConstantArray[True, 5],
      θAOStart = Starter; θAOEnd = Ender; Break[]],
     {Starter, Ender} = NormalizeAngles[{Starts[[n]], Ends[[n]]}, -Pi, Pi];
     θAOTP = NormalizeAngles[θAOSoln, -Pi, Pi];
     If[Table[Starter ≤ θAOTP[[i]] ≤ Ender, {i, 5}] == ConstantArray[True, 5],
      θAOStart = Starter; θAOEnd = Ender; Break[]]];
    n++];
   If[n > Length[Starts], θAOStart = "error"; θAOEnd = "error"];
   Label[end];
   {θAOStart, θAOEnd}]
ListAnimateMech[Soln_] :=
 Module[{Ox, Oy, Bx, By, CA, CB, θδ2, EA, DC, ED, θδ3, TA, θδ1, θδ4, θAOStart, θAOEnd,
    Steps, θAO, acoef, bcoef, ccoef, AO, θCB, Ax, Ay, Cx, Cy, θDC, Acoef, Bcoef, Ccoef, θEA,
    Starts, Ends, RangeTest, θED, Dx, Dy, Ex, Ey, θTA, θT, Tx, Ty, Xrange, Yrange, Xpadding,
    Ypadding, Xmin, Xmax, Ymin, Ymax, ScalingRange, GPSize, PSize, SSize, TPSize, MechLines,
    TriagBCD, TriagATpE, GroundPivotO, GroundPivotB, PivotC, PivotD, PivotE, SliderTrans,
    Sliders, TriagSlider, TaskPosTrans, TaskPosMobile, TaskPos, end, Animation},
   (*First loop constants*)
   Ox = Soln[[1, 1, 1]]; Oy = Soln[[1, 1, 2]]; Bx = Soln[[3, 1, 1]]; By = Soln[[3, 1, 2]];
   CA = VectLength[Soln[[4, 1]], Soln[[2, 1]]];
   CB = VectLength[Soln[[4, 1]], Soln[[3, 1]]];
   θδ2 = VectAngle[Soln[[4, 1]] - Soln[[2, 1]], Soln[[2, 1]] - Soln[[1, 1]]];
   (*Second loop constants*)
   EA = VectLength[Soln[[6, 1]], Soln[[2, 1]]];
   DC = VectLength[Soln[[5, 1]], Soln[[4, 1]]];
   ED = VectLength[Soln[[6, 1]], Soln[[5, 1]]];
   θδ3 = VectAngle[Soln[[4, 1]] - Soln[[3, 1]], Soln[[5, 1]] - Soln[[4, 1]]];
   (*Tool frame constants*)
   TA = VectLength[Soln[[7, 1]], Soln[[2, 1]]];
   θδ1 = Soln[[8, 1]] - VectAngle[Soln[[7, 1]] - Soln[[2, 1]]];
   θδ4 = VectAngle[Soln[[7, 1]] - Soln[[2, 1]], Soln[[6, 1]] - Soln[[2, 1]]];
   (*Input variable*)
   θAOStart = 0; θAOEnd = 2 * Pi;
   Steps = 500;
   θAO = Array[θAOStart + (θAOEnd - θAOStart) / (Steps - 1) * (# - 1) &, Steps - 1];
   (*     First loop variables*)
   acoef = Table[1, {i, Length[θAO]}];
   bcoef = Table[2 * (Ox - Bx) * Cos[θAO[[i]]] +
      2 * (Oy - By) * Sin[θAO[[i]]] + 2 * CA * Cos[θδ2], {i, Length[θAO]}];
   ccoef = Table[(Ox - Bx) ^2 + (Oy - By) ^2 + CA ^2 - CB ^2 + 2 * (Ox - Bx) * CA * Cos[θAO[[i]] + θδ2] +
      2 * (Oy - By) * CA * Sin[θAO[[i]] + θδ2], {i, Length[θAO]}];
   Which[Soln[[12]] == "pos", AO = Table[
      (-bcoef[[i]] + Sqrt[bcoef[[i]] ^2 - 4 * acoef[[i]] * ccoef[[i]]]) / (2 * acoef[[i]]),
      {i, Length[θAO]}], Soln[[12]] == "neg",
    AO = Table[(-bcoef[[i]] - Sqrt[bcoef[[i]] ^2 - 4 * acoef[[i]] * ccoef[[i]]]) /
       (2 * acoef[[i]]), {i, Length[θAO]}]];
   θCB = Table[ArcTan[Ox - Bx + AO[[i]] * Cos[θAO[[i]]] + CA * Cos[θAO[[i]] + θδ2],
      Oy - By + AO[[i]] * Sin[θAO[[i]]] + CA * Sin[θAO[[i]] + θδ2]], {i, Length[θAO]}];
   Ax = Table[Ox + AO[[i]] * Cos[θAO[[i]]], {i, Length[θAO]}];
   Ay = Table[Oy + AO[[i]] * Sin[θAO[[i]]], {i, Length[θAO]}];
   Cx = Table[Bx + CB * Cos[θCB[[i]]], {i, Length[θAO]}];
```

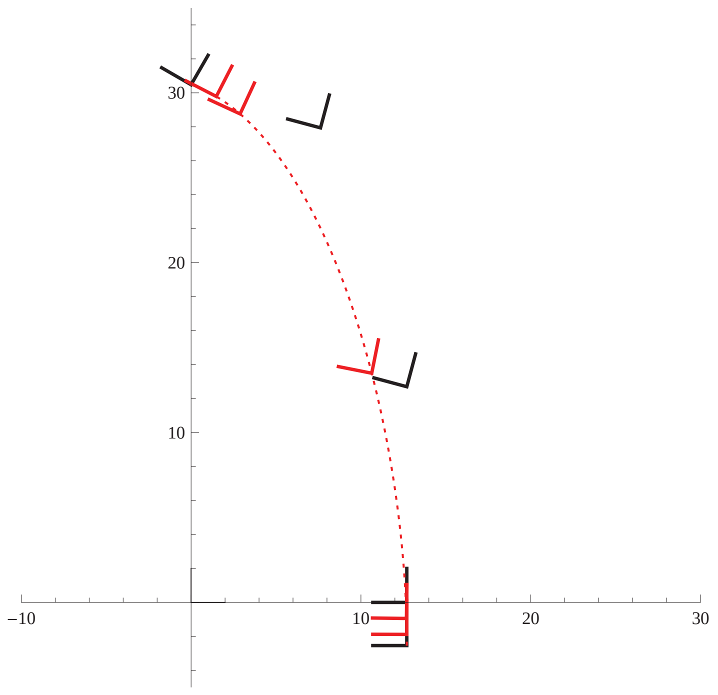
```
Cy = Table[By + CB * Sin[θCB[[i]]], {i, Length[θAO]}];
(*     Second loop variables*)
θDC = Table[θCB[[i]] - θδ3, {i, Length[θAO]}];
Acoef = Table[2 * (Ax[[i]] - Cx[[i]]) * EA - 2 * EA * DC * Cos[θDC[[i]]], {i, Length[θAO]}];
Bcoef = Table[2 * (Ay[[i]] - Cy[[i]]) * EA - 2 * EA * DC * Sin[θDC[[i]]], {i, Length[θAO]}];
Ccoef = Table[(Ax[[i]] - Cx[[i]])^2 + (Ay[[i]] - Cy[[i]])^2 +
   EA^2 + DC^2 - ED^2 - 2 * (Ax[[i]] - Cx[[i]]) * DC * Cos[θDC[[i]]] -
   2 * (Ay[[i]] - Cy[[i]]) * DC * Sin[θDC[[i]]], {i, Length[θAO]}];
Which[Soln[[13]] == "pos", θEA = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] +
    ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[θAO]}],
 Soln[[13]] == "neg", θEA = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] -
    ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[θAO]}]];
{Starts, Ends} = FindRanges[θAO, θEA];
{θAOStart, θAOEnd} = FindRange[Starts, Ends, Soln[[9]]];
If[θAOStart == "error", (*All task positions are not achievable within a single
  range of the input θAO.*)RangeTest = "Fail"; Goto[end], RangeTest = "Pass"];
Steps = 100;
θAO = Array[θAOStart + (θAOEnd - θAOStart) / (Steps - 1) * (# - 1) &, Steps];
If[θAOStart ≠ 0 && θAOEnd ≠ 2 * Pi, θAO = Join[θAO, Reverse[θAO]]];
(*First loop variables*)
acoef = Table[1, {i, Length[θAO]}];
bcoef = Table[2 * (Ox - Bx) * Cos[θAO[[i]]] +
   2 * (Oy - By) * Sin[θAO[[i]]] + 2 * CA * Cos[θδ2], {i, Length[θAO]}];
ccoef = Table[(Ox - Bx)^2 + (Oy - By)^2 + CA^2 - CB^2 + 2 * (Ox - Bx) * CA * Cos[θAO[[i]] + θδ2] +
   2 * (Oy - By) * CA * Sin[θAO[[i]] + θδ2], {i, Length[θAO]}];
Which[Soln[[12]] == "pos", AO = Table[
   (-bcoef[[i]] + Sqrt[bcoef[[i]]^2 - 4 * acoef[[i]] * ccoef[[i]]]) / (2 * acoef[[i]]),
   {i, Length[θAO]}], Soln[[12]] == "neg",
 AO = Table[(-bcoef[[i]] - Sqrt[bcoef[[i]]^2 - 4 * acoef[[i]] * ccoef[[i]]]) /
    (2 * acoef[[i]]), {i, Length[θAO]}]];
If[Im[AO] == ConstantArray[0, Length[AO]], ,
 Print["Imaginary values encountered for length AO"]; Goto[end]];
θCB = Table[ArcTan[Ox - Bx + AO[[i]] * Cos[θAO[[i]]] + CA * Cos[θAO[[i]] + θδ2],
   Oy - By + AO[[i]] * Sin[θAO[[i]]] + CA * Sin[θAO[[i]] + θδ2]], {i, Length[θAO]}];
Ax = Table[Ox + AO[[i]] * Cos[θAO[[i]]], {i, Length[θAO]}];
Ay = Table[Oy + AO[[i]] * Sin[θAO[[i]]], {i, Length[θAO]}];
Cx = Table[Bx + CB * Cos[θCB[[i]]], {i, Length[θAO]}];
Cy = Table[By + CB * Sin[θCB[[i]]], {i, Length[θAO]}];
(*Second loop variables*)
θDC = Table[θCB[[i]] - θδ3, {i, Length[θAO]}];
Acoef = Table[2 * (Ax[[i]] - Cx[[i]]) * EA - 2 * EA * DC * Cos[θDC[[i]]], {i, Length[θAO]}];
Bcoef = Table[2 * (Ay[[i]] - Cy[[i]]) * EA - 2 * EA * DC * Sin[θDC[[i]]], {i, Length[θAO]}];
Ccoef = Table[(Ax[[i]] - Cx[[i]])^2 + (Ay[[i]] - Cy[[i]])^2 +
   EA^2 + DC^2 - ED^2 - 2 * (Ax[[i]] - Cx[[i]]) * DC * Cos[θDC[[i]]] -
   2 * (Ay[[i]] - Cy[[i]]) * DC * Sin[θDC[[i]]], {i, Length[θAO]}];
Which[Soln[[13]] == "pos", θEA = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] +
    ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[θAO]}],
 Soln[[13]] == "neg", θEA = Table[ArcTan[Acoef[[i]], Bcoef[[i]]] -
    ArcCos[-Ccoef[[i]] / Sqrt[Acoef[[i]]^2 + Bcoef[[i]]^2]], {i, Length[θAO]}]];
If[Im[θEA] == ConstantArray[0, Length[θEA]], ,
 Print["Imaginary values encountered for angle θEA"]; Goto[end]];
```

```
θED = Table[ArcTan[Ax[[i]] - Cx[[i]] + EA * Cos[θEA[[i]]] - DC * Cos[θDC[[i]]],
    Ay[[i]] - Cy[[i]] + EA * Sin[θEA[[i]]] - DC * Sin[θDC[[i]]]], {i, Length[θAO]}];
Dx = Table[Cx[[i]] + DC * Cos[θDC[[i]]], {i, Length[θAO]}];
Dy = Table[Cy[[i]] + DC * Sin[θDC[[i]]], {i, Length[θAO]}];
Ex = Table[Ax[[i]] + EA * Cos[θEA[[i]]], {i, Length[θAO]}];
Ey = Table[Ay[[i]] + EA * Sin[θEA[[i]]], {i, Length[θAO]}];
(*Tool frame variables*)
θTA = Table[θEA[[i]] + θδ4, {i, Length[θAO]}];
θT = Table[θTA[[i]] + θδ1, {i, Length[θAO]}];
Tx = Table[Ax[[i]] + TA * Cos[θTA[[i]]], {i, Length[θAO]}];
Ty = Table[Ay[[i]] + TA * Sin[θTA[[i]]], {i, Length[θAO]}];
(*Bounds and scaling*)
Xrange = Max[Ox, Ax, Bx, Cx, Dx, Ex, Tx] - Min[Ox, Ax, Bx, Cx, Dx, Ex, Tx];
Yrange = Max[Oy, Ay, By, Cy, Dy, Ey, Ty] - Min[Oy, Ay, By, Cy, Dy, Ey, Ty];
Which[Xrange ≥ Yrange, Xpadding = .05;
  Ypadding = (Xrange + 2 * Xpadding * Xrange - Yrange) / (2 * Yrange),
  Yrange > Xrange, Ypadding = .05; Xpadding =
   (Yrange + 2 * Ypadding * Yrange - Xrange) / (2 * Xrange)];
Xmin = Min[Ox, Ax, Bx, Cx, Dx, Ex, Tx] - Xpadding * Xrange;
Xmax = Max[Ox, Ax, Bx, Cx, Dx, Ex, Tx] + Xpadding * Xrange;
Ymin = Min[Oy, Ay, By, Cy, Dy, Ey, Ty] - Ypadding * Yrange;
Ymax = Max[Oy, Ay, By, Cy, Dy, Ey, Ty] + Ypadding * Yrange;
Which[Xrange ≥ Yrange, ScalingRange = Xrange, Yrange > Xrange, ScalingRange = Yrange];
GPSize = .03 * ScalingRange;
PSize = .01 * ScalingRange;
SSize = .02 * ScalingRange;
TPSize = .04 * ScalingRange;
(*Graphics*)
MechLines =
  Table[Line[{{Ox, Oy}, {Ax[[i]], Ay[[i]]}, {Ex[[i]], Ey[[i]]}, {Dx[[i]], Dy[[i]]}}],
   {i, Length[θAO]}];
TriagBCD = Table[Polygon[{{Bx, By}, {Cx[[i]], Cy[[i]]}, {Dx[[i]], Dy[[i]]}}],
   {i, Length[θAO]}];
TriagATpE = Table[Polygon[{{Ax[[i]], Ay[[i]]}, {Tx[[i]], Ty[[i]]},
     {Ex[[i]], Ey[[i]]}}], {i, Length[θAO]}];
GroundPivotO = Polygon[{{Ox, Oy}, {Ox, Oy} + {-.6 * GPSize, -GPSize},
    {Ox, Oy} + {.6 * GPSize, -GPSize}}];
GroundPivotB = Polygon[{{Bx, By}, {Bx, By} + {-.6 * GPSize, -GPSize},
    {Bx, By} + {.6 * GPSize, -GPSize}}];
PivotC = Table[Disk[{Cx[[i]], Cy[[i]]}, PSize], {i, Length[θAO]}];
PivotD = Table[Disk[{Dx[[i]], Dy[[i]]}, PSize], {i, Length[θAO]}];
PivotE = Table[Disk[{Ex[[i]], Ey[[i]]}, PSize], {i, Length[θAO]}];
SliderTrans = Table[{{Cos[θAO[[i]]], -Sin[θAO[[i]]], Ax[[i]]},
    {Sin[θAO[[i]]], Cos[θAO[[i]]], Ay[[i]]}}, {i, Length[θAO]}];
Sliders = Table[Polygon[{SliderTrans[[i]].{SSize, .5 * SSize, 1},
     SliderTrans[[i]].{SSize, -.5 * SSize, 1}, SliderTrans[[i]].{-SSize, -.5 * SSize, 1},
     SliderTrans[[i]].{-SSize, .5 * SSize, 1}}], {i, Length[θAO]}];
TriagSlider = Table[Polygon[{SliderTrans[[i]].{SSize, 0, 1},
     SliderTrans[[i]].{-SSize, 0, 1}, {Cx[[i]], Cy[[i]]}}], {i, Length[θAO]}];
TaskPosTrans = Table[{{Cos[θT[[i]]], -Sin[θT[[i]]], Tx[[i]]},
    {Sin[θT[[i]]], Cos[θT[[i]]], Ty[[i]]}}, {i, Length[θAO]}];
```

```
TaskPosMobile = Table[Line[{TaskPosTrans[[i]].{TPSize, 0, 1}, TaskPosTrans[[i]].
      {0, 0, 1}, TaskPosTrans[[i]].{0, TPSize, 1}}], {i, Length[θAO]}];
TaskPos = Table[Line[Transpose[{{1, 0, 0}, {0, 1, 0}}.
      {{Cos[Soln[[8, i]]], -Sin[Soln[[8, i]]], Soln[[7, i, 1]]},
       {Sin[Soln[[8, i]]], Cos[Soln[[8, i]]], Soln[[7, i, 2]]}, {0, 0, 1}}.
      {{TPSize, 0, 0}, {0, 0, TPSize}, {1, 1, 1}}]], {i, 5}];
Animation = ListAnimate[Table[Graphics[{Gray, EdgeForm[Thickness[Medium]],
      TriagATpE[[i]], TriagBCD[[i]], TriagSlider[[i]], Black, Sliders[[i]],
      MechLines[[i]], GroundPivotO, GroundPivotB, PivotC[[i]], PivotD[[i]],
      PivotE[[i]], Thick, TaskPosMobile[[i]], TaskPos}, Axes → True,
     PlotRange → {{Xmin, Xmax}, {Ymin, Ymax}}], {i, Length[θAO]}]];
Label[end];
{RangeTest, Animation}]
```

Execution

```
(*Input task positions (angle in degrees, x-coord, y-coord)*)
TaskP = {{60, 0, 30.48},
   {75, 7.62, 27.94},
   {75, 12.70, 12.70},
   {90, 12.70, 0},
   {90, 12.70, -2.54}};
(*Input RPR parameters*)
Ox = 0;
Oy = 0;
TA = 0;
θδ1 = 0;
(*Bezier Smoothening Option*)
BezOpt = "No";
BezParams = {.05, .1, .5, .9, .95};
BezierTaskGraph[TaskP, BezParams]
```



```
(*Choose tolerance type either "Traditional" or "Point By Point"*)
TolType = "Point By Point";
(*Traditional tolerances*)
Tolϕ = {-2, 2};
Tolx = {-2, 2};
Toly = {-2, 2};
(*Point By Point tolerances*)
TolϕByPoint = {{-1, 1}, {-20, 20}, {-8, 8}, {-2, 2}, {-2, 2}};
TolxByPoint =
  {{-2.54, 2.54}, {-5.08, 5.08}, {-2.54, 2.54}, {-.508, .508}, {-.508, .508}};
TolyByPoint = {{-2.54, 2.54}, {-7.62, 2.54}, {-10.16, 5.08}, {-0, 1.27}, {-1.27, 1.27}};
TolOx = {0, 63.50};
TolOy = {0, 38.10};
TolTA = {0, 25.40};
Tolθδ1 = {-180, 180};
(*Bezier parameters' tolerances*)
TolBez = {{0, .05}, {-.05, .05}, {-.05, .05}, {-.05, .05}, {-.05, 0}};
(*Set number of loops*)
Loops = 1000;

(*Randomization Preview*)
{TaskPRand, OxRand, OyRand, TARand, θδ1Rand} = RandomizeAllParams[TaskP, TolϕByPoint,
   TolxByPoint, TolyByPoint, Ox, Oy, TA, θδ1, TolOx, TolOy, TolTA, Tolθδ1];
If[BezOpt == "Yes", BezierTaskGraph[TaskPRand.{{180 / Pi, 0, 0}, {0, 1, 0}, {0, 0, 1}},
  BezParams + RandomReal /@ TolBez]]

(*Run the code*)
FirstRRSolns = 0;
SecondRRSolns = 0;
FirstRRGoodSolns = {};
FirstRRBadSolns = {};
GoodSolns = {};
```

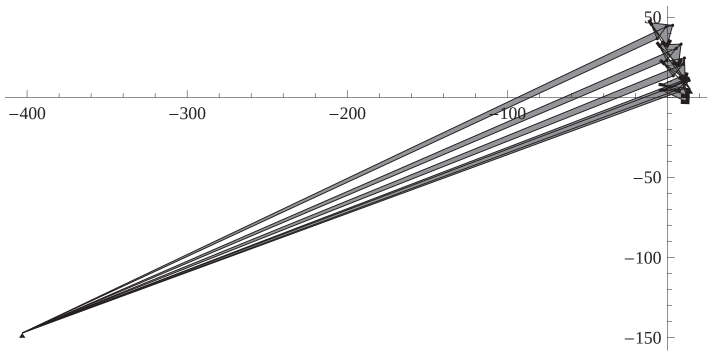
```
BadSolns = {};
n1 = 1;
While[n1 ≤ Loops,
  Which[TolType == "Traditional", TaskPRand = Randomize[TaskP, Tolϕ, Tolx, Toly],
   TolType == "Point By Point",
   {TaskPRand, OxRand, OyRand, TARand, θδ1Rand} = RandomizeAllParams[TaskP, TolϕByPoint,
     TolxByPoint, TolyByPoint, Ox, Oy, TA, θδ1, TolOx, TolOy, TolTA, Tolθδ1]];
  If[BezOpt == "Yes", {BezFunc, TaskPBez} = BezSmooth[
     TaskPRand.{{180 / Pi, 0, 0}, {0, 1, 0}, {0, 0, 1}}, BezParams + RandomReal /@ TolBez]];
  Check[
    Which[TolType == "Traditional", {ptO, ptA, θAO} =
      InverseKinematics[Ox, Oy, TA, θδ1, TaskPRand], TolType == "Point By Point",
     {ptO, ptA, θAO} = InverseKinematics[OxRand, OyRand, TARand, θδ1Rand, TaskPRand]];
    {ptB, ptC, θCB, SolnsFound1} = FirstRRConstraint[ptA, θAO];
    FirstRRSolns = FirstRRSolns + SolnsFound1;
    (*Print["FirstRRSolns = ",SolnsFound1];*)
    n2 = 1;
    While[n2 ≤ Length[ptB],
     {AOmod, AOpos, AOneg, Test1, AOSign} =
      FirstRRConstraintAnalysis[ptO, ptA, ptB[[n2]], ptC[[n2]], θAO, θCB[[n2]]];
     If[Test1 == 1, FirstRRGoodSolns = Join[FirstRRGoodSolns,
        {{ptO, ptA, ptB[[n2]], ptC[[n2]], θCB[[n2]], AOSign}}], FirstRRBadSolns =
       Join[FirstRRBadSolns, {{ptO, ptA, ptB[[n2]], ptC[[n2]], θCB[[n2]], AOSign}}]];
     {ptD, ptE, θEA, θDC, SolnsFound2} = SecondRRConstraint[
       ptB[[n2]], θCB[[n2]], TaskPRand, ptA, ptC[[n2]]];
     If[Test1 == 1, SecondRRSolns = SecondRRSolns + SolnsFound2];
     (*Print["SecondRRSolns = ",SolnsFound2];*)
     If[Test1 == 1, n3 = 1, n3 = Length[ptD] + 1];
     While[n3 ≤ Length[ptD],
      {θEAmod, θEApos, θEAneg, Test2, θEASign} = SecondRRConstraintAnalysis[
        ptA, ptC[[n2]], ptD[[n3]], ptE[[n3]], θEA[[n3]], θDC[[n3]]];
      If[Test2 == 1, GoodSolns = Join[GoodSolns, {{ptO, ptA, ptB[[n2]],
           ptC[[n2]], ptD[[n3]], ptE[[n3]], TaskPRand.{{0, 0}, {1, 0}, {0, 1}},
           TaskPRand[[All, 1]], θAO, θCB[[n2]], θEA[[n3]], AOSign, θEASign}}],
       BadSolns = Join[BadSolns, {{ptO, ptA, ptB[[n2]], ptC[[n2]], ptD[[n3]],
           ptE[[n3]], TaskPRand.{{0, 0}, {1, 0}, {0, 1}}, TaskPRand[[All, 1]],
           θAO, θCB[[n2]], θEA[[n3]], AOSign, θEASign}}]];
      n3++];
     n2++];,
    Print[TaskPRand]]
   n1++];
Print["         Solutions for First RR Constraint = ", FirstRRSolns]
Print[" Dropped Solutions for First RR Constraint = ",
 FirstRRSolns - Length[FirstRRGoodSolns] - Length[FirstRRBadSolns]]
Print["    Good Solutions for First RR Constraint = ", Length[FirstRRGoodSolns]]
Print["     Bad Solutions for First RR Constraint = ", Length[FirstRRBadSolns]]
Print["        Solutions for Second RR Constraint = ", SecondRRSolns]
Print["Dropped solutions for Second RR Constraint = ",
 SecondRRSolns - Length[GoodSolns] - Length[BadSolns]]
Print["                             Good Solutions = ", Length[GoodSolns]]
Print["                              Bad Solutions = ", Length[BadSolns]]
```

```
n4 = Length[GoodSolns];
While[n4 > 0,
  {RangeTest, Animation} = ListAnimateMech[GoodSolns[[n4]]];
  GoodSolns = If[RangeTest == "Pass", GoodSolns, Drop[GoodSolns, {n4}]];
  n4 = n4 - 1];
Print["       Good Solutions without Range Errors = ", Length[GoodSolns]]
```

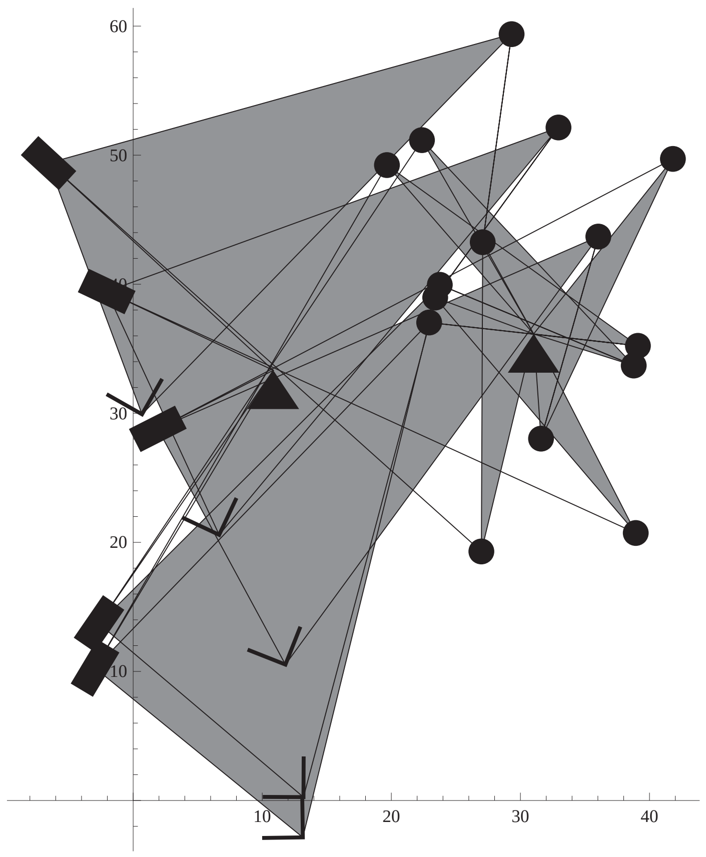
```
         Solutions for First RR Constraint = 5130
 Dropped Solutions for First RR Constraint = 3418
    Good Solutions for First RR Constraint = 364
     Bad Solutions for First RR Constraint = 1348
        Solutions for Second RR Constraint = 2046
Dropped solutions for Second RR Constraint = 1476
                            Good Solutions = 36
                             Bad Solutions = 534
       Good Solutions without Range Errors = 12
```

```
(*Preview solutions*)
GraphMechBatch[GoodSolns, 10, 12]
```

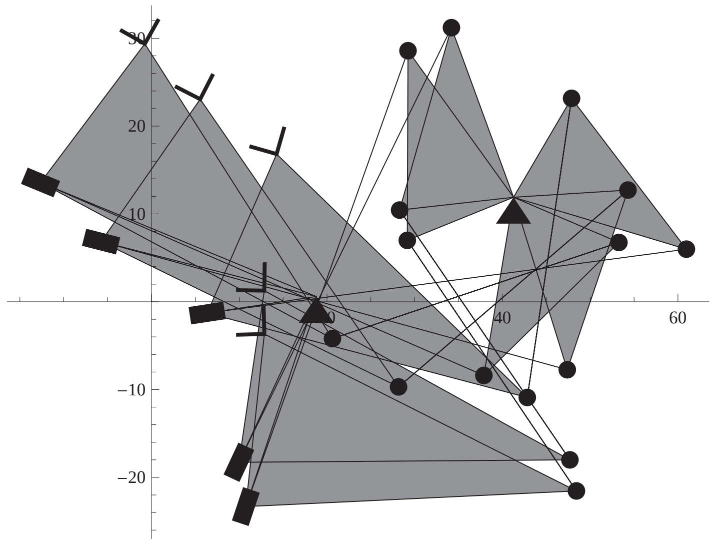
```
Solution No. 10
```



```
Solution No. 11
```



Solution No. 12


30
20
10
−10
−20
20
40
60