UNIVERSITY OF CALIFORNIA,
IRVINE

# The Kinematic Design of Six-bar Linkages Using Polynomial Homotopy Continuation

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Mark Mathew Plecnik

Dissertation Committee:
Professor J. Michael McCarthy, Chair
Professor David J. Reinkensmeyer
Professor Lorenzo Valdevit

2015

# DEDICATION

To my Ata: a man of the greatest character.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I first thank my advisor, Prof. J. Michael McCarthy, for his steadfast commitment to teaching kinematics and supporting his graduate students. I thank the authors of BERTINI for writing such a useful program. I thank the UC Irvine High Performance Computing Cluster which enabled much of the research in this dissertation by making their computing resources available to a wide spectrum of researchers on campus. I thank Prof. Timothy Rupert and Dr. Zhiliang Pan for providing me access to their compute nodes in order to solve the Stephenson II problem, and the San Diego Supercomputer Center of the XSEDE supercomputing network who provided the computing resources necessary in order to solve the Stephenson III problem.

I next thank Prof. David J. Reinkensmeyer who has motivated applications of my research towards stroke rehabilitation, and Prof. Lorenzo Valdevit for his continual interest in my research. I thank my labmates, Kaustubh Sonawale, Brian Parrish, Brandon Tsuge, Yang Liu, Shramana Ghosh, Jeff Glabe, Adam Nilsson, Andrew Brouwer, for their knowledge sharing and good conversation over the past five years.

Finally, I thank the National Science Foundation which provided the research funding that allowed my Ph.D. studies to take place.

# ABSTRACT OF THE DISSERTATION

The Kinematic Design of Six-bar Linkages
Using Polynomial Homotopy Continuation

By

Mark Mathew Plecnik

DOCTOR OF PHILOSOPHY in Mechanical and Aerospace Engineering

University of California, Irvine, 2015

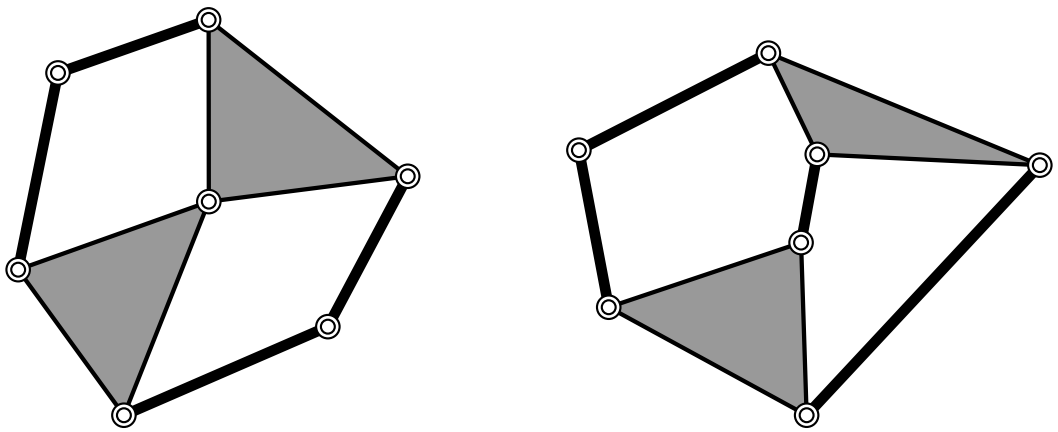
Professor J. Michael McCarthy, Chair

This dissertation presents the kinematic design of six-bar linkages for function, motion, and path generation by means of polynomial homotopy continuation algorithms. When no link dimensions are specified beforehand, the synthesis formulations for each design objective yield polynomial systems of degrees in the millions and billions, suggesting a large number of solutions. Complete solution sets to these systems have not yet been obtained and is the topic of this dissertation. Function generation for eleven positions is explored in most detail, in particular the Stephenson II and III function generators, for which we calculate multihomogeneous degrees of 264,241,152 and 55,050,240. A numerical reduction using homotopy estimates these systems to have 1,521,037 and 834,441 roots, respectively. For motion generation, the Watt I linkage can be specified for eight positions, producing a system of a multihomogeneous degree over 19 billion. However, for this work we focus on the smaller case of six positions, numerically reducing this system to an estimated 5,735 roots. For path generation we take a different approach. The design of path generators is formulated as RR chains constrained to have a single degree-of-freedom by attaching six-bar function generators to them. This enables us to use our results obtained on Stephenson II and III function generators to create four types of eleven position path generators: the Stephenson I linkage, two types of Stephenson II linkages, and the Stephenson III linkage.

# Chapter 1

# Introduction

## 1.1 Six-bar Linkages

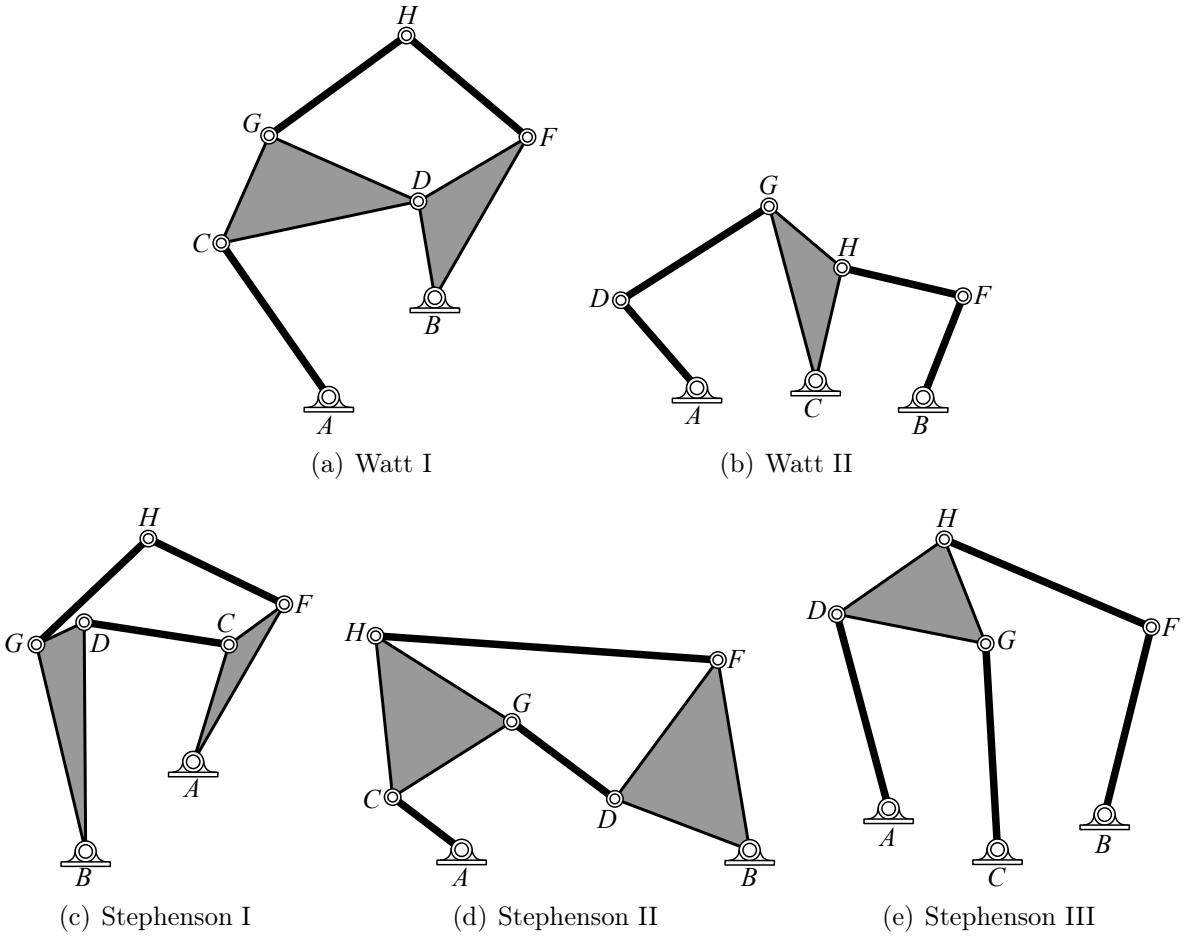


(a) Watt (b) Stephenson

Figure 1.1: Topologies of six-bar linkages.

The six-bar linkages are the next simplest single degree-of-freedom closed kinematic chains after the four-bar linkage. They are organized into two topologies, the Watt chain and the Stephenson chain, shown in Fig. 1.1. Each topology is further classified by which link is chosen as the ground link, forming a total of five different types of six-bar linkages named Watt I, Watt II, Stephenson I, Stephenson II, and Stephenson III, shown in Fig. 1.2. Each

six-bar has four binary links, two ternary links, and seven revolute joints.

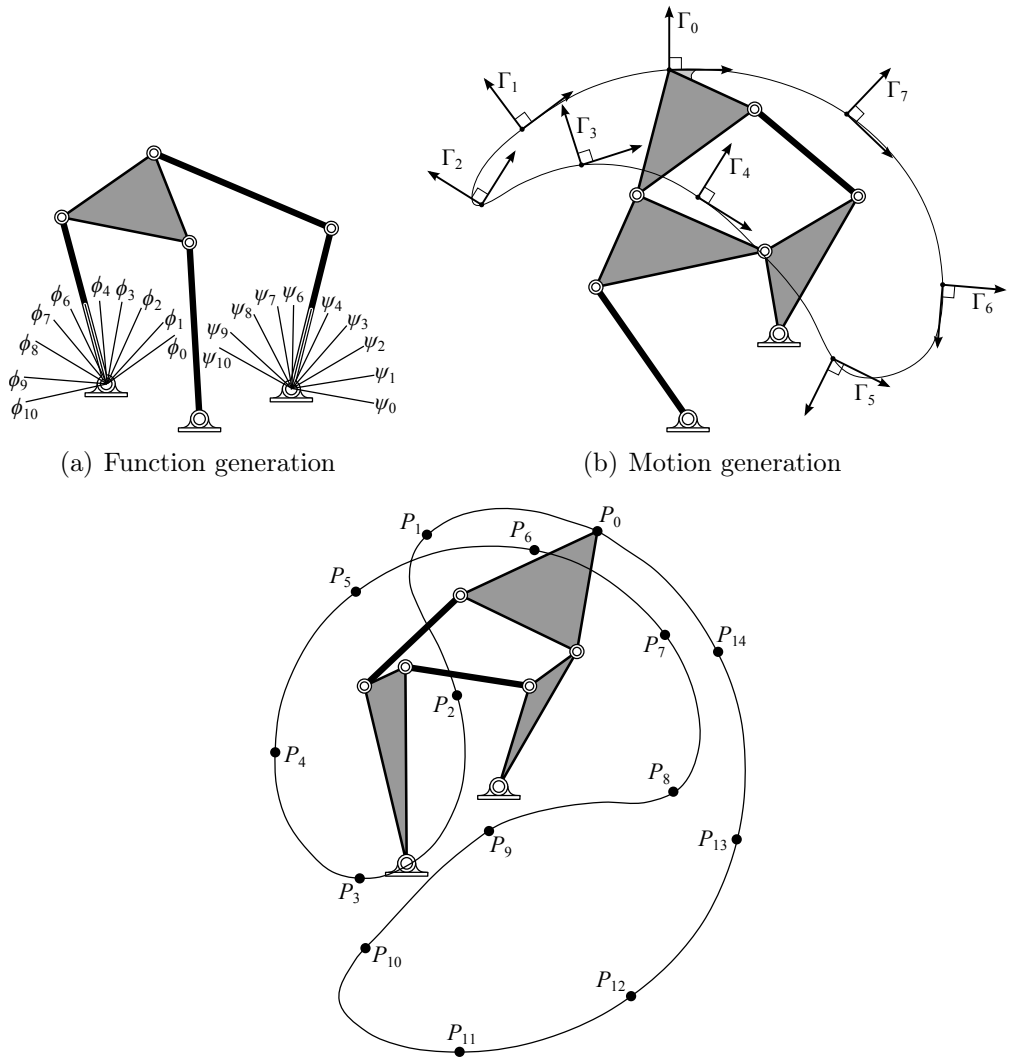



(a) Watt I (b) Watt II

(c) Stephenson I (d) Stephenson II (e) Stephenson III

Figure 1.2: The five types of six-bar linkages.

Although the six-bars represent only one more layer of complexity past the four-bar linkage, the motions they are able to produce are considerably more complicated. For example, a point attached to the coupler link of a four-bar can draw up to a degree 6 curve in the $x$-$y$ plane, while a point attached to a coupler link of a six-bar can draw up to an 18 degree curve [1].




(a) Function generation

(b) Motion generation

(c) Path generation

Figure 1.3: Three types of task specification for kinematic synthesis.

## 1.2 Kinematic Design

The objective of *kinematic design* is to find the dimensions of a linkage so that it produces some specified motion, called a *task*. Tasks can be specified in a variety of ways and usually include that links move through some specified set of angles, points, or both. Past literature [1, 2, 3, 4, 5, 6, 7] has posed three standard task specifications named function generation, motion generation, and path generation. *Function generation* requires that the angles of two links are coordinated in a particular fashion, $(\phi_j, \psi_j)$, $j = 0, \ldots, N-1$, see Fig. 1.3(a).

*Motion generation* requires that an end-effector link moves through a set of task positions, each specified by a translation and orientation, $\Gamma_j = (\mathbf{P}_j, \theta_j)$, $j = 0, \ldots, N-1$, see Fig. 1.3(b). *Path generation* requires that a coupler point traces through a set of points, $P_j$, $j = 0, \ldots, N-1$, see Fig. 1.3(c). In each case, the task specification is defined by $N$ control points. The concepts of timed motion generation and timed path generation have been studied by researchers too [1, 6, 7], where additional requirements are placed on the input crank angle.

Since the six-bar linkages are capable of producing more complex motions than a four-bar linkage, the design of six-bar linkages is considerably more difficult than four-bar linkages. This is mainly due to the high degree of six-bar synthesis equations. For example, the dimensions of a four-bar function generator that coordinates a maximum of five pairs of angles can be found by solving a polynomial system that has 4 roots. On the other hand, the Stephenson II function generator is capable of coordinating eleven pairs of angles where the synthesis equations have a root count ceiling of 264,241,152. The root count is reported as a ceiling because the exact number of roots for the synthesis equations is unknown. However, in this work we numerically estimate that root count to be near 1.5 million. This illustrates the increased complexity moving from four-bar design to six-bar design.

Similar stories emerge when four-bars and six-bars are compared for motion generation and path generation as well. In the paragraph above, the root count ceiling is called the *multihomogeneous degree*, or *Bézout number*. Its computation is combinatoric in nature and represents the maximum number of a roots a polynomial system may have. The estimated root count is found via numeric work applying homotopy algorithms and is referred to as the *homotopy reduced count*. We list the multihomogeneous degrees and homotopy reduced counts reported in this dissertation for various six-bar synthesis problems in Table 1.1.

The synthesis problems explored in this dissertation are function generation, motion generation, and path generation of six-bar linkages. Function generation is presented for the

Table 1.1: A summary of root counts for various synthesis problems.

| Type | Task | Positions | Multihomogeneous degree | Homotopy reduced count |
|---|---|---|---|---|
| Watt II | Function | 9 | 286,720 | to be determined |
| Stephenson II | Function | 11 | 264,241,152 | 1,521,037 |
| Stephenson III | Function | 11 | 55,050,240 | 834,441 |
| Watt I | Motion | 6 | 1,998,720 | 5,735 |
| Watt I | Motion | 8 | 19,447,142,400 | to be determined |

Watt II, Stephenson II, and Stephenson III linkages because these three types of six-bars provide additional capabilities over four-bar linkages. The Watt II is capable of coordinating nine angle pairs and the Stephenson II and III are capable of coordinating eleven angle pairs while the four-bar coordinates five.

Motion generation is presented for the Watt I linkage as it is the only type that can move its end-effector through more task positions that the four-bar. The Watt I can move through eight task positions while the four-bar moves through five.

Path generation is handled differently because of the size of these problems. A six-bar path generator can move through as many as 15 points compared to 9 for the four-bar where the Watt I and Stephenson I–III linkages all provide additional capabilities from the four-bar. In this dissertation, we synthesize for Stephenson I–III linkages capable of moving through 11 points where instead of solving the path generation problem directly, we invert it to function generation of the Stephenson II and III linkages.

The synthesis equations for all of the problems formulated in this dissertation are systems of polynomials. This is the case because all planar linkages are composed of revolute and prismatic constraints between links, that is circles or lines, which are algebraic curves. The roots of these polynomial systems represent linkage design candidates to accomplish the specified task. Although the synthesis equations require each design candidate to produce each control point, things can go wrong in between. That is a number of defects can occur in the

motion of a design candidate that make it unusable, such as branch defects, circuit defects, and order defects. In this dissertation, we invest much effort into removing candidates with branch and circuit defects from our synthesis results, which is the majority of candidates for all examples reported in this work. Furthermore, because of the large number of synthesis results, the methods described in this dissertation are essentially useless without removing defects via a *performance verification* stage.

## 1.3 Polynomial Homotopy Continuation

*Polynomial homotopy continuation*, or simply *homotopy*, refers to a set of solution methods capable of solving high degree polynomial systems. They are based on deforming easily solved start systems into the desired target systems while tracking roots along the way. The six-bar synthesis equations presented in this dissertation pose formidable computational challenges because of the large root counts of each polynomial system. These challenges are overcome by implementing powerful homotopy algorithms made available by the software package Bertini [8, 9], which are run on high performance computing clusters in order to obtain large solution sets.

Once obtained, the advantage of solving these high degree polynomial systems becomes apparent. That is since only a small percentage of design candidates are useful due to linkage defects, a large number of design candidates are needed in order to generate a sufficient pool of useful linkage designs. To illustrate, we preview the results of the numerical example of a Stephenson III function generator presented in Section 5.4. In order to design a specific function, a homotopy reduced system of 834,441 roots was solved where 96 defect-free linkages were found after performance verification. That means 0.01% of roots resulted in useful designs. The percentage is low but the final pool of designs provides a good amount of choices to a linkage designer.

## 1.4 Literature Review

### 1.4.1 Function Generation

Early work on the design of function generating linkages includes a patent written by Svoboda (1944) [10] who designed a Watt II function generator using methods outlined in Svoboda (1948) [11]. Freudenstein (1954, 1959) [12, 13] introduced a new approach that used the loop equations of a four-bar linkage to fit a given set of accuracy points to obtain a four-bar function generator. Also see Hartenberg and Denavit (1964) [14]. McLarnan (1963) [15] formulated the synthesis equations for the Watt II, Stephenson II, and Stephenson III function generators, and used the Newton-Raphson method to solve for mechanisms capable of moving through eight precision points on an IBM 704 computer. Ogawa (1965) [16] formulated the algebraic synthesis equations of the Stephenson III and solved for an eight position function generator. Mohan Rao et al. (1971) [17] used principles of Burmester theory to design six-bar linkages that perform simultaneous function and path generation.

Dhingra et al. (1994) [18] applied multihomogeneous homotopy to the design of Watt II, Stephenson II, and Stephenson III function generators and solved for linkages capable of coordinating nine positions. Li and Yong (2002) [19] applied homotopy methods to the design of eleven position Stephenson III function generators. They concluded that with their formulation and computing power they would be able to track all homotopy paths in 716 days. Kinzel et al. (2007) [20] took a different approach using a graphical method called geometric constraint programming to design a Stephenson III linkage capable of coordinating eleven pairs of angles. In order to find larger solution sets, Luo and Dai (2007) [21] have applied the "patterned bootstrap" method and Luo et al. (2011) [22] have applied the "hyper-chaos Newton downhill" method to the design of Stephenson III function generators capable of coordinating eleven positions. Plecnik and McCarthy (2014) [23] applied homotopy algorithms to the design of Watt II, Stephenson II, and Stephenson III function

generators finding near-complete solution sets to the eight position problems.

Much of the recent work on six-bar synthesis makes use of optimization theory and related algorithms. Shiakolas et al. (2005) [24] apply differential evolution to the design of six-bar dwell linkages. Hwang and Chen (2010) [25] formulate the optimal synthesis of Stephenson II function generators such that order, circuit, and branch defects are avoided. Sancibrian (2011) [26] demonstrated applying the Generalized Reduced Gradient method to the optimal design of function generating linkages. R. R. Bulatović et al. (2013) [27] introduce the Cuckoo Search algorithm to design Stephenson III linkage.

### 1.4.2 Motion Generation

Motion generation for the four-bar linkage for a finite number of task positions is outlined in McCarthy and Soh (2010) [4]. Early work on the design of six-bar motion generators includes Kaufman (1971) [28] who designed for five task positions with prescribed timing of the input crank to yield as many as 36 designs. Chase et al. (1987) [29] introduced the triad, a connected string of three vectors, to design six-bar linkages for five positions. Lin and Erdman (1987) [30] extend triad synthesis to six positions with prescribed timing. Bawab et al. (1996) [31] decompose a Watt I six-bar into dyad and triad vector groups to synthesize defect-free motion generators.

Schreiber et al. (2002) [32] combine a circlepoint search with homotopy methods to design a Stephenson motion generator for five positions. Soh and McCarthy (2008) [33] specify a three degree-of-freedom serial chain then solve for mechanical constraints to create either Watt I, Stephenson I, II, or III linkages capable of moving through five task positions. Also see Plecnik and McCarthy (2013) [34]. Zhao et al. (2013) [35] present a unified method that simultaneously solves for four- and six-bar motion generators capable of moving through five task positions. Furthermore, their design method can solve for either revolute or prismatic

joints by considering the homogeneous version of a circle constraint. Plecnik and McCarthy (2014) [36] formulate the four position synthesis of a Watt I chain where ground pivots and moving pivots are free to be chosen, and a homotopy solution generates up to 1,344 candidate linkages.

### 1.4.3 Path Generation

Path generation was the topic of discussion in some of the early pioneering work in the computational design of linkages. Freudenstein and Sandor (1959, 1961) [13, 37] used a digital computer to design four-bars that guide a point through five positions on a desired trajectory. Roth and Freudenstein (1963) [38] developed the "bootstrap" method for path generating geared five-bar linkages, and later Wampler et al. (1992) [39] used homotopy to find the complete solution set to the four-bar path generation synthesis equations.

Hain (1967) [40] described graphical methods for point path synthesis using six-bar linkages. Kim et al. (1972) [41] formulated the design equations for Watt I and Stephenson I, II and III path generators and found solutions using an optimization technique. Prasad and Bagci (1974) [42] applied a Gaussian relaxation technique, and Bhatia and Bagci (1977) [43] applied a linear partition technique for the error minimization of Stephenson linkages. Liu and McPhee (2007) [44] used genetic algorithms to find path generating six-bar linkages, and Cabrera et al. (2011) [45], Peñuñuri et al. (2011) [46], and Bulatović and Đorđević (2012) [47] have designed six-bar linkages to generate a path using various differential evolution algorithms.

## 1.5 Contribution

In this dissertation, we formulate various synthesis equations and solve them directly using homotopy algorithms made available by the software package BERTINI [8, 9]. Chapter 2 describes the mathematics behind the synthesis formulations, homotopy, and the ensuing linkage analyses. The contributions of this dissertation are the near-complete solution sets generated by the homotopy solver. These solution sets have never been achieved before.

Chapters 3–5 describe the synthesis of Watt II, Stephenson II, and Stephenson III function generators. Homotopy methods were applied for the solution of Stephenson II and III function generators that coordinate eleven pairs of angles. In Chapter 4, the synthesis equations for the Stephenson II are formulated and a multihomogeneous degree of 264,241,152 is calculated, however, instead of tracking 264,241,152 homotopy paths we implement a regeneration homotopy which tracks 24,822,328 paths over 10 levels resulting in 1,521,037 finite nonsingular solutions. This solution set represents a numerical reduction of the system and an estimate of the true number of roots to the system. This chapter follows the work of Plecnik and McCarthy (2015) [48, 49].

The synthesis equations for the Stephenson III function generator are formulated in Chapter 5 and a multihomogeneous degree of 55,050,240 is calculated. In this case, we track 55,050,240 homotopy startpoints, numerically reducing the system to an estimated 834,441 roots. This chapter follows the work of Plecnik and McCarthy (2015) [50].

Chapter 6 formulates and solves the synthesis equations for the Watt I motion generator. For the maximum number of eight task positions, a multihomogeneous degree over 19 billion is calculated. The eight position equations are not fully solved but represent a good candidate to apply regeneration homotopy. A complete solution set is computed for six task positions using regeneration to discover an estimated 5,735 roots. This chapter follows the work of Plecnik et al. (2014) [51].

Chapter 7 presents eleven position synthesis for all types of Stephenson path generators. The synthesis is accomplished by inverting the problem of path generation to function generation by formulating the synthesis equations as the mechanical control of an RR chain. In other words, Stephenson II and III function generators are attached to RR chains, constraining them to a single degree of freedom. There are four ways to attach these six-bar function generators. This synthesis method represents a novel contribution and the work here is described in Plecnik and McCarthy (2015) [52].

# Chapter 2

# Mathematical Foundations

## 2.1 Complex Numbers

Complex numbers naturally represent planar kinematics. Instead of placing the coordinates $(x, y)$ of a point $p$ into the vector $\mathbf{p} = \{x, y\}^T$, we represent $p$ as a complex number,

$$p = x + yi. \tag{2.1}$$

Similar to $\mathbb{R}^2$, the real and imaginary parts of a complex numbers add component-wise. The multiplication of complex numbers performs rotation and scaling operations. An example of a rotation of $p$ is when multiplied by $e^{i\theta}$,

$$e^{i\theta}p = (\cos\theta + i\sin\theta)(x + yi) = (x\cos\theta - y\sin\theta) + i(x\sin\theta + y\cos\theta) \tag{2.2}$$

which is analogous to the multiplication of a rotation matrix $[R(\theta)]$ by vector $\mathbf{p}$,

$$[R(\theta)]\mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{Bmatrix} \tag{2.3}$$

Therefore, the complex number $e^{i\theta}$ operates to rotate $p$ in the complex plane and is called a rotation operator. Note that a complex number is a rotation operator if and only if it has unit magnitude.

Furthermore, all complex numbers can be written in polar form $Me^{i\theta}$ where $M$ is its magnitude and $\theta$ is its angle from the real axis also known as its argument. As an operator, $Me^{i\theta}$ scales by $M$ and rotates by $\theta$, and is called a stretch-rotation operator.

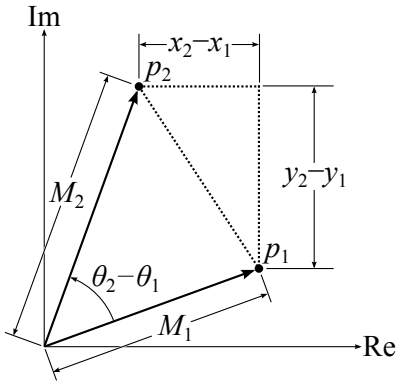



Figure 2.1: The geometric interpretation of the distance between two complex numbers.

The conjugate of a complex number is denoted with an overbar,

$$\bar{p} = x - yi = Me^{-i\theta}. \tag{2.4}$$

For two complex numbers $p_1$ and $p_2$ that are expressed in Cartesian and polar form as

$$\begin{aligned} p_1 &= x_1 + y_1 i = M_1 e^{i\theta_1}, \\ p_2 &= x_2 + y_2 i = M_2 e^{i\theta_2}, \end{aligned} \tag{2.5}$$

the squared distance between them is expressed as

$$(p_2 - p_1)(\bar{p}_2 - \bar{p}_1) = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{2.6}$$

or

$$(p_2 - p_1)(\bar{p}_2 - \bar{p}_1) = M_1^2 + M_2^2 - 2M_1M_2\cos(\theta_2 - \theta_1) \tag{2.7}$$

where the former follows the Pythagorean theorem and the later follows the law of cosines, see Fig. 2.1. The rotation operator associated with the angle between $p_1$ and $p_2$ is computed as

$$e^{i(\theta_2-\theta_1)} = \sqrt{\frac{p_2\bar{p}_1}{\bar{p}_2 p_1}}. \tag{2.8}$$

The complex number $p = x + yi$ represents the coordinates $(x, y)$, however, Wampler (1996) [53] shows that it is beneficial to consider "isotropic coordinates" $(p, \bar{p})$ instead. The $x$-$y$ coordinates are retrieved from isotropic coordinates using a linear transformation,

$$x = \frac{1}{2}(p + \bar{p}), \qquad y = \frac{1}{2i}(p - \bar{p}). \tag{2.9}$$

### 2.1.1 Transformation of a Line Segment

In this section we describe a particular transformation of complex numbers, that is the transformation which moves the line segment defined by endpoints $A$ and $B$ to coincide with a line segment defined by endpoints $A'$ and $B'$, see Fig. 2.2. This transformation is used several times in this dissertation, in particular to compute the scaling, rotation, and translation operations that move a function generator in the plane. This is useful because the coordination of function generation angles is preserved before and after these operations.

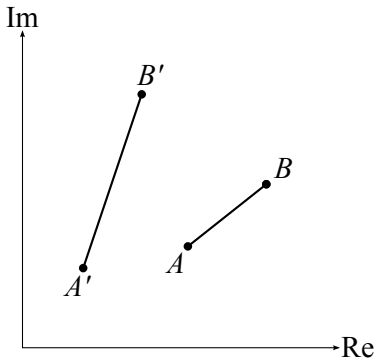



Figure 2.2: Two line segments in the complex plane.

The objective is to transform $line(AB)$ such that it coincides with $line(A'B')$. First, we translate $line(AB)$ to the origin,

$$\mathcal{T}_1(p) = p - A, \tag{2.10}$$

where $p$ is a generic point somewhere along $line(AB)$. Next, we rotate the result so that it is parallel to $line(A'B')$,

$$\mathcal{T}_2(p) = \sqrt{\frac{(B' - A')(\bar{B} - \bar{A})}{(\bar{B}' - \bar{A}')(B - A)}}(p - A), \tag{2.11}$$

where the rotation operator is defined from Eqn. (2.8). Next, we scale the result to the length of $line(A'B')$,

$$\mathcal{T}_3(p) = \frac{|B' - A'|}{|B - A|}\sqrt{\frac{(B' - A')(\bar{B} - \bar{A})}{(\bar{B}' - \bar{A}')(B - A)}}(p - A). \tag{2.12}$$

Finally, we translate the result so that it lies on top of $line(A'B')$,

$$\mathcal{T}(p) = \frac{|B' - A'|}{|B - A|}\sqrt{\frac{(B' - A')(\bar{B} - \bar{A})}{(\bar{B}' - \bar{A}')(B - A)}}(p - A) + A'. \tag{2.13}$$

Transformation $\mathcal{T}(p)$ is simplified by expanding the magnitudes according to Eqn. (2.6) to

obtain

$$\mathcal{T}(p) = \frac{B' - A'}{B - A}(p - A) + A'. \tag{2.14}$$

Eqn. (2.14) can be applied to a linkage with ground link $AB$ in order to move and scale it in the plane while maintaining link length ratios.

## 2.2 Loop Equations

The synthesis and analysis equations presented in this dissertation are based off tracing vector loops. The algebraic elimination of unknown rotation operators from the loop equations forms what are known as constraint equations. To demonstrate this, we begin with the simplest case of a single crank. Then we demonstrate formulating synthesis equations for a four-bar linkage.

### 2.2.1 Synthesis of a Crank

#### Motion Generation: Finding a Pole

The objective of motion generation for a crank is identical to finding the pole between two moving frames. Say we have two task positions $\Gamma_j$, $j = 0, 1$, where each task position is defined by coordinates $(P_j, T_j) = (x_j + y_j i,\ e^{i\theta_j})$, the objective is to find a crank with pivot centered at $A = A_x + A_y i$ such that it can rotate a task frame from $\Gamma_0$ to $\Gamma_1$. Without loss of generality, we can require $T_0 = 1$, that is $\theta_0 = 0$, so that $T_1$ measures the difference in orientation from $\Gamma_0$ to $\Gamma_1$.

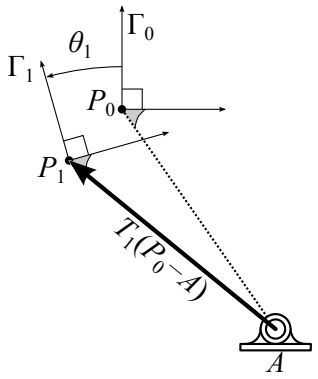



Figure 2.3: A crank placed at the pole of two task positions.

In order to find the location of $A$ we formulate the loop equation,

$$A + T_1(P_0 - A) - P_1 = 0 \tag{2.15}$$

by examining Fig. 2.3. Eqn. (2.15) is linear and can be solved for $A$ to find this pivot, also known as the pole between two task positions.

**Path Generation: Defining a Circle**

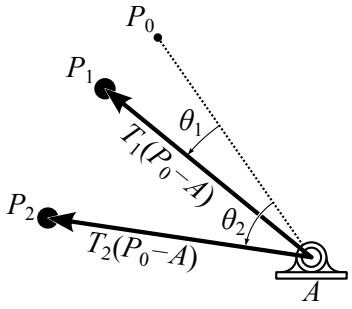



Figure 2.4: A crank placed at the center of a circle defined by three points.

The synthesis of a path generating crank is identical to finding a circle that passes through three points, $P_j$, $j = 0, 1, 2$. This is accomplished using the same loop equation as above,

$$A + T_j(P_0 - A) - P_j = 0, \qquad j = 1, 2, \tag{2.16}$$

except now we have an additional equation for $j = 2$, and $T_1$, $T_2$ are unknowns in addition to $A$. Furthermore, since $T_1$ and $T_2$ are rotation operators, they must have unit magnitude, that is

$$T_j \bar{T}_j = 1, \qquad j = 1, 2. \tag{2.17}$$

Next, we append the conjugate loop equation

$$\bar{A} + \bar{T}_j(\bar{P}_0 - \bar{A}) - \bar{P}_j = 0, \qquad j = 1, 2, \tag{2.18}$$

to form a system of six equations (2.16)–(2.18) in the six unknowns $\left\langle A, \bar{A}, T_1, \bar{T}_1, T_2, \bar{T}_2 \right\rangle$. Eqns. (2.16) and (2.18) can be solved for $T_j$ and $\bar{T}_j$ and subsituted into Eqn. (2.17) to form

$$(P_0 - A)(\bar{P}_0 - \bar{A}) = (P_j - A)(\bar{P}_j - \bar{A}), \qquad j = 1, 2. \tag{2.19}$$

Expansion of Eqn. (2.19) cancels out the $A\bar{A}$ terms and reveals a set of two linear equations in two unknowns $(A, \bar{A})$ of which the solution is the center of a circle that passes through points $P_j$, $j = 0, 1, 2$.

**Constraint Equation of a Crank**

As well, note that Eqn. (2.19) is the constraint equation of a circle which can be seen by substituting

$$A = A_x + A_y i, \qquad P_0 = x_0 + y_0 i, \qquad P_j = x_j + y_j i, \tag{2.20}$$

into Eqn. (2.19) to obtain

$$(x_j - A_x)^2 + (y_j - A_y)^2 = r^2, \tag{2.21}$$

where $(A_x, A_y)$ is the center of the circle and $r = \sqrt{(x_0 - A_x)^2 + (y_0 - A_y)^2}$ is the radius. By dropping the $j$ indices, we algebraically express the trace curve

$$\mathcal{C}(x, y) = 0 = (x - A_x)^2 + (y - A_y)^2 - r^2, \tag{2.22}$$

of a crank in the $x$-$y$ plane. It is much simpler than the coupler curve of a four-bar which is described in Section 2.2.3.

### 2.2.2 Motion Generation of a Four-bar

In Section 2.2.1, we introduced the use of loop equations to design cranks. In this section we extend these concepts to the design of four-bar linkages. The objective of motion generation is to design a four-bar linkage that can move an end-effector attached to its coupler link through $N$ task positions $\Gamma_j = (P_j, T_j) = (x_j + y_j i,\ e^{i\theta_j})$, $j = 0, \ldots, N-1$. In this case, we must find the locations of its two ground pivots $A = A_x + A_y i$ and $B = B_x + B_y i$ and two moving pivots $C = C_x + C_y i$ and $D = D_x + D_y i$. The locations $C$ and $D$ represent moving pivots in position $j = 0$, which when combined with task position $\Gamma_0$, completely defines how the end effector is attached to the coupler link.

A four-bar linkage has three moving links $AC$, $BD$, and $CDP$. The orientation of each link in the $j^{\text{th}}$ position is measured by angles $\phi_j$, $\psi_j$, and $\theta_j$, respectively. These three angles define complex rotation operators $Q_j$, $S_j$, $T_j$,

$$Q_j = e^{i\phi_j}, \qquad S_j = e^{i\psi_j}, \qquad T_j = e^{i\theta_j}, \qquad j = 0, \ldots, N-1. \tag{2.23}$$

Without loss of generality, we define the initial orientations of each link as $Q_0 = S_0 = T_0 = 1$, that is $\phi_0 = \psi_0 = \theta_0 = 0$, such that $Q_j$, $S_j$, $T_j$ measure the change in orientation in the $j^{\text{th}}$ configuration from reference configuration $j = 0$.

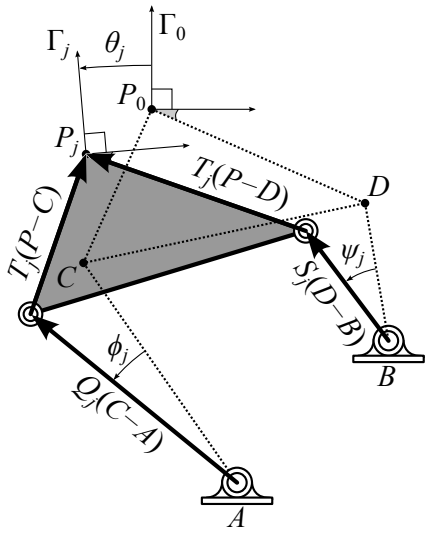



Figure 2.5: A four-bar motion generator displaced to position $j$.

In order to find pivot locations, we examine Fig. 2.5 to form two loop equations for each position,

$$A + Q_j(C - A) + T_j(P_0 - C) - P_j = 0, \tag{2.24}$$

$$B + S_j(D - B) + T_j(P_0 - D) - P_j = 0, \qquad j = 1, \ldots, N-1, \tag{2.25}$$

where the loop equations for reference configuration $j = 0$ are omitted as they are identically zero. The unknown values in Eqns. (2.24) and (2.25) are $A$, $B$, $C$, $D$, and $Q_j$, $S_j$, $j = 1, \ldots, N-1$. Furthermore, the rotation operators $Q_j$ and $S_j$ are required to have unit magnitude,

$$Q_j \bar{Q}_j = 1, \tag{2.26}$$

$$S_j \bar{S}_j = 1, \qquad j = 1, \ldots, N-1. \tag{2.27}$$

As well, we append the conjugate loop equations,

$$\bar{A} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{T}_j(\bar{P}_0 - \bar{C}) - \bar{P}_j = 0, \tag{2.28}$$

$$\bar{B} + \bar{S}_j(\bar{D} - \bar{B}) + \bar{T}_j(\bar{P}_0 - \bar{D}) - \bar{P}_j = 0, \qquad j = 1, \ldots, N-1, \tag{2.29}$$

Eqns. (2.24)–(2.29) form a system of $6(N-1)$ equations in $8 + 4(N-1)$ unknowns,

$$\left\langle A, \bar{A}, B, \bar{B}, C, \bar{C}, D, \bar{D} \right\rangle \quad \text{and} \quad \left\langle Q_j, \bar{Q}_j, S_j, \bar{S}_j \right\rangle, \qquad j = 1, \ldots, N-1, \tag{2.30}$$

which is square for $N = 5$ task positions. Furthermore, Eqns. (2.24), (2.26), (2.28), and Eqns. (2.25), (2.27), (2.29) form two independent systems of the exact same structure in the unknowns $\left\langle A, \bar{A}, C, \bar{C}, Q_j, \bar{Q}_j \right\rangle$ and $\left\langle B, \bar{B}, D, \bar{D}, S_j, \bar{S}_j \right\rangle$, respectively. Therefore, we only need to solve one of these systems, which is known as the synthesis of an RR constraint. Combining two RR constraints forms a four-bar linkage and provides a solution to the entire system Eqns. (2.24)–(2.29).

**Synthesis of an RR Constraint**

Eqns. (2.24), (2.26), (2.28) are the synthesis equations of an RR constraint. The unknowns $(Q_j, \bar{Q}_j)$ can be eliminated by solving (2.24) and (2.28) for $Q_j$ and $\bar{Q}_j$, and substituting into (2.26) to obtain,

$$(C-A)(\bar{C}-\bar{A}) = (A+T_j(P_0-C)+P_j)(\bar{A}+\bar{T}_j(\bar{P}_0-\bar{C})+\bar{P}_j), \qquad j = 1, \ldots, N-1, \tag{2.31}$$

which represents $N-1$ equations in four unknowns $\left\langle A, \bar{A}, C, \bar{C} \right\rangle$. Eqns. (2.31) are bilinear, and for the case $N = 5$, are known to have four solutions, which can be combined in six ways to form four-bar linkages.

### 2.2.3 Path Generation of a Four-bar

The objective of path generation is to find a four-bar linkage that can move a trace point attached to its coupler link through $N$ points $P_j$, $j = 1, \ldots, N-1$. The loop equations are the same as for motion generation (Eqns. (2.24), (2.25), (2.28), (2.29)) and are repeated here,

$$A + Q_j(C - A) + T_j(P_0 - C) - P_j = 0, \tag{2.32}$$

$$B + S_j(D - B) + T_j(P_0 - D) - P_j = 0, \tag{2.33}$$

$$\bar{A} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{T}_j(\bar{P}_0 - \bar{C}) - \bar{P}_j = 0, \tag{2.34}$$

$$\bar{B} + \bar{S}_j(\bar{D} - \bar{B}) + \bar{T}_j(\bar{P}_0 - \bar{D}) - \bar{P}_j = 0, \qquad j = 1, \ldots, N-1. \tag{2.35}$$

Differently from motion generation, $(T_j, \bar{T}_j)$ are now regarded as unknowns which means Eqns. (2.32)–(2.35) cannot be split into two independent systems and must be solved simultaneously. Furthermore, we require unit magnitude of the additional unknowns to form the normalization conditions,

$$Q_j \bar{Q}_j = 1, \tag{2.36}$$

$$S_j \bar{S}_j = 1, \tag{2.37}$$

$$T_j \bar{T}_j = 1, \qquad j = 1, \ldots, N-1. \tag{2.38}$$

Eqns. (2.32)–(2.38) form $7(N-1)$ equations in $8+6(N-1)$ unknowns which is square when $N = 9$ maximum positions $P_j$.

The loop equations can be transformed into constraint equations by algebraic elimination. First, Eqns. (2.32) and (2.34) are solved for $Q_j$, $\bar{Q}_j$ and substituted into Eqn. (2.36), and

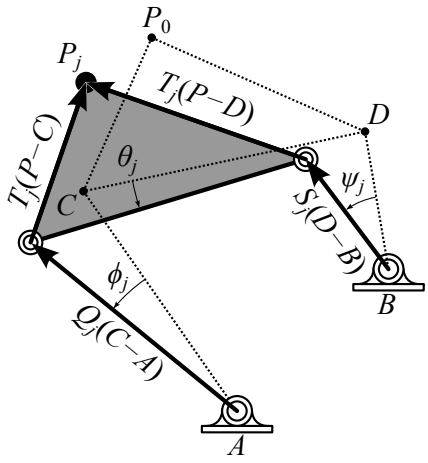



Figure 2.6: A four-bar path generator displaced to position $j$.

Eqns. (2.33) and (2.35) are solved for $S_j$, $\bar{S}_j$ and substituted into Eqn. (2.37) to obtain

$$(C-A)(\bar{C}-\bar{A}) = (A-P_j+T_j(P_0-C))(\bar{A}-\bar{P}_j+\bar{T}_j(\bar{P}_0-\bar{C})), \tag{2.39}$$

$$(D-B)(\bar{D}-\bar{B}) = (B-P_j+T_j(P_0-D))(\bar{B}-\bar{P}_j+\bar{T}_j(\bar{P}_0-\bar{D})), \tag{2.40}$$

$$j = 1, \ldots, N-1,$$

which eliminates $Q_j$, $\bar{Q}_j$, $S_j$, and $\bar{S}_j$. Eqns. (2.38)–(2.40) form $3(N-1)$ equations in $8+2(N-1)$ unknowns $\langle A, \bar{A}, B, \bar{B}, C, \bar{C}, D, \bar{D}\rangle$ and $\langle T_j, \bar{T}_j\rangle$, $j = 1, \ldots, N-1$. Eqns. (2.39) and (2.40) can be expanded and written in matrix form as

$$\begin{bmatrix} a\bar{b}_j & \bar{a}b_j \\ c\bar{d}_j & \bar{c}d_j \end{bmatrix} \begin{Bmatrix} T_j \\ \bar{T}_j \end{Bmatrix} = \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - a\bar{a} - d_j\bar{d}_j \end{Bmatrix}, \tag{2.41}$$

where

$$\begin{aligned} a &= P_0 - C, \qquad & b_j &= A - P_j, \qquad & f &= C - A, \\ c &= P_0 - D, \qquad & d_j &= B - P_j, \qquad & g &= D - B. \end{aligned} \tag{2.42}$$

Eqn. (2.41) can be solved for $T_j$, $\bar{T}_j$ and substituted into Eqn. (2.38) to obtain

$$\begin{aligned}&(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j))\\&\quad\times(\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0.\end{aligned} \tag{2.43}$$

which eliminates $T_j$, $\bar{T}_j$ and forms $N-1$ equations in eight unknowns $\langle A, \bar{A},\ B, \bar{B},\ C, \bar{C},\ D, \bar{D}\rangle$. For the case N=9, these equations are known to have 8,652 solutions that appear in a six-way symmetry group which can be found by homotopy methods, Wampler (1992) [39]. The symmetry exists because points $A$ and $B$, and points $C$ and $D$ can be interchanged to find another solution, and furthermore, each will solution will describe one of three cognate linkages that create the same coupler path.

**Constraint Equation of a Four-bar**

Furthermore, Eqn. (2.43) represents a constraint on the motion of trace point $P$. This is made clear by expanding (2.43) and abandoning the $j$ indices of $P_j$,

$$\begin{aligned}&\mathcal{C}(P,\bar{P}) = 0 =\\&\quad\Big((P_0 - C)(\bar{A} - \bar{P})\xi - (P_0 - D)(\bar{B} - \bar{P})\beta\Big)\Big((\bar{P}_0 - \bar{C})(A - P)\xi - (\bar{P}_0 - \bar{D})(B - P)\beta\Big)\\&\quad+\Big((P_0 - C)(\bar{A} - \bar{P})(\bar{P}_0 - \bar{D})(B - P) - (\bar{P}_0 - \bar{C})(A - P)(P_0 - D)(\bar{B} - \bar{P})\Big)^2\\&\text{where}\quad \beta = (C - A)(\bar{C} - \bar{A}) - (P_0 - C)(\bar{P}_0 - \bar{C}) - (A - P)(\bar{A} - \bar{P})\\&\qquad\ \ \xi = (D - B)(\bar{D} - \bar{B}) - (P_0 - D)(\bar{P}_0 - \bar{D}) - (B - P)(\bar{B} - \bar{P}).\end{aligned} \tag{2.44}$$

Eqn. (2.44) is of degree six with respect to $P$, $\bar{P}$ which means the trace point of a four-bar is constrained to move on a sextic curve defined by its dimensions. The four-bar coupler curve can be written as a constraint in the $x$-$y$ plane by expanding real and imaginary components

$$
\begin{aligned}
&A = A_x + A_y i, \qquad B = B_x + B_y i, \qquad C = C_x + C_y i, \qquad D = D_x + D_y i, \\
&\qquad\qquad P_0 = x_0 + y_0 i, \qquad P = x + y i.
\end{aligned}
\tag{2.45}
$$

to obtain

$$
\begin{aligned}
&\mathcal{C}(x,y) = 0 = \\
&\Big((x_0 - C_x)^2 + (y_0 - C_y)^2\Big)\Big((A_x - x)^2 + (A_y - y)^2\Big)\xi^2 \\
&+\Big((x_0 - D_x)^2 + (y_0 - D_y)^2\Big)\Big((B_x - x)^2 + (B_y - y)^2\Big)\beta^2 \\
&-2\Bigg(\Big((x_0 - C_x)(A_x - x) + (y_0 - C_y)(A_y - y)\Big)\Big((x_0 - D_x)(B_x - x) + (y_0 - D_y)(B_y - y)\Big) \\
&+\Big((x_0 - C_x)(A_y - y) - (y_0 - C_y)(A_x - x)\Big)\Big((x_0 - D_x)(B_y - y) - (y_0 - D_y)(B_x - x)\Big)\Bigg)\beta\xi \\
&-4\Bigg(\Big((x_0 - C_x)(A_x - x) + (y_0 - C_y)(A_y - y)\Big)\Big((x_0 - D_x)(B_y - y) - (y_0 - D_y)(B_x - x)\Big) \\
&-\Big((x_0 - C_x)(A_y - y) - (y_0 - C_y)(A_x - x)\Big)\Big((x_0 - D_x)(B_x - x) + (y_0 - D_y)(B_y - y)\Big)\Bigg)^2 \\
&\text{where} \quad \beta = (C_x - A_x)^2 + (C_y - A_y)^2 - (x_0 - C_x)^2 - (y_0 - C_y)^2 - (A_x - x)^2 - (A_y - y)^2 \\
&\qquad\quad \xi = (D_x - B_x)^2 + (D_y - B_y)^2 - (x_0 - D_x)^2 - (y_0 - D_y)^2 - (B_x - x)^2 - (B_y - y)^2,
\end{aligned}
\tag{2.46}
$$

which is analogous to Eqn. (2.22) for the circle of a crank but substantially more complicated. Eqn. (2.46) can be written compactly by introducing the vector notation,

$$
\begin{aligned}
&\mathbf{A} = \begin{Bmatrix} A_x \\ A_y \end{Bmatrix}, \ \mathbf{B} = \begin{Bmatrix} B_x \\ B_y \end{Bmatrix}, \ \mathbf{C} = \begin{Bmatrix} C_x \\ C_y \end{Bmatrix}, \ \mathbf{D} = \begin{Bmatrix} D_x \\ D_y \end{Bmatrix}, \ \mathbf{P}_0 = \begin{Bmatrix} x_0 \\ y_0 \end{Bmatrix}, \ \mathbf{P} = \begin{Bmatrix} x \\ y \end{Bmatrix}, \\
&\mathbf{a} = \mathbf{P}_0 - \mathbf{C}, \qquad \mathbf{b} = \mathbf{A} - \mathbf{P}, \qquad \mathbf{f} = \mathbf{C} - \mathbf{A}, \\
&\mathbf{c} = \mathbf{P}_0 - \mathbf{D}, \qquad \mathbf{d} = \mathbf{B} - \mathbf{P}, \qquad \mathbf{g} = \mathbf{D} - \mathbf{B}.
\end{aligned}
\tag{2.47}
$$

to obtain

$$\begin{aligned}
&\mathcal{C}(x,y) = 0 = \\
&\quad (\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{b})\xi^2 + (\mathbf{c}\cdot\mathbf{c})(\mathbf{d}\cdot\mathbf{d})\beta^2 - 2\Big((\mathbf{a}\cdot\mathbf{b})(\mathbf{c}\cdot\mathbf{d}) + (\mathbf{a}\times\mathbf{b})(\mathbf{c}\times\mathbf{d})\Big)\beta\xi \\
&\quad -4\Big((\mathbf{a}\cdot\mathbf{b})(\mathbf{c}\times\mathbf{d}) - (\mathbf{a}\times\mathbf{b})(\mathbf{c}\cdot\mathbf{d})\Big)^2 \qquad (2.48)\\
&\text{where} \quad \beta = \mathbf{f}\cdot\mathbf{f} - \mathbf{a}\cdot\mathbf{a} - \mathbf{b}\cdot\mathbf{b} \\
&\qquad\quad \xi = \mathbf{g}\cdot\mathbf{g} - \mathbf{c}\cdot\mathbf{c} - \mathbf{d}\cdot\mathbf{d}
\end{aligned}$$

and "$\cdot$" is the dot product operator and "$\times$" is an operator to similar to the cross product,

$$\begin{Bmatrix} u_x \\ u_y \end{Bmatrix} \times \begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = u_x v_y - u_y v_x. \qquad (2.49)$$

Fig. 2.7 shows a crank constraint curve and two examples of four-bar constraint curves. Four-bar constraint curves are considerably more complex than circles. They may take many different shapes and can have discontinuous sections that would require disassembly and reassembly of the linkage to realize. A green line is drawn in Fig. 2.7(b) that intersects the coupler curve six times, which is possible because Eqn. (2.48) is a sextic. Six-bar mechanisms produce even more complex curves with many examples drawn in Primrose et al. (1967) [54, 55].

### 2.2.4 Function Generation of a Four-bar

The objective of function generation is to coordinate $N$ joint angles $\phi_j$ and $\psi_j$ of a four-bar linkage for $j = 0, \ldots, N-1$, see Fig. 2.8. A four-bar linkage is defined by the locations of its ground pivots $A$ and $B$, and the locations of its moving pivots $C$ and $D$ in a reference configuration $j = 0$. Angles $\phi_j$, $\psi_j$, and coupler angle $\theta_j$ are represented in the equations

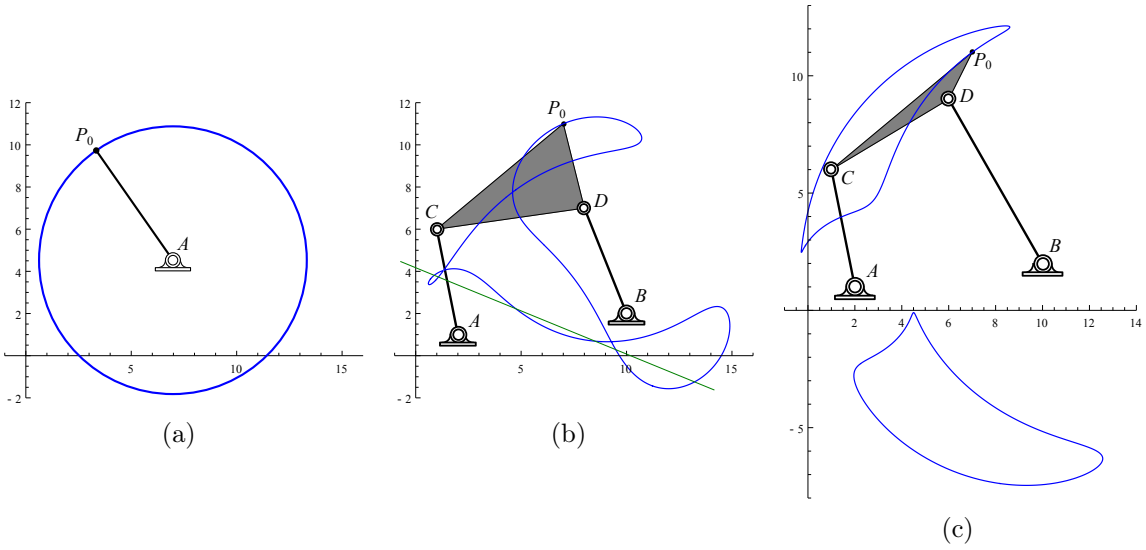



Figure 2.7: (a) A circle constraint, (b) a four-bar constraint with one circuit, and (c) a four-bar constraint with two circuits.

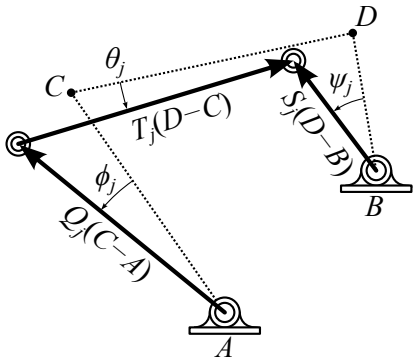



Figure 2.8: A four-bar function generator displaced to position $j$.

below by complex rotation operators $Q_j$, $S_j$, $T_j$,

$$Q_j = e^{i\phi_j}, \qquad S_j = e^{i\psi_j}, \qquad T_j = e^{i\theta_j}, \qquad j = 0, \ldots, N-1. \tag{2.50}$$

Without loss of generality, we define the initial orientations of each link as $Q_0 = S_0 = T_0 = 1$, that is $\phi_0 = \psi_0 = \theta_0 = 0$, such that $Q_j$, $S_j$, $T_j$ measure the change in orientation in the $j^{\text{th}}$ configuration from reference configuration $j = 0$. The loop equations for function generation differ from motion and path generation because we do not keep track of coupler point $P$.

They take the form,

$$A - B + Q_j(C - A) + T_j(D - C) - S_j(D - B) = 0, \qquad j = 1, \ldots, N-1, \tag{2.51}$$

where the loop equation for $j = 0$ is omitted because it is identically zero. In Eqn. (2.51), $Q_j$ and $S_j$ are the specified task and ground pivots $A$ and $B$ are specified in order to set the scale, orientation, and location of the function generator in the plane, leaving $C$, $D$, and $T_j$ as unknowns. Furthermore, since $T_j$ is a rotation operator it must have unit magnitude,

$$T_j \bar{T}_j = 1, \qquad j = 1, \ldots, N-1. \tag{2.52}$$

As well, we append the conjugate loop equations,

$$\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{T}_j(\bar{D} - \bar{C}) - \bar{S}_j(\bar{D} - \bar{B}) = 0, \qquad j = 1, \ldots, N-1, \tag{2.53}$$

Eqns. (2.51)–(2.53) form a system of $3(N-1)$ equations in $4 + 2(N-1)$ unknowns,

$$\left\langle C, \bar{C}, D, \bar{D} \right\rangle \quad \text{and} \quad \left\langle T_j, \bar{T}_j \right\rangle,\ j = 1, \ldots, N-1, \tag{2.54}$$

which is square for $N = 5$ angle pairs. The unknowns $T_j$, $\bar{T}_j$, are eliminated by solving for them in Eqns. (2.51) and (2.53), then substituting into Eqn. (2.52) to obtain

$$(D - C)(\bar{D} - \bar{C}) = (A - B + Q_j(C - A) - S_j(D - B))(\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) - \bar{S}_j(\bar{D} - \bar{B})), \qquad j = 1, \ldots, N-1 \tag{2.55}$$

Eqn. (2.55) are $N-1$ equations in four unknowns $\left\langle C, \bar{C}, D, \bar{D} \right\rangle$ which for the case $N = 5$ has four solutions which are discussed below.

## 2.3 Homotopy

This dissertation uses homotopy continuation to solve highly nonlinear polynomial systems, and in particular the software package BERTINI [8, 9]. Homotopy continuation theory includes the concepts of multihomogeneous homotopy, parameter homotopy, and regeneration homotopy, and makes use of projective geometry, combinatorics, and path tracking algorithms. Rather than describe these concepts generically, we choose to illustrate them on a small example, that is the numerical solution to the synthesis equations of a four-bar function generator, see Eqn. (2.55). Note that closed-form solution methods do exist for this sample problem. However, these methods do not scale to handle the larger problems which are the topic of this dissertation.

### 2.3.1 Overview

The objective is to find all the isolated solutions to the set of quadratic equations shown in Eqn. (2.55). We begin by expanding them to obtain

$$f_j(\mathbf{z}) = q_{3j}C\bar{D} + \bar{q}_{3j}\bar{C}D + q_{2j}C + \bar{q}_{2j}\bar{C} + q_{1j}D + \bar{q}_{1j}\bar{D} + q_{0j} = 0, \qquad j = 1,2,3,4 \quad (2.56)$$

where

$$\begin{aligned}
q_{0j} &= (Q_j\bar{S}_j - 1)A\bar{B} + (\bar{Q}_jS_j - 1)\bar{A}B + (Q_jA - S_jB)(\bar{A} - \bar{B}) \\
&\qquad +(\bar{Q}_j\bar{A} - \bar{S}_j\bar{B})(A - B) - 2(A - B)(\bar{A} - \bar{B}), \\
q_{1j} &= -S_j(\bar{Q}_j - 1)\bar{A} - (S_j - 1)\bar{B}, \qquad & \bar{q}_{1j} &= -\bar{S}_j(Q_j - 1)A - (\bar{S}_j - 1)B, \\
q_{2j} &= -Q_j(\bar{S}_j - 1)\bar{B} - (Q_j - 1)\bar{A}, \qquad & \bar{q}_{2j} &= -\bar{Q}_j(S_j - 1)B - (\bar{Q}_j - 1)A, \\
q_{3j} &= Q_j\bar{S}_j - 1, & \bar{q}_{3j} &= \bar{Q}_jS_j - 1.
\end{aligned} \quad (2.57)$$

The $q$ coefficients are known constants specified by the problem statement and the unknowns to solve for are $\mathbf{z} = \{C, D, \bar{C}, \bar{D}\}$. The quadratic equations $\mathbf{f}(\mathbf{z}) = \{f_1, f_2, f_3, f_4\}$ are referred to as the target system. The general steps of homotopy continuation follow below:

1. Construct an easily solved polynomial system $\mathbf{g}(\mathbf{z})$ that possesses the same or a more general monomial structure as $\mathbf{f}(\mathbf{z})$, then find all solutions of $\mathbf{g}(\mathbf{z})$, called startpoints.
2. Construct a homotopy function $H(\mathbf{z}, t)$ that parameterizes a deformation of $\mathbf{g}(\mathbf{z})$ into $\mathbf{f}(\mathbf{z})$ by $t$.
3. Implement a path tracking method that increments $t$ within $H(\mathbf{z}, t)$ in order to track from startpoints of $\mathbf{g}(\mathbf{z})$ to endpoints which are the solutions of $\mathbf{f}(\mathbf{z})$.

### 2.3.2 Constructing a Start System

The start system $\mathbf{g}(\mathbf{z})$ of a homotopy must possess the same or a more general monomial structure as the target system $\mathbf{f}(\mathbf{z})$ such that the two are members of a family of polynomial systems. By examining the polynomial $f(\mathbf{z})$,

$$f(\mathbf{z}) = q_3 C\bar{D} + \bar{q}_3 \bar{C} D + q_2 C + \bar{q}_2 \bar{C} + q_1 D + \bar{q}_1 \bar{D} + q_0, \tag{2.58}$$

we see that it is a member of the family of polynomials which possess the monomials

$$\left\langle C\bar{D}, \bar{C}D, C, \bar{C}, D, \bar{D}, 1 \right\rangle \tag{2.59}$$

and are parameterized by the seven coefficients

$$\{q_0, q_1, \bar{q}_1, q_2, \bar{q}_2, q_3, \bar{q}_3\} \in \mathbb{C}^7 \tag{2.60}$$

where in this case the overbar notation no longer denotes the conjugate but is meaningless.

**Total-Degree Start System**

The ideal start system would comprise of polynomials $g(\mathbf{z})$ that are members of the same family as $f(\mathbf{z})$, and have solutions which are obtained by some simple means. One way to construct a start system $\mathbf{g}(\mathbf{z})$ is for each equation to be the product of linear polynomials,

$$\mathbf{g}(\mathbf{z}) = \begin{Bmatrix} (b_{01}C + b_{11}D + b_{21}\bar{C} + b_{31}\bar{D} + b_{41})(b_{51}C + b_{61}D + b_{71}\bar{C} + b_{81}\bar{D} + b_{91}) \\ (b_{02}C + b_{12}D + b_{22}\bar{C} + b_{32}\bar{D} + b_{42})(b_{52}C + b_{62}D + b_{72}\bar{C} + b_{82}\bar{D} + b_{92}) \\ (b_{03}C + b_{13}D + b_{23}\bar{C} + b_{33}\bar{D} + b_{43})(b_{53}C + b_{63}D + b_{73}\bar{C} + b_{83}\bar{D} + b_{93}) \\ (b_{04}C + b_{14}D + b_{24}\bar{C} + b_{34}\bar{D} + b_{44})(b_{54}C + b_{64}D + b_{74}\bar{C} + b_{84}\bar{D} + b_{94}) \end{Bmatrix} = 0 \quad (2.61)$$

where all $b$ coefficients are randomly specified complex numbers. Note that each linear polynomial contains every unknown, which constructs the start system for a total-degree homotopy. There are 15 monomials found in Eqn. (2.61) when expanded,

$$\left\langle C^2, D^2, \bar{C}^2, \bar{D}^2, CD, \bar{C}\bar{D}, C\bar{C}, D\bar{D}, C\bar{D}, \bar{C}D, C, D, \bar{C}, \bar{D}, 1 \right\rangle, \quad (2.62)$$

and the solutions are easily obtained since each equation is written as the product of linear polynomials. To see this, rewrite Eqn. (2.61) compactly as,

$$\mathbf{g}(\mathbf{z}) = \begin{Bmatrix} L_1(\mathbf{z})L_2(\mathbf{z}) \\ L_3(\mathbf{z})L_4(\mathbf{z}) \\ L_5(\mathbf{z})L_6(\mathbf{z}) \\ L_7(\mathbf{z})L_8(\mathbf{z}) \end{Bmatrix} = 0 \quad (2.63)$$

such that one linear expression of each product must be equal to zero in order for Eqn. (2.63) to be satisfied. Therefore, the solutions to Eqn. (2.63) must be the unique solutions to the

linear systems,

$$\left\{\begin{array}{c} L_k(\mathbf{z}) \\ L_l(\mathbf{z}) \\ L_m(\mathbf{z}) \\ L_n(\mathbf{z}) \end{array}\right\} = 0, \qquad \begin{array}{r} \text{for } \{k,l,m,n\} = \{1,3,5,7\},\{2,3,5,7\},\{1,4,5,7\},\{1,3,6,7\}, \\ \{1,3,5,8\},\{2,4,5,7\},\{2,3,6,7\},\{2,3,5,8\}, \\ \{1,4,6,7\},\{1,4,5,8\},\{1,3,6,8\},\{2,4,6,7\}, \\ \{2,4,5,8\},\{2,3,6,8\},\{1,4,6,8\},\{2,4,6,8\}, \end{array} \tag{2.64}$$

of which there are 16. This matches the Bézout number of the system $2^4$=16. However, the monomial structure (2.62) of this system is much more general than the monomial structure (2.59) of the target system. The family of polynomials that the target system belongs to is a subset of the family of polynomials that the start system belongs to, where eight coefficients are required to be zero. Eqn. (2.61) is a valid start system, however, there are advantages to creating a start system which is not so general which we illustrate below.

**Multihomogeneous Start System**

A more clever start system can be designed by noting that Eqn. (2.58) is bilinear in terms of the variable groups $\langle C, D\rangle$ and $\langle \bar{C}, \bar{D}\rangle$. Using this knowledge, we construct each $g(\mathbf{z})$ as the linear products,

$$\mathbf{g}(\mathbf{z}) = \left\{\begin{array}{l} (b_{01}C + b_{11}D + b_{21})(b_{31}\bar{C} + b_{41}\bar{D} + b_{51}) \\ (b_{02}C + b_{12}D + b_{22})(b_{32}\bar{C} + b_{42}\bar{D} + b_{52}) \\ (b_{03}C + b_{13}D + b_{23})(b_{33}\bar{C} + b_{43}\bar{D} + b_{53}) \\ (b_{04}C + b_{14}D + b_{24})(b_{34}\bar{C} + b_{44}\bar{D} + b_{54}) \end{array}\right\} = 0 \tag{2.65}$$

where once again the $b$ coefficients are randomly specified complex numbers. The monomial structure of Eqn. (2.65) is

$$\left\langle C\bar{C}, D\bar{D}, C\bar{D}, \bar{C}D, C, D, \bar{C}, \bar{D}, 1\right\rangle \tag{2.66}$$

which only has two more members than (2.59). Eqn. (2.65) is written compactly as

$$\mathbf{g}(\mathbf{z}) = \left\{ \begin{array}{l} L_1(C,D)M_1(\bar{C},\bar{D}) \\ L_2(C,D)M_2(\bar{C},\bar{D}) \\ L_3(C,D)M_3(\bar{C},\bar{D}) \\ L_4(C,D)M_4(\bar{C},\bar{D}) \end{array} \right\} = 0 \tag{2.67}$$

where one linear expression of each product must be zero for Eqn. (2.67) to be true. The solutions are found by pairing linear systems,

$$\left( \left\{ \begin{array}{l} L_k(C,D) \\ L_l(C,D) \end{array} \right\} = 0, \quad \left\{ \begin{array}{l} M_m(\bar{C},\bar{D}) \\ M_n(\bar{C},\bar{D}) \end{array} \right\} = 0 \right) \qquad \begin{array}{r} \text{for } \{k,l,m,n\} = \{1,2,3,4\},\{1,3,2,4\}, \\ \{1,4,2,3\},\{2,3,1,4\}, \\ \{2,4,1,3\},\{3,4,1,2\}, \end{array} \tag{2.68}$$

of which there are six. These six solutions serve as startpoints for the path tracking step and is a reduction from the 16 startpoints found previously, which demonstrates the advantage of examining the monomial structure of the target system, then constructing a start system accordingly. For the high degree systems discussed in Chapters 3–6, this step provides large reductions.

### Computing the Multihomogeneous Degree

The example above demonstrated how the unknowns of a polynomial system can be divided into groups which can be used to count a number of roots lower than the total degree of a system. The root counting scheme is combinatoric in nature and was done manually above, which for larger systems is impractical. However, an algorithmic root counting scheme is available as well which is presented here.

First, variable groupings must be chosen, such as $\langle C, D\rangle$, $\langle \bar{C}, \bar{D}\rangle$. Currently, only heuristic methods exist for finding optimal variable groupings, and the only method for finding the variable grouping which results in the least number of startpoints is to check all possibilities, which is impractical for larger problems. The variable groupings used in the larger problems presented in Chapters 3–6 were found by intuition and checking a handful of possibilities.

Once a variable grouping is chosen, a table is constructed that lists the degree of each equation with respect to each variable group,

$$\begin{array}{c|cc} & \langle C, D\rangle & \langle \bar{C}, \bar{D}\rangle \\ \hline f_1 & d_{1,1} = 1 & d_{1,2} = 1 \\ f_2 & d_{2,1} = 1 & d_{2,2} = 1 \\ f_3 & d_{3,1} = 1 & d_{3,2} = 1 \\ f_4 & d_{4,1} = 1 & d_{4,2} = 1 \end{array} \tag{2.69}$$

Next, we associate each variable group with a dummy variable, say $\alpha_1$ and $\alpha_2$, then construct the polynomial

$$\prod_{j=1}^{4} \sum_{k=1}^{2} d_{j,k}\alpha_k = (\alpha_1 + \alpha_2)^4. \tag{2.70}$$

The root count of $\mathbf{f}(\mathbf{z})$ with respect to the variable grouping is found by expanding Eqn. (2.70) and taking the coefficient in front of the monomial

$$\prod_{k=1}^{2} \alpha_k^{n_k} = \alpha_1^2 \alpha_2^2 \tag{2.71}$$

where $n_k$ is the number of variables in the $k^{\text{th}}$ variable group. This calculation encompasses the combinatorics completed manually in the example above. For the case of a bilinear system with an equal number of variables in each group, this computation simplifies to the binomial coefficient $\binom{N}{N/2}$. The name of the root count calculated with respect to a variable

grouping is called the multihomogeneous degree of that system, or multihomogeneous Bézout number. The reason for this naming is because each group is homogenized as explained in the proceeding section.

### 2.3.3 Constructing a Multihomogeneous Homotopy

The start system displayed in Eqn. (2.65) closely matches the monomial structure of the target system displayed in Eqn. (2.56), but contains two extra monomials, $C\bar{C}$ and $D\bar{D}$. This indicates that there is a good possibility that the target system has less finite roots than the start system, and that some paths may track to infinity. Tracking paths to infinity is computationally expensive and requires a numeric threshold value of infinity to be set. However, the problem can be avoided by homogenizing $\mathbf{f}(\mathbf{z})$ according to each variable group.

#### Projective Coordinates

Homogenizing $\mathbf{f}(\mathbf{z})$ is accomplished by introducing the homogeneous variables $h$ and $\bar{h}$ for each variable group where

$$C = \frac{C_h}{h}, \qquad D = \frac{D_h}{h}, \qquad \bar{C} = \frac{\bar{C}_h}{\bar{h}}, \qquad \bar{D} = \frac{\bar{D}_h}{\bar{h}}, \tag{2.72}$$

and the new homogeneous variables groups are $\langle C_h, D_h, h\rangle$, $\left\langle \bar{C}_h, \bar{D}_h, \bar{h}\right\rangle$. By substituting Eqn. (2.72) into $\mathbf{f}(\mathbf{z})$ and clearing the denominator, we obtain

$$f_j(\mathbf{z}_h) = q_{3j}C_h\bar{D}_h + \bar{q}_{3j}\bar{C}_h D_h + q_{2j}C_h\bar{h} + \bar{q}_{2j}\bar{C}_h h + q_{1j}D_h\bar{h} + \bar{q}_{1j}\bar{D}_h h + q_{0j}h\bar{h} = 0, \qquad j = 1, 2, 3, 4, \tag{2.73}$$

and similarly, the start system $\mathbf{g}(\mathbf{z})$ becomes

$$g_j(\mathbf{z}_h) = (b_{0j}C_h + b_{1j}D_h + b_{2j}h)(b_{3j}\bar{C}_h + b_{4j}\bar{D}_h + b_{5j}\bar{h}) = 0, \qquad j = 1, 2, 3, 4. \tag{2.74}$$

where $\mathbf{z}_h = \{C, D, h, \bar{C}, \bar{D}, \bar{h}\}$.

Eqns. (2.73) and (2.74) are called multihomogeneous because the degree of each term is the same with respect to each variable group. Homogenizing the equations provides an effective means of representing infinity because the values of $(C, D)$ and $(\bar{C}, \bar{D})$ are defined by the ratios of projective coordinates $(C_h, D_h, h)$ and $(\bar{C}_h, \bar{D}_h, \bar{h})$, respectively. That means coordinates for each group are given in projective space $\mathbb{P}^2$ instead of $\mathbb{C}^2$, which can be thought of as lines through the origin in $\mathbb{C}^3$. For example, a line in the projective space of the first group is defined by $(\alpha C_h, \alpha D_h, \alpha h)$, where $\alpha \in \mathbb{C}$. It is when the homogeneous variable $h = 0$ (and either $C_h \neq 0$ or $D_h \neq 0$) that the projective coordinates represent infinity.

However, in order to proceed with numeric path tracking, these lines in $\mathbb{C}^3$ need to be converted to points in $\mathbb{C}^3$. This is accomplished by introducing linear patches, which are planes in $\mathbb{C}^3$. For example, by intersecting the line $(\alpha C_h, \alpha D_h, \alpha h)$ with the linear patch $h = 1$, the projective coordinates directly equal the values of the affine coordinates $(C, D)$. However, this would not be a good patch for representing infinity because it has no intersection with $(\alpha C_h, \alpha D_h, 0)$, which represents infinity. Therefore, instead a random plane in $\mathbb{C}^3$ is used as the linear patch

$$l_1 = a_{01}C_h + a_{11}D_h + a_{21}h - 1 = 0 \tag{2.75}$$

where the $a$ coefficients are random complex numbers. This plane intersects line $(\alpha C_h, \alpha D_h, \alpha h)$

at coordinates

$$\left( \frac{C_h}{a_{01}C_h + a_{11}D_h + a_{21}h} \, , \, \frac{D_h}{a_{01}C_h + a_{11}D_h + a_{21}h} \, , \, \frac{h}{a_{01}C_h + a_{11}D_h + a_{21}h} \right) \tag{2.76}$$

which are valid so long as

$$a_{01}C_h + a_{11}D_h + a_{21}h \neq 0. \tag{2.77}$$

However, with probability one, this intersection will exist because the $a$ coefficients are randomly generated. The random patch for the second homogeneous variable group follows from the first,

$$l_2 = a_{02}\bar{C}_h + a_{12}\bar{D}_h + a_{22}\bar{h} - 1 = 0. \tag{2.78}$$

The startpoints to the affine system are the solutions to $\mathbf{g}(\mathbf{z})$, Eqn (2.68), which are $\{C, D, \bar{C}, \bar{D}\}_k$, $k = 1, \ldots, 6$. The projective patches, Eqns. (2.75) and (2.78), are used to compute the multihomogeneous coordinates of the startpoints,

$$\mathbf{p}_{0,k} = \left\{C_h, D_h, h, \bar{C}_h, \bar{D}_h, \bar{h}\right\}_k = \left\{ \frac{C}{a_{01}C + a_{11}D + a_{21}} \, , \, \frac{D}{a_{01}C + a_{11}D + a_{21}} \, , \right.$$
$$\left. \frac{1}{a_{01}C + a_{11}D + a_{21}} \, , \, \frac{\bar{C}}{a_{02}\bar{C} + a_{12}\bar{D} + a_{22}} \, , \, \frac{\bar{D}}{a_{02}\bar{C} + a_{12}\bar{D} + a_{22}} \, , \, \frac{1}{a_{02}\bar{C} + a_{12}\bar{D} + a_{22}} \right\}_k$$
$$k = 1, \ldots, 6. \tag{2.79}$$

**Homotopy Function**

A homotopy function is a parameterized deformation from start system $\mathbf{g}(\mathbf{z}_h)$ to target system $\mathbf{f}(\mathbf{z}_h)$ which takes the form

$$\mathbf{H}(\mathbf{z}_h, s) = s\mathbf{g}(\mathbf{z}_h) + (1 - s)\mathbf{f}(\mathbf{z}_h), \tag{2.80}$$

so that each homotopy path starts at $s = 1$ and ends at $s = 0$. However, in order for homotopy paths to avoid special failure zones, $s$ does not track along the real line segment, but instead takes on complex values tracing a circular arc in the imaginary plane that begins at $s = 1$ and ends at $s = 0$, see Fig 2.9. This circular arc is parameterized by $t$,

$$s(t) = \frac{\gamma t}{\gamma t + (1 - t)} \tag{2.81}$$

where $\gamma$ is a random complex value with near unit magnitude. Substituting Eqn. (2.81) into Eqn. (2.80) and clearing the denominator, we obtain

$$\mathbf{H}(\mathbf{z}_h, t) = \gamma t \mathbf{g}(\mathbf{z}_h) + (1 - t)\mathbf{f}(\mathbf{z}_h). \tag{2.82}$$

Bates et al. refer to this parameterization of the homotopy path as "the gamma trick". Finally, in order to define unique points to be computed by a path tracking algorithm, we redefine $\mathbf{H}$ to include linear patches $l_1$, $l_2$, as well,

$$\mathbf{H}(\mathbf{z}_h, t) = \begin{Bmatrix} \gamma t g_1(\mathbf{z}_h) + (1 - t) f_1(\mathbf{z}_h) \\ \gamma t g_2(\mathbf{z}_h) + (1 - t) f_2(\mathbf{z}_h) \\ \gamma t g_3(\mathbf{z}_h) + (1 - t) f_3(\mathbf{z}_h) \\ \gamma t g_4(\mathbf{z}_h) + (1 - t) f_4(\mathbf{z}_h) \\ l_1(\mathbf{z}_h) \\ l_2(\mathbf{z}_h) \end{Bmatrix}. \tag{2.83}$$

### 2.3.4 Path Tracking

At this point, we have defined a homogenized start system $\mathbf{g}(\mathbf{z}_h)$, target system $\mathbf{f}(\mathbf{z}_h)$, start-points $\mathbf{p}_{0,k}$ for $k = 1, \ldots, 6$, and a homotopy function $\mathbf{H}(\mathbf{z}_h, t)$, which is all we need to begin path tracking. During path tracking, we increment the path variable from $t = 1$ to $t = 0$

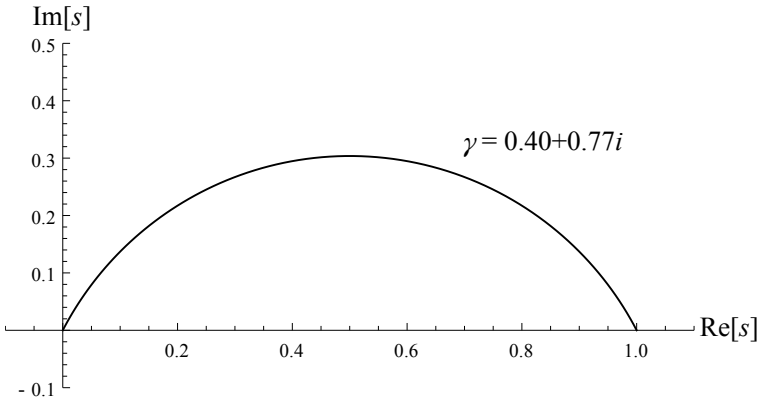



Figure 2.9: The path variable $s$ is parameterized to move along a circular arc in the complex plane.

and track the roots of $\mathbf{H}(\mathbf{z}_h, t)$ from startpoints to endpoints. It is advantageous that each startpoint can be tracked independently which facilitates parallel computations.

There are numerous path tracking methods available, here the Euler-Newton predictor-corrector method is described. Each path begins with a startpoint $\mathbf{p}_0$ at $t_0 = 1$, and iteratively solves for $\mathbf{p}_1, \mathbf{p}_2, \ldots$ at $t_1, t_2, \ldots$. The value of $\mathbf{p}_{i+1}$ is first predicted using Euler's method,

$$\tilde{\mathbf{p}}_{i+1} = \mathbf{p}_i - [J_{\mathbf{H}}(\mathbf{p}_i, t_i)]^{-1} \frac{\partial \mathbf{H}(\mathbf{p}_i, t_i)}{\partial t} \Delta t_i, \tag{2.84}$$

where $[J_{\mathbf{H}}(\mathbf{p}_i, t_i)] = \dfrac{\partial \mathbf{H}}{\partial \mathbf{z}_h}$ and $\Delta t_i = t_{i+1} - t_i$. Geometrically, this is interpreted as following the line tangent to the path from point $\mathbf{p}_i$ at a step $\Delta t_i$. The predicted value $\tilde{\mathbf{p}}_{i+1}$ is then used as the start value for solving $\mathbf{H}(\mathbf{z}_h, t_{i+1}) = 0$ using Newton's method. That is, the variable $\mathbf{z}_{h,0} = \tilde{\mathbf{p}}_{i+1}$ is set, and a few Newton iterations are run,

$$\mathbf{z}_{h,j+1} = \mathbf{z}_{h,j} - [J_{\mathbf{H}}(\mathbf{z}_{h,j}, t_{i+1})]^{-1} \mathbf{H}(\mathbf{z}_{h,j}, t_{i+1}) \tag{2.85}$$

to obtain $\mathbf{p}_{i+1} = \mathbf{z}_{h,j+1}$ as the result of the Newton iterations. The algorithm then returns to Eqn. (2.84) for the next step in the path tracking routine.

The Euler-Newton predictor-corrector method is one of several path tracking methods, and is described above for its simplicity. The software package BERTINI incorporates a wide range of trackers. The default tracking method used by BERTINI is the fifth-order Runge-Kutta-Fehlberg prediction method with a fourth-order error estimation. Furthermore, when tracking towards a singular solution, that is when $|\mathbf{J_H}(\mathbf{z}_h, 0)| = 0$, BERTINI transitions to a special set of path tracking algorithms named *endgames*, which can handle an ill-conditioned Jacobian matrix.

**Numerical Example**

We continue our numerical example by solving $\mathbf{f}(\mathbf{z})$ for the task function values,

| $j$ | $Q_j$ | $S_j$ |
|---|---|---|
| 0 | $e^{i0^\circ}$ | $e^{i0^\circ}$ |
| 1 | $e^{i40^\circ}$ | $e^{i27^\circ}$ |
| 2 | $e^{i80^\circ}$ | $e^{i49^\circ}$ |
| 3 | $e^{i120^\circ}$ | $e^{i68^\circ}$ |
| 4 | $e^{i160^\circ}$ | $e^{i91^\circ}$ |

$$\begin{aligned} A = &\quad 0.9 + 0.8i, \\ B = &\quad 1.6 + 0.15i, \end{aligned} \tag{2.86}$$

which completely defines the coefficients of the target system, see Eqn. (2.57). The coefficients of the start system $\mathbf{g}(\mathbf{z})$ were defined as the random complex numbers, see Eqn.

(2.65),

$$
\begin{aligned}
&b_{01}= 0.82 + 0.78i, && b_{11}= 1.35 - 0.75i, && b_{21}= 1.19 + 0.81i, && b_{31}= -0.63 - 1.19i,\\
&b_{41}= 1.41 + 1.48i, && b_{51}= 0.98 - 1.21i, && b_{02}= 0.56 + 0.61i, && b_{12}= 1.30 + 0.67i,\\
&b_{22}= 1.13 - 0.85i, && b_{32}= 0.91 - 1.18i, && b_{42}= 0.94 + 0.74i, && b_{52}= 0.66 + 0.64i,\\
&b_{03}= 1.46 + 0.80i, && b_{13}= -1.09 - 0.52i, && b_{23}= 0.60 + 1.08i, && b_{33}= 1.16 + 1.18i,\\
&b_{43}= -0.71 + 1.41i, && b_{53}= -1.29 + 0.86i, && b_{04}= -1.10 - 0.76i, && b_{14}= -1.04 + 0.87i,\\
&b_{24}= -0.84 - 0.50i, && b_{34}= 0.75 - 1.10i, && b_{44}= -0.79 + 0.94i, && b_{54}= 0.95 + 1.06i,
\end{aligned}
\tag{2.87}
$$

and the coefficients of the linear patches were defined as the random complex numbers, see Eqns. (2.75) and(2.78),

$$
\begin{aligned}
&a_{01} = 0.27 - 0.54i, && a_{11} = -0.31 + 0.49i, && a_{21} = 0.35 + 0.42i,\\
&a_{02} = -0.38 - 0.45i, && a_{12} = 0.57 + 0.25i, && a_{22} = 0.32 - 0.40i,
\end{aligned}
\tag{2.88}
$$

and the homotopy path $\gamma$ coefficient was defined as the random complex number $\gamma = 0.40 + 0.77i$.

Figs. 2.10 and 2.11 illustrate the six homotopy paths taken by each variable. Each figure contains six paths which correlate to the six startpoints $\mathbf{p}_{0,k}$, $k = 1, \ldots, 6$. Note that for paths 2 and 4 the endpoints of $h$ and $\bar{h}$ are at 0, see Fig. 2.11, which means they represent solutions at infinity. This is not surprising because the monomial structure of our start system was more general than the monomial structure of our target system. These two

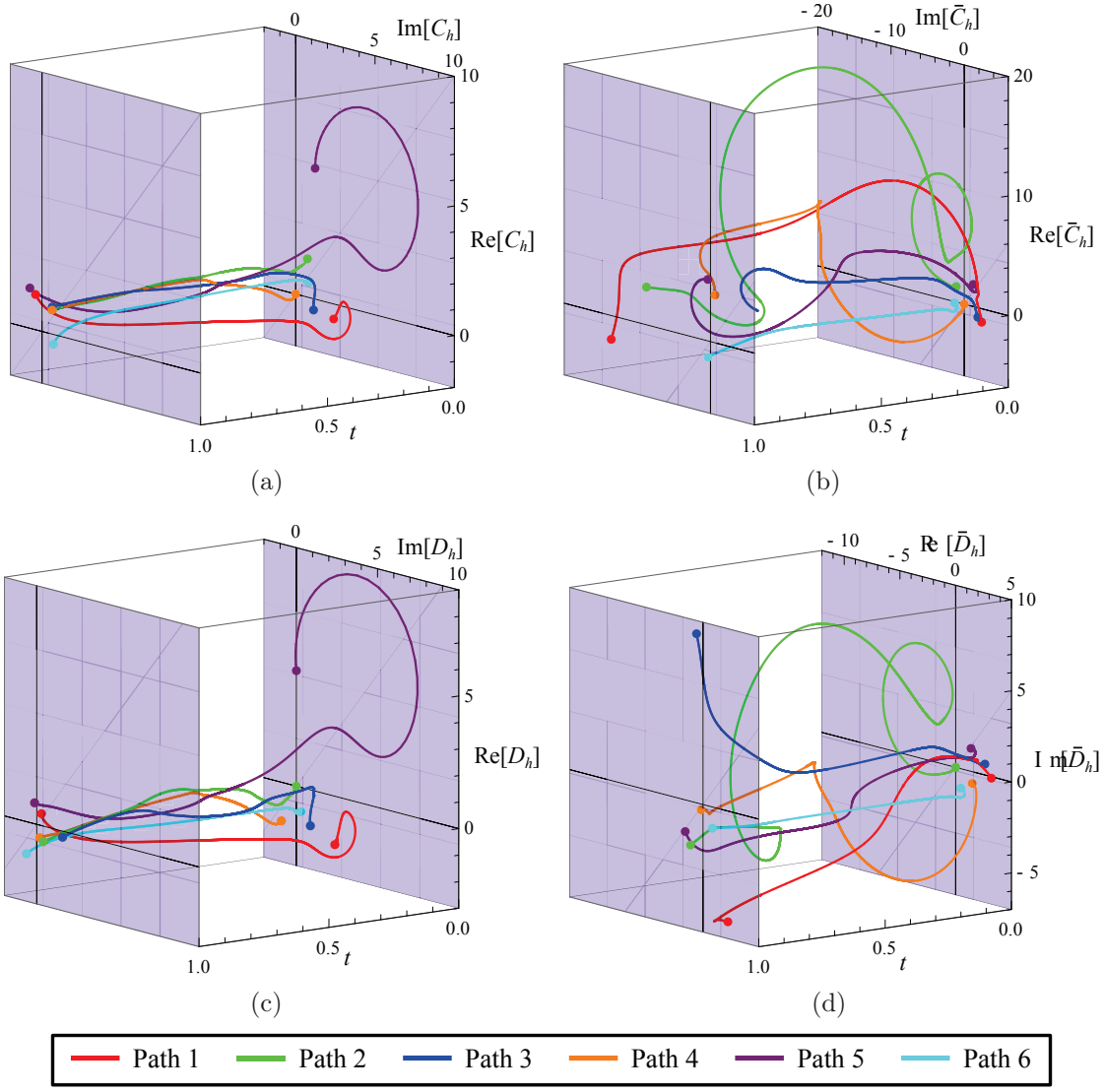



Figure 2.10: The six homotopy paths of $C_h$, $\bar{C}_h$, $D_h$, and $\bar{D}_h$ for the example.

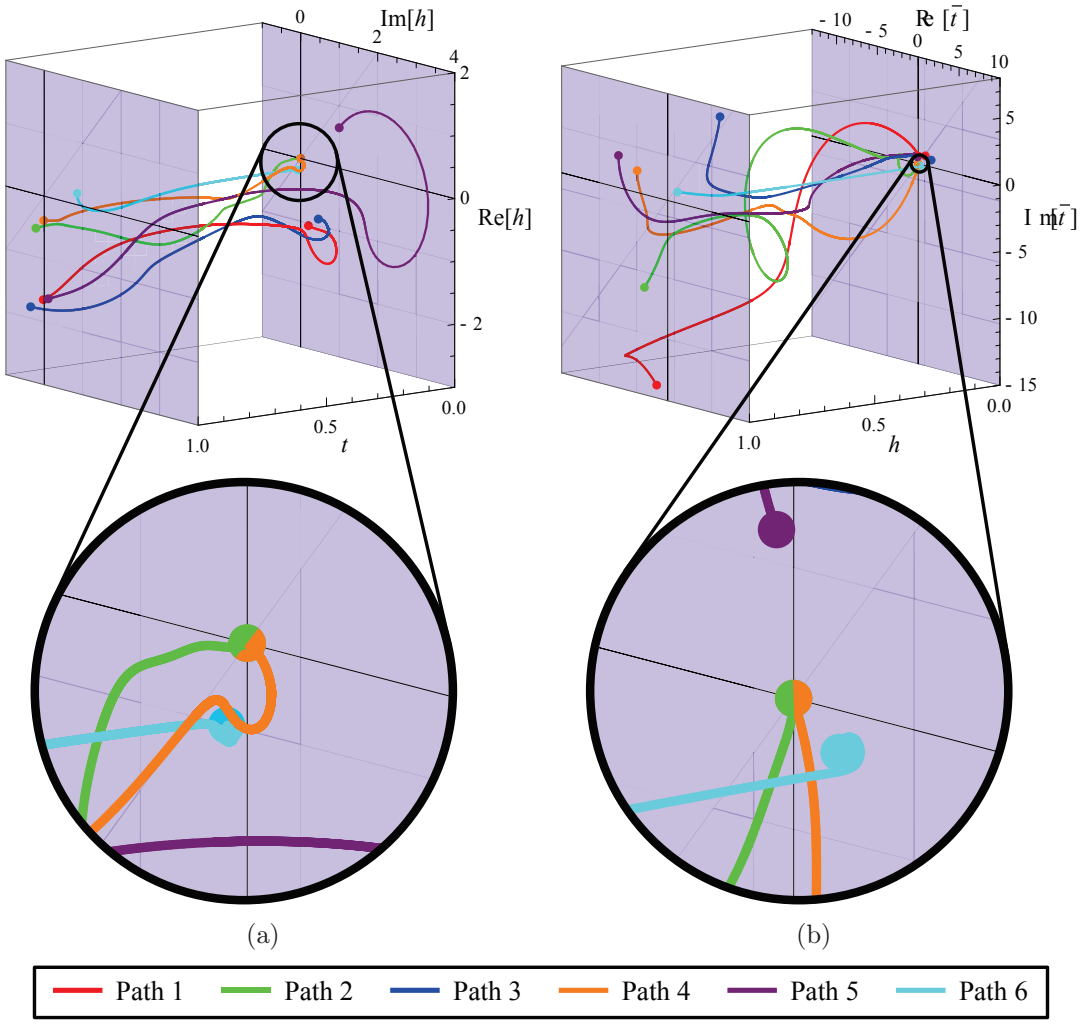



Figure 2.11: The six homotopy paths of the homogeneous variables $h$ and $\bar{h}$ where paths 2 and 4 indicate solutions at infinity.

solutions are dropped. The solutions to the affine system are

$$\begin{Bmatrix} C \\ D \\ \bar{C} \\ \bar{D} \end{Bmatrix} = \begin{Bmatrix} 0.96557746 + 2.12002346i \\ 2.12035427 + 1.87680587i \\ 0.96557746 - 2.12002346i \\ 2.12035427 - 1.87680587i \end{Bmatrix}, \begin{Bmatrix} 0.9 + 0.8i \\ 1.6 + 0.15i \\ 0.9 - 0.8i \\ 1.6 - 1.15i \end{Bmatrix},$$
$$\begin{Bmatrix} 3.16417643 + 2.99734905i \\ 1.98856018 + 3.06682058i \\ 3.16417643 - 2.99734905i \\ 1.98856018 - 3.06682058i \end{Bmatrix}, \begin{Bmatrix} -4.18347015 + 2.83840336i \\ 3.55268302 + 2.71472288i \\ -4.18347015 - 2.83840336i \\ 3.55268302 - 2.71472288i \end{Bmatrix}. \quad (2.89)$$

The second solution is $C = A$ and $D = B$, which is shown to be true by making these substitutions into Eqn. (2.56), however this solution does not describe a four-bar linkage. The other three solutions represent the four-bar linkage design candidates to explore. In general, polynomial systems of the monomial structure of $\mathbf{f}(\mathbf{z})$ have four finite solutions.

### 2.3.5 Parameter Homotopy

Polynomial equations of the form,

$$q_{3j}C\bar{D} + \bar{q}_{3j}\bar{C}D + q_{2j}C + \bar{q}_{2j}\bar{C} + q_{1j}D + \bar{q}_{1j}\bar{D} + q_{0j} = 0, \qquad j = 1, 2, 3, 4, \tag{2.90}$$

where the $q$ coefficients are all random complex numbers, have four finite solutions. This is similar to the case presented above, however the $q$ coefficients were not random but are sufficient for this example. The solutions were found by constructing a homotopy from a start system that included extra monomials $C\bar{C}$ and $D\bar{D}$, see Eqn. (2.66), such that six homotopy paths needed to be tracked.

An ideal start system would include no extra monomials and have only four solutions, which is exactly what the results of Section 2.3.4 provide, that is numerical solutions (2.89) to $\mathbf{f}(\mathbf{z})$ as defined by coefficients $\mathbf{q}$. We now write the target system as a function of both its variables and coefficients, $\mathbf{f}(\mathbf{z}, \mathbf{q})$. Therefore, the system $\mathbf{f}(\mathbf{z}, \mathbf{q})$ can be used as the start system to solve for any other system of the target system's form (2.90) with different $q$ coefficients. This is called a *parameter homotopy* and takes the form

$$\mathbf{H}(\mathbf{z}_h, t) = \gamma t\mathbf{f}(\mathbf{z}, \mathbf{q}_1) + (1 - t)\mathbf{f}(\mathbf{z}, \mathbf{q}_0). \tag{2.91}$$

where $\mathbf{q}_1$ and $\mathbf{q}_0$ are two different coefficient vectors, called *parameters*. The start system $\mathbf{f}(\mathbf{z}, \mathbf{q}_1)$ is used to solve the target system $\mathbf{f}(\mathbf{z}, \mathbf{q}_0)$, where all the roots of $\mathbf{f}(\mathbf{z}, \mathbf{q}_1)$ have been

previously obtained by another homotopy or any other solution method. Note that in this case the parameters $\mathbf{q}$ are identically the coefficients, but in general parameters can be chosen internally to the coefficients as well, i.e. $Q_j$, $\bar{Q}_j$, $S_j$, $\bar{S}_j$, $A$, $\bar{A}$, $B$, $\bar{B}$, see Eqn. (2.57), where the procedure changes slightly.

The advantage of parameter homotopy is that four paths need to be tracked instead of six. A reduction by two homotopy paths does not mean much for this small example, but for the larger problems presented in this dissertation the reductions are much greater and have a significant effect on computing time.

The use of parameter homotopies breaks the solution process of large polynomial systems into two phases: (1) the ab initio phase and (2) the parameter homotopy phase. The ab initio phase consists of a computationally expensive initial solve of a system defined to have parameters specified as random complex numbers. For example, the design equations of the Stephenson III function generator presented in Chapter 5 were solved by forming a multihomogeneous homotopy that tracked 55 million paths. The parameter homotopy phase uses the results of the ab initio phase to track only the finite and/or nonsingular roots for all proceeding solves. For example, the Stephenson III ab initio phase found 834,441 finite nonsingular solutions, so all proceeding solves were required to track only 834,441 paths.

### 2.3.6 Regeneration

Sections 2.3.2 discussed how to examine the monomial structure of a polynomial system in order to construct a multihomogeneous homotopy that tracks a reduced number of paths by choosing variable groupings. One alternative to this method is regeneration homotopy. Regeneration does not require one to specify variable groups, but automatically takes advantage of the sparse monomial structure of the target system by handling each equation one by one.

We begin by picking a start system $\mathbf{g}(\mathbf{z})$ comprised of the products of linear polynomials. In this case, let's choose the less optimal choice from Section 2.3.2, that is Eqn. (2.63),

$$\mathbf{g}(\mathbf{z}) = \begin{Bmatrix} g_1(\mathbf{z}) \\ g_2(\mathbf{z}) \\ g_3(\mathbf{z}) \\ g_4(\mathbf{z}) \end{Bmatrix} = \begin{Bmatrix} L_1(\mathbf{z})L_2(\mathbf{z}) \\ L_3(\mathbf{z})L_4(\mathbf{z}) \\ L_5(\mathbf{z})L_6(\mathbf{z}) \\ L_7(\mathbf{z})L_8(\mathbf{z}) \end{Bmatrix} \tag{2.92}$$

where $L_k$, $k = 1, \ldots, 8$ are linear expressions of random complex coefficients where each unknown appears once. As well, we specify another linear system $\mathbf{l}(\mathbf{z})$,

$$\mathbf{l}(\mathbf{z}) = \begin{Bmatrix} l_1(\mathbf{z}) \\ l_2(\mathbf{z}) \\ l_3(\mathbf{z}) \\ l_4(\mathbf{z}) \end{Bmatrix} \tag{2.93}$$

where $l_j$, $j = 1, \ldots, 4$, are also linear expressions of random complex coefficients that contain all unknowns. Finally, the target system we intend to solve is $\mathbf{f}(\mathbf{z})$,

$$\mathbf{f}(\mathbf{z}) = \begin{Bmatrix} f_1(\mathbf{z}) \\ f_2(\mathbf{z}) \\ f_3(\mathbf{z}) \\ f_4(\mathbf{z}) \end{Bmatrix} \tag{2.94}$$

where $f_j$, $j = 1, \ldots, 4$, are defined in Eqn. (2.56). The objective is to find all the finite nonsingular solutions of the target system. For notational brevity, the argument $\mathbf{z}$ will be omitted from functions in the following explanation.

First, we find the two solutions of $\{g_1, l_2, l_3, l_4\}$ which are the solutions of the linear systems $\{L_1, l_2, l_3, l_4\}$ and $\{L_2, l_2, l_3, l_4\}$. We use those solutions to track two homotopy paths to solve

the system $\{f_1, l_2, l_3, l_4\}$. Let's assume that both paths lead to finite nonsingular solutions. Then we use these two solutions to construct parameter homotopies that solve $\{f_1, L_3, l_3, l_4\}$ and $\{f_1, L_4, l_3, l_4\}$, which provide the four solutions of $\{f_1, g_2, l_3, l_4\}$, which are used to track four homotopy paths to solve $\{f_1, f_2, l_3, l_4\}$. Assuming that all paths lead to finite nonsingular solutions, and continuing this pattern, the final step of the algorithm is to track a 16 path homotopy from $\{f_1, f_2, f_3, g_4\}$ to our target system $\{f_1, f_2, f_3, f_4\}$, see Fig 2.12.

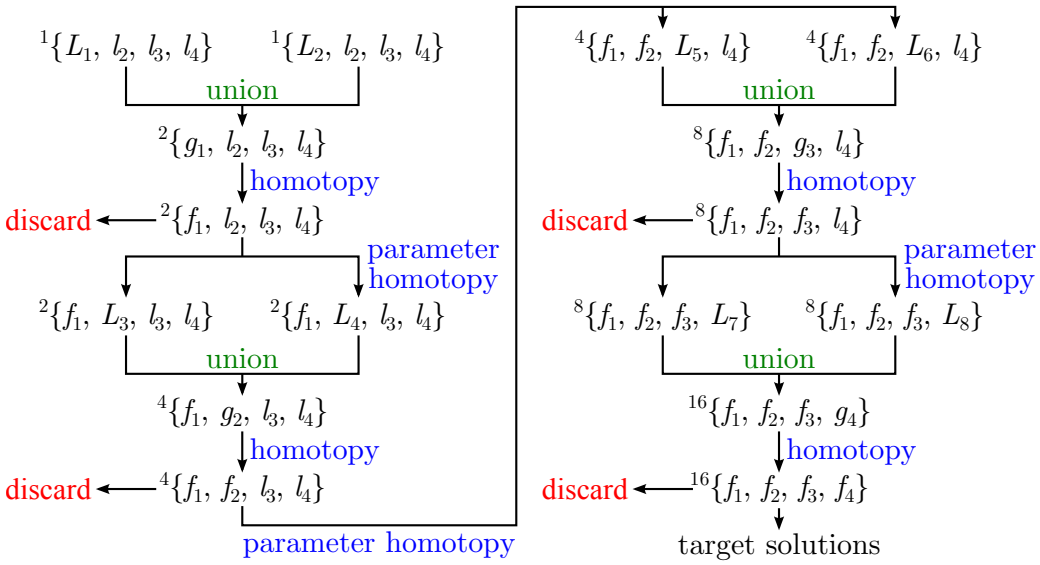



Figure 2.12: Flowchart of the regeneration algorithm.

The assumption that each step of regeneration results in all finite nonsingular solutions creates an algorithm that tracks much more than 16 paths. However, the value of regeneration comes when this assumption is false. Paths that do track to infinity and singular solutions are discarded at each level, removing them and their descendants from the proceeding levels, which is often the case for polynomials with a sparse monomial structure. This is the case for the example at hand, which we know has only four finite nonsingular solutions. In fact, many of the polynomial systems encountered in kinematics have a sparse monomial structure, which makes regeneration a powerful tool.

The regeneration procedure is illustrated in Fig. 2.12. In the flowchart, bracketed { } members refer to a system and the superscript in front refers to the maximum number of

solutions at that point in the regeneration process.

## 2.4 Linkage Analysis

Linkage analysis simulates the motions of a linkage design once its dimensions have been determined from synthesis. Simulating the motions of a linkage entails formulating the forward kinematics equations, solving the forward kinematics equations, and then sorting those solutions onto continuous trajectories. These continuous trajectories represent all the motions that a particular linkage is capable of.

Forward kinematics refers to solving the kinematics equations of a mechanically constrained system in order to determine its output configurations from a set of input joint angle data. All of the mechanisms discussed in this dissertation are single degree-of-freedom linkages, meaning the configuration of the linkage is determined by a single input angle value.

### 2.4.1 Forward Kinematics of a Four-bar

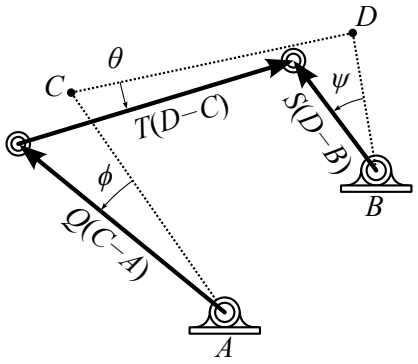



Figure 2.13: A four-bar linkage displaced from its reference configuration.

The kinematics equations of a four-bar linkage are based on its loop equations, which were presented in Section 2.2.4, and are repeated here. A four-bar linkage is defined by the locations of two fixed pivots, $A$, $B$, and the locations of two moving pivots, $C$, $D$, in a

reference position. Connecting these four points draws the parallelogram $ACDB$ which we wish to animate where link $AB$ is fixed. The orientations of links $AC$, $BD$, $CD$ are measured from their reference configuration by angles $\phi$, $\psi$, $\theta$, respectively. We represent these angles with complex rotation operators,

$$Q = e^{i\phi}, \qquad S = e^{i\psi}, \qquad T = e^{i\theta}. \tag{2.95}$$

The loop equation of a four-bar is derived by examining Fig. 2.13,

$$\mathcal{L} = A - B + Q(C - A) + T(D - C) - S(D - B) = 0, \tag{2.96}$$

which is the same equation presented in (2.51), only with task position index $j$ removed. The variables in $\mathcal{L}$ are $Q$, $S$, $T$ and must have unit magnitude as they are rotation operators,

$$\mathcal{N}_Q = Q\bar{Q} - 1 = 0, \qquad \mathcal{N}_S = S\bar{S} - 1 = 0, \qquad \mathcal{N}_T = T\bar{T} - 1 = 0. \tag{2.97}$$

As well, we append the conjugate loop equations,

$$\bar{\mathcal{L}} = \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{T}(\bar{D} - \bar{C}) - \bar{S}(\bar{D} - \bar{B}) = 0. \tag{2.98}$$

Eqns. (2.96)–(2.98) represent five constraints on six variables $\langle Q, \bar{Q}, S, \bar{S}, T, \bar{T} \rangle$, that is a single degree-of-freedom system. If we choose $\mathbf{x} = \{Q, \bar{Q}\}$ to be the specified input parameter and $\mathbf{y} = \{S, \bar{S}, T, \bar{T}\}$ to be the output, then the forward kinematics equations for the four-bar are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{N}_S, \mathcal{N}_T\right\} = 0 \tag{2.99}$$

These equations are adapted easily if we choose $\{S, \bar{S}\}$ or $\{T, \bar{T}\}$ to be the input. Several

methods exist to find the roots $\mathbf{y}$ of $\mathbf{F}(\mathbf{x}, \mathbf{y})$ for a specified $\mathbf{x}$. We choose to use the *NSolve* function built into MATHEMATICA.

### 2.4.2 Forward Kinematics of the Six-bars

Section 2.4.1 illustrates the process of formulating the forward kinematics equations of a linkage. That is to (1) list one loop equation and its conjugate for each independent loop of a mechanism, and (2) append normalization conditions for all unknown output angles. In this section, we apply this procedure to all five types of six-bar linkages: Watt I, Watt II, Stephenson I, Stephenson II, and Stephenson III.

**Watt I**

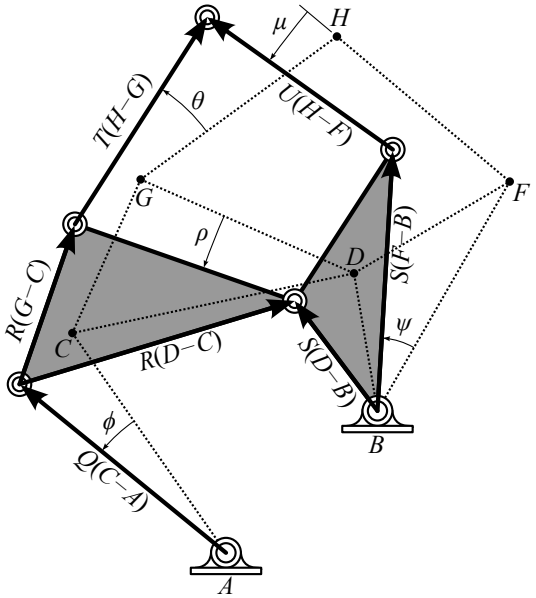



Figure 2.14: A Watt I linkage displaced from its reference configuration.

A Watt I linkage is defined by the locations of two fixed pivots, $A$, $B$, and the locations of five moving pivots $C$, $D$, $F$, $G$, $H$, in a reference position. These seven pivots are connected according to Fig. 2.14 to form five moving links $AC$, $CDG$, $BDF$, $GH$, and $FH$. The

orientations of these links are measured from their reference positions by angles $\phi$, $\rho$, $\psi$, $\theta$, and $\mu$, respectively, which define the rotation operators,

$$Q = e^{i\phi}, \qquad R = e^{i\rho}, \qquad S = e^{i\psi}, \qquad T = e^{i\theta}, \qquad U = e^{i\mu}. \tag{2.100}$$

The loop equations and their conjugates are formed by tracing the vector loops found in Fig. 2.14,

$$\begin{aligned}
\mathcal{L} &= A - B + Q(C - A) + R(D - C) - S(D - B) = 0, \\
\bar{\mathcal{L}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{R}(\bar{D} - \bar{C}) - \bar{S}(\bar{D} - \bar{B}) = 0, \\
\mathcal{M} &= A - B + Q(C - A) + R(G - C) + T(H - G) - S(F - B) - U(H - F) = 0, \\
\bar{\mathcal{M}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{R}(\bar{G} - \bar{C}) + \bar{T}(\bar{H} - \bar{G}) - \bar{S}(\bar{F} - \bar{B}) - \bar{U}(\bar{H} - \bar{F}) = 0.
\end{aligned} \tag{2.101}$$

Note that a six-bar has two independent loops. The variables of Eqns. (2.101) are $\langle Q, \bar{Q}, R, \bar{R}, S, \bar{S}, T, \bar{T}, U, \bar{U} \rangle$ which are rotation operators and must satisfy the conditions,

$$\begin{aligned}
&\mathcal{N}_Q = Q\bar{Q} - 1 = 0, \qquad &&\mathcal{N}_R = R\bar{R} - 1 = 0, \qquad &&\mathcal{N}_S = S\bar{S} - 1 = 0, \\
&\mathcal{N}_T = T\bar{T} - 1 = 0, \qquad &&\mathcal{N}_U = U\bar{U} - 1 = 0.
\end{aligned} \tag{2.102}$$

Eqns. (2.101) and (2.102) represent nine constraints on ten unknowns. If $\mathbf{x} = \{Q, \bar{Q}\}$ is chosen as the input parameter and $\mathbf{y} = \{R, \bar{R}, S, \bar{S}, T, \bar{T}, U, \bar{U}\}$ is the output, then the forward kinematics equations are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{M}, \bar{\mathcal{M}}, \mathcal{N}_R, \mathcal{N}_S, \mathcal{N}_T, \mathcal{N}_U\right\} = 0. \tag{2.103}$$

$\mathbf{F}(\mathbf{x}, \mathbf{y})$ has four roots $\mathbf{y}$ for a specified value of $\mathbf{x} = \{Q, \bar{Q}\}$ which correspond to four configurations of the mechanism. The equations change slightly if a different input $\mathbf{x}$ is

chosen. For $\mathbf{x} = \{S, \bar{S}\}$, there are also four configurations.

**Watt II**

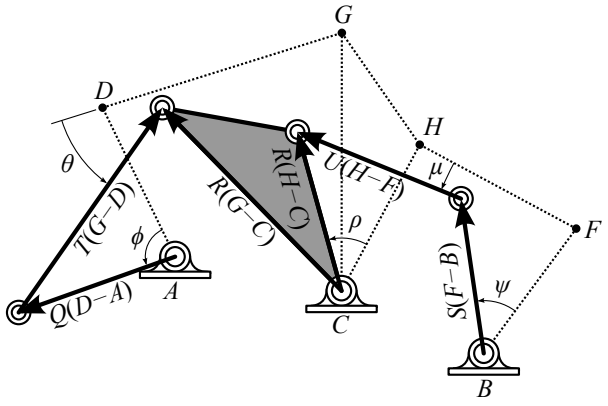



Figure 2.15: A Watt II linkage displaced from its reference configuration.

A Watt II linkage is defined by the locations of three fixed pivots, $A$, $B$, $C$, and the locations of four moving pivots $D$, $F$, $G$, $H$, in a reference position. These seven pivots are connected according to Fig. 2.15 to form five moving links $AD$, $CGH$, $BF$, $DG$, and $FH$. The orientations of these links are measured from their reference positions by angles $\phi$, $\rho$, $\psi$, $\theta$, and $\mu$, respectively, which define the rotation operators $Q$, $R$, $S$, $T$, $U$ as shown in Eqn. (2.100). The loop equations and their conjugates are formed by tracing the vector loops found in Fig. 2.15,

$$\begin{aligned}
\mathcal{L} &= A - C + Q(D - A) + T(G - D) - R(G - C) = 0, \\
\bar{\mathcal{L}} &= \bar{A} - \bar{C} + \bar{Q}(\bar{D} - \bar{A}) + \bar{T}(\bar{G} - \bar{D}) - \bar{R}(\bar{G} - \bar{C}) = 0, \\
\mathcal{M} &= B - C + S(F - B) + U(H - F) - R(H - C) = 0, \\
\bar{\mathcal{M}} &= \bar{B} - \bar{C} + \bar{S}(\bar{F} - \bar{B}) + \bar{U}(\bar{H} - \bar{F}) - \bar{R}(\bar{H} - \bar{C}) = 0.
\end{aligned} \tag{2.104}$$

The variables are the rotation operators which must satisfy the normalization conditions shown in Eqn. (2.102). Eqns. (2.102) and (2.104) are nine constraints on ten unknowns.

If $\mathbf{x} = \{Q, \bar{Q}\}$ is chosen as the input parameter and $\mathbf{y} = \{R,\ \bar{R},\ S,\ \bar{S},\ T,\ \bar{T},\ U,\ \bar{U}\}$ is the output, then the forward kinematics equations are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{M}, \bar{\mathcal{M}}, \mathcal{N}_R, \mathcal{N}_S, \mathcal{N}_T, \mathcal{N}_U\right\} = 0. \tag{2.105}$$

$\mathbf{F}(\mathbf{x}, \mathbf{y})$ has four roots $\mathbf{y}$ for a specified value of $\mathbf{x} = \{Q, \bar{Q}\}$ which correspond to four configurations of the mechanism.

**Stephenson I**

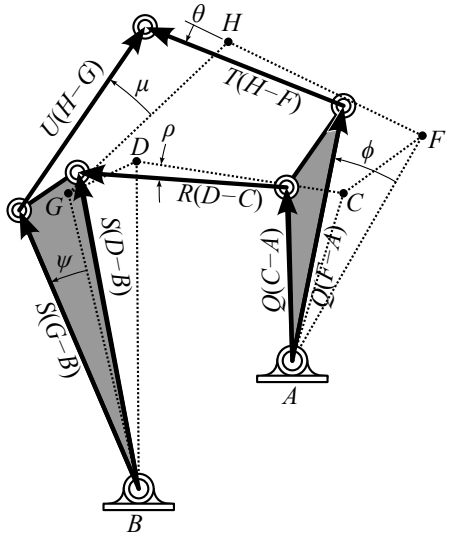



Figure 2.16: A Stephenson I linkage displaced from its reference configuration.

A Stephenson I linkage is defined by the locations of two fixed pivots, $A$, $B$, and the locations of five moving pivots $C$, $D$, $F$, $G$, $H$, in a reference position. These seven pivots are connected according to Fig. 2.16 to form five moving links $ACF$, $CD$, $BDG$, $FH$, and $GH$. The orientations of these links are measured from their reference positions by angles $\phi$, $\rho$, $\psi$, $\theta$, and $\mu$, respectively, which define the rotation operators $Q$, $R$, $S$, $T$, $U$ as shown in Eqn. (2.100). The loop equations and their conjugates are formed by tracing the vector loops

found in Fig. 2.16,

$$\begin{aligned}
\mathcal{L} &= A - B + Q(C - A) + R(D - C) - S(D - B) = 0, \\
\bar{\mathcal{L}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{R}(\bar{D} - \bar{C}) - \bar{S}(\bar{D} - \bar{B}) = 0, \\
\mathcal{M} &= A - B + Q(F - A) + T(H - F) - S(G - B) - U(H - G) = 0, \\
\bar{\mathcal{M}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{F} - \bar{A}) + \bar{T}(\bar{H} - \bar{F}) - \bar{S}(\bar{G} - \bar{B}) - \bar{U}(\bar{H} - \bar{G}) = 0.
\end{aligned} \tag{2.106}$$

The variables are the rotation operators which must satisfy the normalization conditions shown in Eqn. (2.102). Eqns. (2.102) and (2.106) are nine constraints on ten unknowns. If $\mathbf{x} = \{Q, \bar{Q}\}$ is chosen as the input parameter and $\mathbf{y} = \{R,\ \bar{R},\ S,\ \bar{S},\ T,\ \bar{T},\ U,\ \bar{U}\}$ is the output, then the forward kinematics equations are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{M}, \bar{\mathcal{M}}, \mathcal{N}_R, \mathcal{N}_S, \mathcal{N}_T, \mathcal{N}_U\right\} = 0. \tag{2.107}$$

$\mathbf{F}(\mathbf{x}, \mathbf{y})$ has four roots $\mathbf{y}$ for a specified value of $\mathbf{x} = \{Q, \bar{Q}\}$ which correspond to four configurations of the mechanism.

**Stephenson II**

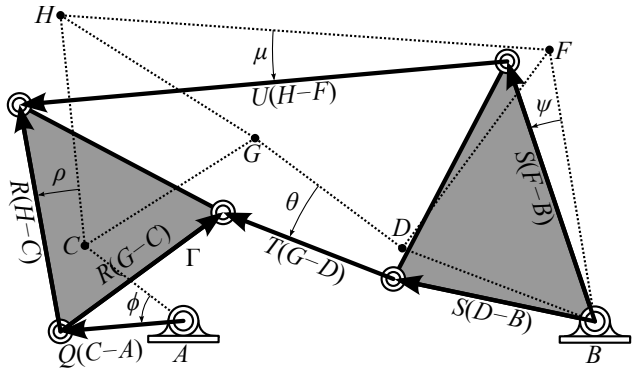



Figure 2.17: A Stephenson II linkage displaced from its reference configuration.

A Stephenson II linkage is defined by the locations of two fixed pivots, $A$, $B$, and the

locations of five moving pivots $C$, $D$, $F$, $G$, $H$, in a reference position. These seven pivots are connected according to Fig. 2.17 to form five moving links $AC$, $CGH$, $BDF$, $DG$, and $FH$. The orientations of these links are measured from their reference positions by angles $\phi$, $\rho$, $\psi$, $\theta$, and $\mu$, respectively, which define the rotation operators $Q$, $R$, $S$, $T$, $U$ as shown in Eqn. (2.100). The loop equations and their conjugates are formed by tracing the vector loops found in Fig. 2.17,

$$\begin{aligned}
\mathcal{L} &= A - B + Q(C - A) + R(G - C) - S(D - B) - T(G - D) = 0, \\
\bar{\mathcal{L}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{R}(\bar{G} - \bar{C}) - \bar{S}(\bar{D} - \bar{B}) - \bar{T}(\bar{G} - \bar{D}) = 0, \\
\mathcal{M} &= A - B + Q(C - A) + R(H - C) - S(F - B) - U(H - F) = 0, \\
\bar{\mathcal{M}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{C} - \bar{A}) + \bar{R}(\bar{H} - \bar{C}) - \bar{S}(\bar{F} - \bar{B}) - \bar{U}(\bar{H} - \bar{F}) = 0.
\end{aligned} \tag{2.108}$$

The variables are the rotation operators which must satisfy the normalization conditions shown in Eqn. (2.102). Eqns. (2.102) and (2.108) are nine constraints on ten unknowns. If $\mathbf{x} = \{Q, \bar{Q}\}$ is chosen as the input parameter and $\mathbf{y} = \{R,\ \bar{R},\ S,\ \bar{S},\ T,\ \bar{T},\ U,\ \bar{U}\}$ is the output, then the forward kinematics equations are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{M}, \bar{\mathcal{M}}, \mathcal{N}_R, \mathcal{N}_S, \mathcal{N}_T, \mathcal{N}_U\right\} = 0. \tag{2.109}$$

$\mathbf{F}(\mathbf{x}, \mathbf{y})$ has six roots $\mathbf{y}$ for a specified value of $\mathbf{x} = \{Q, \bar{Q}\}$ which correspond to six configurations of the mechanism. The equations change slightly if a different input $\mathbf{x}$ is chosen. For $\mathbf{x} = \{S, \bar{S}\}$, there are also six configurations.

### Stephenson III

A Stephenson III linkage is defined by the locations of three fixed pivots, $A$, $B$, $C$, and the locations of four moving pivots $D$, $F$, $G$, $H$, in a reference position. These seven pivots are

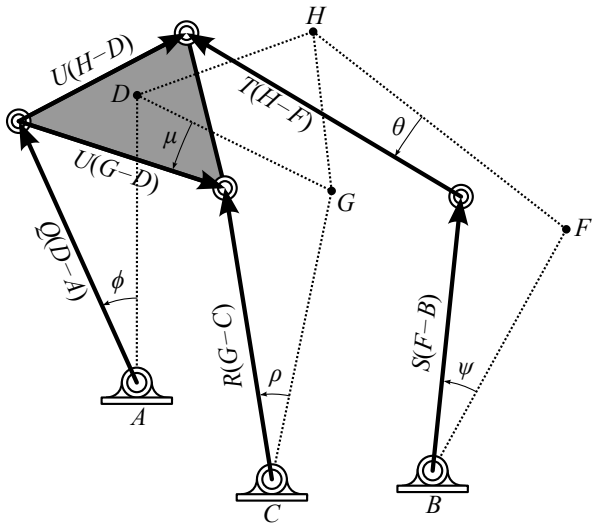



Figure 2.18: A Stephenson III linkage displaced from its reference configuration.

connected according to Fig. 2.18 to form five moving links $AD$, $CG$, $BF$, $FH$, and $DGH$. The orientations of these links are measured from their reference positions by angles $\phi$, $\rho$, $\psi$, $\theta$, and $\mu$, respectively, which define the rotation operators $Q$, $R$, $S$, $T$, $U$ as shown in Eqn. (2.100). The loop equations and their conjugates are formed by tracing the vector loops found in Fig. 2.18,

$$\begin{aligned}
\mathcal{L} &= A - C + Q(D - A) + U(G - D) - R(G - C) = 0, \\
\bar{\mathcal{L}} &= \bar{A} - \bar{C} + \bar{Q}(\bar{D} - \bar{A}) + \bar{U}(\bar{G} - \bar{D}) - \bar{R}(\bar{G} - \bar{C}) = 0, \\
\mathcal{M} &= A - B + Q(D - A) + U(H - D) - S(F - B) - T(H - F) = 0, \\
\bar{\mathcal{M}} &= \bar{A} - \bar{B} + \bar{Q}(\bar{D} - \bar{A}) + \bar{U}(\bar{H} - \bar{D}) - \bar{S}(\bar{F} - \bar{B}) - \bar{T}(\bar{H} - \bar{F}) = 0.
\end{aligned} \tag{2.110}$$

The variables are the rotation operators which must satisfy the normalization conditions shown in Eqn. (2.102). Eqns. (2.102) and (2.110) are nine constraints on ten unknowns. If $\mathbf{x} = \{Q, \bar{Q}\}$ is chosen as the input parameter and $\mathbf{y} = \{R, \bar{R}, S, \bar{S}, T, \bar{T}, U, \bar{U}\}$ is the output, then the forward kinematics equations are

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \left\{\mathcal{L}, \bar{\mathcal{L}}, \mathcal{M}, \bar{\mathcal{M}}, \mathcal{N}_R, \mathcal{N}_S, \mathcal{N}_T, \mathcal{N}_U\right\} = 0. \tag{2.111}$$

$\mathbf{F}(\mathbf{x}, \mathbf{y})$ has four roots $\mathbf{y}$ for a specified value of $\mathbf{x} = \{Q, \bar{Q}\}$ which correspond to four configurations of the mechanism. The equations change slightly if a different input $\mathbf{x}$ is chosen. For $\mathbf{x} = \{S, \bar{S}\}$, there are six configurations.

### 2.4.3 Sorting Solutions onto Trajectories

A six-bar linkage has forward kinematics equations $\mathbf{F}(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is the rotation of the input link and $\mathbf{y}$ lists output rotations. These equations are solved using the *NSolve* function of MATHEMATICA for $n$ input rotations evenly spaced around the unit circle,

$$\mathbf{x}_k = \left\{ \exp\left( (i2\pi) \frac{k-1}{n-1} \right), \exp\left( -(i2\pi) \frac{k-1}{n-1} \right) \right\}, \qquad k = 1, \ldots, n. \tag{2.112}$$

to obtain $m$ output configurations at each position $k$, listed as

$$(\mathbf{x}_k, \mathbf{y}_{k,p}), \qquad k = 1, \ldots, n, \;\; p = 1, \ldots, m, \tag{2.113}$$

which appear in no particular order with respect to index $p$. The objective of this section is to chain configurations together as we increment $k$ from 1 to $n$ so that they form separate trajectories of all the possible motions a mechanism can realize.

A six-bar linkage will have $m = 4$ or $m = 6$ depending on the type and specified input link. The choice of input link provides different parameterizations of the same configuration space and will define different locations of singular configurations in that space. Singular configurations are locations in the configuration space where the Jacobian $\det[J_{\mathbf{F}}(\mathbf{x}, \mathbf{y})] = 0$,

$$[J_{\mathbf{F}}(\mathbf{x}, \mathbf{y})] = \left[ \frac{\partial \mathbf{F}}{\partial y_1} \quad \ldots \quad \frac{\partial \mathbf{F}}{\partial y_8} \right]. \tag{2.114}$$

The configuration space of a mechanism may contain multiple disconnected curves, called *mechanism circuits*. If these curves contain singular points, they will divide them into segments, called *mechanism branches*. *Mechanism circuits* are independent of the selection of input link and *mechanism branches* are dependent on it. Each trajectory we piece together will either be bounded by singularities and indicate a mechanism branch, or contain no singularities and indicate a circuit defined by a fully rotatable crank.

The algorithm initializes by setting the $m$ elements of $\mathcal{C}_1$ as the beginning of $m$ trajectories which are built upon by comparing $\mathcal{C}_k$ to $\mathcal{C}_{k+1}$ and deciphering pairs of connecting configurations,

$$\begin{aligned} \mathcal{C}_k &= \{(\mathbf{x}_k, \mathbf{y}_{k,p}) \mid p = 1, \ldots, m\}, \\ \mathcal{C}_{k+1} &= \{(\mathbf{x}_{k+1}, \mathbf{y}_{k+1,q}) \mid q = 1, \ldots, m\}, \end{aligned} \tag{2.115}$$

where in general configurations $(\mathbf{x}_k, \mathbf{y}_{k,p})$ and $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1,q})$ connect such that $p \neq q$. To decipher connections between $\mathcal{C}_k$ and $\mathcal{C}_{k+1}$, we use Newton's method to solve $\mathbf{F}(\mathbf{x}_{k+1}, \mathbf{y}) = 0$ for $\mathbf{y}$ using start points $\mathbf{y}_{k,p}$, for $p = 1, \ldots, m$. We name these approximate solutions $\tilde{\mathbf{y}}_{k+1,p}$ where,

$$\begin{aligned} \tilde{\mathbf{y}}_{k+1,p} = \mathbf{y}_{k,p} - [J_{\mathbf{F}}(\mathbf{x}_{k+1}, \mathbf{y}_{k,p})]^{-1} \mathbf{F}(\mathbf{x}_{k+1}, \mathbf{y}_{k,p}), \\ p = 1, \ldots, m \end{aligned} \tag{2.116}$$

is calculated from a single Newton iteration. Multiple iterations are used for more accuracy. The approximate configuration set $\tilde{\mathcal{C}}_{k+1}$ is formed from $\tilde{\mathbf{y}}_{k+1,p}$ where

$$\tilde{\mathcal{C}}_{k+1} = \{(\mathbf{x}_{k+1}, \tilde{\mathbf{y}}_{k+1,p}) \mid p = 1, \ldots, m\}. \tag{2.117}$$

Configuration $(\mathbf{x}_k, \mathbf{y}_{k,p})$ of $\mathcal{C}_k$ connects to configuration $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1,q})$ of $\mathcal{C}_{k+1}$ if the following

condition evaluates as true,

$$|\tilde{\mathbf{y}}_{k+1,p} - \mathbf{y}_{k+1,q}| < \text{tol}, \tag{2.118}$$

where tol is a specified threshold value. For most $k$, configurations $\mathcal{C}_k$ and $\mathcal{C}_{k+1}$ will connect in a one to one fashion. However, Eqn. (2.118) allows the possibility that a configuration of $\mathcal{C}_k$ will connect to several or none of the configurations of $\mathcal{C}_{k+1}$, which is often the case near singularities. In these cases, we employ the following logic:

1. If a configuration of $\mathcal{C}_{k+1}$ is not connected to a configuration of $\mathcal{C}_k$, that configuration of $\mathcal{C}_{k+1}$ begins a new trajectory.

2. If a configuration of $\mathcal{C}_k$ connects to multiple configurations of $\mathcal{C}_{k+1}$, the trajectory associated with the configuration of $\mathcal{C}_k$ is duplicated and each duplicate connects to a matching element of $\mathcal{C}_{k+1}$.

3. If a configuration of $\mathcal{C}_k$ does not connect to any configurations of $\mathcal{C}_{k+1}$, the trajectory associated with the configuration of $\mathcal{C}_k$ is concluded.

This procedure is executed for a complete sweep of the unit circle $\mathbf{x}_k$, $k = 1, \ldots, n$, such that $\mathbf{x}_n = \mathbf{x}_1$. The result of this algorithm is a set of connected sequences of configurations that form separate mechanism trajectories. All combinations of these trajectories are checked for connections from $k = n$ to $k = 1$ configurations. If connections are identified, these trajectories are chained together to form longer trajectories. This step is particularly important for analyzing six-bars in which the input crank rotates 360° then begins a different motion on its next rotation. Many six-bars have been observed to be capable of this type of motion.

Finally, configurations that do not correspond to rigid body movement are removed, and the determinant of the Jacobian matrix along each configuration is evaluated. A sign change indicates a change in configuration that can arise from numerical error. Once mechanism

trajectories have been established, we can check if the entire motion requirement is accomplished on a single trajectory or split between several trajectories.

## 2.5 Cognate Linkages

The synthesis methods presented in this dissertation take advantage of sets of cognate linkages. *Cognate linkages* share the same kinematic structure but have different link length ratios that produce an identical motion at their output links. Cognate linkages, or simply cognates, are further classified according to the type of identical motion they produce. Dijksman names these classifications as *function-cognates*, *coupler-cognates*, and *curve-cognates* which refer to mechanisms that produce identical function generation, motion generation, and path generation. In addition, Dijksman lists *timed coupler-cognates* and *timed curve-cognates* which coordinate motion generation and path generation, respectively, with an input crank angle.

### 2.5.1 Four-bar Curve-Cognates

The most famous type of cognates are curve-cognates of the four-bar linkage. Furthermore, these cognates serve as building blocks for more complicated six-bar cognates. Four-bar curve-cognates come in sets of three and were first discovered independently by Roberts (1875) [56] and Chebyshev (1879) [57]. For a given four-bar linkage defined by $ACDB$ and trace point $P_0$, see Fig. 2.19, two other four-bar linkages $AC'D'B'$ and $B'C''D''B$ exist that also have trace point $P_0$ that traces the same coupler curve as $ACDB$. The construction of $AC'D'B'$ and $B'C''D''B$ proceeds below.

Consider the overconstrained mechanism shown in Fig. 2.19. Using the Grübler-Kutzbach criterion, its mobility is $-1$, however, since this mechanism contains special geometry as it

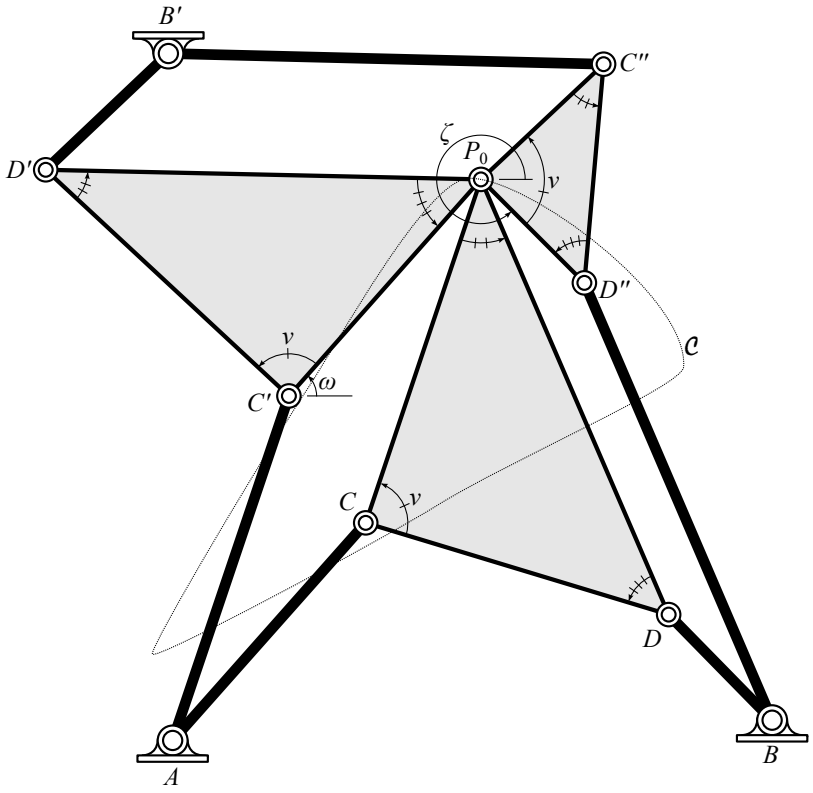



Figure 2.19: An overconstrained mechanism constructed from three four-bar curve-cognates.

is formed by four-bar cognates, it actually moves with mobility 1. The special geometry required for this motion is that $ACP_0C'$, $BDP_0D''$, and $B'D'P_0C''$ form parallelograms, and $P_0CD$, $D'C'P_0$, and $C''P_0D''$ are similar triangles. The mechanism of Fig. 2.19 can be divided into three individual four-bars, $ACDB$, $AC'D'B'$, and $B'C''D''B$, that share point $P_0$ which traces curve $\mathcal{C}$. When disconnected at $P_0$, each four-bar still traces $\mathcal{C}$.

For a given four-bar $ACDB$ that traces a coupler curve $\mathcal{C}$ at $P_0$, we can compute the pivot locations of its curve-cognates $AC'D'B'$ and $B'C''D''B$. We use complex numbers to handle this geometry, see Section 2.1.

The locations of $C'$ and $D''$ are found first through vector addition,

$$\begin{aligned} C' &= A + (P_0 - C), \\ D'' &= B + (P_0 - D). \end{aligned} \tag{2.119}$$

As well, we use vector addition to find expressions for $D'$ and $C''$,

$$\begin{aligned} D' &= C' + WV|D' - C'|, \\ C'' &= P_0 + ZV|C'' - P_0|, \end{aligned} \tag{2.120}$$

where $V = e^{i\nu}$, $W = e^{i\omega}$, and $Z = e^{i\zeta}$. In words, $D'$ is located by vector addition of $C'$ and the length of $D' - C'$ rotated $\omega + \nu$ from horizontal. And $C''$ is located by vector addition of $P_0$ and the length of $C'' - P_0$ rotated $\zeta + \nu$ from horizontal. In order to evaluate (2.120), we must express the complex rotations $V$, $W$, $Z$, and the magnitudes $|D' - C'|$ and $|C'' - P_0|$ in terms of knowns $A$, $B$, $C$, $D$, and $P_0$.

The rotation operator from one complex number to another is expressed in Eqn. (2.8) which is used to express $V$, $W$, $Z$ as

$$\begin{aligned} V = e^{i\nu} &= \sqrt{\frac{(P_0 - C)(\bar{D} - \bar{C})}{(\bar{P}_0 - \bar{C})(D - C)}}\ , \\ W = e^{i\omega} &= \sqrt{\frac{P_0 - C'}{\bar{P}_0 - \bar{C}'}}\ , \\ Z = e^{i\zeta} &= \sqrt{\frac{D'' - P_0}{\bar{D}'' - \bar{P}_0}}\ . \end{aligned} \tag{2.121}$$

The distances $|D' - C'|$ and $|C'' - P_0|$ are related to known parameters through length ratios of similar triangles,

$$\frac{|D' - C'|}{|P_0 - C|} = \frac{|P_0 - C'|}{|D - C|}, \qquad \frac{|C'' - P_0|}{|P_0 - C|} = \frac{|D'' - P_0|}{|D - C|}. \tag{2.122}$$

Using Eqn. (2.6), these distances are written as

$$|D' - C'| = \frac{|P_0 - C||P_0 - C'|}{|D - C|} = \sqrt{\frac{(P_0 - C)(\bar{P}_0 - \bar{C})(P_0 - C')(\bar{P}_0 - \bar{C}')}{(D - C)(\bar{D} - \bar{C})}},$$
$$|C'' - P_0| = \frac{|P_0 - C||D'' - P_0|}{|D - C|} = \sqrt{\frac{(P_0 - C)(\bar{P}_0 - \bar{C})(D'' - P_0)(\bar{D}'' - \bar{P}_0)}{(D - C)(\bar{D} - \bar{C})}}. \quad (2.123)$$

Plugging (2.121) and (2.123) into (2.120), pivot locations $D'$ and $C''$ simplify to

$$D' = C' + \frac{(P_0 - C)(P_0 - C')}{(D - C)}$$
$$C'' = P_0 + \frac{(P_0 - C)(D'' - P_0)}{(D - C)} \quad (2.124)$$

where all conjugate terms cancel out. Eqns. (2.119) are substituted into (2.124) in order to put $D'$ and $C''$ in terms of the original four-bar pivots,

$$D' = P_0 + (C - A)\left(\frac{P_0 - C}{D - C} - 1\right) , \quad (2.125)$$
$$C'' = P_0 + \frac{(B - D)(P_0 - C)}{(D - C)} . \quad (2.126)$$

In order to make these two equations look symmetric, we substitute $1 = \frac{D - C}{D - C}$ into Eqn. (2.125) and rearrange terms,

$$D' = \left(\frac{A - C}{D - C}\right)(D - P_0) + P_0 ,$$
$$C'' = \left(\frac{B - D}{C - D}\right)(C - P_0) + P_0 . \quad (2.127)$$

The final cognate pivot $B'$ is computed through vector addition,

$$B' = D' + (C'' - P_0) \quad (2.128)$$

Substituting in (2.127), we obtain

$$B' = \frac{(A-C)(D-P_0)-(B-D)(C-P_0)}{(D-C)} + P_0 \ . \tag{2.129}$$

To summarize, the cognate pivots are reprinted from Eqns. (2.119), (2.127), and (2.129),

$$\begin{aligned}
C' &= A - C + P_0, \\
D'' &= B - D + P_0, \\
D' &= \left(\frac{A-C}{D-C}\right)(D-P_0) + P_0 \ , \\
C'' &= \left(\frac{B-D}{C-D}\right)(C-P_0) + P_0 \ , \\
B' &= \frac{(A-C)(D-P_0)-(B-D)(C-P_0)}{(D-C)} + P_0 \ .
\end{aligned} \tag{2.130}$$

The four-bars $ACDB$, $AC'D'B'$, and $B'C''D''B$ are curve-cognates. Another important feature of the overconstrained linkage is that several links share same orientation, see Fig. 2.20. Since segments $BD$ and $C''D''$ are two opposing sides of a parallelogram they must share the same orientation $\psi$, therefore links $BD$ and $C''D''P_0$ must share that orientation $\psi$. Also, segments $B'D'$ and $C''P_0$ are opposing sides of a parallelogram, meaning links $B'D'$ and $C''D''P_0$ along with $BD$ share the orientation $\psi$. Likewise, links $AC$, $C'D'P_0$, $B'C''$ share the orientation $\phi$, and links $CDP_0$, $AC'$, $BD''$ share the orientation $\theta$.

Therefore, each set of four-bar curve-cognates yields three pairs of timed curve-cognates for each of angles $\phi$, $\psi$, $\theta$. For example, if $BD$ is the input link of four-bar $ACDB$ and $B'D'$ is the input link of four-bar $AC'D'B'$, then they both parameterize coupler curve $\mathcal{C}$ by $\psi$ and are called timed curve-cognates. As shown in Section 2.5.4, timed curve-cognates are important for determining function-cognates of the the Stephenson III.

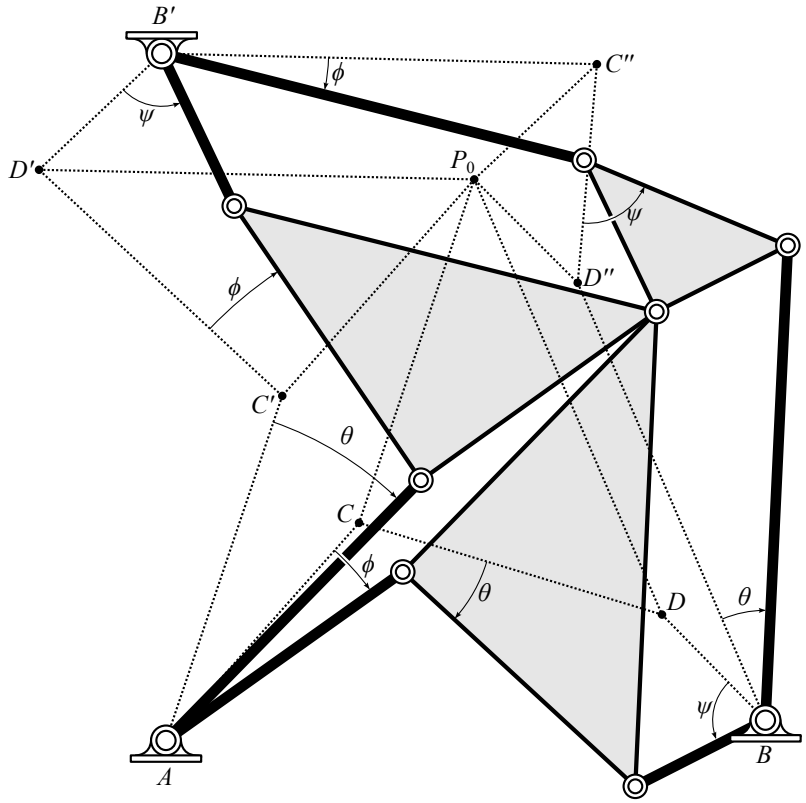



Figure 2.20: An overconstrained four-bar curve-cognate mechanism shown displaced from a reference configuration.

### 2.5.2 Watt II Function-Cognates

A Watt II function generator coordinates the angles $\phi$ and $\psi$ of links $AD$ and $BF$, see Fig. 2.21. This linkage has a two parameter family of function-cognates. To see this, consider the Watt II formed by four-bar loops $ADGC$ and $BFHC$ where, without loss of generality, we set the shared fixed pivot $C$ to be at the origin. The pivots of the second four-bar can be scaled and rotated with respect to the first. This action changes the link length ratios without altering the coordinated angles $(\phi, \psi)$, see Fig. 2.22. In other words, we can multiply pivots $B$, $F$, $H$ by a random complex number $Me^{i\nu}$, $M \neq 0$, without altering the roots of the loop equations.

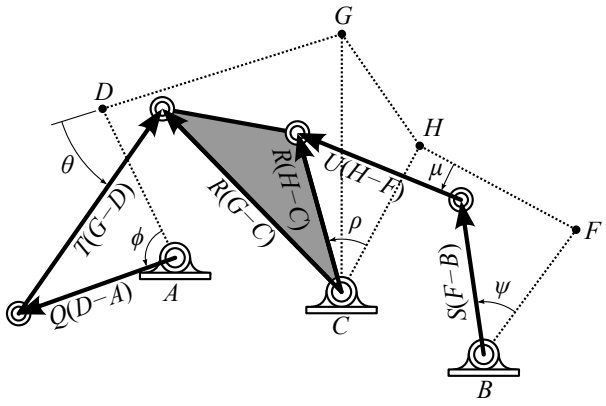



Figure 2.21: A Watt II linkage displaced from a reference configuration.

See that the loop equations before stretch-rotating the second four-bar are

$$\begin{aligned} &A + Q(D - A) + T(G - D) - RG = 0, \\ &B + S(F - B) + U(H - F) - RH = 0, \end{aligned} \tag{2.131}$$

and the loop equations after stretch-rotating the second four-bar are

$$\begin{aligned} &A + Q(D - A) + T(G - D) - RG = 0, \\ &Me^{i\nu}B + S(Me^{i\nu}F - Me^{i\nu}B) + U(Me^{i\nu}H - Me^{i\nu}F) - RMe^{i\nu}H = 0, \end{aligned} \tag{2.132}$$

where $Me^{i\nu}$ factors out and Eqns. (2.131) and (2.132) become identical.

$M$ and $\nu$ are two free parameters and can be chosen such that either $G = Me^{i\nu}H$ or $A = Me^{i\nu}B$. The former case creates a multijoint of the moving pivots on the ternary link (Fig. 2.22(d)), and the latter creates a multijoint of input and output pivots. In the Section 3.1, we specify $G = H$ to reduce the degree of the synthesis equations.

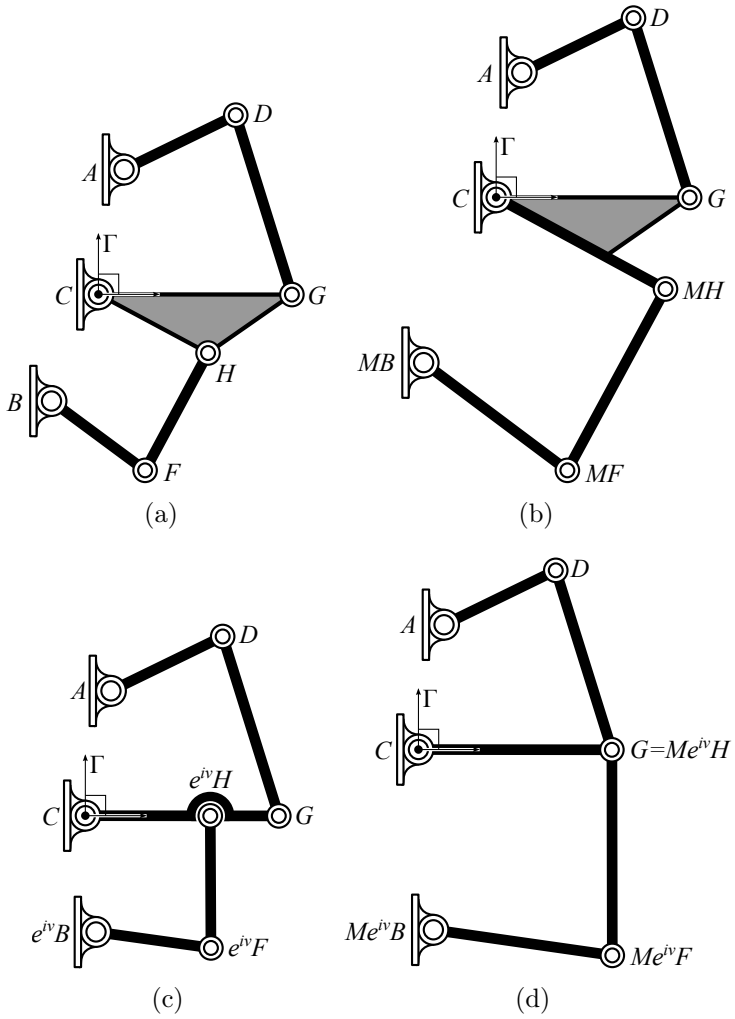
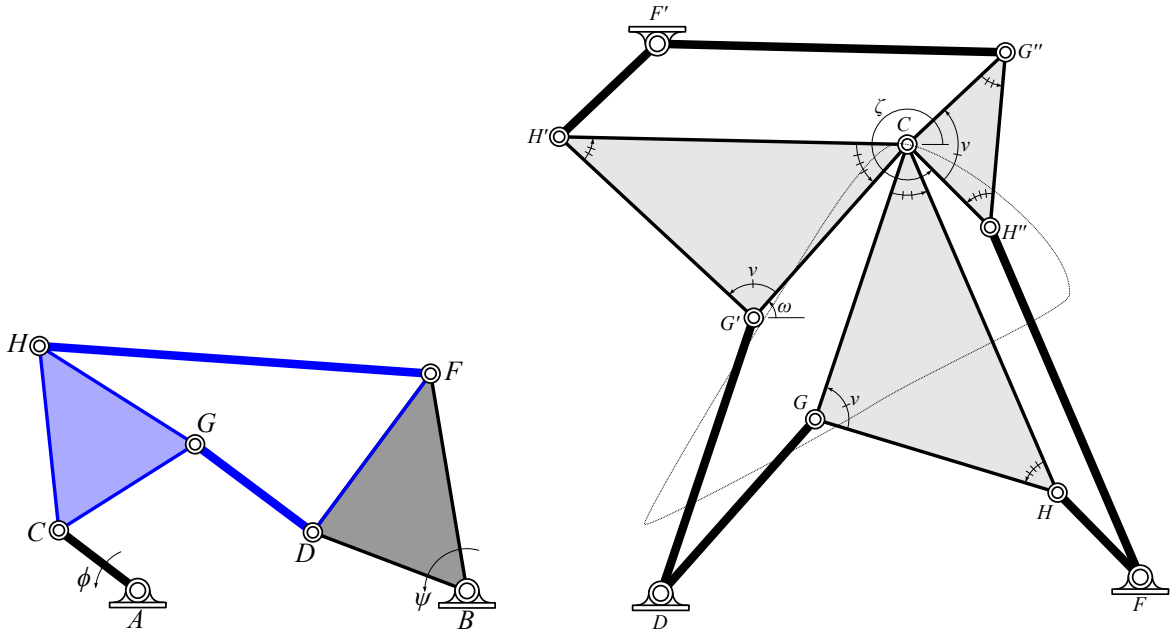



Figure 2.22: Four Watt II function-cognates created by rotating and scaling the pivot locations $B$, $F$, and $H$.

### 2.5.3 Stephenson II Function-Cognates

A Stephenson II function generator coordinates angles $\phi$ and $\psi$ of links $AC$ and $BDF$, see Fig. 2.23(a). Function-cognates of the Stephenson II linkage come in sets of three where each member corresponds to a curve-cognate of the floating four-bar subloop $DGHF$. To see this, note that the movement of pivot $C$ relative to Link $BDF$ creates a path that is traced by the coupler link of four-bar $DGHF$ that is attached to Link $BDF$. Therefore, all four-bar linkages that generate the same coupler curve of $C$ relative to link $BDF$ can replace $DGHF$ and control links $AC$ and $BDF$ in the same manner. These four-bar linkages can

be used to construct Stephenson II function generator cognates.




(a) Stephenson II (b) Four-bar curve-cognates

Figure 2.23: (a) The floating four-bar in a Stephenson II linkage is used to construct (b) three curve-cognates which in turn construct three Stephenson II function-cognates.

The Roberts-Chebyshev theorem says that four-bar curve-cognates come in sets of three. We have shown in Section 2.5.1 how to find the two other cognates from a single four-bar. For convenience, we reproduce Fig. 2.19 with pivot labels updated for the Stephenson II linkage in Fig. 2.23(b). The coupler cognate four-bar linkages are $DGHF$, $DG'H'F'$, and $F'G''H''F$.

Following Eqn. 2.130, the locations of cognate pivots $G'$, $H''$, $H'$, $G''$, $F'$ are

$$
\begin{aligned}
G' &= D - G + C \\
H'' &= F - H + C \\
H' &= \left(\frac{D-G}{H-G}\right)(H-C) + C \\
G'' &= \left(\frac{F-H}{G-H}\right)(G-C) + C \\
F' &= \frac{(D-G)(H-C) - (F-H)(G-C)}{(H-G)} + C
\end{aligned}
\tag{2.133}
$$

A Stephenson II linkage defined by pivots $A$, $B$, $C$, $D$, $G$, $H$, $F$, will have two function cognates with link lengths of different ratios defined by pivots $A$, $B$, $C$, and

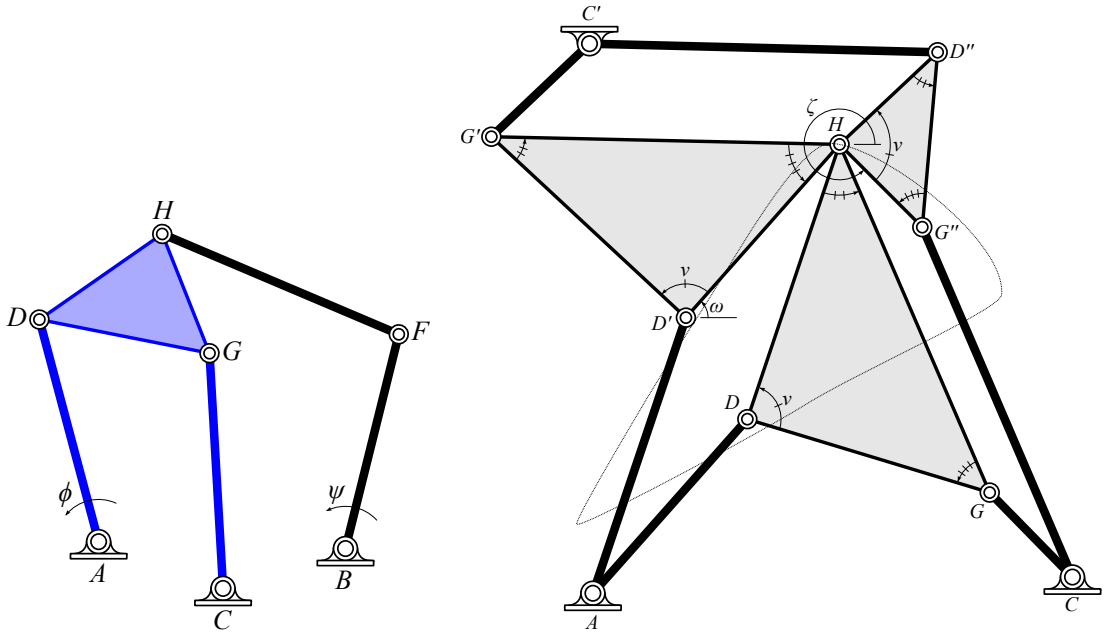
$$
\left(\begin{array}{ll} D_{c1} = D & D_{c1} = F' \\ G_{c1} = G' \quad \text{or} & G_{c1} = H' \\ H_{c1} = H' & H_{c1} = G' \\ F_{c1} = F' & F_{c1} = D \end{array}\right) \quad \text{and} \quad \left(\begin{array}{ll} D_{c2} = F' & D_{c2} = F \\ G_{c2} = G'' \quad \text{or} & G_{c2} = H'' \\ H_{c2} = H'' & H_{c2} = G'' \\ F_{c2} = F & F_{c2} = F' \end{array}\right) \tag{2.134}
$$

Note that the parenthesized "or" pairs are the same linkages with pivot labels swapped.

### 2.5.4 Stephenson III Function-Cognates

A Stephenson III function generator coordinates the angles $\phi$ and $\psi$ of links $AD$ and $BF$. see Fig. 2.24(a). Function-cognates of the Stephenson III linkage come in sets of two which correspond to a pair of timed curve-cognates of the four-bar subloop $ADGC$. To see this, note that a Stephenson III linkage can be viewed as a four-bar linkage $ADGC$ that controls the motion of RR dyad $BF$ by connecting the final link of the RR dyad to the coupler of the four-bar with a revolute joint at $H$. Therefore, all four-bar linkages that generate the same coupler curve at $H$ can replace $ADGC$ and control the motion of dyad $BF$ in the same

manner. There are three four-bar curve-cognates capable of this shown in Fig. 2.24(b). Fig. 2.24(b) is a reproduction of Fig. 2.19 with labelling updated for the Stephenson III.




(a) Watt I (b) Watt II

Figure 2.24: (a) The four-bar in a Stephenson III linkage is used to construct (b) three curve-cognates of which two are used to construct Stephenson III function-cognates.

However, the Stephenson III four-bar loop includes the link of coordinated angle $\phi$. Therefore, in order to form a six-bar function-cognate, $ADGC$ can only be replaced by a four-bar which also parameterizes its coupler curve by input angle $\phi$. The timed curve-cognate $C'D''G''C$ satisfies this criterion as it parameterizes the trajectory of $H$ by the angle $\phi$ of its input link $C'D''$, see Section 2.5.1.

For a given four-bar $ADGC$, the locations of $G''$, $D''$, $C'$, are calculated following Eqn. (2.130),

$$
\begin{aligned}
G'' &= C - G + H, \\
D'' &= \left(\frac{C-G}{D-G}\right)(D-H) + H \ , \\
C' &= \frac{(A-D)(G-H) - (C-G)(D-H)}{(G-D)} + H \ .
\end{aligned} \tag{2.135}
$$

The four-bar linkage $C'D''G''C$ connected to $BF$ at $H$ forms a Stephenson III function-cognate.

However, function generators can scale, rotate, and translate in the plane and maintain coordination between their input-output angles. Furthermore, the synthesis routine presented in Section 5 returns results where pivots $A$ and $D$ are at specified locations. In order to check these results, we need to transform the six-bar defined by $C'$, $D''$, $G''$, $C$, $B$, $F$, $H$ such that pivot $C'$ coincides with point $A$, and pivot $D''$ coincides with point $D$. The transformation which completes this action is derived from Eqn. (2.14),

$$\mathcal{T}(p) = \frac{D - A}{D'' - C'}(p - C') + A. \tag{2.136}$$

First, we apply $\mathcal{T}$ to $C'$ and $D''$,

$$\mathcal{T}(C') = A, \tag{2.137}$$

$$\mathcal{T}(D'') = D, \tag{2.138}$$

which verifies that it is the desired transformation. Next, before $\mathcal{T}$ is applied to $G''$, $C$, $H$, $B$, and $F$, it is convenient to simplify that transformation by substituting Eqns. (2.135) into (2.136). After several cancellations, $\mathcal{T}(p)$ simplifies to

$$\mathcal{T}(p) = \frac{G - D}{H - G}(H - p) + \frac{(C - G)(H - D)}{(H - G)} + D, \tag{2.139}$$

which can be rewritten as

$$\mathcal{T}(p) = \frac{G - D}{H - G}(H - p) + (C - G)\left(\frac{G - D}{H - G} + \frac{H - G}{H - G}\right) + D. \tag{2.140}$$

Substituting in $\frac{H-G}{H-G}=1$, transformation $\mathcal{T}(p)$ simplifies to

$$\mathcal{T}(p)=\frac{D-G}{H-G}(p-C)+C \tag{2.141}$$

Eqn. (2.141) can be interpreted as a stretch rotation around fixed point $C$. Applying it to $G''$, $C$, $H$, $B$, $F$, we obtain

$$\begin{aligned}
&\mathcal{T}(G'')=D-G+C,\\
&\mathcal{T}(C)=C,\\
&\mathcal{T}(H)=\frac{D-G}{H-G}(H-C)+C,\\
&\mathcal{T}(B)=\frac{D-G}{H-G}(B-C)+C,\\
&\mathcal{T}(F)=\frac{D-G}{H-G}(F-C)+C.
\end{aligned} \tag{2.142}$$

A Stephenson III linkage defined by pivots $A$, $C$, $D$, $G$, $H$, $B$, $F$, will have a function-cognate defined by pivots $A$, $C$, $D$, and

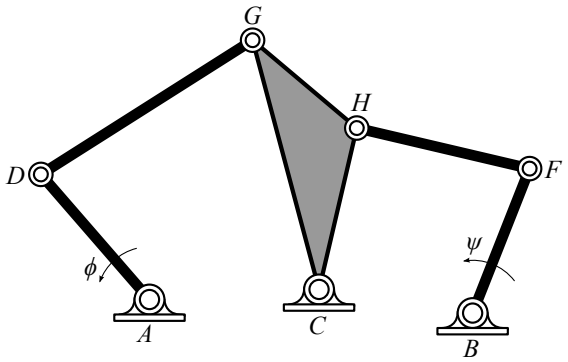
$$\begin{aligned}
&G_c=D-G+C,\\
&H_c=\frac{D-G}{H-G}(H-C)+C,\\
&B_c=\frac{D-G}{H-G}(B-C)+C,\\
&F_c=\frac{D-G}{H-G}(F-C)+C.
\end{aligned} \tag{2.143}$$

# Chapter 3

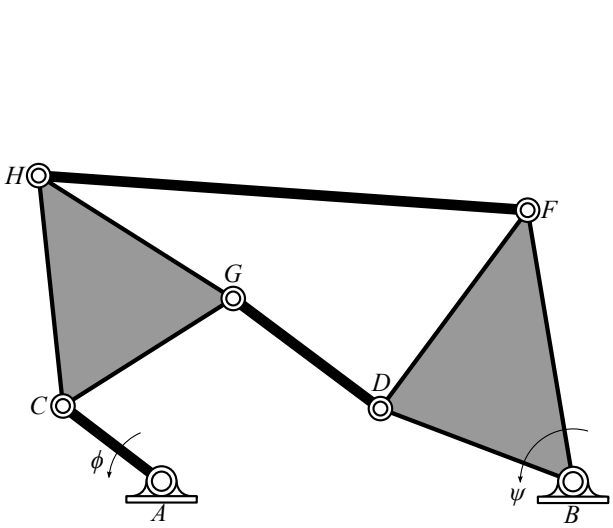
# Watt II Function Generation

The Watt II function generator is one of three types of six-bar linkages that provide capabilities beyond the four-bar linkage. The others are the Stephenson II and III, see Fig, 3.1. The objective of function generation is to coordinate the angles $\phi$ and $\psi$ shown in Fig. 3.1 where $\phi$ and $\psi$ are measured from some initial reference configuration. For a given set of $N$ function points $(x_j, y_j)$, $j = 0, \ldots, N-1$, we assign a set of angle pairs $(\phi_j, \psi_j) = (x_j, y_j)$. For the Stephenson II and III mechanisms, we may also assign $(\psi_j, \phi_j) = (x_j, y_j)$ in order to produce more function generator options.

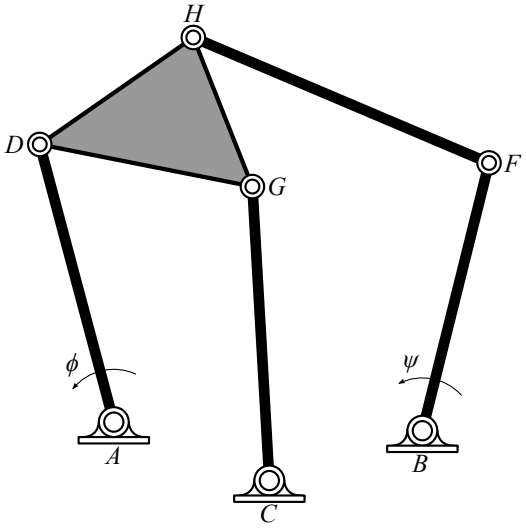
A Watt II mechanism consists of three ground pivots $A$, $B$, $C$, and four moving pivots $D$, $F$, $G$, and $H$, see Fig. 3.2. Their are five moving links $AD$, $BF$, $CGH$, $DG$, and $FH$. The objective is to coordinate the rotations between $\phi$ and $\psi$ of links $AD$ and $BF$. In order to set the scale, orientation, and location of the linkage, the ground pivot $C$ is chosen to be at the origin $C = 0 + i0$, and the initial position of moving pivot $G$ is chosen to be $G = 1 + i0$. Also pivot $H$ must be set in order to specify a unique solution from the two parameter family of cognate linkages that characterize Watt II function generators.




(a) Watt II




(b) Stephenson II




(c) Stephenson III

Figure 3.1: The six-bar linkages useful for function generation.

## 3.1 Synthesis Formulation

Formulation of the synthesis equations begins by constructing the loop closure equations. From Fig. 3.2(b), we construct two sets of loop closure equations,

$$A - C + Q_j(D - A) + T_j(G - D) - R_j(G - C) = 0, \tag{3.1}$$

$$B - C + S_j(F - B) + U_j(H - F) - R_j(H - C) = 0, \quad j = 1, \ldots, N - 1, \tag{3.2}$$

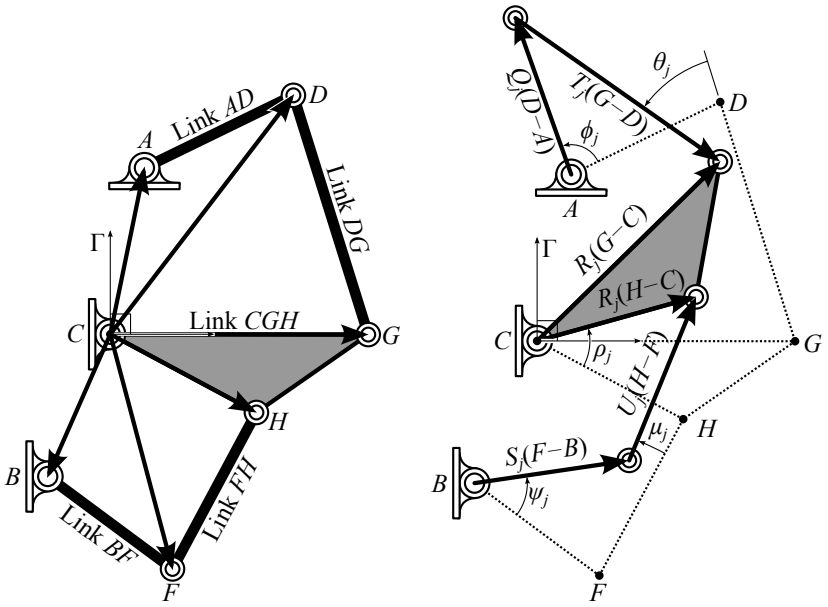



Figure 3.2: A Watt II function generator in (a) its reference configuration and (b) its $j^{\text{th}}$ configuration.

and two sets of conjugate loop closure equations,

$$\bar{A} - \bar{C} + \bar{Q}_j(\bar{D} - \bar{A}) + \bar{T}_j(\bar{G} - \bar{D}) - \bar{R}_j(\bar{G} - \bar{C}) = 0, \tag{3.3}$$

$$\bar{B} - \bar{C} + \bar{S}_j(\bar{F} - \bar{B}) + \bar{U}_j(\bar{H} - \bar{F}) - \bar{R}_j(\bar{H} - \bar{C}) = 0, \quad j = 1, \ldots, N-1, \tag{3.4}$$

where $j$ indexes task positions. These equations contain the complex rotation operators $Q_j$, $R_j$, $S_j$, $T_j$, $U_j$, and their conjugates defined as

$$\begin{aligned} Q_j &= e^{i\phi_j}, \qquad R_j = e^{i\rho_j}, \qquad S_j = e^{i\psi_j}, \qquad T_j = e^{i\theta_j}, \qquad U_j = e^{i\mu_j}, \\ \bar{Q}_j &= e^{-i\phi_j}, \qquad \bar{R}_j = e^{-i\rho_j}, \qquad \bar{S}_j = e^{-i\psi_j}, \qquad \bar{T}_j = e^{-i\theta_j}, \qquad \bar{U}_j = e^{-i\mu_j}. \end{aligned} \tag{3.5}$$

The variables $Q_j$, $\bar{Q}_j$, $S_j$ and $\bar{S}_j$ specify the task. The variables $R_j$, $\bar{R}_j$, $T_j$, $\bar{T}_j$, $U_j$, and $\bar{U}_j$ remain as unknowns. In order for these unknowns to correspond to rotational operators, it

is necessary that they satisfy the normalization conditions,

$$R_j \bar{R}_j = 1, \tag{3.6}$$

$$T_j \bar{T}_j = 1, \tag{3.7}$$

$$U_j \bar{U}_j = 1, \quad j = 1, \ldots, N-1. \tag{3.8}$$

Eqns. (3.1)–(3.4) and (3.6)–(3.8) comprise a system of $7(N-1)$ equations. The unknowns of these equations are

$$\left\langle A, \bar{A}, B, \bar{B}, D, \bar{D}, F, \bar{F} \right\rangle \text{ and } \left\langle R_j, \bar{R}_j, \bar{T}_j, \bar{T}_j, \bar{U}_j, \bar{U}_j, \right\rangle, \quad j = 1, \ldots, N-1. \tag{3.9}$$

Pivot locations $(C, \bar{C})$, $(G, \bar{G})$, and $(H, \bar{H})$ are not included as unknowns. They are specified in order to identify a unique solution. This leaves $8 + 6(N-1)$ unknowns, and the system becomes square for $N = 9$ where the degree is $2^{56} \approx 7.21 \times 10^{16}$.

In order to reduce the degree of the system, the unknowns $T_j$ and $\bar{T}_j$ are eliminated by substituting Eqns. (3.1) and (3.3) into (3.7) to obtain

$$\begin{aligned}&(A - C + Q_j(D - A) - R_j(G - C)) \\ &\quad \times (\bar{A} - \bar{C} + \bar{Q}_j(\bar{D} - \bar{A}) - \bar{R}_j(\bar{G} - \bar{C})) = (G - D)(\bar{G} - \bar{D}).\end{aligned} \tag{3.10}$$

Similarly, $U_j$ and $\bar{U}_j$ are eliminated by substituting Eqns. (3.2) and (3.4) into (3.8) to obtain

$$\begin{aligned}&(B - C + Q_j(F - B) - R_j(H - C)) \\ &\quad \times (\bar{B} - \bar{C} + \bar{Q}_j(\bar{F} - \bar{B}) - \bar{R}_j(\bar{H} - \bar{C})) = (H - F)(\bar{H} - \bar{F}).\end{aligned} \tag{3.11}$$

Eqns. (3.10) and (3.11) are expanded and written in matrix form

$$\begin{bmatrix} a\bar{b}_j & \bar{a}b_j \\ c\bar{d}_j & \bar{c}d_j \end{bmatrix} \begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - a\bar{a} - d_j\bar{d}_j \end{Bmatrix}, \qquad j = 1, \ldots, N-1, \tag{3.12}$$

where

$$\begin{aligned} a &= -(G-C), \qquad & b_j &= A - C + Q_j(D-A), \qquad & f &= G - D, \\ c &= -(H-C), \qquad & d_j &= B - C + S_j(F-B), \qquad & g &= H - F. \end{aligned} \tag{3.13}$$

Note that the variables $b_j$ and $d_j$ take the form

$$\begin{aligned} b_j &= h + Q_j k + S_j l, \\ d_j &= m + Q_j n + S_j o, \end{aligned} \tag{3.14}$$

where $l = n = 0$. Eqn. (3.12) can be solved for $(R_j, \bar{R}_j)$,

$$\begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \frac{1}{a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j} \begin{bmatrix} \bar{c}d_j & -\bar{a}b_j \\ -c\bar{d}_j & a\bar{b}_j \end{bmatrix} \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - c\bar{c} - d_j\bar{d}_j \end{Bmatrix}, \quad j = 1, \ldots, N-1, \tag{3.15}$$

which can be substituted into Eqn. (3.6) to obtain

$$\begin{aligned} &(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) \\ &\quad \times (\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \ldots, N-1. \end{aligned} \tag{3.16}$$

Eqn. (3.16) are the synthesis equations and have degree $8^8 = 16,777,216$ for the maximum number of $N = 9$ task positions.

To further explore the structure of these equations, they are factored into the form

$$\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, N-1. \tag{3.17}$$

where all specified task information is contained in matrix $[\hat{Q}_j]$ and all link dimensions are contained in vector $\mathbf{p}$ which remains constant between task positions. This factorization is described in Appendix A. Matrix $[\hat{Q}_j]$ is defined as

$$[\hat{Q}_j] = \begin{bmatrix} \bar{\mathbf{q}}_j {\mathbf{q}_j}^T & [0] \\ [0] & -\bar{\mathbf{q}}'_j {\mathbf{q}'_j}^T \end{bmatrix} \tag{3.18}$$

where

$$\begin{aligned} \mathbf{q}_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j, Q_j S_j, Q_j^2, S_j^2, Q_j^2\bar{S}_j, \bar{Q}_j S_j^2\right\}^T, \\ \mathbf{q}'_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j\right\}^T. \end{aligned} \tag{3.19}$$

Vector $\mathbf{p}$ is defined with variables

$$\begin{aligned} &a = -(G-C), \quad f = G-D, \quad h = A-C, \quad k = D-A, \quad l = 0, \\ &c = -(H-C), \quad g = H-F, \quad m = B-C, \quad n = 0, \quad o = F-B, \end{aligned} \tag{3.20}$$

from which we observe that

$$l = 0, \qquad n = 0, \qquad -f = a+h+k, \qquad -g = c+m+o \tag{3.21}$$

which we substitute into the components of $\mathbf{p}$ to obtain

$$
\begin{aligned}
p_1 &= \bar{h}a\xi - \bar{m}c\beta & p_{11} &= 0 \\
p_2 &= \bar{k}(a\xi + ch\bar{m}) & p_{12} &= 0 \\
p_3 &= -\bar{o}(c\beta + a\bar{h}m) & p_{13} &= a\bar{c}\bar{h}m - \bar{a}ch\bar{m} \\
p_4 &= \bar{h}\bar{m}ck & p_{14} &= a\bar{c}\bar{k}m \\
p_5 &= -\bar{h}\bar{m}ao & p_{15} &= -\bar{a}ch\bar{o} \\
p_6 &= -\bar{k}\bar{m}ao & p_{16} &= -\bar{a}ck\bar{m} \\
p_7 &= \bar{h}\bar{o}ck & p_{17} &= a\bar{c}\bar{h}o \\
p_8 &= \bar{k}\bar{o}(ch - am) & p_{18} &= a\bar{c}\bar{k}o \\
p_9 &= 0 & p_{19} &= -\bar{a}ck\bar{o} \\
p_{10} &= 0 & & \\
\text{where} \quad \beta &= h\bar{k} + \bar{h}k - a(\bar{f} + \bar{a}) - \bar{a}(f + a) & & \\
\xi &= m\bar{o} + \bar{m}o - a(\bar{g} + \bar{c}) - \bar{a}(g + c) & &
\end{aligned}
\tag{3.22}
$$

As observed in Section 2.5.2, the moving pivots of the ternary link $CGH$ can be specified as a multijoint without loss of generality. Specifying $G = H$ means $a = c$ which simplifies

Eqn. (3.22) to

$$
\begin{aligned}
p_1 &= a(\bar{h}\xi - \bar{m}\beta) & p_{11} &= 0 \\
p_2 &= a(\bar{k}\xi + h\bar{k}\bar{m}) & p_{12} &= 0 \\
p_3 &= a(-\bar{o}\beta - \bar{h}m\bar{o}) & p_{13} &= a\bar{a}(\bar{h}m - h\bar{m}) \\
p_4 &= a\bar{h}k\bar{m} & p_{14} &= a\bar{a}\bar{k}m \\
p_5 &= -a\bar{h}\bar{m}o & p_{15} &= -a\bar{a}h\bar{o} \\
p_6 &= -a\bar{k}\bar{m}o & p_{16} &= -a\bar{a}k\bar{m} \\
p_7 &= a\bar{h}k\bar{o} & p_{17} &= a\bar{a}\bar{h}o \\
p_8 &= a(h\bar{k}\bar{o} - \bar{k}m\bar{o}) & p_{18} &= a\bar{a}\bar{k}o \\
p_9 &= 0 & p_{19} &= -a\bar{a}k\bar{o} \\
p_{10} &= 0
\end{aligned}
\tag{3.23}
$$

In (3.23), we see that $a$ factors out of every element of $\mathbf{p}$ so that $a\bar{a}$ factors out of (3.17) creating a set of degree 6 polynomials with a total degree of $6^8 = 1,679,616$ for the case $N = 9$. Therefore we omit the factored $a$ from $\mathbf{p}$. Next, we define four new variables

$$
\begin{aligned}
r_1 &= h\bar{k}, & r_2 &= m\bar{o}, \\
\bar{r}_1 &= \bar{h}k & \bar{r}_2 &= \bar{m}o
\end{aligned}
\tag{3.24}
$$

and substitute these into (3.23)

$$
\begin{aligned}
p_1 &= \bar{h}\xi - \bar{m}\beta & p_{11} &= 0 \\
p_2 &= \bar{k}\xi + \bar{m}r_1 & p_{12} &= 0 \\
p_3 &= -\bar{o}\beta - \bar{h}r_2 & p_{13} &= \bar{a}(\bar{h}m - h\bar{m}) \\
p_4 &= \bar{m}\bar{r}_1 & p_{14} &= \bar{a}\bar{k}m \\
p_5 &= -\bar{h}\bar{r}_2 & p_{15} &= -\bar{a}h\bar{o} \\
p_6 &= -\bar{k}\bar{r}_2 & p_{16} &= -\bar{a}k\bar{m} \\
p_7 &= \bar{o}\bar{r}_1 & p_{17} &= \bar{a}\bar{h}o \\
p_8 &= \bar{o}r_1 - \bar{k}r_2 & p_{18} &= \bar{a}\bar{k}o \\
p_9 &= 0 & p_{19} &= -\bar{a}k\bar{o} \\
p_{10} &= 0 & & \\
\text{where} \quad \beta &= r_1 + \bar{r}_1 - a(\bar{f} + \bar{a}) - \bar{a}(f + a) & & \\
\xi &= r_2 + \bar{r}_2 - a(\bar{g} + \bar{c}) - \bar{a}(g + c). & &
\end{aligned}
\tag{3.25}
$$

In order to set the scale, orientation, and location of the function generator in the plane, we specify $(C, \bar{C}) = (0, 0)$ and $(G, \bar{G}) = (1, 1)$ which means $(a, \bar{a})$ are known. Therefore every element of $\mathbf{p}$ in (3.25) is degree 2. Eqns. (3.17) then reduce to degree 4 polynomials. Combining (3.17) with (3.24) creates a system of $3 + N$ equations and 12 unknowns which for $N = 9$ has a total degree of $4^8 2^4 = 1,048,576$.

Finally, we substitute in the original design parameters of (3.20) and $(C, \bar{C}) = (0, 0)$ and $(G, \bar{G}) = (1, 1)$ to obtain our final synthesis equations,

$$
\mathbf{p}^T [\hat{Q}_j] \bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, 8, \tag{3.26}
$$

where

$$
\begin{aligned}
p_1 &= \bar{A}\xi - \bar{B}\beta & p_{11} &= 0 \\
p_2 &= (\bar{D} - \bar{A})\xi + \bar{B}r_1 & p_{12} &= 0 \\
p_3 &= -(\bar{F} - \bar{B})\beta - \bar{A}r_2 & p_{13} &= (A\bar{B} - \bar{A}B) \\
p_4 &= \bar{B}\bar{r}_1 & p_{14} &= -(\bar{D} - \bar{A})B \\
p_5 &= -\bar{A}\bar{r}_2 & p_{15} &= A(\bar{F} - \bar{B}) \\
p_6 &= -(\bar{D} - \bar{A})\bar{r}_2 & p_{16} &= (D - A)\bar{B} \\
p_7 &= (\bar{F} - \bar{B})\bar{r}_1 & p_{17} &= -\bar{A}(F - B) \\
p_8 &= (\bar{F} - \bar{B})r_1 - (\bar{D} - \bar{A})r_2 & p_{18} &= -(\bar{D} - \bar{A})(F - B) \\
p_9 &= 0 & p_{19} &= (D - A)(\bar{F} - \bar{B}) \\
p_{10} &= 0 & & \\
\text{where} \quad \beta &= r_1 + \bar{r}_1 - D - \bar{D} & & \\
\xi &= r_2 + \bar{r}_2 - F - \bar{F} & &
\end{aligned}
\tag{3.27}
$$

and

$$
\begin{aligned}
r_1 &= A(\bar{D} - \bar{A}), & r_2 &= B(\bar{F} - \bar{B}), \\
\bar{r}_1 &= \bar{A}(D - A) & \bar{r}_2 &= \bar{B}(F - B).
\end{aligned}
\tag{3.28}
$$

A multihomogeneous root count is performed on the system by selecting the variable groups

$$
\langle A, \bar{A}, D, \bar{D}, r_1, \bar{r}_1 \rangle, \quad \langle B, \bar{B}, F, \bar{F}, r_2, \bar{r}_2 \rangle, \tag{3.29}
$$

which determine the multihomogeneous Bézout number of 286,720. The root count was performed by expanding the polynomial

$$
16\alpha_1^2\alpha_2^2(2\alpha_1 + 2\alpha_2)^8 \tag{3.30}
$$

and taking the coefficient in front of the monomial $\alpha_1^6\alpha_2^6$. This combinatoric procedure is outlined in Section 2.3.2.

The solutions to the synthesis equations can be found using homotopy algorithms. In this dissertation, we only present these synthesis equations and focus more on the Stephenson II and III function generators that are capable of $N = 11$ task positions.

# Chapter 4

# Stephenson II Function Generation

A Stephenson II mechanism consists of two ground pivots $A$, $B$, and five moving pivots $C$, $D$, $F$, $G$, and $H$, see Fig. 4.1. There are five moving links $AC$, $BDF$, $CGH$, $DG$, and $FH$. The objective is to coordinate the rotations between $\phi$ and $\psi$ of links $AC$ and $BDF$. In order to set the scale, orientation, and location of the linkage, the ground pivots $A$ and $B$ are chosen to be at $A = 0 + i0$ and $B = 1 + i0$.

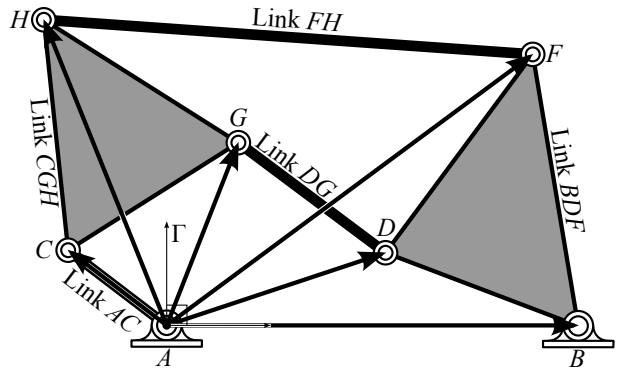
## 4.1 Synthesis Formulation

Formulation of the synthesis equations begins by constructing the loop closure equations. From Fig. 4.1(b), we construct two sets of loop closure equations,

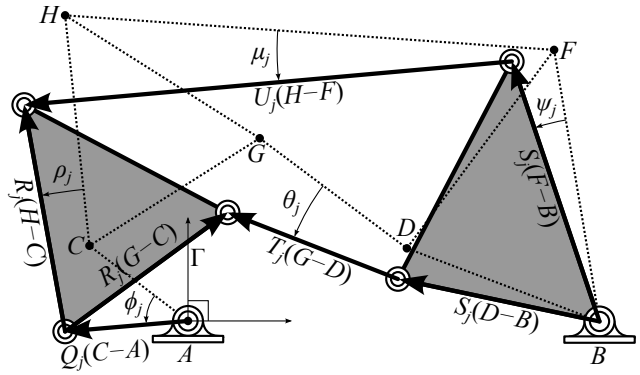
$$A - B + Q_j(C - A) + R_j(G - C) - S_j(D - B) - T_j(G - D) = 0, \tag{4.1}$$

$$A - B + Q_j(C - A) + R_j(H - C) - S_j(F - B) - U_j(H - F) = 0, \tag{4.2}$$

$$j = 1, \ldots, N - 1,$$




(a)




(b)

Figure 4.1: A Stephenson II function generator in (a) its reference configuration and (b) its $j^{\text{th}}$ configuration.

and two sets of conjugate loop equations,

$$\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{R}_j(\bar{G} - \bar{C}) - \bar{S}_j(\bar{D} - \bar{B}) - \bar{T}_j(\bar{G} - \bar{D}) = 0, \tag{4.3}$$

$$\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{R}_j(\bar{H} - \bar{C}) - \bar{S}_j(\bar{F} - \bar{B}) - \bar{U}_j(\bar{H} - \bar{F}) = 0, \tag{4.4}$$

$$j = 1, \ldots, N-1,$$

where $j$ indexes task positions. These equations contain the complex rotation operators $Q_j$, $R_j$, $S_j$, $T_j$, $U_j$, and their conjugates defined as

$$\begin{aligned} Q_j &= e^{i\phi_j}, \qquad R_j = e^{i\rho_j}, \qquad S_j = e^{i\psi_j}, \qquad T_j = e^{i\theta_j}, \qquad U_j = e^{i\mu_j}, \\ \bar{Q}_j &= e^{-i\phi_j}, \qquad \bar{R}_j = e^{-i\rho_j}, \qquad \bar{S}_j = e^{-i\psi_j}, \qquad \bar{T}_j = e^{-i\theta_j}, \qquad \bar{U}_j = e^{-i\mu_j}. \end{aligned} \tag{4.5}$$

The variables $Q_j$, $\bar{Q}_j$, $S_j$ and $\bar{S}_j$ specify the task. The variables $R_j$, $\bar{R}_j$, $T_j$, $\bar{T}_j$, $U_j$, and $\bar{U}_j$ remain as unknowns. In order for these unknowns to correspond to rotational operators, it is necessary that they satisfy the normalization conditions,

$$R_j\bar{R}_j = 1, \tag{4.6}$$

$$T_j\bar{T}_j = 1, \tag{4.7}$$

$$U_j\bar{U}_j = 1, \quad j = 1, \ldots, N-1. \tag{4.8}$$

Eqns. (4.1)–(4.4) and (4.6)–(4.8) comprise a system of $7(N-1)$ equations. The unknowns of these equations are

$$\left\langle C, \bar{C}, D, \bar{D}, F, \bar{F}, G, \bar{G}, H, \bar{H} \right\rangle \text{ and } \left\langle R_j, \bar{R}_j, \bar{T}_j, \bar{T}_j, \bar{U}_j, \bar{U}_j, \right\rangle, \quad j = 1, \ldots, N-1. \tag{4.9}$$

Pivot locations $(A, \bar{A})$ and $(B, \bar{B})$ are not included as unknowns. They are specified in order to set the scale, orientation, and location of the function generator in the plane. This leaves $10 + 6(N-1)$ unknowns, and the system becomes square for $N = 11$ where the degree is $2^{70} \approx 1.18 \times 10^{21}$.

In order to reduce the system, the unknowns $T_j$ and $\bar{T}_j$ are eliminated by substituting Eqns. (4.1) and (4.3) into (4.7) to obtain

$$\begin{aligned}&(A - B + Q_j(C - A) - S_j(D - B) + R_j(G - C)) \\ &\quad \times (\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) - \bar{S}_j(\bar{D} - \bar{B}) + \bar{R}_j(\bar{G} - \bar{C})) = (G - D)(\bar{G} - \bar{D})\end{aligned} \tag{4.10}$$

Similarly, $U_j$ and $\bar{U}_j$ are eliminated by substituting Eqns. (4.2) and (4.4) into (4.8) to obtain

$$\begin{aligned}&(A - B + Q_j(C - A) - S_j(F - B) + R_j(H - C)) \\ &\quad \times (\bar{A} - \bar{B} + \bar{Q}_j(\bar{C} - \bar{A}) - \bar{S}_j(\bar{F} - \bar{B}) + \bar{R}_j(\bar{H} - \bar{C})) = (H - F)(\bar{H} - \bar{F})\end{aligned} \tag{4.11}$$

Eqns. (4.10) and (4.11) are expanded and written in matrix form

$$\begin{bmatrix} a\bar{b}_j & \bar{a}b_j \\ c\bar{d}_j & \bar{c}d_j \end{bmatrix} \begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - a\bar{a} - d_j\bar{d}_j \end{Bmatrix} \tag{4.12}$$

where

$$\begin{aligned} a &= G - C, \qquad & b_j &= A - B + Q_j(C - A) - S_j(D - B), \qquad & f &= G - D, \\ c &= H - C, \qquad & d_j &= A - B + Q_j(C - A) - S_j(F - B), \qquad & g &= H - F. \end{aligned} \tag{4.13}$$

Note that the variables $b_j$ and $d_j$ take the form

$$\begin{aligned} b_j &= h + Q_j k + S_j l, \\ d_j &= m + Q_j n + S_j o \end{aligned} \tag{4.14}$$

where $h = m$ and $k = n$. Eqn.(4.12) can be solved for $(R_j, \bar{R}_j)$,

$$\begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \frac{1}{a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j} \begin{bmatrix} \bar{c}d_j & -\bar{a}b_j \\ -c\bar{d}_j & a\bar{b}_j \end{bmatrix} \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - c\bar{c} - d_j\bar{d}_j \end{Bmatrix}, \quad j = 1, \ldots, N-1, \tag{4.15}$$

which can be substituted into Eqn. (4.6) to obtain

$$\begin{aligned} &(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) \\ &\quad \times (\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \ldots, N-1. \end{aligned} \tag{4.16}$$

Eqn. (4.16) are the synthesis equations and have degree $8^{10} \approx 1.07 \times 10^9$ for the maximum number of $N = 11$ task positions.

To further explore the structure of these equations, they are factored into the form

$$\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, N-1. \tag{4.17}$$

where all specified task information is contained in matrix $[\hat{Q}_j]$ and all link dimensions are contained in vector $\mathbf{p}$ which remains constant between task positions. This factorization is described in Appendix A. Matrix $[\hat{Q}_j]$ is defined as

$$[\hat{Q}_j] = \begin{bmatrix} \bar{\mathbf{q}}_j {\mathbf{q}_j}^T & [0] \\ [0] & -\bar{\mathbf{q}}'_j {\mathbf{q}'_j}^T \end{bmatrix} \tag{4.18}$$

where

$$\begin{aligned} \mathbf{q}_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j, Q_j S_j, Q_j^2, S_j^2, Q_j^2\bar{S}_j, \bar{Q}_j S_j^2\right\}^T, \\ \mathbf{q}'_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j\right\}^T. \end{aligned} \tag{4.19}$$

Vector $\mathbf{p}$ is defined with variables

$$\begin{aligned} &a = G - C, \qquad f = G - D, \qquad h = A - B, \qquad k = C - A, \qquad l = -(D - B), \\ &c = H - C, \qquad g = H - F, \qquad m = A - B, \qquad n = C - A, \qquad o = -(F - B). \end{aligned} \tag{4.20}$$

from which we observe that

$$m = h, \qquad n = k, \qquad f = a + h + k + l, \qquad g = c + h + k + o \tag{4.21}$$

which we substitute into the components of $\mathbf{p}$ to obtain

$$
\begin{aligned}
p_1 &= \bar{h}(k\bar{k}(c-a) + a\xi - c\beta - oa\bar{l} + lc\bar{o}) & p_{11} &= \bar{k}^2(cl - ao) \\
p_2 &= \bar{k}(h\bar{h}(c-a) + a\xi - c\beta - oa\bar{l} + lc\bar{o}) & p_{12} &= -k(\bar{o}a\bar{l} - \bar{l}c\bar{o}) \\
p_3 &= (h\bar{h} + k\bar{k})(c\bar{l} - a\bar{o}) + \xi a\bar{l} - \beta c\bar{o} & p_{13} &= (h\bar{h} + k\bar{k})(a\bar{c} - \bar{a}c) + a\bar{l}\bar{c}o - \bar{a}lc\bar{o} \\
p_4 &= \bar{h}^2 k(c-a) & p_{14} &= h\bar{k}(a\bar{c} - \bar{a}c) \\
p_5 &= \bar{h}^2(cl - oa) & p_{15} &= h(\bar{c}a\bar{l} - \bar{a}c\bar{o}) \\
p_6 &= 2\bar{h}\bar{k}(cl - ao) & p_{16} &= \bar{h}k(a\bar{c} - \bar{a}c) \\
p_7 &= \bar{h}k(c\bar{l} - a\bar{o} - a\bar{l} + c\bar{o}) & p_{17} &= -\bar{h}(c\bar{a}l - a\bar{c}o) \\
p_8 &= h\bar{k}(c\bar{l} - a\bar{o} - a\bar{l} + c\bar{o}) & p_{18} &= -\bar{k}(c\bar{a}l - a\bar{c}o) \\
p_9 &= h\bar{k}^2(c-a) & p_{19} &= k(\bar{c}a\bar{l} - \bar{a}c\bar{o}) \\
p_{10} &= h(-\bar{o}a\bar{l} + \bar{l}c\bar{o}) &&
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad \beta &= a\bar{l} + \bar{a}l + h\bar{k} + \bar{h}k + (h+k)(\bar{f} - \bar{h} - \bar{k}) + (\bar{h} + \bar{k})(f - h - k) \\
\xi &= c\bar{o} + \bar{c}o + h\bar{k} + \bar{h}k + (h+k)(\bar{g} - \bar{h} - \bar{k}) + (\bar{h} + \bar{k})(g - h - k)
\end{aligned}
\tag{4.22}
$$

For the case $N = 11$, we were unable to find any further reductions. The original design

parameters 4.20 and $(A, \bar{A}) = (0, 0)$ and $(B, \bar{B}) = (1, 1)$ are substituted into 4.22 to obtain.

$$
\begin{aligned}
p_1 &= -C\bar{C}(H-G) - (G-C)\xi + (H-C)\beta + (F-1)(G-C)(\bar{D}-1) - (D-1)(H-C)(\bar{F}-1) \\
p_2 &= \bar{C}\Big((H-G) + (G-C)\xi - (H-C)\beta - (F-1)(G-C)(\bar{D}-1) + (D-1)(H-C)(\bar{F}-1)\Big) \\
p_3 &= -(1+C\bar{C})\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1)\Big) - \xi(G-C)(\bar{D}-1) + \beta(H-C)(\bar{F}-1) \\
p_4 &= C(H-G) \\
p_5 &= -(H-C)(D-1) + (F-1)(G-C) \\
p_6 &= 2\bar{C}\Big((H-C)(D-1) - (G-C)(F-1)\Big) \\
p_7 &= C\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1) - (G-C)(\bar{D}-1) + (H-C)(\bar{F}-1)\Big) \\
p_8 &= \bar{C}\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1) - (G-C)(\bar{D}-1) + (H-C)(\bar{F}-1)\Big) \\
p_9 &= -\bar{C}^2(H-G) \\
p_{10} &= (\bar{F}-1)(G-C)(\bar{D}-1) - (\bar{D}-1)(H-C)(\bar{F}-1) \\
p_{11} &= -\bar{C}^2\Big((H-C)(D-1) - (G-C)(F-1)\Big) \\
p_{12} &= -C\Big((\bar{F}-1)(G-C)(\bar{D}-1) - (\bar{D}-1)(H-C)(\bar{F}-1)\Big) \\
p_{13} &= (1+C\bar{C})\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) + (G-C)(\bar{D}-1)(\bar{H}-\bar{C})(F-1) \\
&\quad -(\bar{G}-\bar{C})(D-1)(H-C)(\bar{F}-1) \\
p_{14} &= -\bar{C}\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) \\
p_{15} &= (\bar{H}-\bar{C})(G-C)(\bar{D}-1) - (\bar{G}-\bar{C})(H-C)(\bar{F}-1) \\
p_{16} &= -C\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) \\
p_{17} &= -(H-C)(\bar{G}-\bar{C})(D-1) + (G-C)(\bar{H}-\bar{C})(F-1) \\
p_{18} &= \bar{C}\Big((H-C)(\bar{G}-\bar{C})(D-1) - (G-C)(\bar{H}-\bar{C})(F-1)\Big) \\
p_{19} &= -C\Big((\bar{H}-\bar{C})(G-C)(\bar{D}-1) - (\bar{G}-\bar{C})(H-C)(\bar{F}-1)\Big) \\
\text{where} \quad & \beta = C(\bar{G}-\bar{C}) + \bar{C}(G-C) - D(\bar{G}-1) - \bar{D}(G-1) - 2 \\
& \xi = C(\bar{H}-\bar{C}) + \bar{C}(H-C) - F(\bar{H}-1) - \bar{F}(H-1) - 2
\end{aligned}
\tag{4.23}
$$

**Eleven Position Synthesis**

For the case $N = 11$, a multihomogeneous root count was performed on the system by selecting the variable groups

$$\langle C, D, F, G, H\rangle, \quad \langle \bar{C}, \bar{D}, \bar{F}, \bar{G}, \bar{H}\rangle, \tag{4.24}$$

which determine a multihomogeneous Bézout number of 264,241,152. The root count was performed by expanding the polynomial

$$(4\alpha_1 + 4\alpha_2)^{10}, \tag{4.25}$$

and taking the coefficient in front of the monomial $\alpha_1^5\alpha_2^5$. This combinatoric procedure is outlined in Section 2.3.2.

**Nine Position Synthesis**

However, for the case $N = 9$, it is useful to define the four new variables

$$\begin{aligned} r_1 &= a\bar{l}, & r_2 &= c\bar{o}, \\ \bar{r}_1 &= \bar{a}l, & \bar{r}_2 &= \bar{c}o, \end{aligned} \tag{4.26}$$

and substitute these into (4.22)

$$
\begin{aligned}
p_1 &= \bar{h}(k\bar{k}(c-a)+a\xi-c\beta-or_1+lr_2) & p_{11} &= \bar{k}^2(cl-ao)\\
p_2 &= \bar{k}(h\bar{h}(c-a)+a\xi-c\beta-or_1+lr_2) & p_{12} &= -k(\bar{o}r_1-\bar{l}r_2)\\
p_3 &= (h\bar{h}+k\bar{k})(c\bar{l}-a\bar{o})+\xi r_1-\beta r_2 & p_{13} &= (h\bar{h}+k\bar{k})(a\bar{c}-\bar{a}c)+r_1\bar{r}_2-\bar{r}_1r_2\\
p_4 &= \bar{h}^2k(c-a) & p_{14} &= h\bar{k}(a\bar{c}-\bar{a}c)\\
p_5 &= \bar{h}^2(cl-oa) & p_{15} &= h(\bar{c}r_1-\bar{a}r_2)\\
p_6 &= 2\bar{h}\bar{k}(cl-ao) & p_{16} &= \bar{h}k(a\bar{c}-\bar{a}c)\\
p_7 &= \bar{h}k(c\bar{l}-a\bar{o}-r_1+r_2) & p_{17} &= -\bar{h}(c\bar{r}_1-a\bar{r}_2)\\
p_8 &= h\bar{k}(c\bar{l}-a\bar{o}-r_1+r_2) & p_{18} &= -\bar{k}(c\bar{r}_1-a\bar{r}_2)\\
p_9 &= h\bar{k}^2(c-a) & p_{19} &= k(\bar{c}r_1-\bar{a}r_2)\\
p_{10} &= h(-\bar{o}r_1+\bar{l}r_2)
\end{aligned}
$$

$$
\begin{aligned}
\text{where}\quad \beta &= r_1+\bar{r}_1+h\bar{k}+\bar{h}k+(h+k)(\bar{f}-\bar{h}-\bar{k})+(\bar{h}+\bar{k})(f-h-k)\\
\xi &= r_2+\bar{r}_2+h\bar{k}+\bar{h}k+(h+k)(\bar{g}-\bar{h}-\bar{k})+(\bar{h}+\bar{k})(g-h-k)
\end{aligned}
\tag{4.27}
$$

For the case $N = 9$, there are six parameters to specify. Without loss of generality, we specify $(A, \bar{A}) = (0, 0)$ and $(B, \bar{B}) = (1, 1)$. Also, $(C, \bar{C})$ is specified and since this specification does reduce generality we do not assign it values but simply regard it as known. If $(A, \bar{A})$, $(B, \bar{B})$, $(C, \bar{C})$ are known then so are $(h, \bar{h})$, $(k, \bar{k})$ and every element of $\mathbf{p}$ in (4.27) is degree 2. Eqns. (4.17) then reduce to degree 4 polynomials. Combining (4.17) with (4.26) creates a system of $3+N$ equations and 12 unknowns which for $N = 9$ has a total degree of $4^8 2^4 = 1,048,576$.

Finally, we substitute in the original design parameters of (4.20) and $(A, \bar{A}) = (0, 0)$ and $(B, \bar{B}) = (1, 1)$ to obtain our final synthesis equations,

$$
\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, 8, \tag{4.28}
$$

where

$$
\begin{aligned}
p_1 &= -C\bar{C}(H-G) - (G-C)\xi + (H-C)\beta - (F-1)r_1 + (D-1)r_2 \\
p_2 &= \bar{C}\Big(H - G + (G-C)\xi - (H-C)\beta + (F-1)r_1 - (D-1)r_2\Big) \\
p_3 &= -(1+C\bar{C})\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1)\Big) + \xi r_1 - \beta r_2 \\
p_4 &= C(H-G) \\
p_5 &= -(H-C)(D-1) + (F-1)(G-C) \\
p_6 &= 2\bar{C}\Big((H-C)(D-1) - (G-C)(F-1)\Big) \\
p_7 &= C\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1) + r_1 - r_2\Big) \\
p_8 &= \bar{C}\Big((H-C)(\bar{D}-1) - (G-C)(\bar{F}-1) + r_1 - r_2\Big) \\
p_9 &= -\bar{C}^2(H-G) \\
p_{10} &= -(\bar{F}-1)r_1 + (\bar{D}-1)r_2 \\
p_{11} &= -\bar{C}^2\Big((H-C)(D-1) - (G-C)(F-1)\Big) \\
p_{12} &= C\Big((\bar{F}-1)r_1 - (\bar{D}-1)r_2\Big) \\
p_{13} &= (1+C\bar{C})\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) + r_1\bar{r}_2 - \bar{r}_1 r_2 \\
p_{14} &= -\bar{C}\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) \\
p_{15} &= -(\bar{H}-\bar{C})r_1 + (\bar{G}-\bar{C})r_2 \\
p_{16} &= -C\Big((G-C)(\bar{H}-\bar{C}) - (\bar{G}-\bar{C})(H-C)\Big) \\
p_{17} &= (H-C)\bar{r}_1 - (G-C)\bar{r}_2 \\
p_{18} &= -\bar{C}\Big((H-C)\bar{r}_1 - (G-C)\bar{r}_2\Big) \\
p_{19} &= C\Big((\bar{H}-\bar{C})r_1 - (\bar{G}-\bar{C})r_2\Big) \\
\text{where}\quad \beta &= r_1 + \bar{r}_1 + (C-1)(\bar{G}-\bar{D}-\bar{C}) + (\bar{C}-1)(G-D-C) - 2 \\
\xi &= r_2 + \bar{r}_2 + (C-1)(\bar{H}-\bar{F}-\bar{C}) + (\bar{C}-1)(H-F-C) - 2
\end{aligned}
\tag{4.29}
$$

and

$$
\begin{aligned}
r_1 &= -(G-C)(\bar{D}-1), & r_2 &= -(H-C)(\bar{F}-1), \\
\bar{r}_1 &= -(\bar{G}-\bar{C})(D-1), & \bar{r}_2 &= -(\bar{H}-\bar{C})(F-1)
\end{aligned}
\tag{4.30}
$$

A multihomogeneous root count is performed on the system by selecting the variable groups

$$\langle D, \bar{D}, G, \bar{G}, r_1, \bar{r}_1 \rangle, \quad \langle F, \bar{F}, H, \bar{H}, r_2, \bar{r}_2 \rangle, \tag{4.31}$$

which determine the multihomogeneous Bézout number of 286,720. The root count was performed by expanding the polynomial

$$16\alpha_1^2\alpha_2^2(2\alpha_1 + 2\alpha_2)^8 \tag{4.32}$$

and taking the coefficient in front of the monomial $\alpha_1^6\alpha_2^6$.

## 4.2 Synthesis Solution

The synthesis equations were solved for the maximum $N = 11$ task positions on the UC Irvine High Performance Computing Cluster using the polynomial homotopy solver BERTINI. Rather than specify the requirements for a particular task, the input parameters, $(Q_j, S_j)$, $j = 1, \ldots, 10$, were set to random complex numbers to create a numerically general system. The degree of this polynomial system was too large to directly compute, therefore regeneration homotopy was implemented that solves this system incrementally. Regeneration is described in Section 2.3.6.

The regeneration homotopy required 311 hrs on 256 cores processing at 2.2 GHz to compute, and tracked 24,822,328 paths over 10 levels to find 1,521,037 nonsingular solutions. These nonsingular solutions can be used to construct parameter homotopies for efficient calculation of linkage solutions for a specific sets of 11 coordinated joint angles. Parameter homotopy is discussed in Section 2.3.5. The advantage of parameter homotopy is that the nonsingular endpoints of a general run are used as startpoints of a specific run so that only 1,521,037

paths need to be tracked in order to find all nonsingular solutions of a specific system.

The solutions to the synthesis equations can be grouped by a six-way symmetry. Firstly, if the values of the floating four-bar pivots are swapped such that $D \leftrightarrow F$ and $G \leftrightarrow H$, then the result is another solution to the synthesis equations that describes the same six-bar linkage but with different labelling of the pivots. We call these solutions symmetric pairs and sort the synthesis solutions in order to drop one member of each pair so that the synthesis results do not include duplicate linkages.

Furthermore, the solutions to the synthesis equations can be grouped into sets of three function-cognates. For every Stephenson II linkage that coordinates angles $(\phi, \psi)$, there exist two other Stephenson II linkages that create identical coordinations of $(\phi, \psi)$, called function-cognates. The construction of Stephenson II function-cognates is described in Section 2.5.3. That is, for a Stephenson II linkage defined by $\{A, B, C, D, F, G, H\}$, there exist two function-cognates defined by

$$
\begin{aligned}
A_{c1} &= A, & A_{c2} &= A,\\
B_{c1} &= B, & B_{c2} &= B,\\
C_{c1} &= C, & C_{c2} &= C,\\
D_{c1} &= D, & D_{c2} &= \frac{(D-G)(H-C)-(F-H)(G-C)}{(H-G)} + C,\\
F_{c1} &= \frac{(D-G)(H-C)-(F-H)(G-C)}{(H-G)} + C, & F_{c2} &= F,\\
G_{c1} &= D - G + C, & G_{c2} &= \left(\frac{F-H}{G-H}\right)(G-C) + C,\\
H_{c1} &= \left(\frac{D-G}{H-G}\right)(H-C) + C, & H_{c2} &= F - H + C,
\end{aligned}
\tag{4.33}
$$

Due to the existence of symmetric pairs, these cognates can also be described with the labelling of floating four-bar pivots swapped, that is $D_{c1} \leftrightarrow F_{c1}$, $G_{c1} \leftrightarrow H_{c1}$ and $D_{c2} \leftrightarrow F_{c2}$, $G_{c2} \leftrightarrow H_{c2}$.

The synthesis solutions are then organized into cognate triples. Since the solution method is subject to numerical errors, it is possible that all members of a cognate triple will not be found in the synthesis results. In this case, the missing cognates are constructed and added to the synthesis results.

## 4.3 Analysis

The synthesis solutions represent linkage design candidates. Once design candidates have been sorted into cognate triples, they are analyzed to evaluate the performance of each design. The criteria for a successful design candidate is that the required accuracy points lie on a single trajectory of configurations without any singularities. Trajectories of configurations are determined for each design candidate following the procedure described in Section 2.4. The forward kinematics equations of a Stephenson II six-bar are presented in Eqn. (2.109). The solutions of the forward kinematics equations represent configurations of the linkage, which are sorted into trajectories using the routine outlined in Section 2.4.3.

Once all trajectories have been assembled for a linkage design candidate, each is checked to see which and how many of the specified accuracy points they contain. A successful design candidate will produce a trajectory that moves through all 11 accuracy points. We term these designs 11-point mechanisms.

While linkage designs that contain all 11 accuracy points on a single trajectory is the goal, our design process identifies linkage designs with trajectories that move through less than 11 points as well. It is often the case that these mechanisms only slightly miss some accuracy points and may have other features useful to the designer, such as compact dimensions or reduced link overlap.

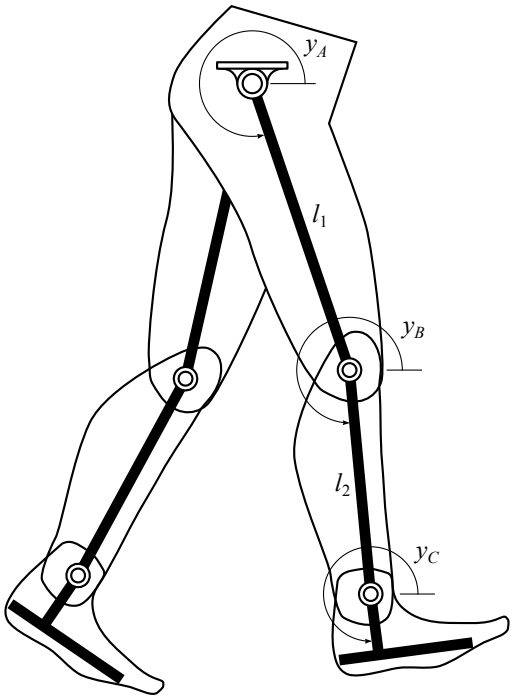



Figure 4.2: A humanoid leg is modelled as a planar 3R chain where $l_1 = 18.7$ and $l_2 = 14$.

## 4.4 Example: Hip, Knee, and Ankle Function Generators

In order to illustrate this design procedure, we specify functions for the movement of the hip, knee, and ankle joints of a humanoid walking gait, see Fig. 4.2, and design Stephenson II function generators to generate these functions from a single constant velocity input. The joint functions were obtained from a video of a walking movement. The resulting joint functions have asymmetries that test the performance of this design system for six-bar function generators, see Fig. 4.3.

The angles at the hip $y_A$, knee $y_B$, and ankle $y_C$ shown in Fig. 4.2 are given by

$$\begin{aligned} y_A &= f_A(t) - 71^\circ, \\ y_B &= f_B(t) - 84.95^\circ, \\ y_C &= f_C(t) - 82.45^\circ, \end{aligned} \tag{4.34}$$

Table 4.1: Fourier coefficients for walking gait joint angle functions.

| | $\Delta y_A = f_A(t)$ | $\Delta y_B = f_B(t)$ | $\Delta y_C = f_C(t)$ |
|---|---|---|---|
| $o$ | 3 | 5 | 5 |
| $a_0$ | −0.23899606 | −0.42317334 | −0.39916633 |
| $a_1$ | 0.26507432 | 0.16530352 | 0.24817028 |
| $b_1$ | 0.04632108 | 0.38229515 | 0.34926232 |
| $a_2$ | 0.00265777 | 0.20929088 | 0.10641896 |
| $b_2$ | −0.05632819 | 0.12673270 | 0.10372243 |
| $a_3$ | −0.02058442 | 0.04930281 | 0.02732644 |
| $b_3$ | 0.01472108 | 0.04075846 | 0.09697237 |
| $a_4$ | 0 | 0.00107466 | 0.01722405 |
| $b_4$ | 0 | 0.01778866 | 0.00426672 |
| $a_5$ | 0 | 0.00429589 | −0.01269033 |
| $b_5$ | 0 | 0.00522743 | −0.01612253 |

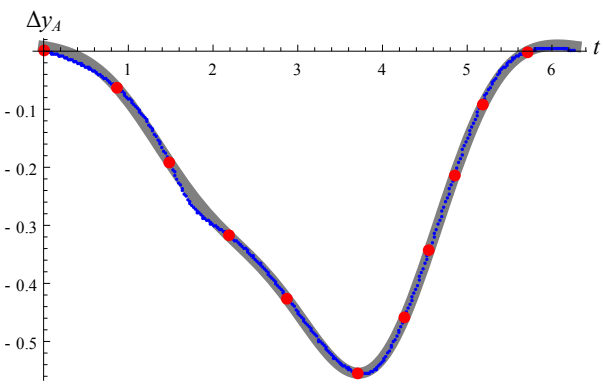
Table 4.2: Task points for each example function. Values below are given in degrees and must be converted to radians in order for $f_A(t)$, $f_B(t)$, and $f_C(t)$ to evaluate properly.

| $j$ | $(t_j,\ \Delta y_{Aj})$ | $(t_j,\ \Delta y_{Bj})$ | $(t_j,\ \Delta y_{Cj})$ |
|---|---|---|---|
| 0 | $\big(0,\ 0\big)$ | $\big(0,\ 0\big)$ | $\big(0,\ 0\big)$ |
| 1 | $\big(50,\ f_A(50)\big)$ | $\big(25,\ f_B(25)\big)$ | $\big(32,\ f_C(32)\big)$ |
| 2 | $\big(85,\ f_A(85)\big)$ | $\big(60,\ f_B(60)\big)$ | $\big(65,\ f_C(65)\big)$ |
| 3 | $\big(125,\ f_A(125)\big)$ | $\big(94,\ f_B(94)\big)$ | $\big(94,\ f_C(94)\big)$ |
| 4 | $\big(165,\ f_A(165)\big)$ | $\big(152,\ f_B(152)\big)$ | $\big(146,\ f_C(146)\big)$ |
| 5 | $\big(212,\ f_A(212)\big)$ | $\big(202,\ f_B(202)\big)$ | $\big(188,\ f_C(188)\big)$ |
| 6 | $\big(244,\ f_A(244)\big)$ | $\big(237,\ f_B(237)\big)$ | $\big(226,\ f_C(226)\big)$ |
| 7 | $\big(261,\ f_A(261)\big)$ | $\big(278,\ f_B(278)\big)$ | $\big(264,\ f_C(264)\big)$ |
| 8 | $\big(278,\ f_A(278)\big)$ | $\big(313,\ f_B(313)\big)$ | $\big(297,\ f_C(297)\big)$ |
| 9 | $\big(297,\ f_A(297)\big)$ | $\big(331,\ f_B(331)\big)$ | $\big(327,\ f_C(327)\big)$ |
| 10 | $\big(327,\ f_A(327)\big)$ | $\big(346,\ f_B(346)\big)$ | $\big(344,\ f_C(344)\big)$ |

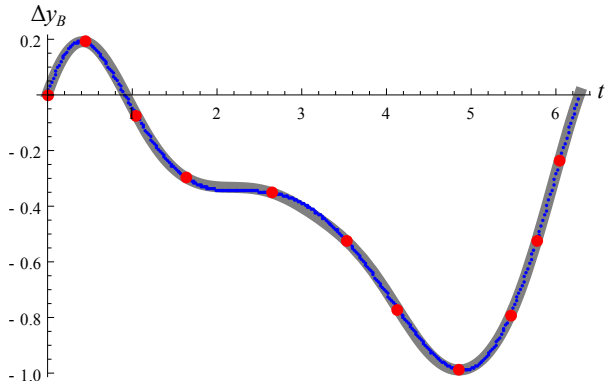
where the three functions $f_A(t)$, $f_B(t)$, and $f_C(t)$ are periodic and computed as the Fourier series,

$$f(t) = \frac{1}{2}a_0 + \sum_{m=1}^{o} \left(a_m \cos(mt) + b_m \sin(mt)\right). \tag{4.35}$$

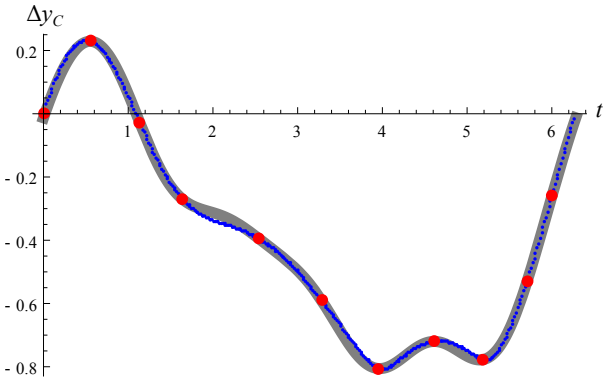
The Fourier coefficients for each of these functions are listed in Table 4.1. The three functions have been made to have periods of length $2\pi$ so that $t$ becomes the angle of a fully rotatable input crank with constant angular velocity that can drive all functions simultaneously. Eleven




(a) Hip angle function $\Delta y_A = f_A(t)$




(b) Knee angle function $\Delta y_B = f_B(t)$




(c) Ankle angle function $\Delta y_C = f_C(t)$

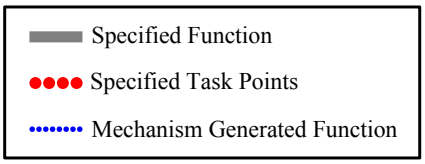
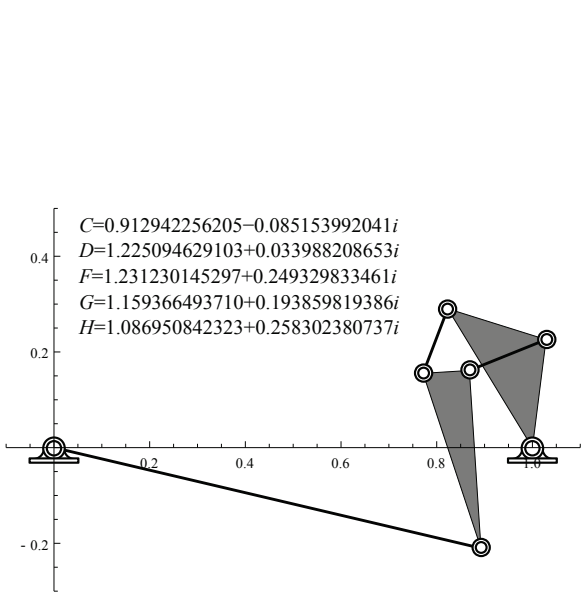



Figure 4.3: The hip, knee, and ankle function specifications include the location of the accuracy points and a trace of the function generated by Stephenson II six-bar linkages.

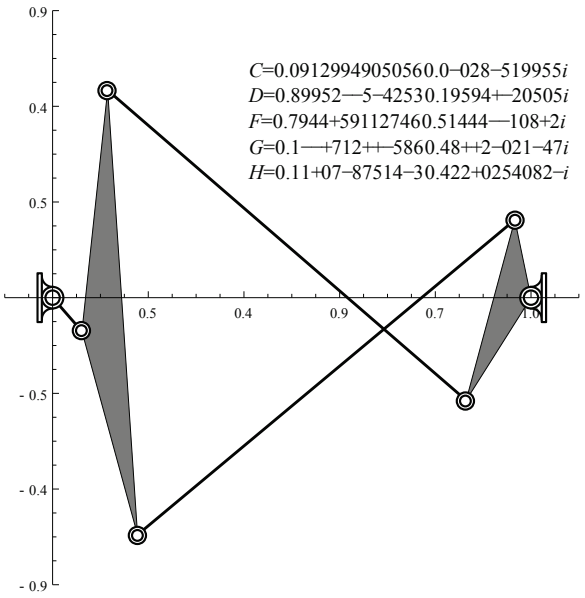
points are selected from each function which are the accuracy points that we synthesize for, displayed in Table 4.2 and Fig. 4.3.

Example designs from the synthesis method are shown in Fig. 4.4. All computations took place on 64×2.2GHz processors. Information on each computation is given in Table 4.4. Notice that the Stephenson II six-bar linkage can have either the binary link or the ternary link as input. The treatment of both cases is nearly identical. Interestingly, for the three example functions designed in this chapter, we found several 11-point mechanisms when the ternary link was taken as the input and no 11-point mechanisms when the binary link was taken as the input.

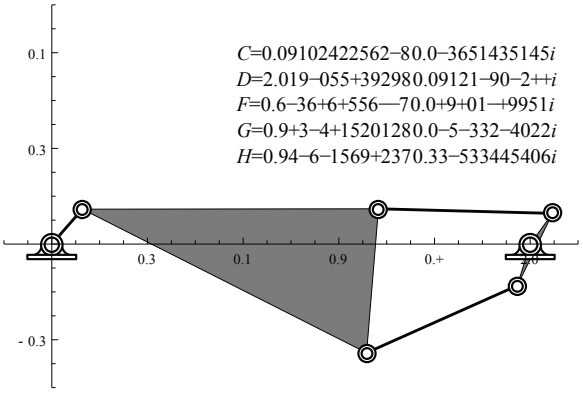
Table 4.4 presents the number of *linkage solutions*, which are the homotopy solutions that




(a) 10-point hip angle $\Delta y_A$ function generator




(b) 11-point knee angle $\Delta y_B$ function generator




(c) 11-point ankle angle $\Delta y_C$ function generator

Figure 4.4: The hip, knee, and ankle Stephenson II six-bar function generators that generate the desired functions.

corresponded to physical linkage designs. *Design candidates* are the linkage solutions where the solution pairs and cognate triples were identified. As well, design candidates were subjected to a maximum and minimum constraint on the size of their link lengths. Each design candidate was analyzed according to the performance verification routine described in Section 2.4. Table 4.4 also lists the number of mechanisms that achieve 11 accuracy points, 11-point mechanisms, as well as those that miss a few accuracy points, 6-, 7-, 8-, 9-, 10-point mechanisms. The *synthesis computation time* reports how long the parameter homotopy

Table 4.3: Synthesis results when (a) the binary link is chosen as the input and (b) the ternary link is chosen as the input.

(a) Input $\phi_A = \phi_B = \phi_C = t$.

| | $\Delta y_A$ | $\Delta y_B$ | $\Delta y_C$ |
|---|---|---|---|
| Linkage solutions | 14,445 | 8,255 | 9,835 |
| Design candidates | 6,686 | 3,830 | 4,791 |
| 11-point mechanisms | 0 | 0 | 0 |
| 10-point mechanisms | 2 | 0 | 0 |
| 9-point mechanisms | 78 | 27 | 17 |
| 8-point mechanisms | 318 | 92 | 49 |
| 7-point mechanisms | 667 | 220 | 196 |
| 6-point mechanisms | 1062 | 447 | 456 |
| Synthesis computation time (hr) | 2.9 | 2.0 | 2.0 |
| Analysis computation time (hr) | 9.4 | 5.2 | 6.4 |

(b) Input $\psi_A = \psi_B = \psi_C = t$.

| | $\Delta y_A$ | $\Delta y_B$ | $\Delta y_C$ |
|---|---|---|---|
| Linkage solutions | 11,175 | 11,459 | 9,913 |
| Design candidates | 4,496 | 5,100 | 4,636 |
| 11-point mechanisms | 51 | 28 | 6 |
| 10-point mechanisms | 119 | 144 | 13 |
| 9-point mechanisms | 129 | 183 | 26 |
| 8-point mechanisms | 219 | 222 | 135 |
| 7-point mechanisms | 320 | 316 | 276 |
| 6-point mechanisms | 557 | 466 | 392 |
| Synthesis computation time (hr) | 2.1 | 1.5 | 1.7 |
| Analysis computation time (hr) | 6.3 | 7 | 6.1 |

took to compute, while *analysis computation time* reports the how long it took to solve forward kinematics, sort configurations into trajectories, and identify the number of accuracy points on each trajectory.

The three linkages shown in Fig. 4.4 were used in the design of a prototype device that creates the desired walking motion, see Fig. 4.5. The three function generators package compactly near the hip where they are driven by a single crankshaft which serves as the input link for all three linkages. The hip joint function generator drives the hip joint directly. The motions of the knee and ankle function generators are passed down the leg chain by parallelogram linkages.

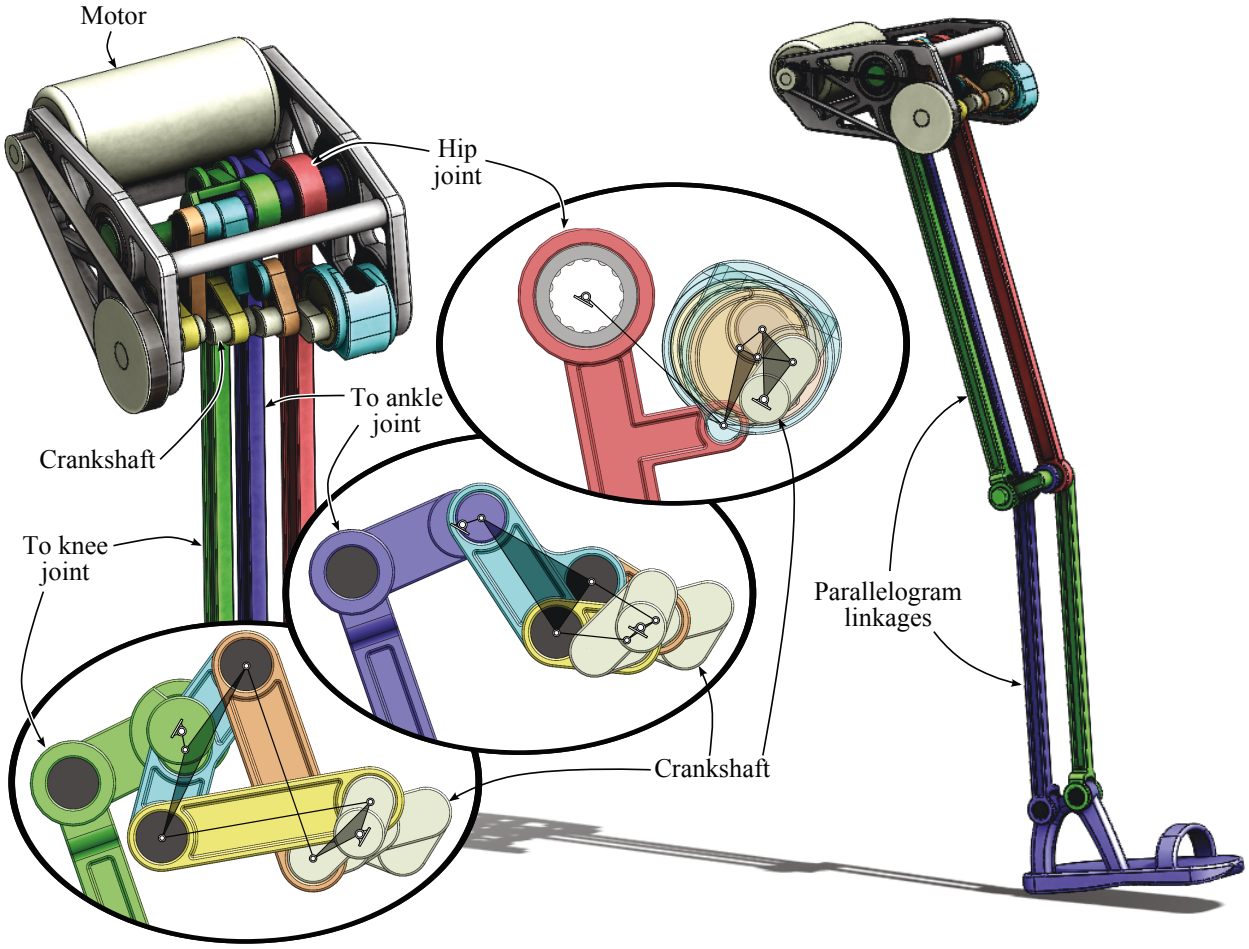



Figure 4.5: Solid model of the hip, knee, and ankle function generators integrated into a humanoid walker. The schematics of Figure 4.4 overlay their physical embodiments for each function generator.

## 4.5 Summary

This chapter presented a computational design procedure for Stephenson II function generators to achieve 11 coordinated input-output angles. The synthesis equations form a system of 10 eighth degree polynomials, that have a multihomogeneous root count of 264,241,152. The polynomial homotopy software BERTINI found a general set of roots for these equations, which was then used to construct a parameter homotopy that efficiently solves the synthesis equations in about two hours on 64×2.2GHz processors.

This parameter homotopy can be executed for any set of 11 accuracy points and yields over

one million solutions from which physical linkage designs, solution pairs, and cognate triples must be identified and then analyzed to determine feasible designs.

This design methodology relies on the solution of a large polynomial system and generates a large number of linkage candidates, most of which are not useful designs. In the example provided, the hip, knee, and ankle functions yielded a total of 11,182, 8,930, and 9,427 candidate designs, respectively, considering both binary and ternary driving cranks. Of these 51, 28, and 6, respectively, were feasible designs that reached the 11 required accuracy points. If the designer accepts feasible designs that reach 9 and 10 accuracy points as well, then the total number of available designs becomes 379, 382, and 62, respectively.

# Chapter 5

# Stephenson III Function Generation

A Stephenson III mechanism consists of three ground pivots $A$, $B$, $C$, and four moving pivots $D$, $F$, $G$, and $H$, see Fig. 5.1. There are five moving links $AD$, $BF$, $CG$, $DGH$, and $FH$. The objective is to coordinate the rotations between $\phi$ and $\psi$ of links $AD$ and $BF$. In order to set the scale, orientation, and location of the linkage, the ground pivot $A$ is chosen to be at the origin $A = 0+i0$, and the initial position of moving pivot $D$ is chosen to be $D = 1+0i$.

## 5.1 Synthesis Formulation

Formulation of the synthesis equations begins by constructing the loop closure equations. From Fig. 5.1(b), we construct two sets of loop closure equations,

$$A - C + Q_j(D - A) + U_j(G - D) - R_j(G - C) = 0, \tag{5.1}$$

$$A - B + Q_j(D - A) + U_j(H - D) - S_j(F - B) - T_j(H - F) = 0, \tag{5.2}$$

$$j = 1, \ldots, N - 1,$$

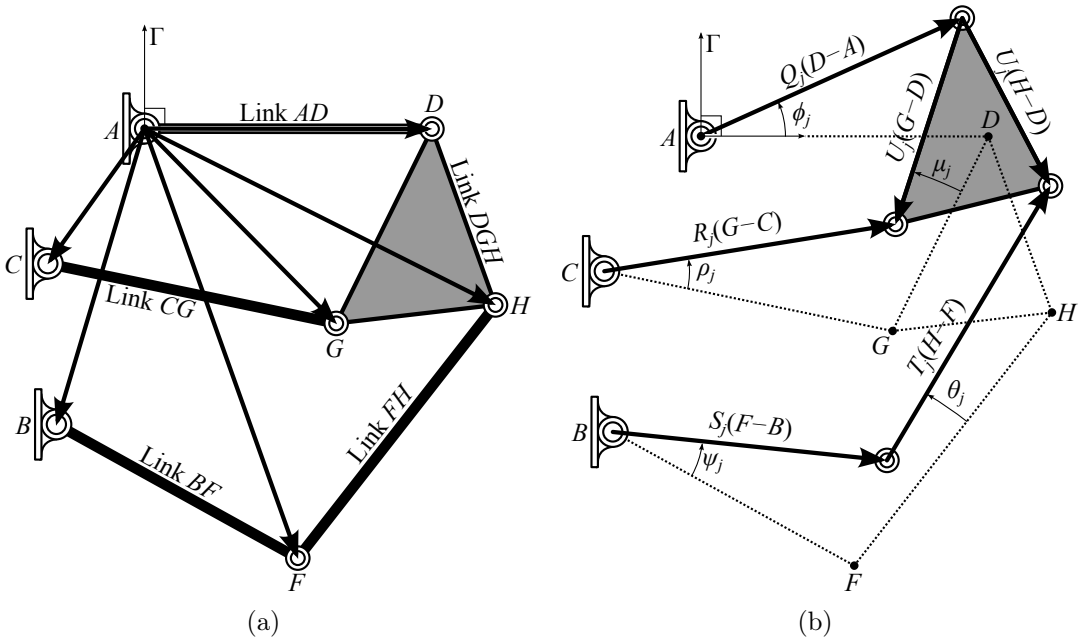



Figure 5.1: A Stephenson III function generator in (a) its reference configuration and (b) its $j^{\text{th}}$ configuration.

and two sets of conjugate loop equations,

$$\bar{A} - \bar{C} + \bar{Q}_j(\bar{D} - \bar{A}) + \bar{U}_j(\bar{G} - \bar{D}) - \bar{R}_j(\bar{G} - \bar{C}) = 0, \tag{5.3}$$

$$\bar{A} - \bar{B} + \bar{Q}_j(\bar{D} - \bar{A}) + \bar{U}_j(\bar{H} - \bar{D}) - \bar{S}_j(\bar{F} - \bar{B}) - \bar{T}_j(\bar{H} - \bar{F}) = 0, \tag{5.4}$$

$$j = 1, \ldots, N-1,$$

where $j$ indexes task positions. These equations contain the complex rotation operators $Q_j$, $R_j$, $S_j$, $T_j$, $U_j$, and their conjugates defined as

$$\begin{aligned} &Q_j = e^{i\phi_j}, \quad &&R_j = e^{i\rho_j}, \quad &&S_j = e^{i\psi_j}, \quad &&T_j = e^{i\theta_j}, \quad &&U_j = e^{i\mu_j}, \\ &\bar{Q}_j = e^{-i\phi_j}, \quad &&\bar{R}_j = e^{-i\rho_j}, \quad &&\bar{S}_j = e^{-i\psi_j}, \quad &&\bar{T}_j = e^{-i\theta_j}, \quad &&\bar{U}_j = e^{-i\mu_j}. \end{aligned} \tag{5.5}$$

The variables $Q_j$, $\bar{Q}_j$, $S_j$ and $\bar{S}_j$ specify the task. The variables $R_j$, $\bar{R}_j$, $T_j$, $\bar{T}_j$, $U_j$, and $\bar{U}_j$ remain as unknowns. In order for these unknowns to correspond to rotational operators, it

is necessary that they satisfy the normalization conditions,

$$R_j \bar{R}_j = 1, \tag{5.6}$$

$$T_j \bar{T}_j = 1, \tag{5.7}$$

$$U_j \bar{U}_j = 1. \quad j = 1, \ldots, N-1. \tag{5.8}$$

Eqns. (5.1)–(5.4) and (5.6)–(5.8) comprise a system of $7(N-1)$ equations. The unknowns of these equations are

$$\left\langle B, \bar{B}, C, \bar{C}, F, \bar{F}, G, \bar{G}, H, \bar{H} \right\rangle \text{ and } \left\langle R_j, \bar{R}_j, \bar{T}_j, \bar{T}_j, \bar{U}_j, \bar{U}_j, \right\rangle, \quad j = 1, \ldots, N-1. \tag{5.9}$$

Pivot locations $(A, \bar{A})$ and $(D, \bar{D})$ are not included as unknowns. They are specified in order to set the scale, orientation, and location of the function generator in the plane. This leaves $10 + 6(N-1)$ unknowns, and the system becomes square for $N = 11$ where the degree is $2^{70} \approx 1.18 \times 10^{21}$.

In order to reduce the system, the unknowns $R_j$ and $\bar{R}_j$ are eliminated by substituting Eqns. (5.1) and (5.3) into (5.6) to obtain

$$\begin{aligned} &(A - C + Q_j(D - A) + U_j(G - D)) \\ &\quad \times (\bar{A} - \bar{C} + \bar{Q}_j(\bar{D} - \bar{A}) + \bar{U}_j(\bar{G} - \bar{D})) = (G - C)(\bar{G} - \bar{C}). \end{aligned} \tag{5.10}$$

Similarly, $T_j$ and $\bar{T}_j$ are eliminated by substituting Eqns. (5.2) and (5.4) into (5.7) to obtain

$$\begin{aligned} &(A - B + Q_j(D - A) - S_j(F - B) + U_j(H - D)) \\ &\quad \times (\bar{A} - \bar{B} + \bar{Q}_j(\bar{D} - \bar{A}) - \bar{S}_j(\bar{F} - \bar{B}) + \bar{U}_j(\bar{H} - \bar{D})) = (H - F)(\bar{H} - \bar{F}) \end{aligned} \tag{5.11}$$

Eqns. (5.10) and (5.11) are expanded and written in matrix form

$$\begin{bmatrix} a\bar{b}_j & \bar{a}b_j \\ c\bar{d}_j & \bar{c}d_j \end{bmatrix} \begin{Bmatrix} U_j \\ \bar{U}_j \end{Bmatrix} = \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - a\bar{a} - d_j\bar{d}_j \end{Bmatrix} \tag{5.12}$$

where

$$\begin{aligned} &a = G - D, \qquad && b_j = A - C + Q_j(D - A), && f = G - C, \\ &c = H - D, \qquad && d_j = A - B + Q_j(D - A) - S_j(F - B), \qquad && g = H - F. \end{aligned} \tag{5.13}$$

Note that the variables $b_j$ and $d_j$ take the form

$$\begin{aligned} b_j &= h + Q_j k + S_j l, \\ d_j &= m + Q_j n + S_j o \end{aligned} \tag{5.14}$$

where $k = n$ and $l = 0$. Eqn. (5.12) can be solved for $(U_j, \bar{U}_j)$,

$$\begin{Bmatrix} U_j \\ \bar{U}_j \end{Bmatrix} = \frac{1}{a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j} \begin{bmatrix} \bar{c}d_j & -\bar{a}b_j \\ -c\bar{d}_j & a\bar{b}_j \end{bmatrix} \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - c\bar{c} - d_j\bar{d}_j \end{Bmatrix}, \quad j = 1, \ldots, N-1, \tag{5.15}$$

which can be substituted into Eqn. (5.8) to obtain

$$\begin{aligned} &(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) \\ &\quad \times (\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \ldots, N-1. \end{aligned} \tag{5.16}$$

Eqn. (5.16) are the design equations and have degree $8^{10} \approx 1.07 \times 10^9$ for the maximum number of $N = 11$ task positions.

To further explore the structure of these equations, they are factored into the form

$$\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, N-1. \tag{5.17}$$

where all specified task information is contained in matrix $[\hat{Q}_j]$ and all link dimensions are contained in vector $\mathbf{p}$ which remains constant between task positions. This factorization is described in Appendix A. Matrix $[\hat{Q}_j]$ is defined as

$$[\hat{Q}_j] = \begin{bmatrix} \bar{\mathbf{q}}_j {\mathbf{q}_j}^T & [0] \\ [0] & -\bar{\mathbf{q}}'_j {\mathbf{q}'_j}^T \end{bmatrix} \tag{5.18}$$

where

$$\begin{aligned} \mathbf{q}_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j, Q_j S_j, Q_j^2, S_j^2, Q_j^2\bar{S}_j, \bar{Q}_j S_j^2\right\}^T, \\ \mathbf{q}'_j &= \left\{1, Q_j, S_j, \bar{Q}_j, \bar{S}_j, Q_j\bar{S}_j, \bar{Q}_j S_j\right\}^T. \end{aligned} \tag{5.19}$$

Vector $\mathbf{p}$ is defined with variables

$$\begin{aligned} &a = G - D, \qquad f = G - C, \qquad h = A - C, \qquad k = D - A, \qquad l = 0, \\ &c = H - D, \qquad g = H - F, \qquad m = A - B, \qquad n = D - A, \qquad o = -(F - B). \end{aligned} \tag{5.20}$$

from which we observe that

$$n = k, \qquad l = 0, \qquad f = a + h + k, \qquad g = c + k + m + o \tag{5.21}$$

which we substitute into the components of $\mathbf{p}$ to obtain

$$
\begin{aligned}
p_1 &= k\bar{k}(c\bar{h} - a\bar{m}) + \xi a\bar{h} - \beta c\bar{m} & p_{11} &= -\bar{k}^2 ao \\
p_2 &= \bar{k}(a\xi - c\beta + hc\bar{m} - ma\bar{h}) & p_{12} &= 0 \\
p_3 &= -k\bar{k}a\bar{o} - \beta c\bar{o} - a\bar{h}m\bar{o} & p_{13} &= k\bar{k}(a\bar{c} - \bar{a}c) + a\bar{h}\bar{c}m - \bar{a}hc\bar{m} \\
p_4 &= k(\bar{h}c\bar{m} - \bar{m}a\bar{h}) & p_{14} &= \bar{k}(a\bar{c}m - c\bar{a}h) \\
p_5 &= -a\bar{h}\bar{m}o & p_{15} &= -\bar{a}hc\bar{o} \\
p_6 &= -\bar{k}(a\bar{m}o + oa\bar{h}) & p_{16} &= -k(\bar{a}c\bar{m} - \bar{c}a\bar{h}) \\
p_7 &= k(\bar{h}c\bar{o} - \bar{o}a\bar{h}) & p_{17} &= a\bar{h}\bar{c}o \\
p_8 &= \bar{k}(hc\bar{o} - am\bar{o}) & p_{18} &= \bar{k}a\bar{c}o \\
p_9 &= \bar{k}^2(ch - am) & p_{19} &= -k\bar{a}c\bar{o} \\
p_{10} &= 0 & &
\end{aligned}
\tag{5.22}
$$

where $\beta = a\bar{h} + \bar{a}h + k(\bar{f} - \bar{k}) + \bar{k}(f - k)$

$\xi = c\bar{m} + \bar{c}m + c\bar{o} + \bar{c}o + m\bar{o} + \bar{m}o + k(\bar{g} - \bar{k}) + \bar{k}(g - k)$

In order to reduce the system, we define eight new variables

$$
\begin{aligned}
r_1 &= a\bar{h} & r_2 &= c\bar{m} & r_3 &= c\bar{o} & r_4 &= m\bar{o} \\
\bar{r}_1 &= \bar{a}h & \bar{r}_2 &= \bar{c}m & \bar{r}_3 &= \bar{c}o & \bar{r}_4 &= \bar{m}o
\end{aligned}
\tag{5.23}
$$

and substitute these into (5.22)

$$
\begin{aligned}
p_1 &= k\bar{k}(c\bar{h} - a\bar{m}) + \xi r_1 - \beta r_2 & p_{11} &= -\bar{k}^2 ao \\
p_2 &= \bar{k}(a\xi - c\beta + hr_2 - mr_1) & p_{12} &= 0 \\
p_3 &= -k\bar{k}a\bar{o} - \beta r_3 - r_1 r_4 & p_{13} &= k\bar{k}(a\bar{c} - \bar{a}c) + r_1\bar{r}_2 - \bar{r}_1 r_2 \\
p_4 &= k(\bar{h}r_2 - \bar{m}r_1) & p_{14} &= \bar{k}(a\bar{r}_2 - c\bar{r}_1) \\
p_5 &= -r_1\bar{r}_4 & p_{15} &= -\bar{r}_1 r_3 \\
p_6 &= -\bar{k}(a\bar{r}_4 + or_1) & p_{16} &= -k(\bar{a}r_2 - \bar{c}r_1) \\
p_7 &= k(\bar{h}r_3 - \bar{o}r_1) & p_{17} &= r_1\bar{r}_3 \\
p_8 &= \bar{k}(hr_3 - ar_4) & p_{18} &= \bar{k}a\bar{r}_3 \\
p_9 &= \bar{k}^2(ch - am) & p_{19} &= -k\bar{a}r_3 \\
p_{10} &= 0 & & \\
\text{where} \quad \beta &= r_1 + \bar{r}_1 + k(\bar{f} - \bar{k}) + \bar{k}(f - k) & & \\
\xi &= r_2 + \bar{r}_2 + r_3 + \bar{r}_3 + r_4 + \bar{r}_4 + k(\bar{g} - \bar{k}) + \bar{k}(g - k) & &
\end{aligned}
\tag{5.24}
$$

In order to set the scale, orientation, and location of the function generator in the plane, we specify $(A, \bar{A}) = (0, 0)$ and $(D, \bar{D}) = (1, 1)$ which means $(k, \bar{k})$ are known. Therefore every element of $\mathbf{p}$ in (5.24) is degree 2. Eqns. (5.17) then reduce to degree 4 polynomials. Combining (5.17) with (5.23) creates a system of $7 + N$ equations and 18 unknowns which for $N = 11$ has a total degree of $4^{10}2^8 = 268,435,456$.

Finally, we substitute in the original design parameters of (5.20) and $(A, \bar{A}) = (0, 0)$ and $(D, \bar{D}) = (1, 1)$ to obtain our final synthesis equations,

$$
\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, N - 1, \tag{5.25}
$$

where

$$
\begin{aligned}
p_1 &= -(H-1)\bar{C} + (G-1)\bar{B} + \xi r_1 - \beta r_2 \\
p_2 &= (G-1)\xi - (H-1)\beta - Cr_2 + Br_1 \\
p_3 &= (G-1)(\bar{F}-\bar{B}) - \beta r_3 - r_1 r_4 \\
p_4 &= -\bar{C}r_2 + \bar{B}r_1 \\
p_5 &= -r_1\bar{r}_4 \\
p_6 &= -(G-1)\bar{r}_4 + (F-B)r_1 \\
p_7 &= -\bar{C}r_3 + (\bar{F}-\bar{B})r_1 \\
p_8 &= -Cr_3 - (G-1)r_4 \\
p_9 &= -(H-1)C + (G-1)B \\
p_{10} &= 0 \\
p_{11} &= (G-1)(F-B) \\
p_{12} &= 0 \\
p_{13} &= (G-1)(\bar{H}-1) - (\bar{G}-1)(H-1) + r_1\bar{r}_2 - \bar{r}_1 r_2 \\
p_{14} &= (G-1)\bar{r}_2 - (H-1)\bar{r}_1 \\
p_{15} &= -\bar{r}_1 r_3 \\
p_{16} &= -(\bar{G}-1)r_2 + (\bar{H}-1)r_1 \\
p_{17} &= r_1\bar{r}_3 \\
p_{18} &= (G-1)\bar{r}_3 \\
p_{19} &= -(\bar{G}-1)r_3
\end{aligned}
\tag{5.26}
$$

$$
\begin{aligned}
\text{where} \quad \beta &= r_1 + \bar{r}_1 + G + \bar{G} - C - \bar{C} - 2 \\
\xi &= r_2 + \bar{r}_2 + r_3 + \bar{r}_3 + r_4 + \bar{r}_4 + H + \bar{H} - F - \bar{F} - 2
\end{aligned}
$$

and

$$
\begin{aligned}
r_1 &= -(G-1)\bar{C}, \qquad r_2 = -(H-1)\bar{B}, \qquad r_3 = -(H-1)(\bar{F}-\bar{B}), \qquad r_4 = B(\bar{F}-\bar{B}), \\
\bar{r}_1 &= -(\bar{G}-1)C, \qquad \bar{r}_2 = -(\bar{H}-1)B, \qquad \bar{r}_3 = -(\bar{H}-1)(F-B), \qquad \bar{r}_4 = \bar{B}(F-B).
\end{aligned}
$$

(5.27)

**Eleven Position Synthesis**

For the case $N = 11$, a multihomogeneous root count was performed by selecting the variable groups

$$\langle C, \bar{C}, G, \bar{G}, r_1, \bar{r}_1 \rangle, \quad \langle B, \bar{B}, F, \bar{F}, H, \bar{H}, r_2, \bar{r}_2, r_3, \bar{r}_3, r_4, \bar{r}_4 \rangle, \tag{5.28}$$

which determine the multihomogeneous Bézout number of 55,050,240. The root count was performed by expanding the polynomial

$$256\alpha_1^2\alpha_2^6(2\alpha_1 + 2\alpha_2)^{10} \tag{5.29}$$

and taking the coefficient in front of the monomial $\alpha_1^6\alpha_2^{12}$. This combinatoric procedure is outlined in Section 2.3.2.

**Nine Position Synthesis**

For the case $N = 9$, we take the coordinates of the second ground pivot $B$, $\bar{B}$ to be known. A multihomogeneous root count was performed by selecting the variable groups

$$\langle C, \bar{C}, G, \bar{G}, r_1, \bar{r}_1 \rangle, \quad \langle F, \bar{F}, H, \bar{H}, r_2, \bar{r}_2, r_3, \bar{r}_3, r_4, \bar{r}_4 \rangle, \tag{5.30}$$

which determine the multihomogeneous Bézout number of 286,720. The root count was performed by expanding the polynomial

$$16\alpha_1^2\alpha_2^6(2\alpha_1 + 2\alpha_2)^8 \tag{5.31}$$

and taking the coefficient in front of the monomial $\alpha_1^6 \alpha_2^{10}$.

## 5.2 Synthesis Solution

The synthesis equations were solved for the maximum $N = 11$ task positions on the Gordon cluster at the San Diego Supercomputer Center of the XSEDE supercomputering network using the polynomial homotopy software BERTINI. Rather than specify the requirements for a particular task, the input parameters, $(Q_j, S_j)$, $j = 1, \ldots, 10$, were set to random complex numbers to create a numerically general system. Homotopy paths were tracked over 40 hours on $512 \times 2.6$GHz cores. Nonsingular solutions were sorted by the Jacobian condition number of which 834,441 were found.

The roots for the general system need only be computed once, and can then be used as the start system for a parameter homotopy for any particular set of input parameters. Parameter homotopy is discussed in Section 2.3.5. The advantage of parameter homotopy is that the nonsingular endpoints of a general run are used as startpoints of a specific run so that only 834,441 paths need to be tracked in order to find all nonsingular solutions of a specific system.

The solutions to the synthesis equations can be grouped into pairs of function-cognates. For every Stephenson III linkage that coordinates angles $(\phi, \psi)$, there exists one other Stephenson III linkage that creates an identical coordination of $(\phi, \psi)$, called a function-cognate. The construction of Stephenson III function-cognates is described in Section 2.5.4. That is, for a Stephenson III linkage defined by $\{A, B, C, D, F, G, H\}$, there exists a function-cognate

defined by

$$
\begin{aligned}
A_c &= A, \\
B_c &= \frac{D-G}{H-G}(B-C)+C, \\
C_c &= C, \\
D_c &= D, \\
F_c &= \frac{D-G}{H-G}(F-C)+C, \\
G_c &= D-G+C, \\
H_c &= \frac{D-G}{H-G}(H-C)+C.
\end{aligned} \tag{5.32}
$$

The synthesis solutions are organized into cognate pairs. Since the solution method is subject to numerical errors, it is possible that the cognate of a solution will not be found. In this case, the missing cognate is constructed and added to the synthesis results.

## 5.3 Analysis

The synthesis solutions represent linkage design candidates. Once design candidates have been sorted into cognate pairs, they are analyzed to evaluate the performance of each design. The criteria for a successful design candidate is that the required accuracy points lie on a single trajectory of configurations without any singularities. Trajectories of configurations are determined for each design candidate following the procedure described in Section 2.4. The forward kinematics equations of a Stephenson III six-bar are presented in Eqn.(2.111). The solutions of the forward kinematics equations represent configurations of the linkage, which are sorted into trajectories using the routine outlined in Section 2.4.3.

Once all trajectories have been assembled for a linkage design candidate, each is checked to

see which and how many of the specified accuracy points they contain. A successful design candidate will produce a trajectory that moves through all 11 accuracy points. We term these designs 11-point mechanisms.

While linkage designs that contain all 11 accuracy points on a single trajectory is the goal, our design process identifies linkage designs with trajectories that move through less than 11 points as well. It is often the case that these mechanisms only slightly miss some accuracy points and may have other features useful to the designer, such as compact dimensions or reduced link overlap.

## 5.4 Example: Design of a Torque Cancelling Linkage

Survivors of strokes often suffer from a muscle control disorder called spasticity which causes increased stiffness in muscles of joints such as the wrist. Measurement data of intrinsic wrist stiffness can be found in [58], and shown in Fig. 5.2. The goal is to design a six-bar linkage that generates a specified torque profile which cancels spastic wrist stiffness.

### 5.4.1 Obtaining the Input Torque Profile

The torque profile that the Stephenson III is to reproduce, and then cancel, is derived from the data collected by Mirbagheri and Settle (2010) [58], who measured the intrinsic stiffness profile in the wrists of 21 stroke survivors. The data was taken from Fig. 4 of [58] and was

least-squares fit with the following fifth degree polynomial,

$$\begin{aligned} S(x) &= -0.3403347740344527x^5 + 2.3767146714792213x^4 \\ &+ 1.4329074166324411x^3 - 0.21211179259258692x^2 \\ &+ 0.5381754676253262x + 1.903537638831755. \end{aligned} \tag{5.33}$$

Because stiffness is the rate of change of a spring torque with respect to angular deflection, we integrate $S(x)$ to obtain the torque profile,

$$\begin{aligned} T(x) &= \int S(x)dx + c_0 = -0.056722462339075x^6 + 0.475342934295844x^5 \\ &+ 0.358226854158110x^4 - 0.070703930864196x^3 + 0.269087733812663x^2 \\ &+ 1.903537638831755x + 1.859723104149862. \end{aligned} \tag{5.34}$$

Eqns. (5.33) and (5.34) are shown in Fig. 5.2. The integration constant of the torque profile was set such that $T(-\pi/3) = 0$ which requires an unstable equilibrium at $x = -\pi/3$ rad. That means when the input link is rotated in the positive direction, a positive torque will act on it and move the link away from the equilibrium position. Thus, this linkage will behave like a spring with negative stiffness.

Table 5.1: Task position data as displayed in Fig. 5.3.

| $j$ | $x_j$ | $y_j$ |
|---|---|---|
| 1 | $-90°$ | $-6.665566873543°$ |
| 2 | $-68°$ | $-19.185437363846°$ |
| 3 | $-45°$ | $-18.280827185669°$ |
| 4 | $-22°$ | $-11.558672810110°$ |
| 0 | $0°$ | $0°$ |
| 5 | $19°$ | $15.000375068384°$ |
| 6 | $37°$ | $34.756261256578°$ |
| 7 | $54°$ | $61.184689516866°$ |
| 8 | $70°$ | $99.804701760596°$ |
| 9 | $82°$ | $148.305746675651°$ |
| 10 | $90°$ | $203.021804302295°$ |

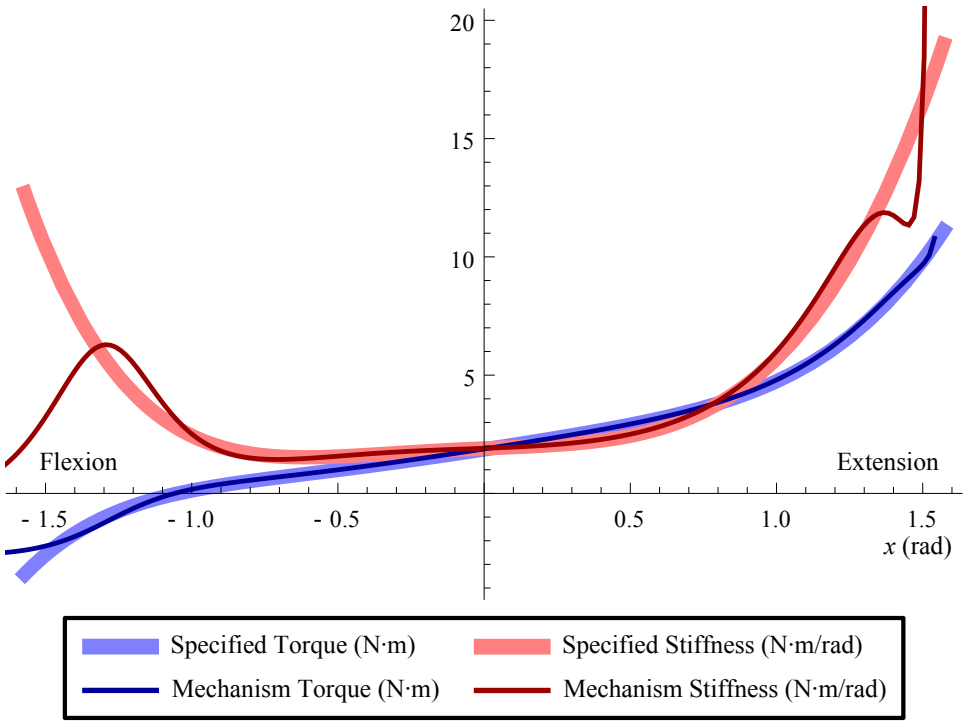



Figure 5.2: The desired torque and stiffness profiles as derived from Fig. 4 of [58], and the torque and stiffness profiles produced by the example mechanism. As well, the resultant torque taken by negating the mechanism generated torque from the desired torque.

### 5.4.2 Input-Output Function

The use of a function generator to provide a required input torque profile begins with the assumption that there are no losses from friction, wear, and dynamic effects, this yields the power balance,

$$T_{in}\dot{x} = T_{out}\dot{y}, \tag{5.35}$$

where $\dot{x}$ denotes the angular velocity of the input crank, and $\dot{y}$ is the angular velocity of the output crank.

For this design, the output torque $T_{out}$ is generated by a torsion spring with stiffness $k$ and

equilibrium angle $y_e$, therefore the input torque is given by,

$$T_{in} = -k(y - y_e)\frac{\dot{y}}{\dot{x}} = -k(y - y_e)\frac{dy}{dx}, \tag{5.36}$$

which is a function of the input angle $x$. Eqn. (5.36) can be solved for $y = f(x)$ to obtain the set of input-output angles needed to design a Stephenson III function generator.

Separate variables and integrate to obtain,

$$-\frac{1}{k}\int T_{in}(x)dx = \frac{1}{2}y^2 - y_e y \tag{5.37}$$

and then solve for $y$ to obtain,

$$y = f(x) = \pm\sqrt{-\frac{2}{k}\int T_{in}(x)dx + y_e^2} + y_e. \tag{5.38}$$

The "+" and "−" solutions are two different functions that produce the desired torque profile for given spring parameters $k$ and $y_e$.

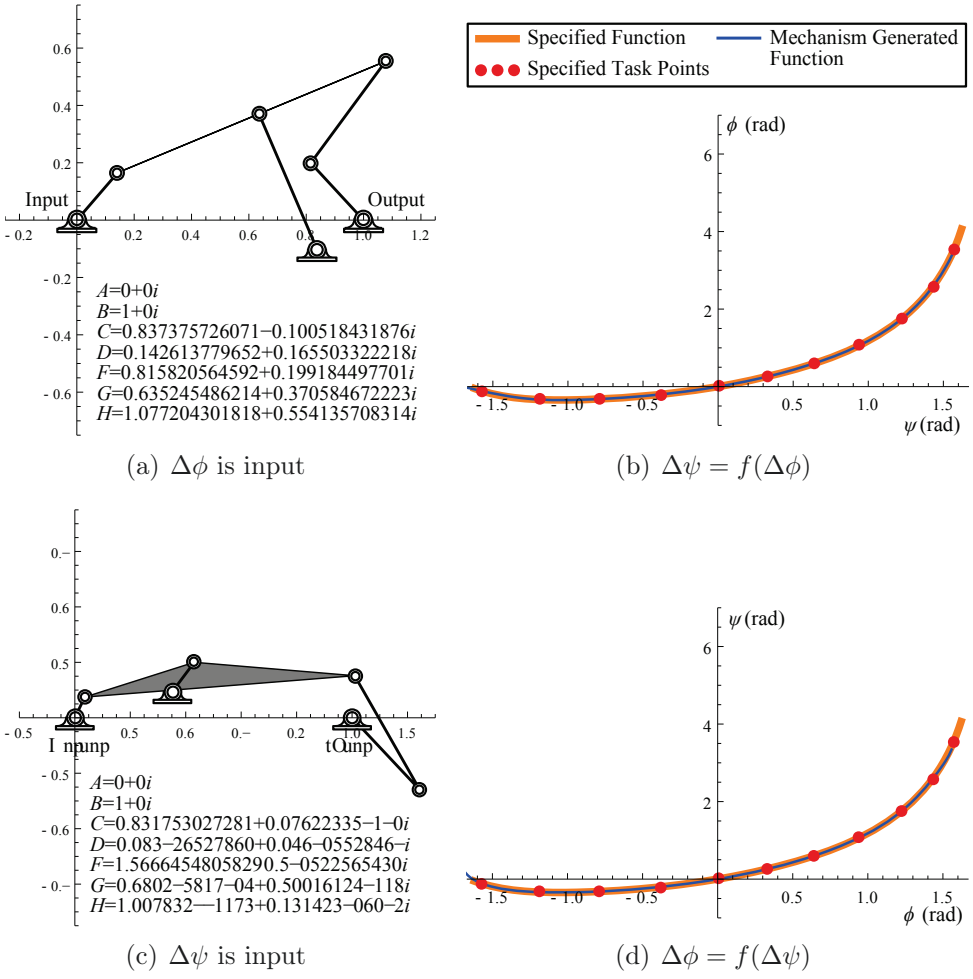
Table 5.2: Synthesis results for the cases of actuating Link $AD$, $\psi = f(\phi)$, and actuating Link $BF$, $\phi = f(\psi)$.

| | $\psi = f(\phi)$ | $\phi = f(\psi)$ |
|---|---|---|
| Linkage solutions | 8341 | 8583 |
| Cognates added | 712 | 647 |
| Linkages analyzed | 4547 | 5323 |
| 11 point mechanisms | 96 | 109 |
| 10 point mechanisms | 225 | 131 |
| 9 point mechanisms | 352 | 333 |
| 8 point mechanisms | 450 | 596 |
| 7 point mechanisms | 793 | 887 |
| 6 point mechanisms | 1389 | 1104 |
| Synthesis computation time (hr) | 0.8 | 0.9 |
| Analysis computation time (hr) | 4.0 | 6.1 |

The input-output function for the synthesis of the Stephenson III function generator is obtained by substituting (5.34) into (5.38), with the requirement that $k = 0.45$ N·m/rad and

$y_e = 2\pi$ rad. The "$-$" solution was taken to calculate the input-output $y = f(x)$ function shown in Figs. 5.3(b) and 5.3(d).

This input-output function was evaluated at 11 positions of $x$ to obtain the coordinated angles shown in Table. 5.1. We investigate producing this function using both $AD$ as the input, $(x, y) = (\Delta\phi, \Delta\psi)$, and $BF$ as the input, $(x, y) = (\Delta\psi, \Delta\phi)$. The use of BERTINI to obtain solutions to the synthesis equations is the same for both cases.




(a) $\Delta\phi$ is input

(b) $\Delta\psi = f(\Delta\phi)$

(c) $\Delta\psi$ is input

(d) $\Delta\phi = f(\Delta\psi)$

Figure 5.3: (a) A Stephenson III linkage actuated by Link $AD$ and (b) its mechanized function and (c) a Stephenson III linkage actuated by Link $BF$ and (d) its mechanized function.

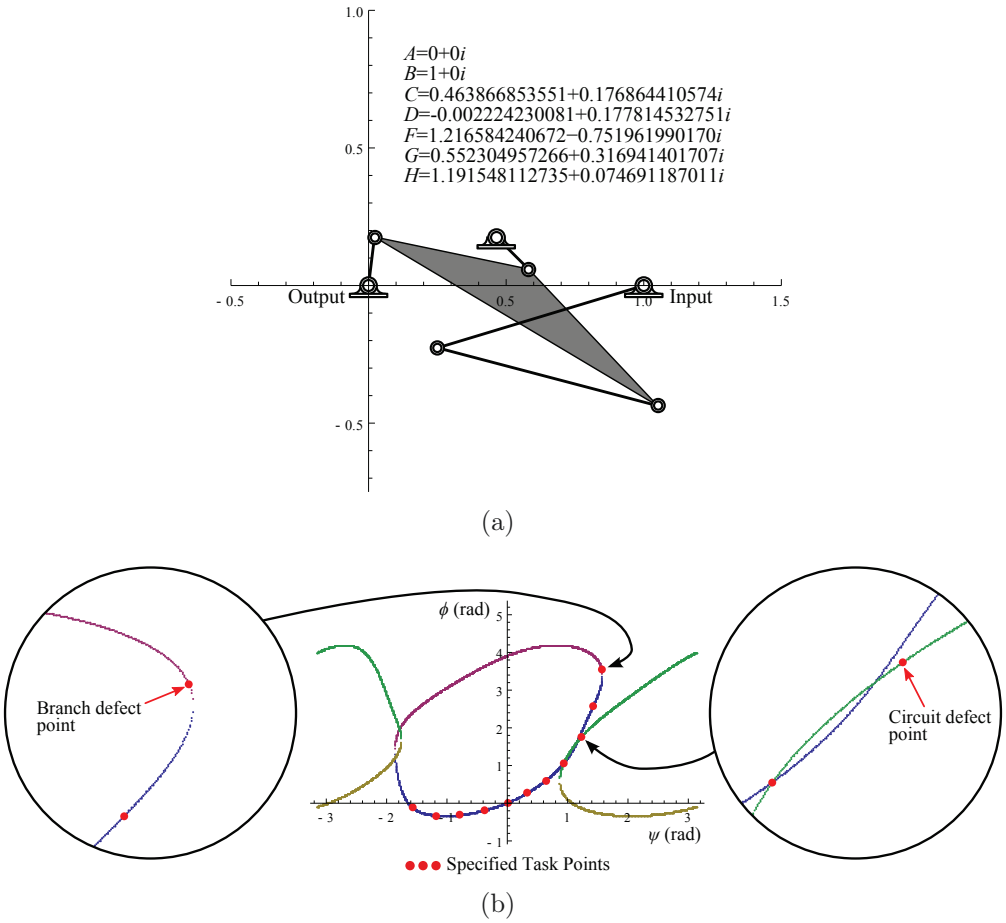



Figure 5.4: (a) A defective linkage and (b) its configuration space. Separate singularity free trajectories are indicated with different colors. A branch and circuit defect is illustrated.

### 5.4.3 Successful Linkage Designs

A summary of synthesis results is shown in Table 5.2. For the cases with $\phi$ as the input and $\psi$ as the input, BERTINI found 8,341 and 8,583 solutions that corresponded to physical linkages, respectively. Each solution set was then processed to add cognate solutions and remove solutions with very small or large link lengths such that 4,547 and 5,323 solutions were prepared for each case. The performance of these linkages was analyzed in order to categorize mechanisms by the number of accuracy points they can achieve in a singularity-free trajectory from 6 to 11 points. For $\phi$ as the input and $\psi$ as the input, there were 96 and 109 mechanisms, respectively, that passed through all 11 points. The total computation

time for each case was 5 hrs and 7 hrs performed on 64 × 2.2GHz nodes of the UC Irvine High Performance Computing Cluster.

Fig. 5.3 shows two 11–point mechanisms and their input-output functions, one with $\phi$ as the input and one with $\psi$ as the input. Fig. 5.2 shows the torque and stiffness profiles produced by the linkage shown in Fig. 5.3(a).

Fig. 5.4 illustrates some common defects found in mechanisms that achieve less than 11 accuracy points. The figure depicts the configuration of a 9–point mechanism. Notice that one accuracy point is on a separate trajectory yielding a circuit defect. However, in the second case the accuracy point is on the same trajectory but separated from the others by a singularity, known as a branch defect. Despite these defects, the mechanism tracks the desired trajectory very closely and is useful to the designer.

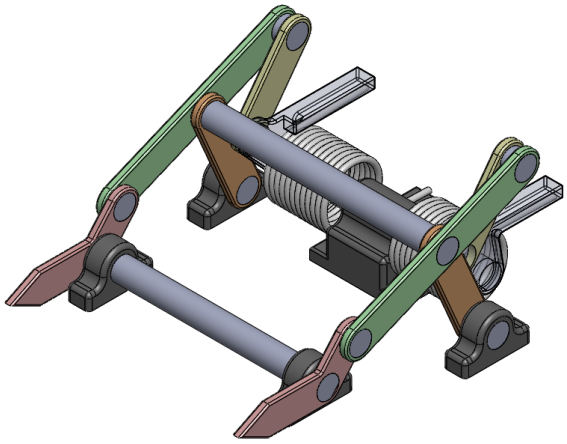


Figure 5.5: A solid model of the design shown in Fig. 5.3(a).

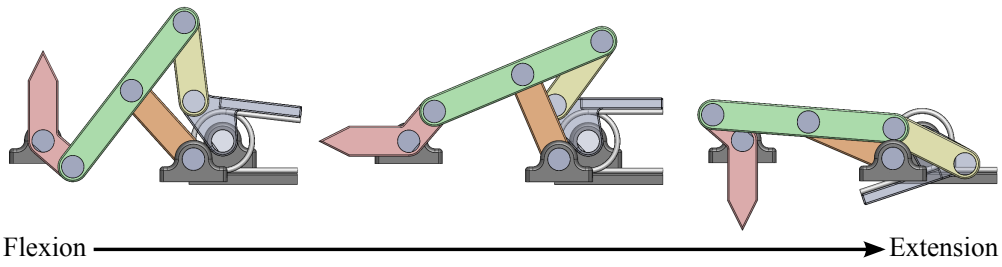



Figure 5.6: A solid model of the design shown in Fig. 5.3(a) shown in three poses.

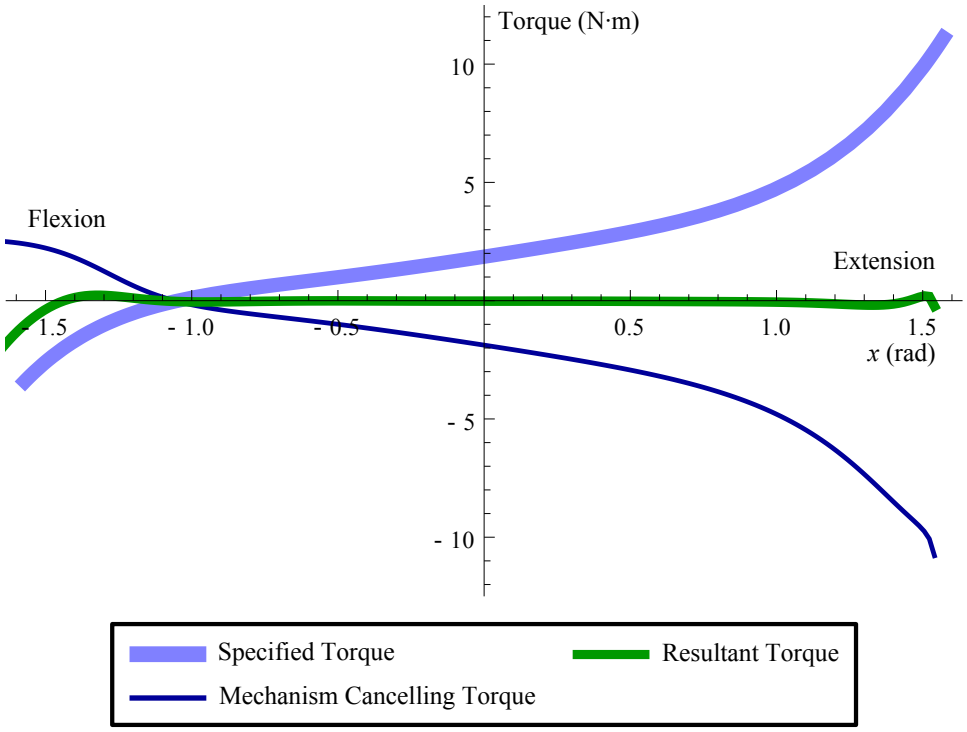



Figure 5.7: The resultant of the desired torque and the torque generated by the mechanism shown in Fig. 5.3(a) is near zero.

### 5.4.4 Example Design

A solid model of the 11–point mechanism shown in Fig. 5.3(a) was produced and is shown in Figs. 5.5 and 5.6. This device can be used to cancel the torque that arises from the intrinsic stiffness of stroke survivors that suffer from spasticity. The negation of the mechanism generated torque from the stroke survivor data yields a resultant torque near zero shown in Fig. 5.7.

## 5.5 Summary

This chapter presented a synthesis procedure for Stephenson III six-bar function generators that achieve 11 coordinated input and output angles. It is shown that the structure of the synthesis equations yields a polynomial system with multihomogeneous degree of 55,050,240.

The polynomial homotopy software BERTINI was used to compute 834,441 nonsingular solutions for a general set of parameters. These solutions were then used in a parameter homotopy to obtain the dimensions of Stephenson III six-bar function generators. Each solution is analyzed to ensure its performance, and it is shown that successful designs may have small errors at specific accuracy points. An example design uses the Stephenson III linkage to transform a linear spring on the output link into a torque profile on the input link that cancels the intrinsic stiffness of the wrist of a stroke survivor.

# Chapter 6

# Watt I Motion Generation

This chapter presents the synthesis equations for a Watt I six-bar linkage, Fig. 6.1, to guide a rigid body through $N$ specified task positions. This is a generalization of the motion generation problem for four-bar linkages, [4, 7, 14].

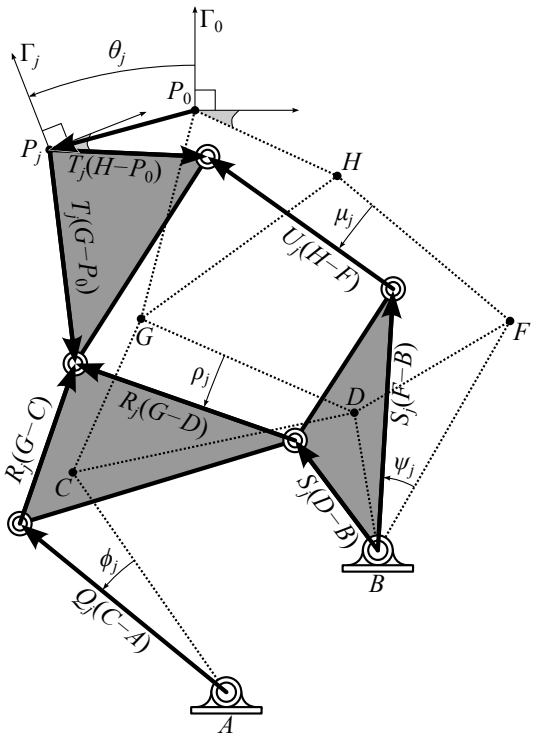



Figure 6.1: Cartesian and polar coordinates of a complex number

Our formulation yields 28 equations in 28 unknowns for the maximum number of task positions, $N = 8$, which has a 13-homogeneous Bézout degree of $1.94 \times 10^{10}$. The size of this problem seems to be beyond the ability of polynomial continuation software to formulate a start system. In what follows, we present the solution for six task positions with the locations of the two ground pivots specified.

## 6.1 Synthesis Formulation

A Watt I six-bar linkage, Fig. 6.1, can be viewed as a four-bar linkage sitting on top of another four-bar linkage. The base four-bar linkage consists of the ground link $AB$, the cranks $AC$ and $BDF$, and the floating link $CDG$. The second four-bar linkage is attached to $BDF$, has cranks $CDG$ and $FH$, and the floating link $GHP$. The floating link $GHP$ is considered to be the end-effector of the system. The goal is to place a task frame attached to $GHP$ in $N$ positions $\Gamma_j$ for $j = 0, \ldots, N-1$.

A task frame $\Gamma_j$ is defined by the coordinates of its origin $P_j$ and its orientation angle measured relative to a fixed frame. Complex coordinates are used to define these parameters, so

$$\Gamma_j = (x_j + iy_j, e^{i\theta_j}) = (P_j, T_j). \tag{6.1}$$

For convenience, choose the fixed frame to coincide with the first task frame, so $\Gamma_0 = (0, 1)$, Fig. 6.1.

The coordinates of the seven pivots are identified in position $j = 0$ by the complex numbers $A$, $B$, $C$, $D$, $F$, $G$, $H$ as shown in Fig. 6.1. The coordinates of these numbers are 14 unknowns of the linkage synthesis problem. Following Wampler (1996) [53], we consider these numbers and their complex conjugates as separate unknowns, which yields a total of

28 unknowns.

The synthesis equations are obtained from three vector loop equations for the Watt I six-bar linkage formulated in each of the specified task positions. The rotations of each joint relative to the initial configuration are defined by the rotation operators and their conjugates,

$$(Q_j, \bar{Q}_j) = (e^{i\phi_j}, e^{-i\phi_j}), \quad (R_j, \bar{R}_j) = (e^{i\rho_j}, e^{-i\rho_j}),$$
$$(S_j, \bar{S}_j) = (e^{i\psi_j}, e^{-i\psi_j}), \quad (U_j, \bar{U}_j) = (e^{i\mu_j}, e^{-i\mu_j}). \tag{6.2}$$

The overbar denotes the complex conjugate. Notice that these pairs must satisfy the normalization conditions

$$Q_j\bar{Q}_j = 1, \quad R_j\bar{R}_j = 1, \quad S_j\bar{S}_j = 1, \quad U_j\bar{U}_j = 1, \qquad j = 1, \ldots, N-1. \tag{6.3}$$

This yields $4(N-1)$ equations in the unknown rotation operators. Using these rotation operators, we obtain three sets of loop equations from Fig. 6.1,

$$\mathcal{A}_j = \begin{cases} A + Q_j(C-A) + R_j(G-C) - T_jG &= P_j \\ \bar{A} + \bar{Q}_j(\bar{C}-\bar{A}) + \bar{R}_j(\bar{G}-\bar{C}) - \bar{T}_j\bar{G} &= \bar{P}_j \end{cases} \quad j = 1, \ldots, N-1, \tag{6.4}$$

$$\mathcal{B}_j = \begin{cases} B + S_j(D-B) + R_j(G-D) - T_jG &= P_j \\ \bar{B} + \bar{S}_j(\bar{D}-\bar{B}) + \bar{R}_j(\bar{G}-\bar{D}) - \bar{T}_j\bar{G} &= \bar{P}_j \end{cases} \quad j = 1, \ldots, N-1, \tag{6.5}$$

$$\mathcal{C}_j = \begin{cases} B + S_j(F-B) + U_j(H-F) - T_jH &= P_j \\ \bar{B} + \bar{S}_j(\bar{F}-\bar{B}) + \bar{U}_j(\bar{H}-\bar{F}) - \bar{T}_j\bar{H} &= \bar{P}_j \end{cases} \quad j = 1, \ldots, N-1. \tag{6.6}$$

This is $6(N-1)$ equations in the pivot location and rotation operator unknowns.

The collection of the joint normalization conditions (6.3) and the three sets of loop equations

$\mathcal{A}_j$, $\mathcal{B}_j$, and $\mathcal{C}_j$ yields $10(N-1)$ equations in the $8N+6$ unknowns,

$$\left\langle A, \bar{A}, B, \bar{B}, C, \bar{C}, D, \bar{D}, F, \bar{F}, G, \bar{G}, H, \bar{H} \right\rangle \left\langle Q_j, \bar{Q}_j, R_j, \bar{R}_j, S_j, \bar{S}_j, U_j, \bar{U}_j \right\rangle$$
$$j = 1, \ldots, N-1. \tag{6.7}$$

For $N = 8$, this system is square and has a total degree $2^{70} \approx 1.18 \times 10^{21}$.

The joint rotation angles $(Q_j, \bar{Q}_j)$ can be eliminated by solving the loop equations $\mathcal{A}_j$ and substituting the result into its normalization condition of (6.3). Similarly, $(U_j, \bar{U}_j)$ are eliminated using the loop equations $\mathcal{C}_j$ and the normalization condition of (6.3). The resulting equations are

$$(P_j - A + R_j(C-G) + T_jG)(\bar{P}_j - \bar{A} + \bar{R}_j(\bar{C}-\bar{G}) + \bar{T}_j\bar{G}) = (C-A)(\bar{C}-\bar{A}), \tag{6.8}$$
$$(P_j - B + S_j(B-F) + T_jH)(\bar{P}_j - \bar{B} + \bar{S}_j(\bar{B}-\bar{F}) + \bar{T}_j\bar{H}) = (H-F)(\bar{H}-\bar{F}), \tag{6.9}$$
$$j = 1, \ldots, N-1.$$

This yields a system of $6(N-1)$ equations in $4N+10$ unknowns. The degree of this system is $(2^4 \cdot 3^2)^{N-1}$, which for $N = 8$ is approximately $1.28 \times 10^{15}$.

Now, eliminate the variables $(R_j, \bar{R}_j)$ by solving the loop equations $\mathcal{B}_j$. Substitute the result into the normalization conditions of (6.3) to obtain,

$$(P_j - B + S_j(B-D) + T_jG)(\bar{P}_j - \bar{B} + \bar{S}_j(\bar{B}-\bar{D}) + \bar{T}_j\bar{G}) = (G-D)(\bar{G}-\bar{D}),$$
$$j = 1, \ldots, N-1. \tag{6.10}$$

Substitution of $(R_j, \bar{R}_j)$ into (6.8) gives,

$$((G-D)(P_j - A + T_j G) + w_j(C-G))((\bar{G}-\bar{D})(\bar{P}_j - \bar{A} + \bar{T}_j\bar{G}) + \bar{w}_j(\bar{C}-\bar{G}))$$
$$= (C-A)(\bar{C}-\bar{A})(G-D)(\bar{G}-\bar{D}), \quad j = 1, \ldots, N-1, \tag{6.11}$$

where

$$w_j = P_j - B + S_j(B-D) + T_j G, \quad \bar{w}_j = \bar{P}_j - \bar{B} + \bar{S}_j(\bar{B}-\bar{D}) + \bar{T}_j\bar{G}. \tag{6.12}$$

After this elimination process, the system consists of the normalization conditions, $S_j\bar{S}_j = 1$ and (6.9), (6.10), and (6.11). which tallies to $4(N-1)$ equations in $2N+12$ unknowns. The total degree of these equations is $(2\cdot 3^2\cdot 5)^{N-1}$, which for $N = 8$ is approximately $4.78\times 10^{13}$. To compute the multihomgeneous degree, we introduce the 13 groups,

$$\langle A, C\rangle, \langle \bar{A}, \bar{C}\rangle, \langle B, D, G\rangle, \langle \bar{B}, \bar{D}, \bar{G}\rangle, \langle F, H\rangle, \langle \bar{F}, \bar{H}\rangle,$$
$$\text{and } \langle S_j, \bar{S}_j\rangle, \quad j = 1, \ldots, N-1. \tag{6.13}$$

For $N = 8$ this grouping yields a 13-homogeneous Bézout degree of $1.94\times 10^{10}$.

Table 6.1: Eight task positions for Watt I motion generation.

| $j$ | $x_j$ | $y_j$ | $\theta_j$ (°) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | −0.08849958 | 0.63143282 | 5.95778240 |
| 2 | −0.49237057 | 1.35531439 | 17.35017046 |
| 3 | −1.14387347 | 1.93359099 | 31.25403761 |
| 4 | −1.92567709 | 2.25400993 | 45.59910917 |
| 5 | −2.68188520 | 2.29738239 | 58.34343392 |
| 6 | −3.27440782 | 2.13240439 | 68.00251408 |
| 7 | −3.61649767 | 1.86582938 | 73.93850323 |

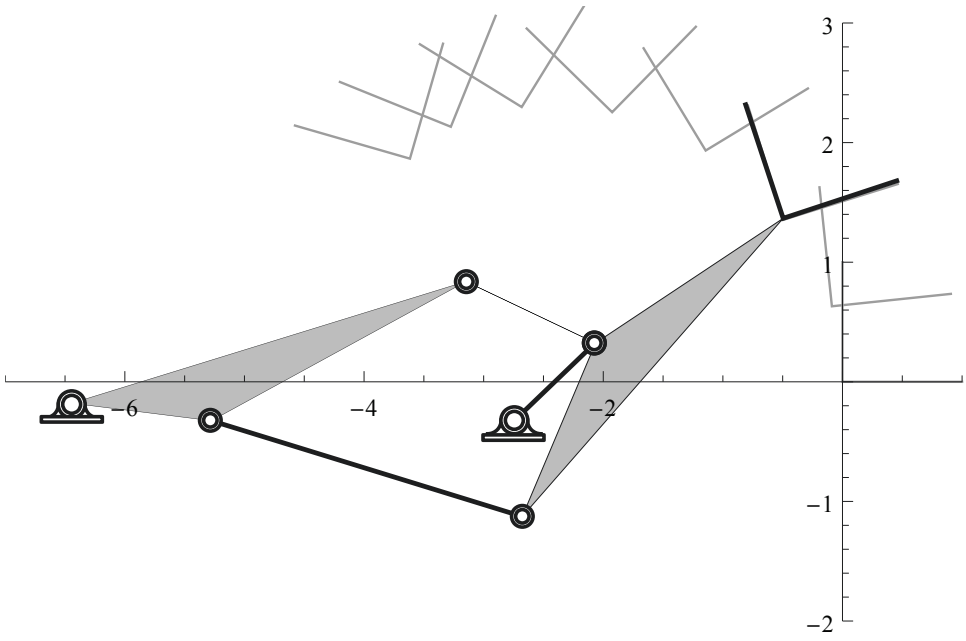



Figure 6.2: Watt I six-bar linkage that passes through eight specified task positions.

## 6.2 Solution for Eight Positions by the Newton-Raphson Method

The synthesis equations were validated by generating a few solutions using an implementation of Newton's method provided by the MATHEMATICA computational software package called *FindRoot*. We solved for the task positions listed in Table 6.1. Example solutions are listed in Table 6.2. Solution 1 is shown in Fig. 6.2.

Newton's method was used to solve the synthesis equations for a randomized set of 100,000 of start points. The computation took 46 minutes. All computations of this chapter were done in parallel on a 64 core machine. The start point values of the isotropic coordinate pairs $(A, \bar{A}), \ldots, (H, \bar{H})$ were randomized within a $10 \times 10$ box centered on the origin of the complex plane. The start point values of $(S_j, \bar{S}_j)$, $j = 1, \ldots, 7$ were specified to be random complex numbers of unit magnitude. All isotropic coordinate start points maintained conjugate relationships throughout randomization.

/[htb]

Table 6.2: A sample of solutions to the eight position synthesis equations.

| | Solution 1 | | Solution 2 | | Solution 3 | | Solution 4 | |
|---|---|---|---|---|---|---|---|---|
| | Re | Im | Re | Im | Re | Im | Re | Im |
| $A$ | −2.750557 | −0.331168 | −2.688674 | −0.294929 | −3.464722 | −0.981841 | −2.133969 | −1.241728 |
| $B$ | −6.443286 | −0.179046 | −2.918281 | −1.066570 | −2.979378 | −1.882345 | −2.456924 | −1.199599 |
| $C$ | −1.819766 | −0.499668 | −1.695131 | −0.453426 | −2.776606 | −1.762104 | −1.973392 | −1.046694 |
| $D$ | −2.995680 | −0.283615 | −3.054520 | −1.255878 | −2.796190 | −2.001717 | −2.233036 | −1.203616 |
| $F$ | −5.386180 | −0.695399 | −3.134324 | −1.075385 | −2.965783 | −2.550331 | −2.403520 | −1.686455 |
| $G$ | −1.815717 | −0.498056 | −1.492112 | −0.390568 | −3.157675 | −1.868198 | −2.884410 | −1.164443 |
| $H$ | −2.852650 | −1.683542 | −3.191124 | −1.473735 | −2.619687 | −2.233975 | −2.041865 | −1.797497 |
| $S_1$ | 0.999690 | −0.024890 | −0.384794 | −0.923002 | 0.686117 | 0.727491 | 0.915234 | 0.402922 |
| $S_2$ | 0.999171 | −0.040701 | −0.764101 | 0.645097 | 0.236524 | 0.971626 | 0.386722 | 0.922197 |
| $S_3$ | 0.998217 | −0.059687 | −0.985651 | −0.168793 | −0.110517 | 0.993874 | −0.931545 | 0.363626 |
| $S_4$ | 0.838386 | 0.545077 | −0.997586 | −0.069446 | −0.405476 | 0.914105 | −0.664287 | 0.747478 |
| $S_5$ | 0.831441 | 0.555613 | −0.965251 | 0.261323 | −0.617109 | 0.786878 | −0.979973 | 0.199132 |
| $S_6$ | 0.986016 | −0.166650 | −0.777383 | 0.629027 | −0.824702 | 0.565568 | −0.980540 | −0.196322 |
| $S_7$ | 0.915030 | 0.403386 | 0.789645 | 0.613565 | −0.870772 | 0.491688 | −0.847467 | −0.530849 |

This yielded 28,890 solutions, of which only 64 were suitable for linkage analysis. Solutions suitable for linkage analysis are those whose isotropic pairs are truly conjugates and that have link lengths $AB$, $AC$, $BD$, $DF$, $CD$, $DG$, $FH$, $GH$ greater than 0.1. Only two of the suitable solutions corresponded to defect-free linkages, Solutions 1 and 4 in Table 6.2.

## 6.3 Solution for Six Positions by Homotopy

If only $N = 6$ positions are specified, then we may specify four conditions on the design parameters. One choice is to specify the ground pivot locations $(A, \bar{A})$ and $(B, \bar{B})$. This is natural for design purposes and also leads to considerable simplification in the numerical treatment of the problem. With $N = 6$ there are 20 synthesis equations in 20 unknowns with a total degree of $(2 \cdot 3^2 \cdot 5)^5 \approx 5.90 \times 10^9$. In this case, however, the unknowns separate into the ten groups

$$\langle C, \bar{C} \rangle, \langle D, \bar{D} \rangle, \langle F, \bar{F} \rangle, \langle G, \bar{G} \rangle, \langle H, \bar{H} \rangle, \text{and } \langle S_j, \bar{S}_j \rangle, \quad j = 1, \ldots, 5, \tag{6.14}$$

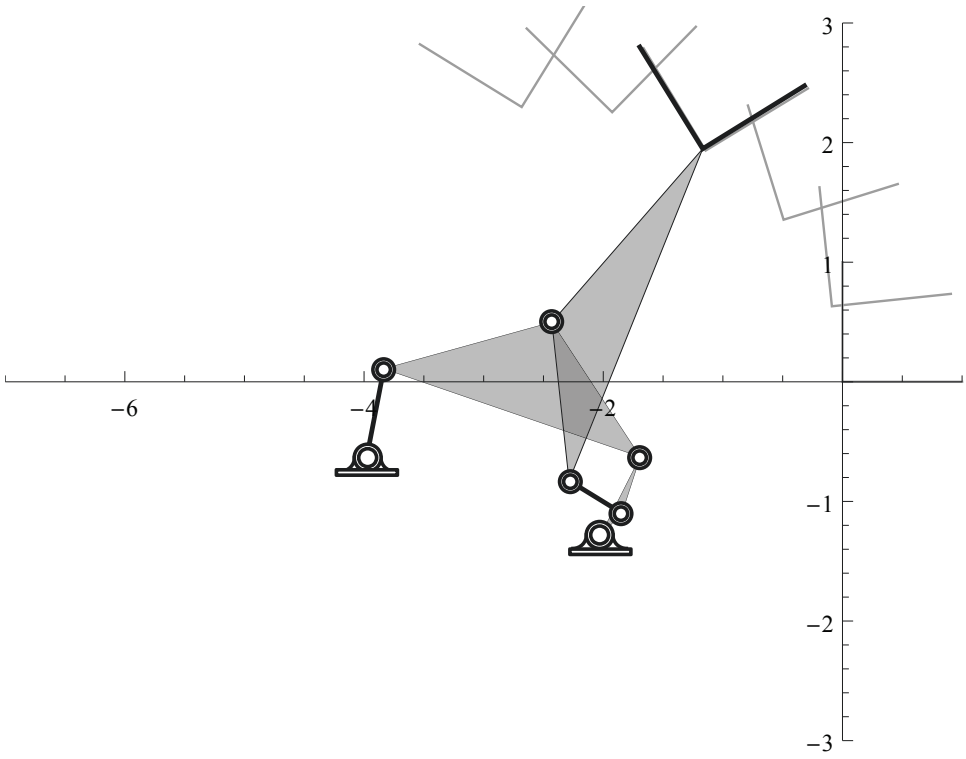



Figure 6.3: A Watt I six-bar linkage that reaches six task positions.

which yields the ten-homogeneous Bézout degree 1,998,720. The Bertini numerical continuation software was used to obtain a complete solution to this problem.

We solved this system using regeneration homotopy, which is described in Section 2.3.6 and in more detail in Bates et al. (2013) [9]. Our level by level results of regeneration are shown in Table 6.3. Finally, 5,735 nonsingular solutions were found after tracking a total of 51,000 paths. The computation took about 7 hours.

The solution set and parameters from the regeneration run provided a parameter homotopy for the synthesis equations. The base pivots were specified to be $A = -3.976225 - 0.623063i$ and $B = -2.024139 - 1.285906i$. This computation took about 5 minutes and yielded 5,556 nonsingular solutions of which 243 were suitable to analyze and 43 were defect-free. Table 6.4 displays defect-free solutions. Fig. 6.3 shows Solution 1.

Table 6.3: Paths tracked and nonsingular endpoints found at each level of regeneration.

| Level | Paths | Non-singular Endpoints | Level | Paths | Non-singular Endpoints | Level | Paths | Non-singular Endpoints |
|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 3 | 7 | 2062 | 980 | 14 | 2042 | 808 |
| 1 | 9 | 9 | 8 | 2022 | 868 | 15 | 1616 | 1548 |
| 2 | 25 | 25 | 9 | 1346 | 500 | 16 | 3095 | 2880 |
| 3 | 63 | 61 | 10 | 900 | 900 | 17 | 5760 | 4867 |
| 4 | 138 | 130 | 11 | 1600 | 1600 | 18 | 9734 | 6638 |
| 5 | 548 | 437 | 12 | 2400 | 2341 | 19 | 13275 | 5735 |
| 6 | 1421 | 788 | 13 | 2941 | 2042 | | | |

Table 6.4: A sample of solutions to the six position synthesis equations.

| | Solution 1 | | Solution 2 | | Solution 3 | | Solution 4 | |
|---|---|---|---|---|---|---|---|---|
| | Re | Im | Re | Im | Re | Im | Re | Im |
| $C$ | −3.277663 | −0.360479 | −2.438554 | −0.627923 | −2.672919 | −1.937923 | −2.787652 | −3.185862 |
| $D$ | −1.612335 | −1.898356 | −0.432594 | −0.991854 | −1.172132 | −2.538575 | −3.547796 | −1.100739 |
| $F$ | −1.948622 | −1.535115 | −0.322911 | −2.589864 | −2.175566 | −1.663265 | −6.427970 | −0.222094 |
| $G$ | −1.835116 | −0.569740 | −1.204924 | −0.030339 | −2.786399 | −2.523568 | −2.103420 | −1.644206 |
| $H$ | −2.410335 | −1.772735 | 0.116380 | −2.016269 | −4.636286 | −1.395320 | −2.387046 | −2.239104 |
| $S_1$ | 0.626133 | 0.779716 | 0.956723 | 0.290999 | 0.989602 | 0.143835 | −0.978732 | 0.205142 |
| $S_2$ | 0.040843 | 0.999166 | 0.783349 | 0.621582 | −0.876928 | −0.480621 | 0.999287 | −0.037744 |
| $S_3$ | −0.473696 | 0.880688 | 0.493733 | 0.869614 | −0.916843 | −0.399247 | −0.905977 | −0.423328 |
| $S_4$ | −0.822092 | 0.569355 | 0.144621 | 0.989487 | 0.403962 | 0.914776 | 0.998733 | 0.050325 |
| $S_5$ | −0.978809 | 0.204776 | −0.194393 | 0.980924 | −0.977576 | −0.210581 | 0.985576 | 0.169231 |

## 6.4 Summary

The chapter derives the synthesis equations for Watt I six-bar mechanisms that guides a body through $N$ task positions. For $N = 8$, we were unable to solve this problem using polynomial continuation but did obtain solutions via repeated trials of Newton's method.

For the simpler problem of six positions with specified ground pivots, we computed a complete solution using polynomial regeneration. This gives a set of 5,735 solutions that serve as the start points for a parameter homotopy to solve any particular case in about five minutes on a 64 core parallel computer.

# Chapter 7

# Stephenson Path Generation

This chapter describes the use of Stephenson II and Stephenson III six-bar function generators to control an RR chain so that it generates a desired trajectory. Eleven configurations of the chain defined by points $P_j$, $j = 0, \ldots, 10$, along the trajectory provide a set of coordinated joint angles that are used as the accuracy points for the design of a Stephenson II and III function generators as described in Chapters 4 and 5. The ground link of the function generator becomes the first link of the RR chain, thus the resulting linkage design is a scaled kinematic inversion of the function generator.

The Stephenson II and III six-bar linkages can be attached to the RR chain each in two different ways. The Stephenson II function generator yields a Stephenson III path generator and a Stephenson II path generator with the trace point on the ternary ternary link. The Stephenson III function generator yields a Stephenson I path generator and a Stephenson II path generator with the trace point on a binary floating link. Thus, we obtain four Stephenson path generators, Fig. 7.1.

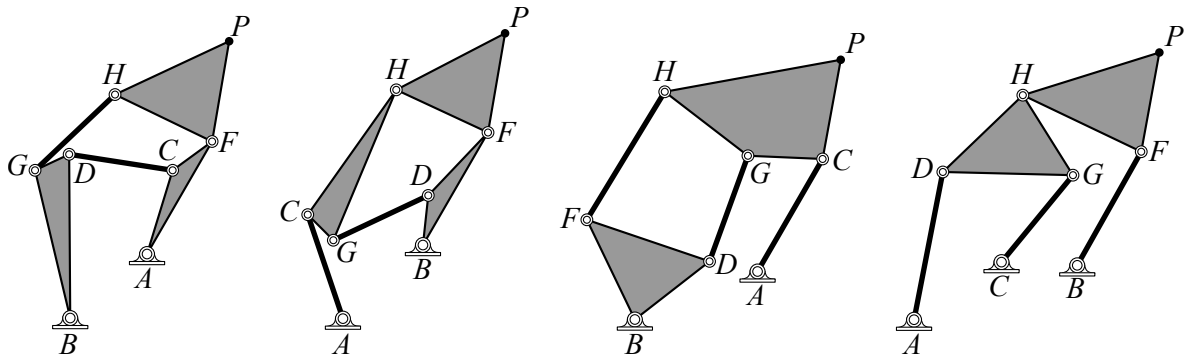



Figure 7.1: The four Stephenson six-bar path generators that guide the trace point $P$. (a) the Stephenson I, (b) the Stephenson II with the trace point on the binary floating link, (c) the Stephenson II with the trace point on the ternary floating link, and (d) the Stephenson III.

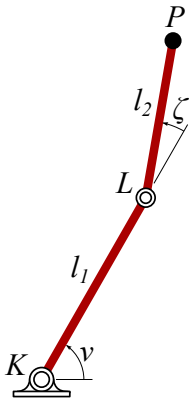



Figure 7.2: The user defined RR chain with a trace point that defines the required trajectory.

## 7.1 The RR Chain

The synthesis procedure begins with the specification of a RR chain, see Fig. 7.2. The ground pivot, moving pivot, and trace point are located by $K$, $L$, and $P$, respectively,

$$\begin{aligned} K &= K_x + K_y i, \\ L &= L_x + L_y i, \\ P &= P_x + P_y i. \end{aligned} \tag{7.1}$$

The angle $\nu$ of the first link of the RR chain is measured from global horizontal. The angle $\zeta$ of the second link is measured relative to the first link. The exponential rotation operators defined by $\nu$ and $\zeta$ are

$$V = e^{i\nu}, \qquad Z = e^{i\zeta}, \tag{7.2}$$

The location of the trace point $P$ can then be written as

$$P = K + Vl_1 + VZl_2. \tag{7.3}$$

Next, we solve for coordinated angle pairs $(\nu, \zeta)$ that move the trace point through a set of specified points $P_j$, for $j = 0, \ldots, 10$. The angle $\zeta$ is related to the joint coordinates by the the law of cosines, where

$$(P - K)(\bar{P} - \bar{K}) = l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - \zeta), \tag{7.4}$$

where the overbar notation denotes the complex conjugate. Instead of solving for $\zeta$ directly, it is convenient to substitute,

$$Z + \bar{Z} = -2\cos(\pi - \zeta), \tag{7.5}$$

which is shown to be true using Eqn. (7.2) and Euler's formula. The terms are rearranged to obtain,

$$l_1 l_2 Z + l_1^2 + l_2^2 - (P - K)(\bar{P} - \bar{K}) + l_1 l_2 \bar{Z} = 0. \tag{7.6}$$

Since $\bar{Z} = \frac{1}{Z}$, Eqn. (7.6) is quadratic when multiplied by $Z$ and can be solved by the quadratic formula to obtain two solutions,

$$Z^{\pm} = \frac{1}{2l_1 l_2}\Big((P-K)(\bar{P}-\bar{K}) - l_1^2 - l_2^2 \pm \sqrt{((P-K)(\bar{P}-\bar{K}) - l_1^2 - l_2^2)^2 - 4l_1^2 l_2^2}\Big) \tag{7.7}$$

Eqn. (7.7) indicates "+" and "−" solutions which correspond to elbow up and elbow down configurations of the RR chain. The corresponding solutions of $V$ are found from solving Eqn. (7.3),

$$V^{\pm} = \frac{P-K}{l_1 + (Z^{\pm})l_2} \tag{7.8}$$

Eqns. (7.7) and (7.8) are computed for $P = P_j$, for $j = 0, \ldots, 10$, to obtain values $(V_j, Z_j)$. For each position, it is the free choice of the designer to select an elbow up or elbow down configuration. The coordinated pairs $(V_j, Z_j)$ correspond to angle pairs $(\nu_j, \zeta_j)$ for $j = 0, \ldots, 10$. Therefore, in order to find mechanisms that move the trace point through $P_j$, we must constrain the RR chain such that it achieves coordinated angle pairs $(\nu_j, \zeta_j)$ for $j = 0, \ldots, 10$.

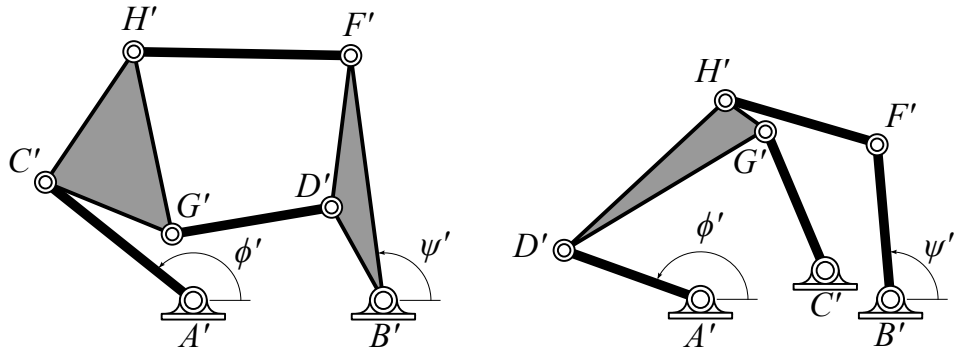



Figure 7.3: (a) Stephenson II function generator, and (b) Stephenson III function generator.

## 7.2 Inversion to Function Generation

Coordinating the angle pairs $(\nu_j, \zeta_j)$, $j = 0, \ldots, 10$, is the objective of function generation. We choose to coordinate these angles with six-bar function generators of which only the Stephenson II and III are capable of achieving 11 accuracy points at their ground pivots. Stephenson II and III function generators are shown in Fig. 7.3 where their pivot locations are labelled with primed notation $A'$, $B'$, $C'$, $D'$, $F'$, $G'$, and $H'$. Furthermore, each of these linkages can be connected to the RR chain (Fig. 7.2) either one of two ways to produce the desired coordination, that is, either joint $A'$ lies coincident to $K$ and $B'$ lies coincident to $L$ or vice versa, see Figs. 7.4–7.7. All four resulting types of path generators are described in detail below.

### 7.2.1 Stephenson I Path Generator

A Stephenson I path generator is formed by attaching a Stephenson III function generator to the RR chain $KLP$ such that $A'$ lies coincident to $K$ and $B'$ lies coincident to $L$ as shown in Fig. 7.4. In this case, ground Link $A'B'C'$ of the function generator is allowed to pivot about $A'$ and Link $A'D'$ is set as the new ground link. The transformation that scales, rotates, and translates the function generator such that $A'$ and $B'$ line up with $K$ and $L$ is $\mathcal{T}$ defined from Eqn. (2.14) as

$$\mathcal{T}(p) = \frac{L - K}{B' - A'}(p - A') + K \tag{7.9}$$

where the pivots $A$, $B$, $C$, $D$, $F$, $G$, $H$ of the new path generating linkage are

$$\begin{aligned} &A = \mathcal{T}(A'), \quad B = \mathcal{T}(D'), \quad C = \mathcal{T}(C'), \quad D = \mathcal{T}(G'), \\ &F = \mathcal{T}(B'), \quad G = \mathcal{T}(H'), \quad H = \mathcal{T}(F'). \end{aligned} \tag{7.10}$$

Trace point $P$ is then rigidly attached to Link $FH$ and the Stephenson I path generator is formed. Fig. 7.4 shows the angles $\nu$ and $\zeta$ are related to $\phi'$ and $\psi'$ of the function generator by

$$(\Delta\phi'_j, \Delta\psi'_j) = (-\Delta\nu_j, \Delta\zeta_j), \quad j = 1, \ldots, 10, \tag{7.11}$$

where

$$\Delta\phi'_j = \phi'_j - \phi'_0, \quad \Delta\psi'_j = \psi'_j - \psi'_0, \quad \Delta\nu_j = \nu_j - \nu_0, \quad \Delta\zeta_j = \zeta_j - \zeta_0. \tag{7.12}$$

Eqn. (7.11) sets up the function generation synthesis problem and Eqn. (7.10) is used to transform the synthesis results of the Stephenson III function generator to Stephenson I path generator results.

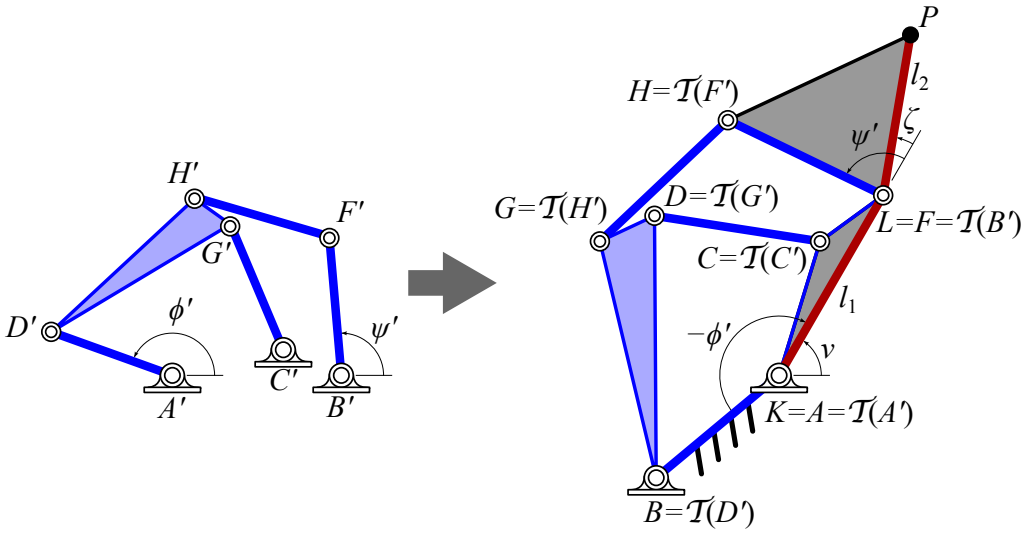



Figure 7.4: A Stephenson I path generator obtained by coordinating the RR joints with a Stephenson III function generator.

### 7.2.2 Stephenson II Path Generator with Trace Point on Binary Link

A Stephenson II path generator is formed by attaching a Stephenson III function generator to the RR chain $KLP$ such that $A'$ lies coincident to $L$ and $B'$ lies coincident to $K$ as shown in Fig. 7.5. In this case, ground Link $A'B'C'$ of the function generator is allowed to pivot about $B'$ and Link $B'F'$ is set as the new ground link. The transformation that scales, rotates, and translates the function generator such that $A'$ and $B'$ line up with $L$ and $K$ is $\mathcal{T}$ defined from Eqn. (2.14) as

$$\mathcal{T}(p) = \frac{K - L}{B' - A'}(p - A') + L \tag{7.13}$$

where the pivots $A$, $B$, $C$, $D$, $F$, $G$, $H$ of the new path generating linkage are

$$\begin{aligned} &A = \mathcal{T}(F'), \quad B = \mathcal{T}(B'), \quad C = \mathcal{T}(H'), \quad D = \mathcal{T}(C'), \\ &F = \mathcal{T}(A'), \quad G = \mathcal{T}(G'), \quad H = \mathcal{T}(D'). \end{aligned} \tag{7.14}$$

Trace point $P$ is then rigidly attached to Link $FH$ and the Stephenson II path generator is formed. Fig. 7.5 shows the angles $\nu$ and $\zeta$ are related to $\phi'$ and $\psi'$ of the function generator by

$$(\Delta\phi'_j, \Delta\psi'_j) = (\Delta\zeta_j, -\Delta\nu_j), \quad j = 1, \ldots, 10. \tag{7.15}$$

Eqn. (7.15) sets up the function generation synthesis problem and Eqn. (7.14) is used to transform the synthesis results of the Stephenson III function generator to Stephenson II path generator results.

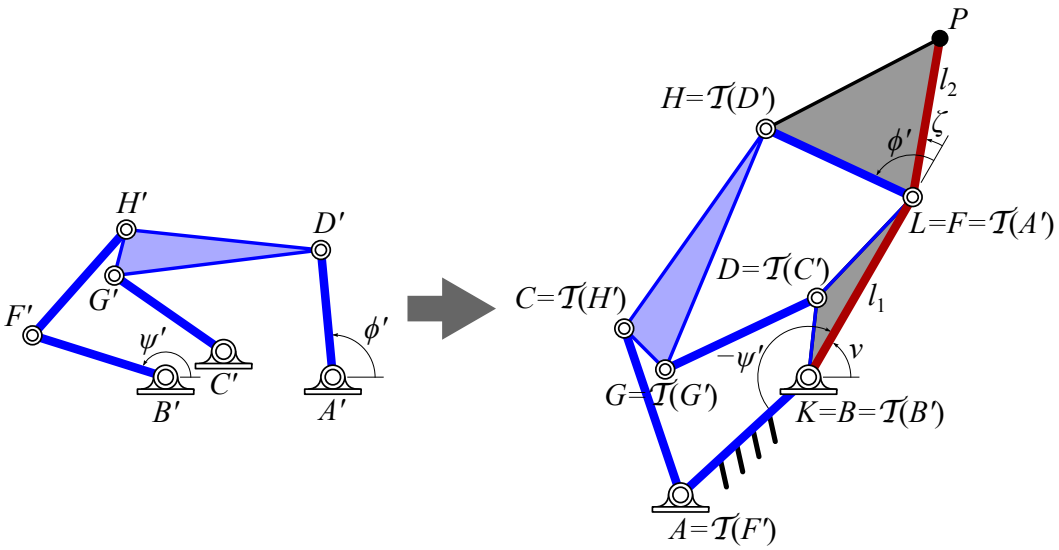



Figure 7.5: A Stephenson II path generator with trace point on a binary floating link obtained by coordinating the RR joints with a Stephenson III function generator.

### 7.2.3 Stephenson II Path Generator with Trace Point on Ternary Link

A Stephenson II path generator is formed by attaching a Stephenson II function generator to the RR chain $KLP$ such that $A'$ lies coincident to $K$ and $B'$ lies coincident to $L$ as shown in Fig. 7.6. In this case, ground Link $A'B'$ of the function generator is allowed to pivot about $A'$ and Link $A'C'$ is set as the new ground link. The transformation that scales, rotates, and translates the function generator such that $A'$ and $B'$ line up with $K$ and $L$ is $\mathcal{T}$ defined from Eqn. (2.14) as

$$\mathcal{T}(p) = \frac{L-K}{B'-A'}(p-A') + K \tag{7.16}$$

where the pivots $A$, $B$, $C$, $D$, $F$, $G$, $H$ of the new path generating linkage are

$$\begin{aligned} &A = \mathcal{T}(A'), \quad B = \mathcal{T}(C'), \quad C = \mathcal{T}(B'), \quad D = \mathcal{T}(G'), \\ &F = \mathcal{T}(H'), \quad G = \mathcal{T}(D'), \quad H = \mathcal{T}(F'). \end{aligned} \tag{7.17}$$

Trace point $P$ is then rigidly attached to Link $CGH$ and the Stephenson II path generator is formed. Fig. 7.6 shows the angles $\nu$ and $\zeta$ are related to $\phi'$ and $\psi'$ of the function generator by

$$(\Delta\phi'_j, \Delta\psi'_j) = (-\Delta\nu_j, \Delta\zeta_j), \quad j = 1, \ldots, 10. \tag{7.18}$$

Eqn. (7.18) sets up the function generation synthesis problem and Eqn. (7.17) is used to transform the synthesis results of the Stephenson II function generator to Stephenson II path generator results.

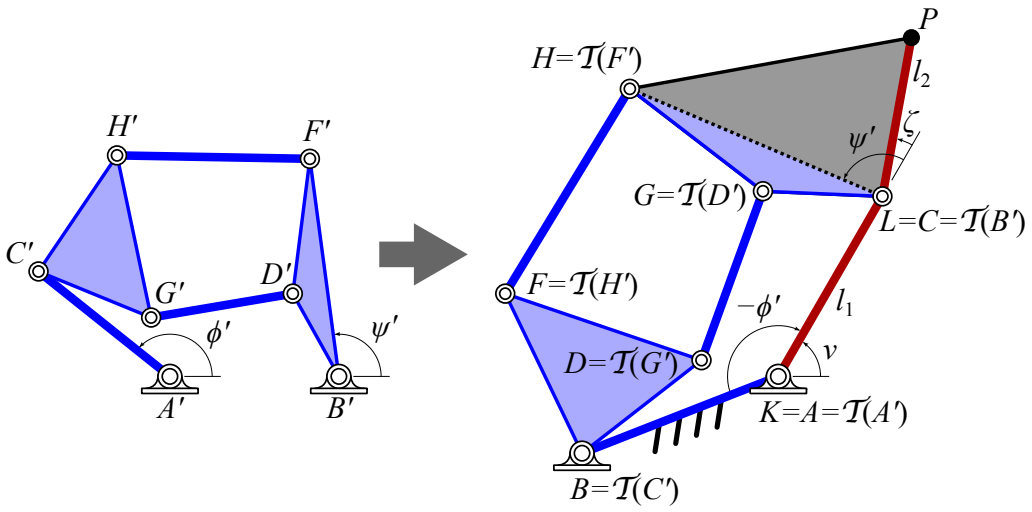



Figure 7.6: A Stephenson II path generator with trace point on the ternary floating link obtained by coordinating the RR joints with a Stephenson II function generator.

### 7.2.4 Stephenson III Path Generator

A Stephenson III path generator is formed by attaching a Stephenson II function generator to the RR chain $KLP$ such that $A'$ lies coincident to $L$ and $B'$ lies coincident to $K$ as shown in Fig. 7.7. In this case, ground Link $A'B'$ of the function generator is allowed to pivot about $B'$ and Link $B'D'F'$ is set as the new ground link. The transformation that scales, rotates, and translates the function generator such that $A'$ and $B'$ line up with $L$ and $K$ is

$\mathcal{T}$ defined from Eqn. (2.14) as

$$\mathcal{T}(p) = \frac{K - L}{B' - A'}(p - A') + L \tag{7.19}$$

where the pivots $A$, $B$, $C$, $D$, $F$, $G$, $H$ of the new path generating linkage are

$$\begin{aligned} &A = \mathcal{T}(F'), \quad B = \mathcal{T}(B'), \quad C = \mathcal{T}(D'), \quad D = \mathcal{T}(H'), \\ &F = \mathcal{T}(A'), \quad G = \mathcal{T}(G'), \quad H = \mathcal{T}(C'). \end{aligned} \tag{7.20}$$

Trace point $P$ is then rigidly attached to Link $FH$ and the Stephenson III path generator is formed. Fig. 7.7 shows the angles $\nu$ and $\zeta$ are related to $\phi'$ and $\psi'$ of the function generator by

$$(\Delta\phi'_j, \Delta\psi'_j) = (\Delta\zeta_j, -\Delta\nu_j), \quad j = 1, \ldots, 10. \tag{7.21}$$

Eqn. (7.21) sets up the function generation synthesis problem and Eqn. (7.20) is used to transform the synthesis results of the Stephenson II function generator to Stephenson III path generator results.

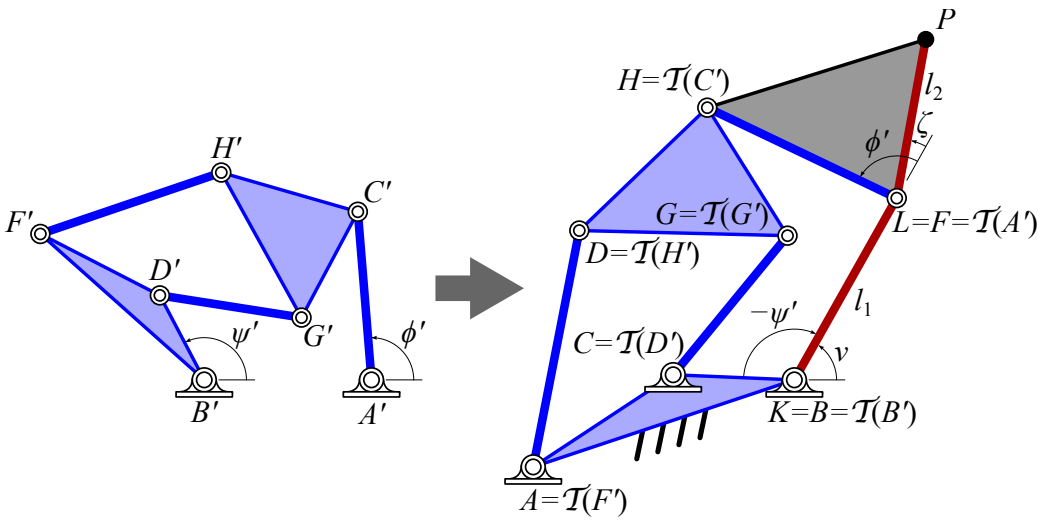



Figure 7.7: A Stephenson III path generator obtained by coordinating the RR joints with a Stephenson II function generator.

## 7.3 Synthesis of Function Generators

Section 7.2 describes four cases in which the problem of path generation is inverted to the problem of function generation. The four cases require solutions for Stephenson II and III function generators that coordinate the angles $(\Delta\phi'_j, \Delta\psi'_j)$, $j = 0, \ldots, 10$. The formulation and solution of these synthesis equations is described in Chapters 4 and 5. For each case, BERTINI was used to form parameter homotopies from the general solution sets found during this dissertation work to obtain solutions to the examples in this chapter in a couple of hours.

Stephenson II and III function generators are displayed in Fig. 7.3. The synthesis equations for both cases take the form,

$$\begin{aligned}&(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j))\\ &\quad\times(\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0. \qquad (7.22)\end{aligned}$$

For the Stephenson II case, joint locations and rotation operators substitute into Eqn. (7.22) in the following manner,

$$\begin{aligned} a &= G' - C', & b_j &= A' - B' + Q'_j(C' - A') - S'_j(D' - B'),\\ c &= H' - C', & d_j &= A' - B' + Q'_j(C' - A') - S'_j(F' - B'),\\ f &= G' - D', & g &= H' - F'. \end{aligned} \qquad (7.23)$$

And for the Stephenson III case, the following substitutions into Eqn. (7.22) are made,

$$\begin{aligned} a &= G' - D', & b_j &= A' - C' + Q'_j(D' - A'),\\ c &= H' - D', & d_j &= A' - B' + Q'_j(D' - A') - S'_j(F' - B'),\\ f &= G' - C', & g &= H' - F'. \end{aligned} \qquad (7.24)$$

## 7.4 Analysis

Once function generator solutions are obtained, they are transformed into path generators according to Section 7.2, forming a set of design candidates. Design candidates must be analyzed to evaluate their performance. The criteria for a successful design is that a path generator reaches all specified points $P_j$, $j = 0, \ldots, 10$, on a single trajectory of configurations without passing through a singular configuration. If all points cannot be reached on a single trajectory, then the design has a circuit defect. If the design must pass through a singularity to reach all points, then the design has a branch defect. Singularities are those points where the determinant of the Jacobian of the loop equations is equal to zero.

Each design candidate was analyzed according to the procedure described in Section 2.4. That is the forward kinematics equations for solved for incremented input values, then the resulting configurations were sorted onto mechanism trajectories.

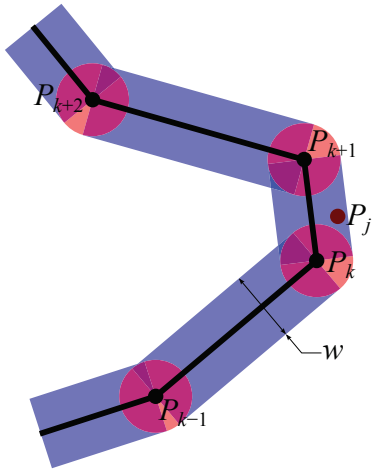



Figure 7.8: Performance verification requires the linkage candidate trace a trajectory that lies within a small region around the required task points.

Once trajectories have been assembled for a design candidate, each is checked to determine how many of the task points $P_j$, $j = 0, \ldots, 10$, it moves through. Trajectories are represented by discrete points generated from a numerical procedure which makes it somewhat difficult to determine whether a task point $P_j$ belongs to a trajectory $P_k$, $k = 1, \ldots, n$. Therefore,

each trajectory is "thickened" by constructing a series of circles and boxes at and between each trajectory point $P_k$ and $P_{k+1}$ as shown in Fig 7.8. If $P_j$ lies in either a circle or a box, it is said to belong to that trajectory.

The condition for $P_j$ to be contained in a circle of diameter $w$ centered at $P_k$ is

$$(P_j - P_k)(\bar{P}_j - \bar{P}_k) < \frac{w^2}{4}. \tag{7.25}$$

The conditions for $P_j$ to be contained in a box with midline segment defined by $P_k$ and $P_{k+1}$ and width $w$ are

$$\begin{aligned} 0 < (P_j - P_k)(\bar{P}_{k+1} - \bar{P}_k) + (\bar{P}_j - \bar{P}_k)(P_{k+1} - P_k) \\ < 2(P_{k+1} - P_k)(\bar{P}_{k+1} - \bar{P}_k), \\ -w\sqrt{(P_{k+1} - P_k)(\bar{P}_{k+1} - \bar{P}_k)} \\ < i\Big((P_{k+1} - P_k)(\bar{P}_j - \bar{P}_k) - (\bar{P}_{k+1} - \bar{P}_k)(P_j - P_k)\Big) \\ < w\sqrt{(P_{k+1} - P_k)(\bar{P}_{k+1} - \bar{P}_k)}. \end{aligned} \tag{7.26}$$

Ideally, a design candidate will possess a trajectory that moves through all 11 points, however, we have noticed that design candidates that move through less than 11 points may still have practical value. It is often the case that these mechanisms only slightly miss a task point. Therefore, we keep track of mechanisms that move through 6–11 points.

## 7.5 Example: Leg Mechanism

The synthesis method described in this chapter is illustrated by the design of a leg mechanism for a walking machine that provides a cyclic motion for the foot. The dimensions of the RR chain to be guided were specified as $K = 0 + 0i$, $l_1 = 2$, and $l_2 = 2.5$. The task points to

Table 7.1: Required task points $P_j$ and associated RR joint angles, $(\nu_j, \zeta_j)$.

| $j$ | $P_j$ | $\nu_j$ (deg) | $\zeta_j$ (deg) | Solution Family |
|---|---|---|---|---|
| 0 | $-1.75-4.00i$ | $-97.93$ | $-28.21$ | $-$ |
| 1 | $-1.20-4.00i$ | $-82.11$ | $-44.03$ | $-$ |
| 2 | $-0.60-4.00i$ | $-69.24$ | $-52.34$ | $-$ |
| 3 | $0-4.00i$ | $-59.25$ | $-54.90$ | $-$ |
| 4 | $0.60-4.00i$ | $-52.17$ | $-52.34$ | $-$ |
| 5 | $1.20-4.00i$ | $-48.71$ | $-44.03$ | $-$ |
| 6 | $1.75-4.00i$ | $-50.67$ | $-28.21$ | $-$ |
| 7 | $1.60-3.60i$ | $-98.67$ | 58.20 | $+$ |
| 8 | $0.60-3.40i$ | $-125.55$ | 80.39 | $+$ |
| 9 | $-0.60-3.40i$ | $-145.56$ | 80.39 | $+$ |
| 10 | $-1.60-3.60i$ | $-146.60$ | 58.20 | $+$ |

Table 7.2: The number of design candidates and successful designs for the leg mechanism of a walking machine: Stephenson I, Stephenson II (binary floating link), Stephenson II (ternary floating link), and Stephenson III.

| | SI | SII (binary) | SII (ternary) | SIII |
|---|---|---|---|---|
| Linkage solutions | 27,000 | 31,469 | 14,358 | 15,545 |
| Design candidates | 14,973 | 17,634 | 5,486 | 4,066 |
| 11-point mechanisms | 36 | 98 | 37 | 4 |
| 10-point mechanisms | 74 | 276 | 54 | 11 |
| 9-point mechanisms | 141 | 469 | 84 | 65 |
| 8-point mechanisms | 473 | 692 | 234 | 176 |
| 7-point mechanisms | 1,328 | 2,996 | 660 | 1,053 |
| 6-point mechanisms | 3,058 | 2,473 | 1,004 | 1,023 |

trace through and RR joint angles are displayed in Table 7.1. The synthesis algorithm was run for all four Stephenson path generators described in Section 7.2. Statistics on the results for each appear in Table 7.2. The combined number of 11-point mechanisms was 175, and including 10- and 9-point mechanisms provides 1349 linkage design alternatives.

Example leg mechanisms for each case of the Stephenson path generators are shown in Fig. 7.9; corresponding $(\nu, \zeta)$ functions for each example design appear in Fig. 7.9(e); and solid models of each example design appear in Fig. 7.10. A solid model of a six-legged alternating tripod gait robot that uses the design shown in Fig. 7.9(b) appears in Fig. 7.11.

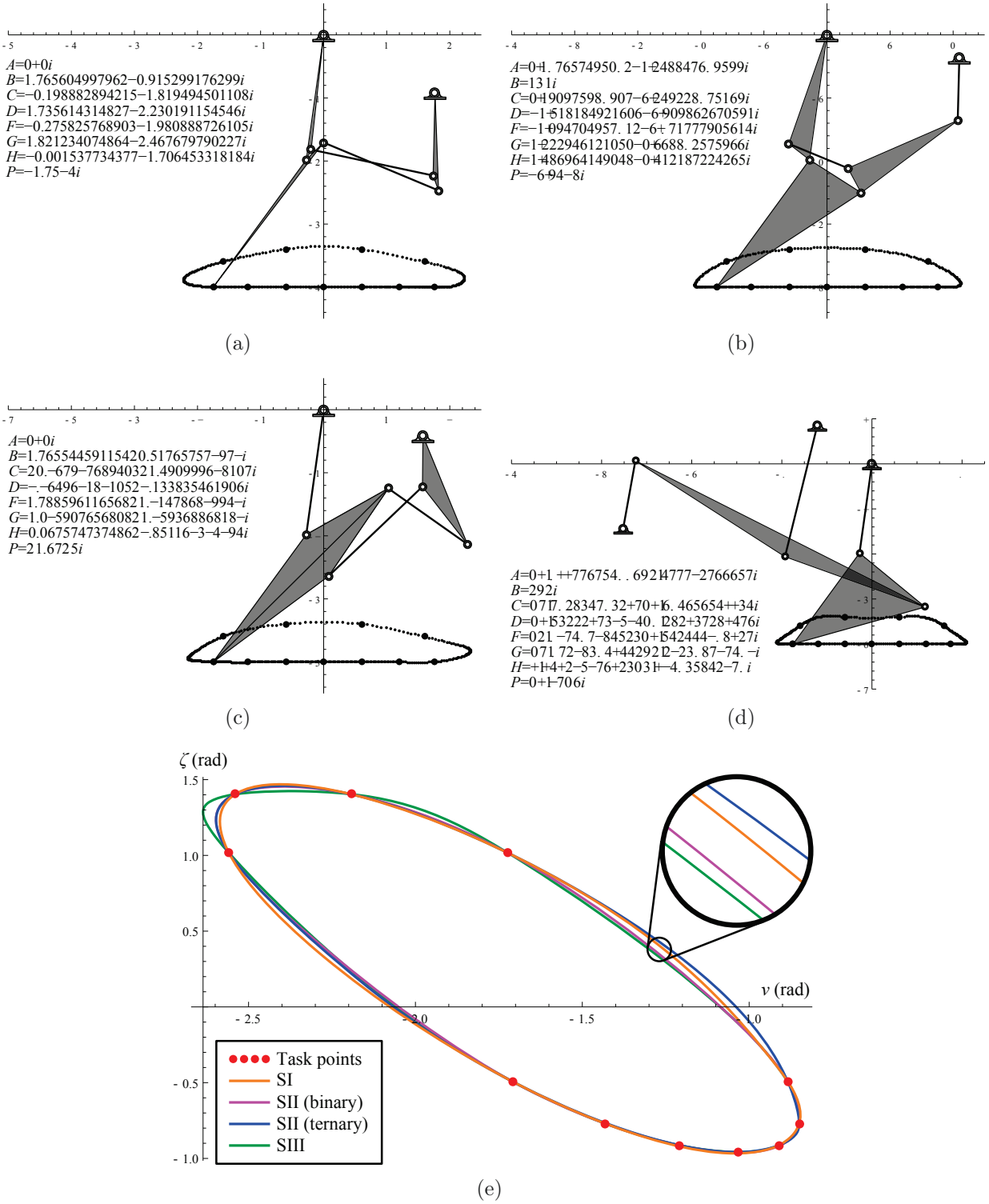



Figure 7.9: Successful leg mechanisms for each of the path generators: (a) the Stephenson I, (b) the Stephenson II (binary link trace point), (c) the Stephenson II (ternary link trace point), and (d) the Stephenson III. Joint angles trajectories $(\nu, \zeta)$ for the RR chain are shown in (e) for each mechanism.

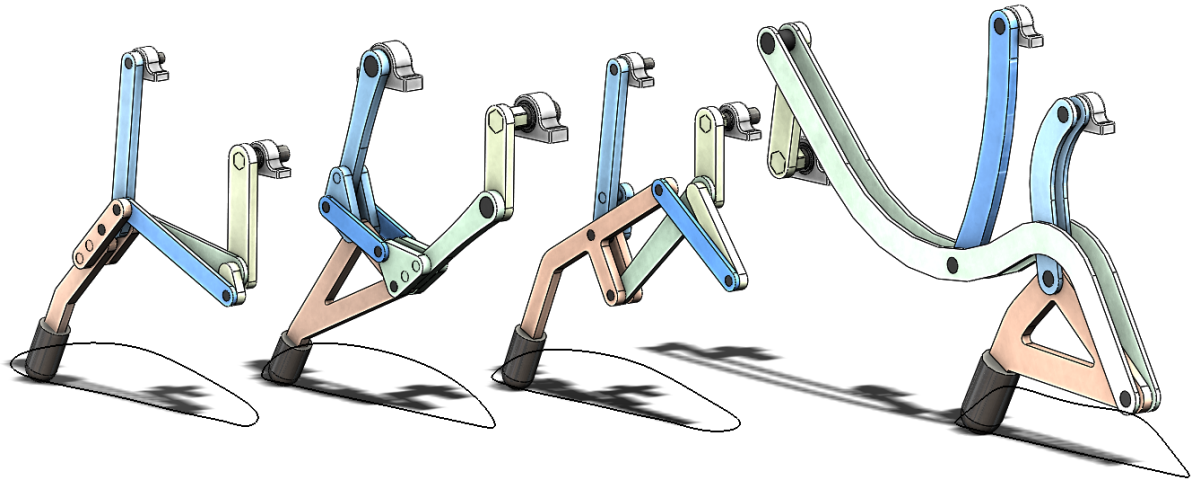


Figure 7.10: Solid models showing an implementation of the leg mechanisms. From left to right: the Stephenson I, the Stephenson II (binary link trace point), a Stephenson II (ternary link trace point), a Stephenson III.

## 7.6 Summary

This chapter presented a synthesis method for six-bar linkages to trace a required trajectory. It uses the synthesis of a function generator to control 11 coordinated angles of an RR chain that position a trace point on the trajectory. The ground link of the function generator becomes the first link of the RR chain and coordinates its two joints. The result is the ability to design linkages that have increased accuracy in tracing curves in the plane. The technique is demonstrated with the design of a leg mechanism for a walking robot. The example demonstrates the ability of this technique to generate a large number of design alternatives.

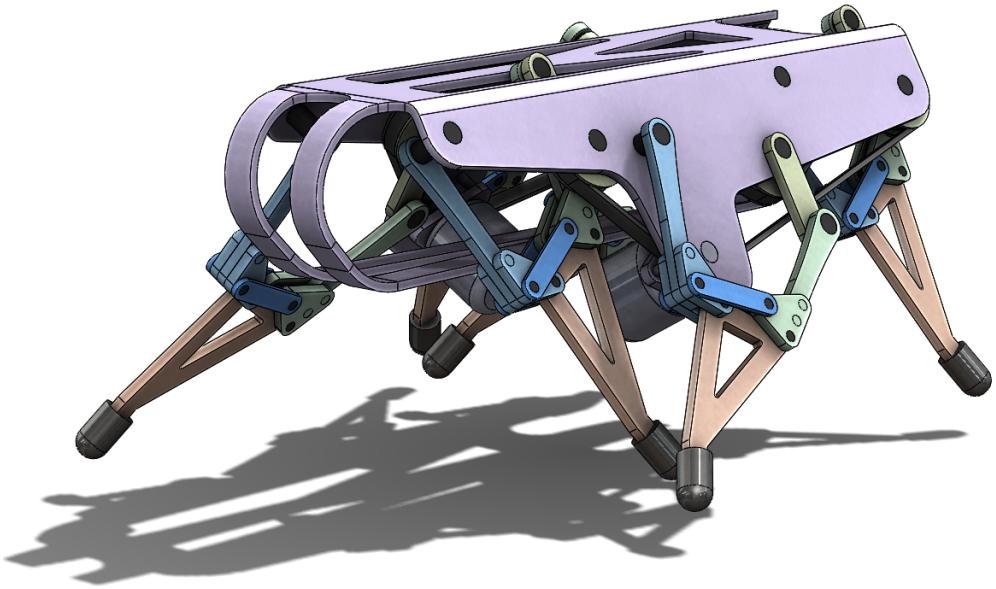


Figure 7.11: A solid model of a two degree-of-freedom walker, one motor on each side, that uses the Stephenson II (binary link trace point) leg mechanism shown in Fig. 7.10.

# Chapter 8

# Conclusion

In this dissertation we presented the kinematic synthesis of six-bar linkages for function, motion, and path generation. In each case, the synthesis equations were formulated as a system of polynomials and solved using homotopy continuation. For function generation, the Watt II, Stephenson II, and Stephenson III linkages were presented. For motion generation, the Watt I linkage was presented. These six-bars were selected because they provide design capabilities beyond the four-bar linkage. For path generation, the Watt I and Stephenson I–III linkages all provide design capabilities beyond the four-bar linkage but we focused on just the Stephenson I–III. In particular, the path generation of Stephenson linkages is formulated as the control of a 2R chain via inverted Stephenson function generators.

The synthesis equations for the problems explored in this dissertation have Bézout numbers in the millions and beyond. The challenge of finding complete solution sets was approached using the homotopy continuation algorithms made available through the software package Bertini in combination with modern day performance computing technology, in particular the UC Irvine High Performance Computing Cluster and the Gordon cluster at the San Diego Supercomputer Center.

Finding the solution set to a numerically general set of synthesis equations is a numerical reduction of that system. This dissertation presents particular success with respect to the numerical reduction of the synthesis equations for the Stephenson II and III function generators, reporting estimated root counts of 1,521,037 and 834,441 nonsingular solutions. These two solutions sets can be used to construct parameter homotopies to solve the Stephenson II and III function generator synthesis equations in one or two hours on a 2.2×64GHz machine. The numerical reduction of the Watt I motion generator problem for six task positions resulted in 5,735 roots which computes with parameter homotopy in about five minutes.

As well, this dissertation illustrates a particular advantage of working with large solution sets. That is, since each solution represents a linkage design candidate, and only a small percentage of design candidates will be shown to be useful after performance verification, then a large number of solutions must be found in order to generate a pool of design choices. This advantage was the case in all the examples presented in this dissertation. To make this paragraph clear, we revisit one of the example functions presented for the Stephenson III function generator. That is 834,441 homotopy paths were tracked of which 0.01% lead to roots which represented 11-point mechanisms. This percentage sounds low but the final result is 96 good linkage designs. A linkage designer can then manually assess each design to find which will package most conveniently for his/her application.

## 8.1 Future Work

The numerically reduced root counts listed above are presented as estimates. That is because the homotopy solution process used to obtain these root counts is subject to numerical errors. However, a more meticulous handling of this solution process would produce better estimates and can even provide a convincing argument for the exact number of roots.

Another opportunity for research is finding the complete solution for the eight position Watt I motion generator. It is the conjecture of the author that combining regeneration homotopy with today's high performance computing power will result in a good estimate of this solution set for use with parameter homotopies.

Furthermore, the synthesis of 15 position path generating six-bars remains as an outstanding problem. It is the conjecture of the author that it in order to make progress on this front a new clever reduction or solution technique must be devised.

Another route for future research is to formulate these synthesis problems to have one dimensional solution sets as opposed to isolated solution points. This would be analogous to the cubic circle- and center-point curves formulated for the synthesis of four-bar linkages [4]. The BERTINI software package contains an entire set of tools for constructing positive dimensional solution sets.

Finally, we note that a useful pairing exists between homotopy and optimization techniques to be used in a two stage coarse-fine solution process. Homotopy algorithms provide the advantage that they find all, or at least most, of the solutions for a particular design problem. But they solve a formulation which requires that a linkage pass exactly through a discrete number of control points when approximate synthesis is often good enough. Optimization techniques accommodate approximate synthesis well but need a well picked start point and will only yield one solution for that start point. The two techniques make up for each other's weaknesses. A coarse solution stage can take place using the formulation and homotopy techniques presented in this chapter. Then a fine solution stage can take place where optimization is applied to useful design candidates to accommodate approximate synthesis requirements.

# Bibliography


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

[2] H. H. Mabie and C. F. Reinholtz. *Mechanisms and Dynamics of Machinery.* Wiley, New York, Fourth edition, 1987.

[3] K. J. Waldron and G. L. Kinzel. *Kinematics, Dynamics, and Design of Machinery.* Wiley, Second edition, 2004.

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

[5] J. J. Uicker Jr., G. R. Pennock, and J. E. Shigley. *Theory of Machines and Mechanisms.* Oxford University Press, New York, Fourth edition, 2011.

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

[7] A. G. Erdman, G. N. Sandor, and S. Kota. *Mechanism Design: Analysis and Synthesis.* Prentice Hall, Upper Saddle River, NJ, Fourth edition, 2001.

[8] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5.

[9] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. *Numerically Solving Polynomial Systems with Bertini.* SIAM Press, Philadelphia, 2013.

[10] A. Svoboda. Mechanism for use in computing apparatus. U. S. Patent No. 2,340,350, Feb. 1, 1944.

[11] A. Svoboda. *Computing Mechanisms and Linkages.* McGraw-Hill, New York, 1948.

[12] F. Freudenstein. An analytical approach to the design of four-link mechanisms. *Transactions of the ASME*, 76:483–492, 1954.

[13] F. Freudenstein and G. N. Sandor. Synthesis of path generating mechanisms by means of a programmed digital computer. *Journal of Engineering for Industry*, 81(2):159–168, 1959.

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

[15] C. W. McLarnan. Synthesis of six-link plane mechanisms by numerical analysis. *Journal of Engineering for Industry*, 85(1):5–10, 1963.

[16] K. Ogawa. Researches on the synthesis of the six-bar linkage: Part 1, as regards the type A3. *Bulletin of JSME*, 8(30):264–273, 1965.

[17] A. V. Mohan Rao, A. G. Erdman, G. N. Sandor, V. Raghunathan, D. E. Nigbora, L. E. Brown, E. F. Mahardy, and E. D. Enderle. Synthesis of multi-loop, dual-purpose planar mechanisms utilizing Burmester theory. In *Proceedings of the 2nd OSU Applied Mechanism Conference*, Paper No. 7, Stillwater, Oklahoma, October 1971.

[18] A. K. Dhingra, J. C. Cheng, and D. Kohli. Synthesis of six-link, slider-crank and four-link mechanisms for function, path and motion generation using homotopy with m-homogenization. *Journal of Mechanical Design*, 116(4):1122–1131, 1994.

[19] L. Li and C. Yong. Synthesis of Stephenson III six-link mechanism for function generation using an improved homotopy method. *Journal of Southwest Jiaotong University*, 10(2):161–166, 2002.

[20] 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.

[21] Z. Luo and J. S. Dai. Patterned bootstrap: A new method that gives efficiency for some precision position synthesis problems. *Journal of Mechanical Design*, 129(2):173–183, 2007.

[22] Y. Luo, Q. Liu, and X. Che. Synthesis of planar Stephenson III six-link mechanism for function generation based on hyper-chaos Newton downhill method. *Key Engineering Materials*, 467:421–426, 2011.

[23] M. Plecnik and J. M. McCarthy. Numerical synthesis of six-bar linkages for mechanical computation. *Journal of Mechanisms and Robotics*, 6(3):031012, 2014.

[24] 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.

[25] W. M. Hwang and Y. J. Chen. Defect-free synthesis of Stephenson II function generators. *Journal of Mechanisms and Robotics*, 2(4):041012, 2010.

[26] R. Sancibrian. Improved GRG method for the optimal synthesis of linkages in function generation problems. *Mechanism and Machine Theory*, 46(10):1350–1375, 2011.

[27] R. R. Bulatović, S. R. Đorđević, and V. S. Đorđević. Cuckoo search algorithm: A metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. *Mechanism and Machine Theory*, 61:1–13, 2013.

[28] R. E. Kaufman. Synthesis of the Watt-1 six-bar linkage for five position higher coupler motion correlated with input crank rotations. In *Proceedings of the 2nd OSU Applied Mechanism Conference*, Paper No. 28, Stillwater, Oklahoma, October 1971.

[29] T. R. Chase, A. G. Erdman, and D. R. Riley. Triad synthesis for up to five design positions with application to the design of arbitrary planar mechanisms. *Journal of Mechanisms, Transmissions and Automation in Design*, 109(4):426–434, 1987.

[30] C. S. Lin and A. G. Erdman. Dimensional synthesis of planar triads: motion generation with prescribed timing for six precision positions. *Mechanism and Machine Theory*, 22(5):411–419, 1987.

[31] 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, 1996.

[32] H. Schreiber, K. Meer, and B. J. Schmitt. Dimensional synthesis of planar Stephenson mechanisms for motion generation using circlepoint search and homotopy methods. *Mechanism and Machine Theory*, 37(7):717–737, 2002.

[33] 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, 2008.

[34] M. Plecnik and J. M. McCarthy. Synthesis of a Stephenson II six-bar function generator for eight precision positions. In *Proceedings of the ASME 2013 IDETC/CIE Conference*, DETC2013-12763, Portland, Oregon, August 2013.

[35] P. Zhao, A. Purwar, and Q. J. Ge. Task driven unified synthesis of planar four-bar and six-bar linkages with revolute and prismatic joints for five position synthesis. In *Proceedings of the ASME 2013 IDETC/CIE Conference*, DETC2013-13168, Portland, Oregon, August 2013.

[36] M. Plecnik and J. M. McCarthy. Vehicle suspension design based on a six-bar linkage. In *Proceedings of the ASME 2014 IDETC/CIE Conference*, DETC2014-35374, Buffalo, New York, August 2014.

[37] F. Freudenstein and G. N. Sandor. On the burmester points of a plane. *Journal of Applied Mechanics*, 28(1):41–49, 1961.

[38] B. Roth and F. Freudenstein. Synthesis of path-generating mechanisms by numerical methods. *Journal of Engineering for Industry*, 85(3):298–304, 1963.

[39] C. W. Wampler, A. J. Sommese, and A. P. Morgan. Complete solution of the nine-point path synthesis problem for four-bar linkages. *Journal of Mechanical Design*, 114(1):153–159, 1992.

[40] K. Hain. *Applied Kinematics*. McGraw-Hill, New York, 1967.

[41] H. S. Kim, S. Hamid, and A. H. Soni. Synthesis of six-link mechanisms for point path generation. *Journal of Mechanisms*, 6(4):447–461, 1972.

[42] K. N. Prasad and C. Bagci. Minimum error synthesis of multiloop plane mechanisms for rigid body guidance. *Journal of Engineering for Industry*, 96(1):107–116, 1974.

[43] D. H. Bhatia and C. Bagci. Optimum synthesis of multiloop planar mechanisms for the generation of paths and rigid-body positions by the linear partition of design equations. *Journal of Engineering for Industry*, 99(1):116–123, 1977.

[44] Y. Liu and J. McPhee. Automated kinematic synthesis of planar mechanisms with revolute joints. *Mechanics Based Design of Structures and Machines*, 35(4):405–445, 2007.

[45] J. A. Cabrera, A Ortiz, F. Nadal, and J. J. Castillo. An evolutionary algorithm for path synthesis of mechanisms. *Mechanism and Machine Theory*, 46(2):127–141, 2011.

[46] F. Peñuñuri, R. Peón-Escalante, C. Villanueva, and D. Pech-Oy. Synthesis of mechanisms for single and hybrid tasks using differential evolution. *Mechanism and Machine Theory*, 46(10):1335–1349, 2011.

[47] R. R. Bulatović and S. R. Đorđević. Optimal synthesis of a path generator six-bar linkage. *Journal of Mechanical Science and Technology*, 26(12):4027–4040, 2012.

[48] M. Plecnik and J. M. McCarthy. Computational design of Stephenson II six-bar function generators for 11 accuracy points. *Journal of Mechanisms and Robotics*, Accepted for publication, 2015.

[49] M. Plecnik and J. M. McCarthy. Controlling the movement of a TRR spatial chain with coupled six-bar function generators for biomimetic motion. In *Proceedings of the ASME 2015 IDETC/CIE Conference*, DETC2015-47876, Boston, Massachusetts, August 2015.

[50] M. Plecnik and J. M. McCarthy. Kinematic synthesis of Stephenson III six-bar function generators. *Mechanism and Machine Theory*, Under review, 2015.

[51] M. Plecnik, J. M. McCarthy, and C. W. Wampler. Kinematic synthesis of a Watt I six-bar linkage for body guidance. In *Advances in Robot Kinematics*, pages 317–325. Springer International Publishing, 2014.

[52] M. Plecnik and J. M. McCarthy. Design of Stephenson linkages that guide a point along a specified trajectory. *Mechanism and Machine Theory*, Under review, 2015.

[53] C. W. Wampler. Isotropic coordinates, circularity, and Bézout numbers: planar kinematics from a new perspective. In *Proceedings of the ASME 1996 DETC/CIE Conference*, 96-DETC/MECH-1210, Irvine, CA, August 1996.

[54] E. J. Primrose, F. Freudenstein, and B. Roth. Six-bar motion I. the Watt mechanism. *Archive for Rational Mechanics and Analysis*, 24(1):22–41, 1967.

[55] E. J. Primrose, F. Freudenstein, and B. Roth. Six-bar motion II. the Stephenson-1 and Stephenson-2 mechanisms. *Archive for Rational Mechanics and Analysis*, 24(1):42–72, 1967.

[56] S. Roberts. Three-bar motion in plane space. *Proceedings of the London Mathematical Society*, s1-7(1):14–23.

[57] P. L. Chebyshev. Sur les parallélogrammes composés de trois éléments quelconques. *Mémoires de l'Académie des Sciences de Saint-Pétersbourg*, 36:suppl. 3, 1879.

[58] M. M. Mirbagheri and K. Settle. Neuromuscular properties of different spastic human joints vary systematically. In *32nd Annual International Conference of the IEEE EMBS*, pp. 468–486, Buenos Aires, Argentina, August 2010.

# Appendices

## A Factoring Function Generator Synthesis Equations

The synthesis equations of the Watt II, Stephenson II and III function generators can be written in the form

$$\begin{bmatrix} a\bar{b}_j & \bar{a}b_j \\ c\bar{d}_j & \bar{c}d_j \end{bmatrix} \begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - c\bar{c} - d_j\bar{d}_j \end{Bmatrix} \tag{A.1}$$

where

$$\begin{aligned} b_j &= h + Q_j k + S_j l, \\ d_j &= m + Q_j n + S_j o. \end{aligned} \tag{A.2}$$

There are 10 linkage parameters $a$, $c$, $f$, $g$, $h$, $k$, $l$, $m$, $n$, and $o$ that are defined for each linkage. Note these parameters are not independent. This section describes how to put (A.1) and (A.2) into the form

$$\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0, \tag{A.3}$$

where $\mathbf{p}$ is a vector constructed from the linkage design parameters and $[\hat{Q}_j]$ is a Hermitian matrix constructed from the specified task. Eqn. (A.3) provides a convenient form for ad hoc simplification specific to each type.

First, Eqn. (A.1) is solved for the unknown rotation parameters

$$\begin{Bmatrix} R_j \\ \bar{R}_j \end{Bmatrix} = \frac{1}{a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j} \begin{bmatrix} \bar{c}d_j & -\bar{a}b_j \\ -c\bar{d}_j & a\bar{b}_j \end{bmatrix} \begin{Bmatrix} f\bar{f} - a\bar{a} - b_j\bar{b}_j \\ g\bar{g} - c\bar{c} - d_j\bar{d}_j \end{Bmatrix}, \tag{A.4}$$

which are substituted into their corresponding normalization condition,

$$R_j\bar{R}_j = 1 \tag{A.5}$$

to obtain

$$\begin{aligned} &(a\bar{b}_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - c\bar{d}_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) \\ &\quad \times (\bar{a}b_j(g\bar{g} - c\bar{c} - d_j\bar{d}_j) - \bar{c}d_j(f\bar{f} - a\bar{a} - b_j\bar{b}_j)) + (a\bar{b}_j\bar{c}d_j - \bar{a}b_jc\bar{d}_j)^2 = 0. \end{aligned} \tag{A.6}$$

The terms of (A.6) can be factored into vector inner products

$$\begin{aligned} &\left(\begin{Bmatrix} a(g\bar{g} - c\bar{c}) \\ c(f\bar{f} - a\bar{a}) \\ a \\ c \end{Bmatrix}^T \begin{Bmatrix} \bar{b}_j \\ -\bar{d}_j \\ -\bar{b}_jd_j\bar{d}_j \\ b_j\bar{b}_j\bar{d}_j \end{Bmatrix}\right)\left(\begin{Bmatrix} b_j \\ -d_j \\ -b_jd_j\bar{d}_j \\ b_j\bar{b}_jd_j \end{Bmatrix}^T \begin{Bmatrix} \bar{a}(g\bar{g} - c\bar{c}) \\ \bar{c}(f\bar{f} - a\bar{a}) \\ \bar{a} \\ \bar{c} \end{Bmatrix}\right) \\ &\qquad - \left(\begin{Bmatrix} a\bar{c} \\ \bar{a}c \end{Bmatrix}^T \begin{Bmatrix} \bar{b}_jd_j \\ -b_j\bar{d}_j \end{Bmatrix}\right)\left(\begin{Bmatrix} b_j\bar{d}_j \\ -\bar{b}_jd_j \end{Bmatrix}^T \begin{Bmatrix} \bar{a}c \\ a\bar{c} \end{Bmatrix}\right) = 0 \end{aligned} \tag{A.7}$$

taking the form

$$\mathbf{a}^T\bar{\mathbf{b}}\mathbf{b}^T\bar{\mathbf{a}} - \mathbf{c}^T\bar{\mathbf{d}}\mathbf{d}^T\bar{\mathbf{c}} = 0 \tag{A.8}$$

where

$$\mathbf{a} = \begin{Bmatrix} a(g\bar{g} - c\bar{c}) \\ c(f\bar{f} - a\bar{a}) \\ a \\ c \end{Bmatrix}, \quad \mathbf{b}_j = \begin{Bmatrix} b_j \\ -d_j \\ -b_j d_j \bar{d}_j \\ b_j \bar{b}_j d_j \end{Bmatrix}, \quad \mathbf{c} = \begin{Bmatrix} a\bar{c} \\ \bar{a}c \end{Bmatrix}, \quad \mathbf{d}_j = \begin{Bmatrix} b_j\bar{d}_j \\ -\bar{b}_j d_j \end{Bmatrix}. \tag{A.9}$$

Eqns. (A.2) are substituted into $\mathbf{b}_j$ and $\mathbf{d}_j$,

$$\begin{aligned} \mathbf{b}_j &= \begin{Bmatrix} h + Q_j k + S_j l \\ -(m + Q_j n + S_j o) \\ -(h + Q_j k + S_j l)(m + Q_j n + S_j o)(\bar{m} + \bar{Q}_j \bar{n} + \bar{S}_j \bar{o}) \\ (h + Q_j k + S_j l)(\bar{h} + \bar{Q}_j \bar{k} + \bar{S}_j \bar{l})(m + Q_j n + S_j o) \end{Bmatrix} \\ \mathbf{d}_j &= \begin{Bmatrix} (h + Q_j k + S_j l)(\bar{m} + \bar{Q}_j \bar{n} + \bar{S}_j \bar{o}) \\ -(\bar{h} + \bar{Q}_j \bar{k} + \bar{S}_j \bar{l})(m + Q_j n + S_j o) \end{Bmatrix} \end{aligned} \tag{A.10}$$

Eqns. (A.10) are expanded and the task specification parameters $Q_j$, $\bar{Q}_j$, $S_j$, and $\bar{S}_j$ are factored out to create the equations

$$\begin{aligned} \mathbf{b}_j &= [B]\mathbf{q}_j, \\ \mathbf{d}_j &= [D]\mathbf{q}'_j, \end{aligned} \tag{A.11}$$

where

$$[B]^T = \begin{bmatrix} h & -m & -m(k\bar{n}+l\bar{o})-h(m\bar{m}+n\bar{n}+o\bar{o}) & h(\bar{k}n+\bar{l}o)+m(h\bar{h}+k\bar{k}+l\bar{l}) \\ k & -n & -n(h\bar{m}+l\bar{o})-k(m\bar{m}+n\bar{n}+o\bar{o}) & k(\bar{h}m+\bar{l}o)+n(h\bar{h}+k\bar{k}+l\bar{l}) \\ l & -o & -o(h\bar{m}+k\bar{n})-l(m\bar{m}+n\bar{n}+o\bar{o}) & l(\bar{h}m+\bar{k}n)+o(h\bar{h}+k\bar{k}+l\bar{l}) \\ 0 & 0 & -hm\bar{n} & hm\bar{k} \\ 0 & 0 & -hm\bar{o} & hm\bar{l} \\ 0 & 0 & -\bar{o}(hn+km) & \bar{l}(hn+km) \\ 0 & 0 & -\bar{n}(ho+lm) & \bar{k}(ho+lm) \\ 0 & 0 & -\bar{m}(ko+ln) & \bar{h}(ko+ln) \\ 0 & 0 & -kn\bar{m} & kn\bar{h} \\ 0 & 0 & -lo\bar{m} & lo\bar{h} \\ 0 & 0 & -kn\bar{o} & kn\bar{l} \\ 0 & 0 & -lo\bar{n} & lo\bar{k} \end{bmatrix} \tag{A.12}$$

and

$$[D]^T = \begin{bmatrix} h\bar{m}+k\bar{n}+l\bar{o} & -(\bar{h}m+\bar{k}n+\bar{l}o) \\ k\bar{m} & \bar{h}n \\ l\bar{m} & \bar{h}o \\ h\bar{n} & \bar{k}m \\ h\bar{o} & \bar{l}m \\ k\bar{o} & \bar{l}n \\ l\bar{n} & \bar{k}o \end{bmatrix}, \qquad \mathbf{q}_j = \begin{Bmatrix} 1 \\ Q_j \\ S_j \\ \bar{Q}_j \\ \bar{S}_j \\ Q_j\bar{S}_j \\ \bar{Q}_j S_j \\ Q_j S_j \\ Q_j^2 \\ S_j^2 \\ Q_j^2\bar{S}_j \\ \bar{Q}_j S_j^2 \end{Bmatrix}, \qquad \mathbf{q}'_j = \begin{Bmatrix} 1 \\ Q_j \\ S_j \\ \bar{Q}_j \\ \bar{S}_j \\ Q_j\bar{S}_j \\ \bar{Q}_j S_j \end{Bmatrix}. \tag{A.13}$$

Eqn. (A.11) is substituted into (A.8) to obtain

$$\mathbf{a}^T[\bar{B}]\bar{\mathbf{q}}_j{\mathbf{q}_j}^T[B]^T\bar{\mathbf{a}} - \mathbf{c}^T[\bar{D}]\bar{\mathbf{q}}'_j{\mathbf{q}'_j}^T[D]^T\bar{\mathbf{c}} = 0. \tag{A.14}$$

Next, the outer product $\bar{\mathbf{q}}_j {\mathbf{q}_j}^T$ is evaluated and simplified according to $Q_j \bar{Q}_j = S_j \bar{S}_j = 1$,

$$[Q_j] = \bar{\mathbf{q}}_j {\mathbf{q}_j}^T =$$

$$\begin{bmatrix}
1 & Q & S & \bar{Q} & \bar{S} & Q\bar{S} & \bar{Q}S & QS & Q^2 & S^2 & Q^2\bar{S} & \bar{Q}S^2 \\
\bar{Q} & 1 & \bar{Q}S & \bar{Q}^2 & \bar{Q}\bar{S} & \bar{S} & \bar{Q}^2S & S & Q & \bar{Q}S^2 & Q\bar{S} & \bar{Q}^2S^2 \\
\bar{S} & Q\bar{S} & 1 & \bar{Q}\bar{S} & \bar{S}^2 & Q\bar{S}^2 & \bar{Q} & Q & Q^2\bar{S} & S & Q^2\bar{S}^2 & \bar{Q}S \\
Q & Q^2 & QS & 1 & Q\bar{S} & Q^2\bar{S} & S & Q^2S & Q^3 & QS^2 & Q^3\bar{S} & S^2 \\
S & QS & S^2 & \bar{Q}S & 1 & Q & \bar{Q}S^2 & QS^2 & Q^2S & S^3 & Q^2 & \bar{Q}S^3 \\
\bar{Q}S & S & \bar{Q}S^2 & \bar{Q}^2S & \bar{Q} & 1 & \bar{Q}^2S^2 & S^2 & QS & \bar{Q}S^3 & Q & \bar{Q}^2S^3 \\
Q\bar{S} & Q^2\bar{S} & Q & \bar{S} & Q\bar{S}^2 & Q^2\bar{S}^2 & 1 & Q^2 & Q^3\bar{S} & QS & Q^3\bar{S}^2 & S \\
\bar{Q}\bar{S} & \bar{S} & \bar{Q} & \bar{Q}^2\bar{S} & \bar{Q}\bar{S}^2 & \bar{S}^2 & \bar{Q}^2 & 1 & Q\bar{S} & \bar{Q}S & Q\bar{S}^2 & \bar{Q}^2S \\
\bar{Q}^2 & \bar{Q} & \bar{Q}^2S & \bar{Q}^3 & \bar{Q}^2\bar{S} & \bar{Q}\bar{S} & \bar{Q}^3S & \bar{Q}S & 1 & \bar{Q}^2S^2 & \bar{S} & \bar{Q}^3S^2 \\
\bar{S}^2 & Q\bar{S}^2 & \bar{S} & \bar{Q}\bar{S}^2 & \bar{S}^3 & Q\bar{S}^3 & \bar{Q}\bar{S} & Q\bar{S} & Q^2\bar{S}^2 & 1 & Q^2\bar{S}^3 & \bar{Q} \\
\bar{Q}^2S & \bar{Q}S & \bar{Q}^2S^2 & \bar{Q}^3S & \bar{Q}^2 & \bar{Q} & \bar{Q}^3S^2 & \bar{Q}S^2 & S & \bar{Q}^2S^3 & 1 & \bar{Q}^3S^3 \\
Q\bar{S}^2 & Q^2\bar{S}^2 & Q\bar{S} & \bar{S}^2 & Q\bar{S}^3 & Q^2\bar{S}^3 & \bar{S} & Q^2\bar{S} & Q^3\bar{S}^2 & Q & Q^3\bar{S}^3 & 1
\end{bmatrix} \tag{A.15}$$

Note that the task positions index subscripts $j$ have been removed from the matrix elements for brevity. The outer product $\bar{\mathbf{q}}'_j {\mathbf{q}'_j}^T$ is computed in a similar manner

$$[Q'_j] = \bar{\mathbf{q}}'_j {\mathbf{q}'_j}^T, \tag{A.16}$$

where $[Q'_j]$ evaluates to the upper left seven rows and columns of (A.15). Eqns. (A.15) and (A.16) are substituted into (A.14)

$$\mathbf{a}^T[\bar{B}][Q_j][B]^T\bar{\mathbf{a}} - \mathbf{c}^T[\bar{D}][Q'_j][D]^T\bar{\mathbf{c}} = 0 \tag{A.17}$$

which can be rewritten as

$$\begin{Bmatrix} \mathbf{a}^T & \mathbf{c}^T \end{Bmatrix} \begin{bmatrix} [\bar{B}] & [0] \\ [0] & [\bar{D}] \end{bmatrix} \begin{bmatrix} [Q_j] & [0] \\ [0] & -[Q'_j] \end{bmatrix} \begin{bmatrix} [B]^T & [0] \\ [0] & [D]^T \end{bmatrix} \begin{Bmatrix} \bar{\mathbf{a}} \\ \bar{\mathbf{c}} \end{Bmatrix} = 0. \tag{A.18}$$

The edges of this product can be collapsed into a vector $\mathbf{p}$ such that

$$\mathbf{p} = \begin{Bmatrix} p_1 \\ \vdots \\ p_{19} \end{Bmatrix} = \begin{bmatrix} [\bar{B}]^T & [0] \\ [0] & [\bar{D}]^T \end{bmatrix} \begin{Bmatrix} \mathbf{a} \\ \mathbf{c} \end{Bmatrix} \tag{A.19}$$

which expands to

$$\begin{aligned}
p_1 &= \bar{h}(a\xi + c(k\bar{n} + l\bar{o})) - \bar{m}(c\beta + a(\bar{k}n + \bar{l}o)) & p_{11} &= \bar{k}\bar{n}(cl - ao) \\
p_2 &= \bar{k}(a\xi + c(h\bar{m} + l\bar{o})) - \bar{n}(c\beta + a(\bar{h}m + \bar{l}o)) & p_{12} &= \bar{l}\bar{o}(ck - an) \\
p_3 &= \bar{l}(a\xi + c(h\bar{m} + k\bar{n})) - \bar{o}(c\beta + a(\bar{h}m + \bar{k}n)) & p_{13} &= a\bar{c}(\bar{h}m + \bar{k}n + \bar{l}o) - \bar{a}c(h\bar{m} + k\bar{n} + l\bar{o}) \\
p_4 &= \bar{h}\bar{m}(ck - an) & p_{14} &= a\bar{c}\bar{k}m - \bar{a}ch\bar{n} \\
p_5 &= \bar{h}\bar{m}(cl - ao) & p_{15} &= a\bar{c}\bar{l}m - \bar{a}ch\bar{o} \\
p_6 &= (\bar{k}\bar{m} + \bar{h}\bar{n})(cl - ao) & p_{16} &= -\bar{a}ck\bar{m} + a\bar{c}\bar{h}n \\
p_7 &= (\bar{l}\bar{m} + \bar{h}\bar{o})(ck - an) & p_{17} &= -\bar{a}cl\bar{m} + a\bar{c}\bar{h}o \\
p_8 &= (\bar{l}\bar{n} + \bar{k}\bar{o})(ch - am) & p_{18} &= a\bar{c}\bar{k}o - \bar{a}cl\bar{n} \\
p_9 &= \bar{k}\bar{n}(ch - am) & p_{19} &= -\bar{a}ck\bar{o} + a\bar{c}\bar{l}n \\
p_{10} &= \bar{l}\bar{o}(ch - am) & & \\
\text{where} \quad \beta &= f\bar{f} - a\bar{a} - h\bar{h} - k\bar{k} - l\bar{l} & & \\
\xi &= g\bar{g} - c\bar{c} - m\bar{m} - n\bar{n} - o\bar{o} & &
\end{aligned} \tag{A.20}$$

The matrix $[\hat{Q}_j]$ is defined

$$[\hat{Q}_j] = \begin{bmatrix} [Q_j] & [0] \\ [0] & -[Q_j'] \end{bmatrix} \tag{A.21}$$

and (A.19) and (A.21) are substituted into (A.18) to create the desired form,

$$\mathbf{p}^T[\hat{Q}_j]\bar{\mathbf{p}} = 0 \qquad j = 1, \ldots, N-1. \tag{A.22}$$

The Hermitian matrix $[\hat{Q}_j]$ is constructed from the task specification and is pre- and post-multiplied by the 19×1 vector $\mathbf{p}$ and its conjugate. Each term in $\mathbf{p}$ is degree 4 and thus Eqns. (A.22) are degree 8.

# B Mathematica code: Function Generation Formulation

```
General Synthesis Equations
(*Synthesis equations take this form*)
eqns1=(a*bc*(g*gc-c*cc-d*dc)-c*dc*(f*fc-a*ac-b*bc))*(ac*b*(g*gc-c*cc-d*dc)
    -cc*d*(f*fc-a*ac-b*bc))+(a*bc*cc*d-ac*b*c*dc)^2;
(*Here is another way to write those equations with dot products*)
eqns2={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c}.{bc,-dc,-bc*d*dc,b*bc*dc}*{b,-d,-b
    *d*dc,b*bc*d}.{ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc}-{a*cc,ac*c}.{bc*d,-
    b*dc}*{b*dc,-bc*d}.{ac*c,a*cc};
Expand[eqns1]==Expand[eqns2]
True
(*b=Bmat.q*)
Bmat={{h,k,l,0,0,0,0,0,0,0,0,0},{-m,-n,-o,0,0,0,0,0,0,0,0,0},
{-m*(k*nc+l*oc)-h*(m*mc+n*nc+o*oc),-n*(h*mc+l*oc)-k*(m*mc+n*nc+o*oc),-o*(h
    *mc+k*nc)-l*(m*mc+n*nc+o*oc),-h*m*nc,-h*m*oc,-oc*(h*n+k*m),-nc*(h*o+l*m
    ),-mc*(k*o+l*n),-k*n*mc,-l*o*mc,-k*n*oc,-l*o*nc},
{h*(kc*n+lc*o)+m*(h*hc+k*kc+l*lc),k*(hc*m+lc*o)+n*(h*hc+k*kc+l*lc),l*(hc*m
    +kc*n)+o*(h*hc+k*kc+l*lc),h*m*kc,h*m*lc,lc*(h*n+k*m),kc*(h*o+l*m),hc*(k
    *o+l*n),k*n*hc,l*o*hc,k*n*lc,l*o*kc}};
Bcmat={{hc,kc,lc,0,0,0,0,0,0,0,0,0},{-mc,-nc,-oc,0,0,0,0,0,0,0,0,0},
{-mc*(kc*n+lc*o)-hc*(m*mc+n*nc+o*oc),-nc*(hc*m+lc*o)-kc*(m*mc+n*nc+o*oc),-
    oc*(hc*m+kc*n)-lc*(m*mc+n*nc+o*oc),-hc*mc*n,-hc*mc*o,-o*(hc*nc+kc*mc),-
    n*(hc*oc+lc*mc),-m*(kc*oc+lc*nc),-kc*nc*m,-lc*oc*m,-kc*nc*o,-lc*oc*n},
{hc*(k*nc+l*oc)+mc*(h*hc+k*kc+l*lc),kc*(h*mc+l*oc)+nc*(h*hc+k*kc+l*lc),lc
    *(h*mc+k*nc)+oc*(h*hc+k*kc+l*lc),hc*mc*k,hc*mc*l,l*(hc*nc+kc*mc),k*(hc*
    oc+lc*mc),h*(kc*oc+lc*nc),kc*nc*h,lc*oc*h,kc*nc*l,lc*oc*k}};
(*d=Dmat.q'*)
Dmat={{h*mc+k*nc+l*oc,k*mc,l*mc,h*nc,h*oc,k*oc,l*nc},{-(hc*m+kc*n+lc*o),-
    hc*n,-hc*o,-kc*m,-lc*m,-lc*n,-kc*o}};
Dcmat={{hc*m+kc*n+lc*o,kc*m,lc*m,hc*n,hc*o,kc*o,lc*n},{-(h*mc+k*nc+l*oc),-
    h*nc,-h*oc,-k*mc,-l*mc,-l*nc,-k*oc}};
avect={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c};
aCvect={ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc};
cvect={a*cc,ac*c};
cCvect={ac*c,a*cc};
(*BDmat={{Bmat,0},{0,Dmat}}*)
BDmat=Table[0,{6},{19}];
BDmat[[1;;4,1;;12]]=Bmat;
BDmat[[5;;6,13;;19]]=Dmat;
BcDcmat=Table[0,{6},{19}];
BcDcmat[[1;;4,1;;12]]=Bcmat;
BcDcmat[[5;;6,13;;19]]=Dcmat;
(*acvect={a,c}*)
acvect=Table[,{6}];
acvect[[1;;4]]=avect;
acvect[[5;;6]]=cvect;
aCcCvect=Table[,{6}];
aCcCvect[[1;;4]]=aCvect;
aCcCvect[[5;;6]]=cCvect;
\[ScriptP]=Transpose[BcDcmat].acvect;
```

```
\[ScriptP]c=Transpose[BDmat].aCcCvect;
\[ScriptP]={hc*(a*\[Xi]+c*(k*nc+l*oc))-mc*(c*\[Beta]+a*(kc*n+lc*o)),
kc*(a*\[Xi]+c*(h*mc+l*oc))-nc*(c*\[Beta]+a*(hc*m+lc*o)),
lc*(a*\[Xi]+c*(h*mc+k*nc))-oc*(c*\[Beta]+a*(hc*m+kc*n)),
hc*mc*(c*k-a*n),
hc*mc*(c*l-a*o),
(kc*mc+hc*nc)*(c*l-a*o),
(lc*mc+hc*oc)*(c*k-a*n),
(lc*nc+kc*oc)*(c*h-a*m),
kc*nc*(c*h-a*m),
lc*oc*(c*h-a*m),
kc*nc*(c*l-a*o),
lc*oc*(c*k-a*n),
a*cc*(hc*m+kc*n+lc*o)-ac*c*(h*mc+k*nc+l*oc),
a*cc*kc*m-ac*c*h*nc,
a*cc*lc*m-ac*c*h*oc,
-ac*c*k*mc+a*cc*hc*n,
-ac*c*l*mc+a*cc*hc*o,
a*cc*kc*o-ac*c*l*nc,
-ac*c*k*oc+a*cc*lc*n};
\[ScriptP]c={h*(ac*\[Xi]+cc*(kc*n+lc*o))-m*(cc*\[Beta]+ac*(k*nc+l*oc)),
k*(ac*\[Xi]+cc*(hc*m+lc*o))-n*(cc*\[Beta]+ac*(h*mc+l*oc)),
l*(ac*\[Xi]+cc*(hc*m+kc*n))-o*(cc*\[Beta]+ac*(h*mc+k*nc)),
h*m*(cc*kc-ac*nc),
h*m*(cc*lc-ac*oc),
(k*m+h*n)*(cc*lc-ac*oc),
(l*m+h*o)*(cc*kc-ac*nc),
(l*n+k*o)*(cc*hc-ac*mc),
k*n*(cc*hc-ac*mc),
l*o*(cc*hc-ac*mc),
k*n*(cc*lc-ac*oc),
l*o*(cc*kc-ac*nc),
ac*c*(h*mc+k*nc+l*oc)-a*cc*(hc*m+kc*n+lc*o),
ac*c*k*mc-a*cc*hc*n,
ac*c*l*mc-a*cc*hc*o,
-a*cc*kc*m+ac*c*h*nc,
-a*cc*lc*m+ac*c*h*oc,
ac*c*k*oc-a*cc*lc*n,
-a*cc*kc*o+ac*c*l*nc};
\[Beta]\[Xi]subs={\[Beta]->f*fc-a*ac-h*hc-k*kc-l*lc,\[Xi]->g*gc-c*cc-m*mc-
   n*nc-o*oc};
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
   i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
qvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]],Q[[j]]*S[[j]],Q[[j]]^2,S[[j]]^2,Q[[j]]^2*Sc[[j]],Qc[[j]]*S[[j]]^2},{
```

```
   j,npos-1}];
qpvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]]},{j,npos-1}];
qcvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]],Qc[[j]]*Sc[[j]],Qc[[j]]^2,Sc[[j]]^2,Qc[[j]]^2*S[[j]],Q[[j]]*Sc[[j
   ]]^2},{j,npos-1}];
qcpvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]]},{j,npos-1}];
isoSimplify[expression_,isopairs_]:=Module[{isoSubs,expression2},
expression2=expression;
isoSubs=Flatten[Table[{exp=i+1-j,expc=j}->{If[exp-expc>0,exp-expc,0],If[
   expc-exp>0,expc-exp,0]},{i,5},{j,Range[i]}]];
Do[
expression2=CoefficientRules[expression2,isopairs[[i]]];
expression2=expression2/.isoSubs;
expression2=FromCoefficientRules[expression2,isopairs[[i]]];
,{i,Length[isopairs]}];
expression2]
(*Qmat=qcvect.qvect\[Transpose]*)
Qmat=Table[Outer[Times,qcvect[[j]],qvect[[j]]],{j,npos-1}];
Qmat=isoSimplify[Qmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
Qpmat=Table[Outer[Times,qcpvect[[j]],qpvect[[j]]],{j,npos-1}];
Qpmat=isoSimplify[Qpmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
(*QmatBIG={{Qmat,0},{0,-Qpmat}}*)
QmatBIG=Table[0,{npos-1},{19},{19}];
Do[
QmatBIG[[j,1;;12,1;;12]]=Qmat[[j]];
QmatBIG[[j,13;;19,13;;19]]=-Qpmat[[j]];
,{j,npos-1}];
eqns3=Table[\[ScriptP].QmatBIG[[j]].\[ScriptP]c,{j,npos-1}];
(*The final product is p, pc, and QmatBIG[[j]]*)
eqns4=Table[(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)
   *(mc+Qc[[j]]*nc+Sc[[j]]*oc))-c*(mc+Qc[[j]]*nc+Sc[[j]]*oc)*(f*fc-a*ac-(h
   +Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc[[j]]*lc)))*(ac*(h+Q[[j]]*k+S[[j
   ]]*l)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)*(mc+Qc[[j]]*nc+Sc[[j]]*oc))-cc*(
   m+Q[[j]]*n+S[[j]]*o)*(f*fc-a*ac-(h+Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc
   [[j]]*lc)))+(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*cc*(m+Q[[j]]*n+S[[j]]*o)-ac*(
   h+Q[[j]]*k+S[[j]]*l)*c*(mc+Qc[[j]]*nc+Sc[[j]]*oc))^2,{j,npos-1}];
\[ScriptP] Vector For WII, SIII, SII
(*\[ScriptP] simplified for Watt II, l->0, n->0*)
\[ScriptP]WII={hc*\[Xi]-mc*\[Beta],
kc*\[Xi]+h*kc*mc,
-oc*\[Beta]-hc*m*oc,
hc*k*mc,
-hc*mc*o,
-kc*mc*o,
hc*k*oc,
h*kc*oc-kc*m*oc,
0,
0,
0,
```

```
0,
ac*(hc*m-h*mc),
ac*kc*m,
-ac*h*oc,
-ac*k*mc,
ac*hc*o,
ac*kc*o,
-ac*k*oc};
\[ScriptP]cWII={h*\[Xi]-m*\[Beta],
k*\[Xi]+hc*k*m,
-o*\[Beta]-h*mc*o,
h*kc*m,
-h*m*oc,
-k*m*oc,
h*kc*o,
hc*k*o-k*mc*o,
0,
0,
0,
0,
a*(h*mc-hc*m),
a*k*mc,
-a*hc*o,
-a*kc*m,
a*h*oc,
a*k*oc,
-a*kc*o};
\[Beta]\[Xi]subsWII={\[Beta]->h*kc+hc*k-a*(fc+ac)-ac*(f+a),\[Xi]->m*oc+mc*
   o-a*(gc+cc)-ac*(g+c)};
(*\[ScriptP] simplified for Watt II with r subs*)
\[ScriptP]WII={hc*\[Xi]-mc*\[Beta],
kc*\[Xi]+mc*r1,
-oc*\[Beta]-hc*r2,
mc*rc1,
-hc*rc2,
-kc*rc2,
oc*rc1,
oc*r1-kc*r2,
0,
0,
0,
0,
ac*(hc*m-h*mc),
ac*kc*m,
-ac*h*oc,
-ac*k*mc,
ac*hc*o,
ac*kc*o,
-ac*k*oc};
\[ScriptP]cWII={h*\[Xi]-m*\[Beta],
k*\[Xi]+m*rc1,
```

```
-o*\[Beta]-h*rc2,
m*r1,
-h*r2,
-k*r2,
o*r1,
o*rc1-k*rc2,
0,
0,
0,
0,
a*(h*mc-hc*m),
a*k*mc,
-a*hc*o,
-a*kc*m,
a*h*oc,
a*k*oc,
-a*kc*o};
\[Beta]\[Xi]subsWII={\[Beta]->r1+rc1-a*(fc+ac)-ac*(f+a),\[Xi]->r2+rc2-a*(
   gc+cc)-ac*(g+c)};
auxeqnsWII={h*kc-r1,hc*k-rc1,m*oc-r2,mc*o-rc2};
(*Some final equations that appear in the paper*)
\[ScriptP]WIIforPaper={(Ac-CCc)*\[Xi]-(Bc-CCc)*\[Beta],
(DDc-Ac)*\[Xi]+(Bc-CCc)*r1,
-(Fc-Bc)*\[Beta]-(Ac-CCc)*r2,
(Bc-CCc)*rc1,
-(Ac-CCc)*rc2,
-(DDc-Ac)*rc2,
(Fc-Bc)*rc1,
(Fc-Bc)*r1-(DDc-Ac)*r2,
0,
0,
0,
0,
-(Gc-CCc)*((Ac-CCc)*(B-CC)-(A-CC)*(Bc-CCc)),
-(Gc-CCc)*(DDc-Ac)*(B-CC),
(Gc-CCc)*(A-CC)*(Fc-Bc),
(Gc-CCc)*(DD-A)*(Bc-CCc),
-(Gc-CCc)*(Ac-CCc)*(F-B),
-(Gc-CCc)*(DDc-Ac)*(F-B),
(Gc-CCc)*(DD-A)*(Fc-Bc)};
\[Beta]\[Xi]subsWIIforPaper={\[Beta]->r1+rc1+(G-CC)*(CCc-DDc)+(Gc-CCc)*(CC
   -DD),\[Xi]->r2+rc2+(G-CC)*(CCc-Fc)+(Gc-CCc)*(CC-F)};
auxeqnsWIIforPaper={(A-CC)*(DDc-Ac)-r1,(Ac-CCc)*(DD-A)-rc1,(B-CC)*(Fc-Bc)-
   r2,(Bc-CCc)*(F-B)-rc2};
(*Some final equations that appear in the paper even more simplified*)
\[ScriptP]WIIforPaper1={Ac*\[Xi]-Bc*\[Beta],
(DDc-Ac)*\[Xi]+Bc*r1,
-(Fc-Bc)*\[Beta]-Ac*r2,
Bc*rc1,
-Ac*rc2,
-(DDc-Ac)*rc2,
```

```
(Fc-Bc)*rc1,
(Fc-Bc)*r1-(DDc-Ac)*r2,
0,
0,
0,
0,
(A*Bc-Ac*B),
-(DDc-Ac)*B,
A*(Fc-Bc),
(DD-A)*Bc,
-Ac*(F-B),
-(DDc-Ac)*(F-B),
(DD-A)*(Fc-Bc)};
\[Beta]\[Xi]subsWIIforPaper1={\[Beta]->r1+rc1-DD-DDc,\[Xi]->r2+rc2-F-Fc};
auxeqnsWIIforPaper1={A*(DDc-Ac)-r1,Ac*(DD-A)-rc1,B*(Fc-Bc)-r2,Bc*(F-B)-rc2
   };
(*\[ScriptP] simplified for Stephenson III, l->0, n->k*)
\[ScriptP]SIII={k*kc*(c*hc-a*mc)+\[Xi]*a*hc-\[Beta]*c*mc,
kc*(a*\[Xi]-c*\[Beta]+h*c*mc-m*a*hc),
-k*kc*a*oc-\[Beta]*c*oc-a*hc*m*oc,
k*(hc*c*mc-mc*a*hc),
-a*hc*mc*o,
-kc*(a*mc*o+o*a*hc),
k*(hc*c*oc-oc*a*hc),
kc*(h*c*oc-a*m*oc),
kc^2*(c*h-a*m),
0,
-kc^2*a*o,
0,
k*kc*(a*cc-ac*c)+a*hc*cc*m-ac*h*c*mc,
kc*(a*cc*m-c*ac*h),
-ac*h*c*oc,
-k*(ac*c*mc-cc*a*hc),
a*hc*cc*o,
kc*a*cc*o,
-k*ac*c*oc};
\[ScriptP]cSIII={k*kc*(cc*h-ac*m)+\[Xi]*ac*h-\[Beta]*cc*m,
k*(ac*\[Xi]-cc*\[Beta]+hc*cc*m-mc*ac*h),
-k*kc*ac*o-\[Beta]*cc*o-ac*h*mc*o,
kc*(h*cc*m-m*ac*h),
-ac*h*m*oc,
-k*(ac*m*oc+oc*ac*h),
kc*(h*cc*o-o*ac*h),
k*(hc*cc*o-ac*mc*o),
k^2*(cc*hc-ac*mc),
0,
-k^2*ac*oc,
0,
k*kc*(ac*c-a*cc)+ac*h*c*mc-a*hc*cc*m,
k*(ac*c*mc-cc*a*hc),
-a*hc*cc*o,
```

```
-kc*(a*cc*m-c*ac*h),
ac*h*c*oc,
k*ac*c*oc,
-kc*a*cc*o};
\[Beta]\[Xi]subsSIII={\[Beta]->a*hc+ac*h+k*(fc-kc)+kc*(f-k),\[Xi]->c*mc+cc
    *m+c*oc+cc*o+m*oc+mc*o+k*(gc-kc)+kc*(g-k)};
(*\[ScriptP] simplified for Stephenson III with r subs*)
\[ScriptP]SIII={k*kc*(c*hc-a*mc)+\[Xi]*r1-\[Beta]*r2,
kc*(a*\[Xi]-c*\[Beta]+h*r2-m*r1),
-k*kc*a*oc-\[Beta]*r3-r1*r4,
k*(hc*r2-mc*r1),
-r1*rc4,
-kc*(a*rc4+o*r1),
k*(hc*r3-oc*r1),
kc*(h*r3-a*r4),
kc^2*(c*h-a*m),
0,
-kc^2*a*o,
0,
k*kc*(a*cc-ac*c)+r1*rc2-rc1*r2,
kc*(a*rc2-c*rc1),
-rc1*r3,
-k*(ac*r2-cc*r1),
r1*rc3,
kc*a*rc3,
-k*ac*r3};
\[ScriptP]cSIII={k*kc*(cc*h-ac*m)+\[Xi]*rc1-\[Beta]*rc2,
k*(ac*\[Xi]-cc*\[Beta]+hc*rc2-mc*rc1),
-k*kc*ac*o-\[Beta]*rc3-rc1*rc4,
kc*(h*rc2-m*rc1),
-rc1*r4,
-k*(ac*r4+oc*rc1),
kc*(h*rc3-o*rc1),
k*(hc*rc3-ac*rc4),
k^2*(cc*hc-ac*mc),
0,
-k^2*ac*oc,
0,
k*kc*(ac*c-a*cc)+rc1*r2-r1*rc2,
k*(ac*r2-cc*r1),
-r1*rc3,
-kc*(a*rc2-c*rc1),
rc1*r3,
k*ac*r3,
-kc*a*rc3};
\[Beta]\[Xi]subsSIII={\[Beta]->r1+rc1+k*(fc-kc)+kc*(f-k),\[Xi]->r2+rc2+r3+
    rc3+r4+rc4+k*(gc-kc)+kc*(g-k)};
auxeqnsSIII={a*hc-r1,ac*h-rc1,c*mc-r2,cc*m-rc2,c*oc-r3,cc*o-rc3,m*oc-r4,mc
    *o-rc4};
(*Some final equations that appear in the paper*)
\[ScriptP]SIIIforPaper={(DD-A)*(DDc-Ac)*((H-DD)*(Ac-CCc)-(G-DD)*(Ac-Bc))
```

```
   +\[Xi]*r1-\[Beta]*r2,
(DDc-Ac)*((G-DD)*\[Xi]-(H-DD)*\[Beta]+(A-CC)*r2-(A-B)*r1),
(DD-A)*(DDc-Ac)*(G-DD)*(Fc-Bc)-\[Beta]*r3-r1*r4,
(DD-A)*((Ac-CCc)*r2-(Ac-Bc)*r1),
-r1*rc4,
-(DDc-Ac)*((G-DD)*rc4-(F-B)*r1),
(DD-A)*((Ac-CCc)*r3+(Fc-Bc)*r1),
(DDc-Ac)*((A-CC)*r3-(G-DD)*r4),
(DDc-Ac)^2*((H-DD)*(A-CC)-(G-DD)*(A-B)),
0,
(DDc-Ac)^2*(G-DD)*(F-B),
0,
(DD-A)*(DDc-Ac)*((G-DD)*(Hc-DDc)-(Gc-DDc)*(H-DD))+r1*rc2-rc1*r2,
(DDc-Ac)*((G-DD)*rc2-(H-DD)*rc1),
-rc1*r3,
-(DD-A)*((Gc-DDc)*r2-(Hc-DDc)*r1),
r1*rc3,
(DDc-Ac)*(G-DD)*rc3,
-(DD-A)*(Gc-DDc)*r3};
\[Beta]\[Xi]subsSIIIforPaper={\[Beta]->r1+rc1+(DD-A)*(Gc-CCc-DDc+Ac)+(DDc-
   Ac)*(G-CC-DD+A),\[Xi]->r2+rc2+r3+rc3+r4+rc4+(DD-A)*(Hc-Fc-DDc+Ac)+(DDc-
   Ac)*(H-F-DD+A)};
auxeqnsSIIIforPaper={(G-DD)*(Ac-CCc)-r1,(Gc-DDc)*(A-CC)-rc1,(H-DD)*(Ac-Bc)
   -r2,(Hc-DDc)*(A-B)-rc2,-(H-DD)*(Fc-Bc)-r3,-(Hc-DDc)*(F-B)-rc3,-(A-B)*(
   Fc-Bc)-r4,-(Ac-Bc)*(F-B)-rc4};
(*Some final equations that appear in the paper even more simplified*)
\[ScriptP]SIIIforPaper1={-(H-1)*CCc+(G-1)*Bc+\[Xi]*r1-\[Beta]*r2,
(G-1)*\[Xi]-(H-1)*\[Beta]-CC*r2+B*r1,
(G-1)*(Fc-Bc)-\[Beta]*r3-r1*r4,
-CCc*r2+Bc*r1,
-r1*rc4,
-(G-1)*rc4+(F-B)*r1,
-CCc*r3+(Fc-Bc)*r1,
-CC*r3-(G-1)*r4,
-(H-1)*CC+(G-1)*B,
0,
(G-1)*(F-B),
0,
(G-1)*(Hc-1)-(Gc-1)*(H-1)+r1*rc2-rc1*r2,
(G-1)*rc2-(H-1)*rc1,
-rc1*r3,
-(Gc-1)*r2+(Hc-1)*r1,
r1*rc3,
(G-1)*rc3,
-(Gc-1)*r3};
\[Beta]\[Xi]subsSIIIforPaper1={\[Beta]->r1+rc1+G+Gc-CC-CCc-2,\[Xi]->r2+rc2
   +r3+rc3+r4+rc4+H+Hc-F-Fc-2};
auxeqnsSIIIforPaper1={-(G-1)*CCc-r1,-(Gc-1)*CC-rc1,-(H-1)*Bc-r2,-(Hc-1)*B-
   rc2,-(H-1)*(Fc-Bc)-r3,-(Hc-1)*(F-B)-rc3,B*(Fc-Bc)-r4,Bc*(F-B)-rc4};
(*The new SII*)
(*\[ScriptP] simplified for Stephenson II, m->h, n->k*)
```

```
\[ScriptP]SII={hc*(k*kc*(c-a)+a*\[Xi]-c*\[Beta]-o*a*lc+l*c*oc),
kc*(h*hc*(c-a)+a*\[Xi]-c*\[Beta]-o*a*lc+l*c*oc),
(h*hc+k*kc)*(c*lc-a*oc)+\[Xi]*a*lc-\[Beta]*c*oc,
hc^2*k*(c-a),
hc^2*(c*l-o*a),
2*hc*kc*(c*l-a*o),
hc*k*(c*lc-a*oc-a*lc+c*oc),
h*kc*(c*lc-a*oc-a*lc+c*oc),
h*kc^2*(c-a),
h*(-oc*a*lc+lc*c*oc),
kc^2*(c*l-a*o),
-k*(oc*a*lc-lc*c*oc),
(h*hc+k*kc)*(a*cc-ac*c)+a*lc*cc*o-ac*l*c*oc,
h*kc*(a*cc-ac*c),
h*(cc*a*lc-ac*c*oc),
hc*k*(a*cc-ac*c),
-hc*(c*ac*l-a*cc*o),
-kc*(c*ac*l-a*cc*o),
k*(cc*a*lc-ac*c*oc)};
\[ScriptP]cSII={h*(k*kc*(cc-ac)+ac*\[Xi]-cc*\[Beta]-oc*ac*l+lc*cc*o),
k*(h*hc*(cc-ac)+ac*\[Xi]-cc*\[Beta]-oc*ac*l+lc*cc*o),
(h*hc+k*kc)*(cc*l-ac*o)+\[Xi]*ac*l-\[Beta]*cc*o,
h^2*kc*(cc-ac),
h^2*(cc*lc-oc*ac),
2*h*k*(cc*lc-ac*oc),
h*kc*(cc*l-ac*o-ac*l+cc*o),
hc*k*(cc*l-ac*o-ac*l+cc*o),
hc*k^2*(cc-ac),
hc*(-o*ac*l+l*cc*o),
k^2*(cc*lc-ac*oc),
-kc*(o*ac*l-l*cc*o),
(h*hc+k*kc)*(ac*c-a*cc)+ac*l*c*oc-a*lc*cc*o,
hc*k*(ac*c-a*cc),
hc*(c*ac*l-a*cc*o),
h*kc*(ac*c-a*cc),
-h*(cc*a*lc-ac*c*oc),
-k*(cc*a*lc-ac*c*oc),
kc*(c*ac*l-a*cc*o)};
(*\[Beta]\[Xi]subsSII={\[Beta]->a*lc+ac*l-h*kc-hc*k+h*(fc-hc)+hc*(f-h)+k*(
    fc-kc)+kc*(f-k),
\[Xi]->c*oc+cc*o-h*kc-hc*k+h*(gc-hc)+hc*(g-h)+k*(gc-kc)+kc*(g-k)};*)
\[Beta]\[Xi]subsSII={\[Beta]->a*lc+ac*l+h*kc+hc*k+(h+k)*(fc-hc-kc)+(hc+kc)
    *(f-h-k),
\[Xi]->c*oc+cc*o+h*kc+hc*k+(h+k)*(gc-hc-kc)+(hc+kc)*(g-h-k)};
(*For paper*)
\[ScriptP]SIIforPaper={(Ac-Bc)*((CC-A)*(CCc-Ac)*(H-G)+(G-CC)*\[Xi]-(H-CC)
    *\[Beta]-(F-B)*(G-CC)*(DDc-Bc)+(DD-B)*(H-CC)*(Fc-Bc)),
(CCc-Ac)*((A-B)*(Ac-Bc)*(H-G)+(G-CC)*\[Xi]-(H-CC)*\[Beta]-(F-B)*(G-CC)*(
    DDc-Bc)+(DD-B)*(H-CC)*(Fc-Bc)),
((A-B)*(Ac-Bc)+(CC-A)*(CCc-Ac))*(-(H-CC)*(DDc-Bc)+(G-CC)*(Fc-Bc))-\[Xi]*(G
    -CC)*(DDc-Bc)+\[Beta]*(H-CC)*(Fc-Bc),
```

```
(Ac-Bc)^2*(CC-A)*(H-G),
(Ac-Bc)^2*(-(H-CC)*(DD-B)+(F-B)*(G-CC)),
2*(Ac-Bc)*(CCc-Ac)*(-(H-CC)*(DD-B)+(G-CC)*(F-B)),
(Ac-Bc)*(CC-A)*(-(H-CC)*(DDc-Bc)+(G-CC)*(Fc-Bc)+(G-CC)*(DDc-Bc)-(H-CC)*(Fc
    -Bc)),
(A-B)*(CCc-Ac)*(-(H-CC)*(DDc-Bc)+(G-CC)*(Fc-Bc)+(G-CC)*(DDc-Bc)-(H-CC)*(Fc
    -Bc)),
(A-B)*(CCc-Ac)^2*(H-G),
(A-B)*(-(Fc-Bc)*(G-CC)*(DDc-Bc)+(DDc-Bc)*(H-CC)*(Fc-Bc)),
(CCc-Ac)^2*(-(H-CC)*(DD-B)+(G-CC)*(F-B)),
(CC-A)*(-(Fc-Bc)*(G-CC)*(DDc-Bc)+(DDc-Bc)*(H-CC)*(Fc-Bc)),
((A-B)*(Ac-Bc)+(CC-A)*(CCc-Ac))*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC))+(G-CC)*(
    DDc-Bc)*(Hc-CCc)*(F-B)+-(Gc-CCc)*(DD-B)*(H-CC)*(Fc-Bc),
(A-B)*(CCc-Ac)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
-(A-B)*((Hc-CCc)*(G-CC)*(DDc-Bc)-(Gc-CCc)*(H-CC)*(Fc-Bc)),
(Ac-Bc)*(CC-A)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
(Ac-Bc)*((H-CC)*(Gc-CCc)*(DD-B)-(G-CC)*(Hc-CCc)*(F-B)),
(CCc-Ac)*((H-CC)*(Gc-CCc)*(DD-B)-(G-CC)*(Hc-CCc)*(F-B)),
-(CC-A)*((Hc-CCc)*(G-CC)*(DDc-Bc)-(Gc-CCc)*(H-CC)*(Fc-Bc))};
\[Beta]\[Xi]subsSIIforPaper={\[Beta]->-(G-CC)*(DDc-Bc)-(Gc-CCc)*(DD-B)-(A-
    B)*(CCc-Ac)-(Ac-Bc)*(CC-A)+(A-B)*(Gc-DDc-Ac+Bc)+(Ac-Bc)*(G-DD-A+B)+(CC-
    A)*(Gc-DDc-CCc+Ac)+(CCc-Ac)*(G-DD-CC+A),
\[Xi]->-(H-CC)*(Fc-Bc)-(Hc-CCc)*(F-B)-(A-B)*(CCc-Ac)-(Ac-Bc)*(CC-A)+(A-B)
    *(Hc-Fc-Ac+Bc)+(Ac-Bc)*(H-F-A+B)+(CC-A)*(Hc-Fc-CCc+Ac)+(CCc-Ac)*(H-F-CC
    +A)};
(*For paper again*)
\[ScriptP]SIIforPaper1={-CC*CCc*(H-G)-(G-CC)*\[Xi]+(H-CC)*\[Beta]+(F-1)*(G
    -CC)*(DDc-1)-(DD-1)*(H-CC)*(Fc-1),
CCc*((H-G)+(G-CC)*\[Xi]-(H-CC)*\[Beta]-(F-1)*(G-CC)*(DDc-1)+(DD-1)*(H-CC)
    *(Fc-1)),
-(1+CC*CCc)*((H-CC)*(DDc-1)-(G-CC)*(Fc-1))-\[Xi]*(G-CC)*(DDc-1)+\[Beta]*(H
    -CC)*(Fc-1),
CC*(H-G),
-(H-CC)*(DD-1)+(F-1)*(G-CC),
2*CCc*((H-CC)*(DD-1)-(G-CC)*(F-1)),
CC*((H-CC)*(DDc-1)-(G-CC)*(Fc-1)-(G-CC)*(DDc-1)+(H-CC)*(Fc-1)),
CCc*((H-CC)*(DDc-1)-(G-CC)*(Fc-1)-(G-CC)*(DDc-1)+(H-CC)*(Fc-1)),
-CCc^2*(H-G),
(Fc-1)*(G-CC)*(DDc-1)-(DDc-1)*(H-CC)*(Fc-1),
-CCc^2*((H-CC)*(DD-1)-(G-CC)*(F-1)),
-CC*((Fc-1)*(G-CC)*(DDc-1)-(DDc-1)*(H-CC)*(Fc-1)),
(1+CC*CCc)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC))+(G-CC)*(DDc-1)*(Hc-CCc)*(F-1)
    -(Gc-CCc)*(DD-1)*(H-CC)*(Fc-1),
-CCc*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
(Hc-CCc)*(G-CC)*(DDc-1)-(Gc-CCc)*(H-CC)*(Fc-1),
-CC*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
-(H-CC)*(Gc-CCc)*(DD-1)+(G-CC)*(Hc-CCc)*(F-1),
CCc*((H-CC)*(Gc-CCc)*(DD-1)-(G-CC)*(Hc-CCc)*(F-1)),
-CC*((Hc-CCc)*(G-CC)*(DDc-1)-(Gc-CCc)*(H-CC)*(Fc-1))};
(*\[Beta]\[Xi]subsSIIforPaper1={\[Beta]->-(G-CC)*(DDc-1)-(Gc-CCc)*(DD-1)+(
    CC-1)*(Gc-DDc-CCc)+(CCc-1)*(G-DD-CC)-2,
```

```
\[Xi]->-(H-CC)*(Fc-1)-(Hc-CCc)*(F-1)+(CC-1)*(Hc-Fc-CCc)+(CCc-1)*(H-F-CC)
    -2};*)
\[Beta]\[Xi]subsSIIforPaper1={\[Beta]->CC*(Gc-CCc)+CCc*(G-CC)-DD*(Gc-1)-
    DDc*(G-1)-2,
\[Xi]->CC*(Hc-CCc)+CCc*(H-CC)-F*(Hc-1)-Fc*(H-1)-2};
(*\[ScriptP] simplified for Stephenson II with r subs*)
\[ScriptP]SIIAlt1={hc*(k*kc*(c-a)+a*\[Xi]-c*\[Beta]-o*r1+l*r2),
kc*(h*hc*(c-a)+a*\[Xi]-c*\[Beta]-o*r1+l*r2),
(h*hc+k*kc)*(c*lc-a*oc)+\[Xi]*r1-\[Beta]*r2,
hc^2*k*(c-a),
hc^2*(c*l-o*a),
2*hc*kc*(c*l-a*o),
hc*k*(c*lc-a*oc-r1+r2),
h*kc*(c*lc-a*oc-r1+r2),
h*kc^2*(c-a),
h*(-oc*r1+lc*r2),
kc^2*(c*l-a*o),
-k*(oc*r1-lc*r2),
(h*hc+k*kc)*(a*cc-ac*c)+r1*rc2-rc1*r2,
h*kc*(a*cc-ac*c),
h*(cc*r1-ac*r2),
hc*k*(a*cc-ac*c),
-hc*(c*rc1-a*rc2),
-kc*(c*rc1-a*rc2),
k*(cc*r1-ac*r2)};
\[ScriptP]cSIIAlt1={h*(k*kc*(cc-ac)+ac*\[Xi]-cc*\[Beta]-oc*rc1+lc*rc2),
k*(h*hc*(cc-ac)+ac*\[Xi]-cc*\[Beta]-oc*rc1+lc*rc2),
(h*hc+k*kc)*(cc*l-ac*o)+\[Xi]*rc1-\[Beta]*rc2,
h^2*kc*(cc-ac),
h^2*(cc*lc-oc*ac),
2*h*k*(cc*lc-ac*oc),
h*kc*(cc*l-ac*o-rc1+rc2),
hc*k*(cc*l-ac*o-rc1+rc2),
hc*k^2*(cc-ac),
hc*(-o*rc1+l*rc2),
k^2*(cc*lc-ac*oc),
-kc*(o*rc1-l*rc2),
(h*hc+k*kc)*(ac*c-a*cc)+rc1*r2-r1*rc2,
hc*k*(ac*c-a*cc),
hc*(c*rc1-a*rc2),
h*kc*(ac*c-a*cc),
-h*(cc*r1-ac*r2),
-k*(cc*r1-ac*r2),
kc*(c*rc1-a*rc2)};
(*\[Beta]\[Xi]subsSIIAlt1={\[Beta]->r1+rc1-h*kc-hc*k+h*(fc-hc)+hc*(f-h)+k
    *(fc-kc)+kc*(f-k),
\[Xi]->r2+rc2-h*kc-hc*k+h*(gc-hc)+hc*(g-h)+k*(gc-kc)+kc*(g-k)};*)
\[Beta]\[Xi]subsSIIAlt1={\[Beta]->r1+rc1+h*kc+hc*k+(h+k)*(fc-hc-kc)+(hc+kc
    )*(f-h-k),
\[Xi]->r2+rc2+h*kc+hc*k+(h+k)*(gc-hc-kc)+(hc+kc)*(g-h-k)};
auxeqnsSIIAlt1={a*lc-r1,ac*l-rc1,c*oc-r2,cc*o-rc2};
```

```
\[ScriptP]SIIAlt1forPaper={(Ac-Bc)*((CC-A)*(CCc-Ac)*(H-G)+(G-CC)*\[Xi]-(H-
   CC)*\[Beta]+(F-B)*r1-(DD-B)*r2),
(CCc-Ac)*((A-B)*(Ac-Bc)*(H-G)+(G-CC)*\[Xi]-(H-CC)*\[Beta]+(F-B)*r1-(DD-B)*
   r2),
((A-B)*(Ac-Bc)+(CC-A)*(CCc-Ac))*(-(H-CC)*(DDc-Bc)+(G-CC)*(Fc-Bc))+\[Xi]*r1
   -\[Beta]*r2,
(Ac-Bc)^2*(CC-A)*(H-G),
-(Ac-Bc)^2*((H-CC)*(DD-B)-(F-B)*(G-CC)),
-2*(Ac-Bc)*(CCc-Ac)*((H-CC)*(DD-B)-(G-CC)*(F-B)),-(Ac-Bc)*(CC-A)*((H-CC)(
   DDc-Bc)-(G-CC)*(Fc-Bc)+r1-r2),
-(A-B)*(CCc-Ac)*((H-CC)*(DDc-Bc)-(G-CC)*(Fc-Bc)+r1-r2),
(A-B)*(CCc-Ac)^2*(H-G),
(A-B)*((Fc-Bc)*r1-(DDc-Bc)*r2),
-(CCc-Ac)^2*((H-CC)(DD-B)-(G-CC)*(F-B)),
(CC-A)*((Fc-Bc)*r1-(DDc-Bc)*r2),
((A-B)*(Ac-Bc)+(CC-A)*(CCc-Ac))*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC))+r1*rc2-
   rc1*r2,
(A-B)*(CCc-Ac)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
(A-B)*((Hc-CCc)*r1-(Gc-CCc)*r2),
(Ac-Bc)*(CC-A)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
-(Ac-Bc)*((H-CC)*rc1-(G-CC)*rc2),
-(CCc-Ac)*((H-CC)*rc1-(G-CC)*rc2),
(CC-A)*((Hc-CCc)*r1-(Gc-CCc)*r2)};
\[Beta]\[Xi]subsSIIAlt1forPaper={\[Beta]->r1+rc1-(A-B)*(CCc-Ac)-(Ac-Bc)*(
   CC-A)+(A-B)*(Gc-DDc-Ac+Bc)+(Ac-Bc)*(G-DD-A+B)+(CC-A)*(Gc-DDc-CCc+Ac)+(
   CCc-Ac)*(G-DD-CC+A),
\[Xi]->r2+rc2-(A-B)*(CCc-Ac)-(Ac-Bc)*(CC-A)+(A-B)*(Hc-Fc-Ac+Bc)+(Ac-Bc)*(H
   -F-A+B)+(CC-A)*(Hc-Fc-CCc+Ac)+(CCc-Ac)*(H-F-CC+A)};
auxeqnsSIIAlt1forPaper={-(G-CC)(DDc-Bc)-r1,-(Gc-CCc)*(DD-B)-rc1,-(H-CC)*(
   Fc-Bc)-r2,-(Hc-CCc)*(F-B)-rc2};
(*Some final equations that appear in the paper even more simplified*)
\[ScriptP]SIIAlt1forPaper1={-CC*CCc*(H-G)-(G-CC)*\[Xi]+(H-CC)*\[Beta]-(F
   -1)*r1+(DD-1)*r2,
CCc*(H-G+(G-CC)*\[Xi]-(H-CC)*\[Beta]+(F-1)*r1-(DD-1)*r2),
-(1+CC*CCc)*((H-CC)*(DDc-1)-(G-CC)*(Fc-1))+\[Xi]*r1-\[Beta]*r2,
CC*(H-G),
-(H-CC)*(DD-1)+(F-1)*(G-CC),
2*CCc*((H-CC)*(DD-1)-(G-CC)*(F-1)),
CC*((H-CC)(DDc-1)-(G-CC)*(Fc-1)+r1-r2),
CCc*((H-CC)*(DDc-1)-(G-CC)*(Fc-1)+r1-r2),
-CCc^2*(H-G),
-(Fc-1)*r1+(DDc-1)*r2,
-CCc^2*((H-CC)(DD-1)-(G-CC)*(F-1)),
CC*((Fc-1)*r1-(DDc-1)*r2),
(1+CC*CCc)*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC))+r1*rc2-rc1*r2,
-CCc*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
-(Hc-CCc)*r1+(Gc-CCc)*r2,
-CC*((G-CC)*(Hc-CCc)-(Gc-CCc)*(H-CC)),
(H-CC)*rc1-(G-CC)*rc2,
-CCc*((H-CC)*rc1-(G-CC)*rc2),
CC*((Hc-CCc)*r1-(Gc-CCc)*r2)};
```

```
\[Beta]\[Xi]subsSIIAlt1forPaper1={\[Beta]->r1+rc1+(CC-1)*(Gc-DDc-CCc)+(CCc
   -1)*(G-DD-CC)-2,
\[Xi]->r2+rc2+(CC-1)*(Hc-Fc-CCc)+(CCc-1)*(H-F-CC)-2};
auxeqnsSIIAlt1forPaper1={-(G-CC)(DDc-1)-r1,-(Gc-CCc)*(DD-1)-rc1,-(H-CC)*(
   Fc-1)-r2,-(Hc-CCc)*(F-1)-rc2};
(*\[ScriptP] simplified for Stephenson II, m->h, n->k*)
\[ScriptP]SIIAlt2={hc*(a*\[Xi]-c*\[Beta]+k*c*kc+l*c*oc-k*a*kc-o*a*lc),
h*hc*(c*kc-a*kc)+\[Xi]*a*kc-\[Beta]*c*kc+kc*l*c*oc-a*lc*kc*o,
h*hc(c*lc-a*oc)+\[Xi]*a*lc-\[Beta]*c*oc+k*lc*c*kc-a*kc*k*oc,
hc^2*k*(c-a),
hc^2*(c*l-a*o),
2*hc*(c*kc*l-a*kc*o),
hc*(c*k*lc+k*c*oc-k*a*lc-a*k*oc),
h*(lc*c*kc-kc*a*lc+kc*c*oc-oc*a*kc),
h*kc*(c*kc-a*kc),
h*(lc*c*oc-oc*a*lc),
kc*l*c*kc-a*kc*kc*o,
k*lc*c*oc-a*lc*k*oc,
h*hc*(a*cc-ac*c)+a*kc*cc*k-ac*k*c*kc+a*lc*cc*o-ac*l*c*oc,
h*(cc*a*kc-ac*c*kc),
h*(cc*a*lc-ac*c*oc),
-hc*(c*ac*k-a*cc*k),
-hc*(c*ac*l-a*cc*o),
a*kc*cc*o-ac*l*c*kc,
-ac*k*c*oc+a*lc*cc*k};
\[ScriptP]cSIIAlt2={h*(ac*\[Xi]-cc*\[Beta]+kc*cc*k+lc*cc*o-kc*ac*k-oc*ac*l
   ),
h*hc*(cc*k-ac*k)+\[Xi]*ac*k-\[Beta]*cc*k+k*lc*cc*o-ac*l*k*oc,
h*hc(cc*l-ac*o)+\[Xi]*ac*l-\[Beta]*cc*o+kc*l*cc*k-ac*k*kc*o,
h^2*kc*(cc-ac),
h^2*(cc*lc-ac*oc),
2*h*(cc*k*lc-ac*k*oc),
h*(cc*kc*l+kc*cc*o-kc*ac*l-ac*kc*o),
hc*(l*cc*k-k*ac*l+k*cc*o-o*ac*k),
hc*k*(cc*k-ac*k),
hc*(l*cc*o-o*ac*l),
k*lc*cc*k-ac*k*k*oc,
kc*l*cc*o-ac*l*kc*o,
h*hc*(ac*c-a*cc)+ac*k*c*kc-a*kc*cc*k+ac*l*c*oc-a*lc*cc*o,
hc*(c*ac*k-a*cc*k),
hc*(c*ac*l-a*cc*o),
-h*(cc*a*kc-ac*c*kc),
-h*(cc*a*lc-ac*c*oc),
ac*k*c*oc-a*lc*cc*k,
-a*kc*cc*o+ac*l*c*kc};
\[Beta]\[Xi]subsSIIAlt2={\[Beta]->a*kc+ac*k+a*lc+ac*l+k*lc+kc*l+h*(fc-hc)+
   hc*(f-h),\[Xi]->c*kc+cc*k+c*oc+cc*o+k*oc+kc*o+h*(gc-hc)+hc*(g-h)};
(*\[ScriptP] simplified for Stephenson II with r subs*)
\[ScriptP]SIIAlt2={hc*(a*\[Xi]-c*\[Beta]+k*r4+l*r5-k*r1-o*r2),
h*hc*(r4-r1)+\[Xi]*r1-\[Beta]*r4+rc3*r5-r2*rc6,
h*hc*(c*lc-a*oc)+\[Xi]*r2-\[Beta]*r5+r3*r4-r1*r6,
```

```
hc^2*k*(c-a),
hc^2*(c*l-a*o),
2*hc*(l*r4-o*r1),
hc*(c*r3+k*r5-k*r2-a*r6),
h*(lc*r4-kc*r2+kc*r5-oc*r1),
h*kc*(r4-r1),
h*(lc*r5-oc*r2),
rc3*r4-r1*rc6,
r3*r5-r2*r6,
h*hc*(a*cc-ac*c)+r1*rc4-rc1*r4+r2*rc5-rc2*r5,
h*(cc*r1-ac*r4),
h*(cc*r2-ac*r5),
-hc*(c*rc1-a*rc4),
-hc*(c*rc2-a*rc5),
r1*rc5-rc2*r4,
-rc1*r5+r2*rc4};
\[ScriptP]cSIIAlt2={h*(ac*\[Xi]-cc*\[Beta]+kc*rc4+lc*rc5-kc*rc1-oc*rc2),
h*hc*(rc4-rc1)+\[Xi]*rc1-\[Beta]*rc4+r3*rc5-rc2*r6,
h*hc*(cc*l-ac*o)+\[Xi]*rc2-\[Beta]*rc5+rc3*rc4-rc1*rc6,
h^2*kc*(cc-ac),
h^2*(cc*lc-ac*oc),
2*h*(lc*rc4-oc*rc1),
h*(cc*rc3+kc*rc5-kc*rc2-ac*rc6),
hc*(l*rc4-k*rc2+k*rc5-o*rc1),
hc*k*(rc4-rc1),
hc*(l*rc5-o*rc2),
r3*rc4-rc1*r6,
rc3*rc5-rc2*rc6,
h*hc*(ac*c-a*cc)+rc1*r4-r1*rc4+rc2*r5-r2*rc5,
hc*(c*rc1-a*rc4),
hc*(c*rc2-a*rc5),
-h*(cc*r1-ac*r4),
-h*(cc*r2-ac*r5),
rc1*r5-r2*rc4,
-r1*rc5+rc2*r4};
\[Beta]\[Xi]subsSIIAlt2={\[Beta]->r1+rc1+r2+rc2+r3+rc3+h*(fc-hc)+hc*(f-h)
    ,\[Xi]->r4+rc4+r5+rc5+r6+rc6+h*(gc-hc)+hc*(g-h)};
auxeqnsSIIAlt2={a*kc-r1,ac*k-rc1,a*lc-r2,ac*l-rc2,k*lc-r3,kc*l-rc3,c*kc-r4
    ,cc*k-rc4,c*oc-r5,cc*o-rc5,k*oc-r6,kc*o-rc6};
Substitutions for SII, SIII, and WII
SIIsynthsubs={a->G-CC,
ac->Gc-CCc,
(*b->A-B+Q*(CC-A)-S*(DD-B),
bc->Ac-Bc+Qc*(CCc-Ac)-Sc*(DDc-Bc),*)
c->H-CC,
cc->Hc-CCc,
(*d->A-B+Q*(CC-A)-S*(F-B),
dc->Ac-Bc+Qc*(CCc-Ac)-Sc*(Fc-Bc),*)
f->G-DD,
fc->Gc-DDc,
g->H-F,
```

```
gc->Hc-Fc,
h->A-B,
hc->Ac-Bc,
k->CC-A,
kc->CCc-Ac,
l->-(DD-B),
lc->-(DDc-Bc),
m->A-B,
mc->Ac-Bc,
n->CC-A,
nc->CCc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
SIIIsynthsubs={a->G-DD,
ac->Gc-DDc,
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->H-DD,
cc->Hc-DDc,
(*d->A-B+Q*(DD-A)-S*(F-B),
dc->Ac-Bc+Qc*(DDc-Ac)-Sc*(Fc-Bc),*)
f->G-CC,
fc->Gc-CCc,
g->H-F,
gc->Hc-Fc,
h->A-CC,
hc->Ac-CCc,
k->DD-A,
kc->DDc-Ac,
l->0,
lc->0,
m->A-B,
mc->Ac-Bc,
n->DD-A,
nc->DDc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
WIIsynthsubs={a->-(G-CC),
ac->-(Gc-CCc),
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->-(H-CC),
cc->-(Hc-CCc),
(*d->B-CC+S*(F-B),
dc->Bc-CCc+Sc*(Fc-Bc),*)
f->(G-DD),
fc->(Gc-DDc),
g->(H-F),
gc->(Hc-Fc),
h->A-CC,
hc->Ac-CCc,
```

```
k->DD-A,
kc->DDc-Ac,
l->0,
lc->0,
m->B-CC,
mc->Bc-CCc,
n->0,
nc->0,
o->F-B,
oc->Fc-Bc};
Test Tasks
SIItasksubs={
Q1->Exp[I*20*Degree],S1->Exp[I*34.69248500*Degree],
Q2->Exp[I*40*Degree],S2->Exp[I*42.32954184*Degree],
Q3->Exp[I*60*Degree],S3->Exp[I*48.77116588*Degree],
Q4->Exp[I*80*Degree],S4->Exp[I*54.04707336*Degree],
Q5->Exp[I*100*Degree],S5->Exp[I*58.28925652*Degree],
Q6->Exp[I*120*Degree],S6->Exp[I*61.25960059*Degree],
Q7->Exp[I*140*Degree],S7->Exp[I*62.32035606*Degree],
Q8->Exp[I*160*Degree],S8->Exp[I*60.84473912*Degree],
Q9->Exp[I*180*Degree],S9->Exp[I*56.44096182*Degree],
Q10->Exp[I*200*Degree],S10->Exp[I*48.88334587*Degree]};
SIIsolnsubs={
A->0,
B->1,
CC->0+.27*I,
DD->.78+.3*I,
F->1.2+.47*I,
G->.75+.07*I,
H->.63+.56*I(*,
R1->Exp[I*-.98969140*Degree],
R2->Exp[I*.16333134*Degree],
R3->Exp[I*3.03571875*Degree],
R4->Exp[I*6.93101057*Degree],
R5->Exp[I*11.17319373*Degree],
R6->Exp[I*15.52415347*Degree],
R7->Exp[I*20.15414556*Degree],
R8->Exp[I*25.16256272*Degree],
R9->Exp[I*30.21915024*Degree],
R10->Exp[I*34.56376215*Degree]*)};
SIItasksubs=SIItasksubs~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S
   \[Conjugate]]]/.SIItasksubs);
SIIsolnsubs=SIIsolnsubs~Join~(Join[{Ac->A\[Conjugate],Bc->B\[Conjugate],
   CCc->CC\[Conjugate],DDc->DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[
   Conjugate],Hc->H\[Conjugate]}(*,Thread[Rc->R\[Conjugate]]*)]/.
   SIIsolnsubs);
SIIItasksubs={
Q1->Exp[I*25*Degree],S1->Exp[I*14.76476953*Degree],
Q2->Exp[I*50*Degree],S2->Exp[I*31.08130612*Degree],
Q3->Exp[I*75*Degree],S3->Exp[I*46.07726706*Degree],
Q4->Exp[I*100*Degree],S4->Exp[I*59.89540592*Degree],
```

```
Q5->Exp[I*125*Degree],S5->Exp[I*73.70852150*Degree],
Q6->Exp[I*150*Degree],S6->Exp[I*87.23594859*Degree],
Q7->Exp[I*175*Degree],S7->Exp[I*97.28337403*Degree],
Q8->Exp[I*200*Degree],S8->Exp[I*103.91657141*Degree],
Q9->Exp[I*225*Degree],S9->Exp[I*109.97999370*Degree],
Q10->Exp[I*250*Degree],S10->Exp[I*117.89152259*Degree]};
SIIIsolnsubs={
A->1.21412169+1.27756077*I,
B->3.32317282+1.6897844*I,
CC->1.57841234+1.53639886*I,
DD->0.92536993+2.3859124*I,
F->3.36307665+4.6000394*I,
G->1.98424242+1.92615158*I,
H->2.46131102+3.66734059*I};
SIIItasksubs=SIIItasksubs~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->
    S\[Conjugate]]]/.SIIItasksubs);
SIIIsolnsubs=SIIIsolnsubs~Join~(Join[{Ac->A\[Conjugate],Bc->B\[Conjugate],
    CCc->CC\[Conjugate],DDc->DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[
    Conjugate],Hc->H\[Conjugate]}]/.SIIIsolnsubs);
WIItasksubs={
Q1->Exp[I*35*Degree],S1->Exp[I*68.93766307*Degree],
Q2->Exp[I*70*Degree],S2->Exp[I*101.61252528*Degree],
Q3->Exp[I*105*Degree],S3->Exp[I*118.70961619*Degree],
Q4->Exp[I*140*Degree],S4->Exp[I*123.91020834*Degree],
Q5->Exp[I*175*Degree],S5->Exp[I*124.19148235*Degree],
Q6->Exp[I*210*Degree],S6->Exp[I*124.17373651*Degree],
Q7->Exp[I*245*Degree],S7->Exp[I*123.86766508*Degree],
Q8->Exp[I*280*Degree],S8->Exp[I*120.79823078*Degree]};
WIIsolnsubs={
A->1.96338562+0.93536854*I,
B->4.53401907+0.70732847*I,
CC->3.49747333+1.59875782*I,
DD->2.6599402+2.37199304*I,
F->5.4179342-0.08670631*I,
G->4.63279604+2.43292296*I,
H->4.63279604+2.43292296*I};
WIItasksubs=WIItasksubs~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S
    \[Conjugate]]]/.WIItasksubs);
WIIsolnsubs=WIIsolnsubs~Join~(Join[{Ac->A\[Conjugate],Bc->B\[Conjugate],
    CCc->CC\[Conjugate],DDc->DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[
    Conjugate],Hc->H\[Conjugate]}]/.WIIsolnsubs);
Miscellaneous
mHomogCount[F_,vars_]:=Module[{n,m,(*degrees,\[Alpha],poly,mono*)},
n=Length[F];
m=Length[vars];
ctr=0;
SetSharedVariable[ctr];
Print[Dynamic[ctr]];
Print[ProgressIndicator[Dynamic[ctr],{0,n*m}]];
degrees=Table[{Max[Map[Total,CoefficientRules[eqn,set][[All,1]]]],ctr
    ++}[[1]],{eqn,F},{set,vars}];
```

```
(*degrees indexed
        eqn no
                variable group no*)
\[Alpha]=Table[ToExpression["\[Alpha]"<>ToString[j]],{j,m}];
poly=Product[Sum[degrees[[i,j]]*\[Alpha][[j]],{j,m}],{i,n}];
mono=Product[\[Alpha][[j]]^Length[vars[[j]]],{j,m}];
Coefficient[poly,mono]]
Synthesis Equations
(*Synthesis Equations from general form*)
SIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIsynthsubs),{j,npos-1}];
SIIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIIsynthsubs),{j,npos-1}];
WIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.WIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.WIIsynthsubs),{j,npos-3}];
SIIeqns1/.SIItasksubs/.SIIsolnsubs
SIIIeqns1/.SIIItasksubs/.SIIIsolnsubs
WIIeqns1/.WIItasksubs/.WIIsolnsubs
(*Synthesis equations from compact form*)
SIIeqns2=eqns4/.SIIsynthsubs;
SIIIeqns2=eqns4/.SIIIsynthsubs;
WIIeqns2=eqns4[[1;;8]]/.WIIsynthsubs;
SIIeqns2/.SIItasksubs/.SIIsolnsubs
SIIIeqns2/.SIIItasksubs/.SIIIsolnsubs
WIIeqns2/.WIItasksubs/.WIIsolnsubs
(*Stephenson II with unknowns {CC,CCc,DD,DDc,F,Fc,G,Gc,H,Hc}*)
SIIeqns3=Table[(\[ScriptP]SII/.\[Beta]\[Xi]subsSII/.SIIsynthsubs).QmatBIG
   [[j]].(\[ScriptP]cSII/.\[Beta]\[Xi]subsSII/.SIIsynthsubs),{j,npos-1}];
SIIeqns3/.SIItasksubs/.SIIsolnsubs
(*Stephenson II with unknowns {A||B||CC,Ac||Bc||CCc,DD,DDc,F,Fc,G,Gc,H,Hc,
   r1,rc1,r2,rc2}*)
SIIeqns3Alt1a=Table[(\[ScriptP]SIIAlt1/.\[Beta]\[Xi]subsSIIAlt1/.
   SIIsynthsubs).QmatBIG[[j]].(\[ScriptP]cSIIAlt1/.\[Beta]\[Xi]subsSIIAlt1
   /.SIIsynthsubs),{j,npos-1}];
SIIeqns3Alt1b=auxeqnsSIIAlt1/.SIIsynthsubs;
SIIeqns3Alt1=Join[SIIeqns3Alt1a,SIIeqns3Alt1b];
SIIeqns3Alt1/.{r1->a*lc,rc1->ac*l,r2->c*oc,rc2->cc*o}/.SIIsynthsubs/.
   SIItasksubs/.SIIsolnsubs
(*Stephenson II with unknowns {CC,CCc,DD,DDc,F,Fc,G,Gc,H,Hc,r1,rc1,r2,rc2,
   r3,rc3,r4,rc4,r5,rc5,r6,rc6}*)
SIIeqns3Alt2a=Table[(\[ScriptP]SIIAlt2/.\[Beta]\[Xi]subsSIIAlt2/.
   SIIsynthsubs).QmatBIG[[j]].(\[ScriptP]cSIIAlt2/.\[Beta]\[Xi]subsSIIAlt2
   /.SIIsynthsubs),{j,npos-1}];
SIIeqns3Alt2b=auxeqnsSIIAlt2/.SIIsynthsubs;
SIIeqns3Alt2=Join[SIIeqns3Alt2a,SIIeqns3Alt2b];
SIIeqns3Alt2/.{r1->a*kc,rc1->ac*k,r2->a*lc,rc2->ac*l,r3->k*lc,rc3->kc*l,r4
   ->c*kc,rc4->cc*k,r5->c*oc,rc5->cc*o,r6->k*oc,rc6->kc*o}/.SIIsynthsubs/.
   SIItasksubs/.SIIsolnsubs
SIIIeqns3a=Table[(\[ScriptP]SIII/.\[Beta]\[Xi]subsSIII/.SIIIsynthsubs).
   QmatBIG[[j]].(\[ScriptP]cSIII/.\[Beta]\[Xi]subsSIII/.SIIIsynthsubs),{j,
   npos-1}];
```

```
SIIIeqns3b=auxeqnsSIII/.SIIIsynthsubs;
SIIIeqns3=Join[SIIIeqns3a,SIIIeqns3b];
SIIIeqns3/.{r1->a*hc,rc1->ac*h,r2->c*mc,rc2->cc*m,r3->c*oc,rc3->cc*o,r4->m
   *oc,rc4->mc*o}/.SIIIsynthsubs/.SIIItasksubs/.SIIIsolnsubs
WIIeqns3a=Table[(\[ScriptP]WII/.\[Beta]\[Xi]subsWII/.WIIsynthsubs).QmatBIG
   [[j]].(\[ScriptP]cWII/.\[Beta]\[Xi]subsWII/.WIIsynthsubs),{j,npos-3}];
WIIeqns3b=auxeqnsWII/.WIIsynthsubs;
WIIeqns3=Join[WIIeqns3a,WIIeqns3b];
WIIeqns3/.{r1->h*kc,rc1->hc*k,r2->m*oc,rc2->mc*o}/.WIIsynthsubs/.
   WIItasksubs/.WIIsolnsubs
SIIIeqns4a=Table[((r1+a*kc*Qc[[j]])*(r2+rc2+r3+rc3+r4+rc4+k*(gc-kc)+kc*(g-
   k)-r4*Sc[[j]]-rc4*S[[j]]-k*Q[[j]]*(mc+Sc[[j]]*oc)-kc*Qc[[j]]*(m+S[[j]]*
   o))
-(r2+c*kc*Qc[[j]]+r3*Sc[[j]])*(r1+rc1+k*(fc-kc)+kc*(f-k)-h*kc*Qc[[j]]-hc*k
   *Q[[j]]))
*((rc1+ac*k*Q[[j]])*(r2+rc2+r3+rc3+r4+rc4+k*(gc-kc)+kc*(g-k)-r4*Sc[[j]]-
   rc4*S[[j]]-k*Q[[j]]*(mc+Sc[[j]]*oc)-kc*Qc[[j]]*(m+S[[j]]*o))
-(rc2+cc*k*Q[[j]]+rc3*S[[j]])*(r1+rc1+k*(fc-kc)+kc*(f-k)-h*kc*Qc[[j]]-hc*k
   *Q[[j]]))
+((r1+a*kc*Qc[[j]])*(rc2+cc*k*Q[[j]]+rc3*S[[j]])-(rc1+ac*k*Q[[j]])*(r2+c*
   kc*Qc[[j]]+r3*Sc[[j]]))^2,{j,npos-1}];
SIIIeqns4=Join[SIIIeqns4a,auxeqnsSIII]/.SIIIsynthsubs;
SIIIeqns4/.{r1->a*hc,rc1->ac*h,r2->c*mc,rc2->cc*m,r3->c*oc,rc3->cc*o,r4->m
   *oc,rc4->mc*o}/.SIIIsynthsubs/.SIIItasksubs/.SIIIsolnsubs
Root Counts- 11 positions
mHomogCount[SIIeqns3,{{CC,DD,F,G,H},{CCc,DDc,Fc,Gc,Hc}}]
mHomogCount[SIIeqns3Alt1,{{DD,DDc,G,Gc,r1,rc1},{F,Fc,H,Hc,r2,rc2},{A},{Ac
   }}]
mHomogCount[SIIIeqns4,{{CC,CCc,G,Gc,r1,rc1},{B,Bc,F,Fc,H,Hc,r2,rc2,r3,rc3,
   r4,rc4}}]
mHomogCount[Join[SIIIeqns3a/.{r2->c*mc,rc2->cc*m,r3->c*oc,rc3->cc*o}/.
   SIIIsynthsubs,SIIIeqns3b[[{1,2,7,8}]]],{{CC,CCc,G,Gc,r1,rc1},{B,Bc,F,Fc
   ,r4,rc4},{H,Hc}}]
Root Counts- 9 positions
mHomogCount[Join[SIIeqns3Alt1a[[1;;8]],SIIeqns3Alt1b],{{DD,DDc,G,Gc,r1,rc1
   },{F,Fc,H,Hc,r2,rc2}}]
mHomogCount[Join[SIIIeqns3a[[1;;8]],SIIIeqns3b],{{CC,CCc,G,Gc,r1,rc1},{F,
   Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4}}]
mHomogCount[WIIeqns3,{{A,Ac,DD,DDc,r1,rc1},{B,Bc,F,Fc,r2,rc2}}]
Stephenson II 11 Position Bertini Export
p=Table[ToExpression["p"<>ToString[i]],{i,19}];
pc=Table[ToExpression["pc"<>ToString[i]],{i,19}];
additionalEqns1=MapThread[#1<>"␣=␣"<>ToString[InputForm[#2]]<>";"&,{{"bet"
   ,"xi"},{\[Beta],\[Xi]}/.\[Beta]\[Xi]subsSII/.SIIsynthsubs/.{A->0,Ac->0,
   B->1,Bc->1}}];
additionalEqns2=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>";
   "&,{p,\[ScriptP]SII/.{\[Beta]->bet,\[Xi]->xi}/.SIIsynthsubs/.{A->0,Ac
   ->0,B->1,Bc->1}}];
additionalEqns2c=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>"
   ;"&,{pc,\[ScriptP]cSII/.{\[Beta]->bet,\[Xi]->xi}/.SIIsynthsubs}/.{A->0,
   Ac->0,DD->1,DDc->1}];
```

```
additionalEqns=Join[additionalEqns1,additionalEqns2,additionalEqns2c];
mainEqns=Table[p.QmatBIG[[j]].pc,{j,npos-1}];
\[ScriptCapitalF]=mainEqns;
(*vars={{CC,DD,F,G,H},{CCc,DDc,Fc,Gc,Hc}};*)
vars={{CC,CCc,DD,DDc,F,Fc,G,Gc,H,Hc}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY: 2;",
"SECURITYLEVEL: 0;",
"SECURITYMAXNORM: 1e20;",
"PathTruncationThreshold: 1e20;",
"MPTYPE: 0;",
"CONDNUMTHRESHOLD: 1e20;",
"PrintPathProgress: 100000;",
"UseRegeneration: 0;",
"END;"};
variableGroups={Map["variable_group "<>StringTake[ToString[#],{2,-2}]<>";"
   &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random "<>StringTake[ToString[Join[Q,S]],{2,-2}]<>", A, Ac
   , B, Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
   Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>" =1/"<>ToString[pairs
   [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}},{j,npos-1}]];
function={"function "<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
   ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter "<>StringTake[ToString[Join[Q,S]],{2,-2}]<>(*", A,
   Ac, DD, DDc"<>*)";"};
equations=Table["f"<>ToString[i]<>" ="<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
   NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^"~~a:(Shortest[___]~~NumberString):>"
   e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
   ,*)parameter,extraISOeqns,additionalEqns,equations,{"END;"}];
Column[inputBertini];
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar
   Synthesis\\Function\\Stephenson III"];
Export["something.txt",inputBertini]
Stephenson II 9 Position Bertini Export
p=Table[ToExpression["p"<>ToString[i]],{i,19}];
pc=Table[ToExpression["pc"<>ToString[i]],{i,19}];
additionalEqns1=MapThread[#1<>" ="<>ToString[InputForm[#2]]<>";"&,{{"bet"
   ,"xi"},{\[Beta],\[Xi]}/.\[Beta]\[Xi]subsSIIAlt1/.SIIsynthsubs/.{A->0,Ac
   ->0,B->1,Bc->1}}];
additionalEqns2=MapThread[ToString[#1]<>" ="<>ToString[InputForm[#2]]<>";
```

```
    "&,{p,\[ScriptP]SIIAlt1/.{\[Beta]->bet,\[Xi]->xi}/.SIIsynthsubs/.{A->0,
    Ac->0,B->1,Bc->1}}];
additionalEqns2c=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>"
    ;"&,{pc,\[ScriptP]cSIIAlt1/.{\[Beta]->bet,\[Xi]->xi}/.SIIsynthsubs}/.{A
    ->0,Ac->0,B->1,Bc->1}];
additionalEqns=Join[additionalEqns1,additionalEqns2,additionalEqns2c];
mainEqns=Table[p.QmatBIG[[j]].pc,{j,npos-3}];
\[ScriptCapitalF]=Join[mainEqns,SIIeqns3Alt1b/.{A->0,Ac->0,B->1,Bc->1}];
vars={{DD,DDc,G,Gc,r1,rc1},{F,Fc,H,Hc,r2,rc2}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY:␣1;",
"SECURITYLEVEL:␣0;",
"SECURITYMAXNORM:␣1e20;",
"PathTruncationThreshold:␣1e20;",
"MPTYPE:␣2;",
"CONDNUMTHRESHOLD:␣1e20;",
"PrintPathProgress:␣10000;",
"UseRegeneration:␣0;",
"END;"};
variableGroups={Map["variable_group␣"<>StringTake[ToString[#],{2,-2}]<>";"
    &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>",␣A,␣Ac
    ,␣B,␣Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
    Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>"␣=␣1/"<>ToString[pairs
    [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}},{j,npos-3}]];
function={"function␣"<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
    ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter␣"<>StringTake[ToString[Join[Q[[1;;8]],S
    [[1;;8]]]],{2,-2}]<>",␣CC,␣CCc"<>";"};
equations=Table["f"<>ToString[i]<>"␣=␣"<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
    NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^""~~a:(Shortest[___]~~NumberString):>"
    e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
    ,*)parameter,extraISOeqns,additionalEqns,equations,{"END;"}];
Column[inputBertini];
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
    Synthesis\\Function\\Stephenson␣II\\Homotopies\\9␣Position\\141009
    parameterhomotopy"];
Export["funcgenSII.txt",inputBertini]
funcgenSII.txt
Stephenson III 11 Position Bertini Export
```

```
p=Table[ToExpression["p"<>ToString[i]],{i,19}];
pc=Table[ToExpression["pc"<>ToString[i]],{i,19}];
additionalEqns1=MapThread[#1<>"␣=␣"<>ToString[InputForm[#2]]<>";"&,{{"bet"
    ,"xi"},{\[Beta],\[Xi]}/.\[Beta]\[Xi]subsSIII/.SIIIsynthsubs/.{A->0,Ac
    ->0,DD->1,DDc->1}}];
additionalEqns2=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>";
    "&,{p,\[ScriptP]SIII/.{\[Beta]->bet,\[Xi]->xi}/.SIIIsynthsubs/.{A->0,Ac
    ->0,DD->1,DDc->1}}];
additionalEqns2c=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>"
    ;"&,{pc,\[ScriptP]cSIII/.{\[Beta]->bet,\[Xi]->xi}/.SIIIsynthsubs}/.{A
    ->0,Ac->0,DD->1,DDc->1}];
additionalEqns=Join[additionalEqns1,additionalEqns2,additionalEqns2c];
mainEqns=Table[p.QmatBIG[[j]].pc,{j,npos-1}]/.{p10->0,p12->0,pc10->0,pc12
    ->0};
\[ScriptCapitalF]=Join[mainEqns,SIIIeqns3b/.{A->0,Ac->0,DD->1,DDc->1}];
vars={{CC,CCc,G,Gc,r1,rc1},{B,Bc,F,Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY:␣1;",
"SECURITYLEVEL:␣0;",
"SECURITYMAXNORM:␣1e20;",
"PathTruncationThreshold:␣1e20;",
"MPTYPE:␣0;",
"CONDNUMTHRESHOLD:␣1e20;",
"PrintPathProgress:␣100000;",
"UseRegeneration:␣0;",
"END;"};
variableGroups={Map["variable_group␣"<>StringTake[ToString[#],{2,-2}]<>";"
    &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>",␣A,␣Ac
    ,␣B,␣Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
    Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>"␣=␣1/"<>ToString[pairs
    [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}},{j,npos-1}]];
function={"function␣"<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
    ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>(*", A,
    Ac, DD, DDc"<>*)";"};
equations=Table["f"<>ToString[i]<>"␣=␣"<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
    NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^"~~a:(Shortest[___]~~NumberString):>"
    e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
    ,*)parameter,extraISOeqns,additionalEqns,equations,{"END;"}];
```

```
Column[inputBertini];
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
   Synthesis\\Function\\Stephenson␣III"];
Export["funcgenstephIII2.txt",inputBertini]
funcgenstephIII2.txt
Stephenson III 9 Position Bertini Export
p=Table[ToExpression["p"<>ToString[i]],{i,19}];
pc=Table[ToExpression["pc"<>ToString[i]],{i,19}];
additionalEqns1=MapThread[#1<>"␣=␣"<>ToString[InputForm[#2]]<>";"&,{{"bet"
   ,"xi"},{\[Beta],\[Xi]}/.\[Beta]\[Xi]subsSIII/.SIIIsynthsubs/.{A->0,Ac
   ->0,DD->1,DDc->1}}];
additionalEqns2=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>";
   "&,{p,\[ScriptP]SIII/.{\[Beta]->bet,\[Xi]->xi}/.SIIIsynthsubs/.{A->0,Ac
   ->0,DD->1,DDc->1}}];
additionalEqns2c=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>"
   ;"&,{pc,\[ScriptP]cSIII/.{\[Beta]->bet,\[Xi]->xi}/.SIIIsynthsubs}/.{A
   ->0,Ac->0,DD->1,DDc->1}];
additionalEqns=Join[additionalEqns1,additionalEqns2,additionalEqns2c];
mainEqns=Table[p.QmatBIG[[j]].pc,{j,npos-3}]/.{p10->0,p12->0,pc10->0,pc12
   ->0};
\[ScriptCapitalF]=Join[mainEqns,SIIIeqns3b/.{A->0,Ac->0,DD->1,DDc->1}];
vars={{CC,CCc,G,Gc,r1,rc1},{F,Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY:␣1;",
"SECURITYLEVEL:␣0;",
"SECURITYMAXNORM:␣1e20;",
"PathTruncationThreshold:␣1e20;",
"MPTYPE:␣2;",
"CONDNUMTHRESHOLD:␣1e20;",
"PrintPathProgress:␣10000;",
"UseRegeneration:␣0;",
"END;"};
variableGroups={Map["variable_group␣"<>StringTake[ToString[#],{2,-2}]<>";"
   &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>",␣A,␣Ac
   ,␣B,␣Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
   Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>"␣=␣1/"<>ToString[pairs
   [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}},{j,npos-3}]];
function={"function␣"<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
   ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter␣"<>StringTake[ToString[Join[Q[[1;;8]],S
   [[1;;8]]]],{2,-2}]<>",␣B,␣Bc"<>";"};
equations=Table["f"<>ToString[i]<>"␣=␣"<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
```

```
   NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^"~~a:(Shortest[___]~~NumberString):>"
   e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
   ,*)parameter,extraISOeqns,additionalEqns,equations,{"END;"}];
Column[inputBertini];
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
   Synthesis\\Function\\Stephenson␣III\\Homotopies\\9␣Position\\141009
   parameterhomotopy"];
Export["funcgenSIII.txt",inputBertini]
funcgenSIII.txt
Watt II 9 Position Bertini Export
p=Table[ToExpression["p"<>ToString[i]],{i,19}];
pc=Table[ToExpression["pc"<>ToString[i]],{i,19}];
additionalEqns1=MapThread[#1<>"␣=␣"<>ToString[InputForm[#2]]<>";"&,{{"bet"
   ,"xi"},{\[Beta],\[Xi]}/.\[Beta]\[Xi]subsWII/.WIIsynthsubs/.{CC->0,CCc
   ->0,G->1,Gc->1}}];
additionalEqns2=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>";
   "&,{p,\[ScriptP]WII/.{\[Beta]->bet,\[Xi]->xi}/.WIIsynthsubs/.{CC->0,CCc
   ->0,G->1,Gc->1}}];
additionalEqns2c=MapThread[ToString[#1]<>"␣=␣"<>ToString[InputForm[#2]]<>"
   ;"&,{pc,\[ScriptP]cWII/.{\[Beta]->bet,\[Xi]->xi}/.WIIsynthsubs/.{CC->0,
   CCc->0,G->1,Gc->1}}];
additionalEqns=Join[additionalEqns1,additionalEqns2,additionalEqns2c];
mainEqns=Table[p.QmatBIG[[j]].pc,{j,npos-3}]/.{p9->0,p10->0,p11->0,p12->0,
   pc9->0,pc10->0,pc11->0,pc12->0};
\[ScriptCapitalF]=Join[mainEqns,WIIeqns3b/.{CC->0,CCc->0,G->1,Gc->1}];
vars={{A,Ac,DD,DDc,r1,rc1},{B,Bc,F,Fc,r2,rc2}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY:␣1;",
"SECURITYLEVEL:␣0;",
"SECURITYMAXNORM:␣1e20;",
"PathTruncationThreshold:␣1e20;",
"MPTYPE:␣2;",
"CONDNUMTHRESHOLD:␣1e20;",
"PrintPathProgress:␣10000;",
"UseRegeneration:␣0;",
"END;"};
variableGroups={Map["variable_group␣"<>StringTake[ToString[#],{2,-2}]<>";"
   &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>",␣A,␣Ac
   ,␣B,␣Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
   Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>"␣=␣1/"<>ToString[pairs
```

```
    [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}[[All,All,1;;8]]},{j,npos-3}]];
function={"function␣"<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
    ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter␣"<>StringTake[ToString[Join[Q[[1;;8]],S
    [[1;;8]]]],{2,-2}]<>(*", CC, CCc, G, Gc"<>*)";"};
equations=Table["f"<>ToString[i]<>"␣=␣"<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
    NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^"~~a:(Shortest[___]~~NumberString):>"
    e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
    ,*)parameter,extraISOeqns,additionalEqns,equations,{"END;"}];
Column[inputBertini];
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
    Synthesis\\Function\\Watt␣II\\Homotopies\\9␣Position\\141009
    parameterhomotopy"];
Export["funcgenWII.txt",inputBertini]
funcgenWII.txt
Stephenson III 11 Position Bertini Export -NEW 3/13/15
\[ScriptCapitalF]=SIIIeqns4/.{A->0,Ac->0,DD->1,DDc->1};
vars={{CC,CCc,G,Gc,r1,rc1},{B,Bc,F,Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4}};
n\[ScriptCapitalF]=Length[\[ScriptCapitalF]];
fvars=Flatten[vars];
(*params=Join[Rest[P],Rest[Pc],Rest[T5](*,{A,Ac,B,Bc}*)];
paramSubs=Thread[params->(params/.tasks)];*)
config={"CONFIG",
"PARAMETERHOMOTOPY:␣2;",
"SECURITYLEVEL:␣0;",
"SECURITYMAXNORM:␣1e20;",
"PathTruncationThreshold:␣1e20;",
"MPTYPE:␣0;",
"CONDNUMTHRESHOLD:␣1e20;",
"PrintPathProgress:␣10000;",
"UseRegeneration:␣0;",
"END;"};
variableGroups={Map["variable_group␣"<>StringTake[ToString[#],{2,-2}]<>";"
    &,vars,{Depth[vars]-2}]};
variableGroups=Flatten[variableGroups];
randomnumbers={"random␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>",␣A,␣Ac
    ,␣B,␣Bc"<>";"};
(*extraISOeqns=MapThread[ToString[#2]<>" = 1/"<>ToString[#1]<>";"&,Rest/@{
    Q,Qc,S,Sc}];*)
extraISOeqns=Flatten[Table[ToString[pairs[[2,j]]]<>"␣=␣1/"<>ToString[pairs
    [[1,j]]]<>";",{pairs,{{Q,Qc},{S,Sc}}},{j,npos-1}]];
function={"function␣"<>StringTake[ToString[Table["f"<>ToString[i],{i,n\[
    ScriptCapitalF]}]],{2,-2}]<>";"};
parameter={"parameter␣"<>StringTake[ToString[Join[Q,S]],{2,-2}]<>(*", A,
    Ac, DD, DDc"<>*)";"};
equations=Table["f"<>ToString[i]<>"␣=␣"<>
ToString[InputForm[Chop[SetPrecision[\[ScriptCapitalF][[i]],50]],
```

```
    NumberMarks->False]]
<>";",{i,n\[ScriptCapitalF]}];
equations=StringReplace[equations,"*^"~~a:(Shortest[___]~~NumberString):>"
    e"<>ToString[a]];
inputBertini=Join[config,{"INPUT"},variableGroups,function,(*randomnumbers
    ,*)parameter,extraISOeqns,(*additionalEqns,*)equations,{"END;"}];
Column[inputBertini]
SetDirectory["C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
    Synthesis\\Function\\Stephenson␣III"];
Export["funcgenstephIII3.txt",inputBertini]
funcgenstephIII3.txt
```

# C Mathematica code: Function Generation Specification

```
General Synthesis Equations
(*Synthesis equations take this form*)
eqns1=(a*bc*(g*gc-c*cc-d*dc)-c*dc*(f*fc-a*ac-b*bc))*(ac*b*(g*gc-c*cc-d*dc)
    -cc*d*(f*fc-a*ac-b*bc))+(a*bc*cc*d-ac*b*c*dc)^2;
(*Here is another way to write those equations with dot products*)
eqns2={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c}.{bc,-dc,-bc*d*dc,b*bc*dc}*{b,-d,-b
    *d*dc,b*bc*d}.{ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc}-{a*cc,ac*c}.{bc*d,-
    b*dc}*{b*dc,-bc*d}.{ac*c,a*cc};
Expand[eqns1]==Expand[eqns2]
True
(*b=Bmat.q*)
Bmat={{h,k,l,0,0,0,0,0,0,0,0,0},{-m,-n,-o,0,0,0,0,0,0,0,0,0},
{-m*(k*nc+l*oc)-h*(m*mc+n*nc+o*oc),-n*(h*mc+l*oc)-k*(m*mc+n*nc+o*oc),-o*(h
    *mc+k*nc)-l*(m*mc+n*nc+o*oc),-h*m*nc,-h*m*oc,-oc*(h*n+k*m),-nc*(h*o+l*m
    ),-mc*(k*o+l*n),-k*n*mc,-l*o*mc,-k*n*oc,-l*o*nc},
{h*(kc*n+lc*o)+m*(h*hc+k*kc+l*lc),k*(hc*m+lc*o)+n*(h*hc+k*kc+l*lc),l*(hc*m
    +kc*n)+o*(h*hc+k*kc+l*lc),h*m*kc,h*m*lc,lc*(h*n+k*m),kc*(h*o+l*m),hc*(k
    *o+l*n),k*n*hc,l*o*hc,k*n*lc,l*o*kc}};
Bcmat={{hc,kc,lc,0,0,0,0,0,0,0,0,0},{-mc,-nc,-oc,0,0,0,0,0,0,0,0,0},
{-mc*(kc*n+lc*o)-hc*(m*mc+n*nc+o*oc),-nc*(hc*m+lc*o)-kc*(m*mc+n*nc+o*oc),-
    oc*(hc*m+kc*n)-lc*(m*mc+n*nc+o*oc),-hc*mc*n,-hc*mc*o,-o*(hc*nc+kc*mc),-
    n*(hc*oc+lc*mc),-m*(kc*oc+lc*nc),-kc*nc*m,-lc*oc*m,-kc*nc*o,-lc*oc*n},
{hc*(k*nc+l*oc)+mc*(h*hc+k*kc+l*lc),kc*(h*mc+l*oc)+nc*(h*hc+k*kc+l*lc),lc
    *(h*mc+k*nc)+oc*(h*hc+k*kc+l*lc),hc*mc*k,hc*mc*l,l*(hc*nc+kc*mc),k*(hc*
    oc+lc*mc),h*(kc*oc+lc*nc),kc*nc*h,lc*oc*h,kc*nc*l,lc*oc*k}};
(*d=Dmat.q'*)
Dmat={{h*mc+k*nc+l*oc,k*mc,l*mc,h*nc,h*oc,k*oc,l*nc},{-(hc*m+kc*n+lc*o),-
    hc*n,-hc*o,-kc*m,-lc*m,-lc*n,-kc*o}};
Dcmat={{hc*m+kc*n+lc*o,kc*m,lc*m,hc*n,hc*o,kc*o,lc*n},{-(h*mc+k*nc+l*oc),-
    h*nc,-h*oc,-k*mc,-l*mc,-l*nc,-k*oc}};
avect={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c};
aCvect={ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc};
cvect={a*cc,ac*c};
cCvect={ac*c,a*cc};
(*BDmat={{Bmat,0},{0,Dmat}}*)
BDmat=Table[0,{6},{19}];
BDmat[[1;;4,1;;12]]=Bmat;
BDmat[[5;;6,13;;19]]=Dmat;
BcDcmat=Table[0,{6},{19}];
BcDcmat[[1;;4,1;;12]]=Bcmat;
BcDcmat[[5;;6,13;;19]]=Dcmat;
(*acvect={a,c}*)
acvect=Table[,{6}];
acvect[[1;;4]]=avect;
acvect[[5;;6]]=cvect;
aCcCvect=Table[,{6}];
aCcCvect[[1;;4]]=aCvect;
aCcCvect[[5;;6]]=cCvect;
\[ScriptP]=Transpose[BcDcmat].acvect;
```

```
\[ScriptP]c=Transpose[BDmat].aCcCvect;
\[ScriptP]={hc*(a*\[Xi]+c*(k*nc+l*oc))-mc*(c*\[Beta]+a*(kc*n+lc*o)),
kc*(a*\[Xi]+c*(h*mc+l*oc))-nc*(c*\[Beta]+a*(hc*m+lc*o)),
lc*(a*\[Xi]+c*(h*mc+k*nc))-oc*(c*\[Beta]+a*(hc*m+kc*n)),
hc*mc*(c*k-a*n),
hc*mc*(c*l-a*o),
(kc*mc+hc*nc)*(c*l-a*o),
(lc*mc+hc*oc)*(c*k-a*n),
(lc*nc+kc*oc)*(c*h-a*m),
kc*nc*(c*h-a*m),
lc*oc*(c*h-a*m),
kc*nc*(c*l-a*o),
lc*oc*(c*k-a*n),
a*cc*(hc*m+kc*n+lc*o)-ac*c*(h*mc+k*nc+l*oc),
a*cc*kc*m-ac*c*h*nc,
a*cc*lc*m-ac*c*h*oc,
-ac*c*k*mc+a*cc*hc*n,
-ac*c*l*mc+a*cc*hc*o,
a*cc*kc*o-ac*c*l*nc,
-ac*c*k*oc+a*cc*lc*n};
\[ScriptP]c={h*(ac*\[Xi]+cc*(kc*n+lc*o))-m*(cc*\[Beta]+ac*(k*nc+l*oc)),
k*(ac*\[Xi]+cc*(hc*m+lc*o))-n*(cc*\[Beta]+ac*(h*mc+l*oc)),
l*(ac*\[Xi]+cc*(hc*m+kc*n))-o*(cc*\[Beta]+ac*(h*mc+k*nc)),
h*m*(cc*kc-ac*nc),
h*m*(cc*lc-ac*oc),
(k*m+h*n)*(cc*lc-ac*oc),
(l*m+h*o)*(cc*kc-ac*nc),
(l*n+k*o)*(cc*hc-ac*mc),
k*n*(cc*hc-ac*mc),
l*o*(cc*hc-ac*mc),
k*n*(cc*lc-ac*oc),
l*o*(cc*kc-ac*nc),
ac*c*(h*mc+k*nc+l*oc)-a*cc*(hc*m+kc*n+lc*o),
ac*c*k*mc-a*cc*hc*n,
ac*c*l*mc-a*cc*hc*o,
-a*cc*kc*m+ac*c*h*nc,
-a*cc*lc*m+ac*c*h*oc,
ac*c*k*oc-a*cc*lc*n,
-a*cc*kc*o+ac*c*l*nc};
\[Beta]\[Xi]subs={\[Beta]->f*fc-a*ac-h*hc-k*kc-l*lc,\[Xi]->g*gc-c*cc-m*mc-
   n*nc-o*oc};
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
   i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
qvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]],Q[[j]]*S[[j]],Q[[j]]^2,S[[j]]^2,Q[[j]]^2*Sc[[j]],Qc[[j]]*S[[j]]^2},{
```

```
   j,npos-1}];
qpvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]]},{j,npos-1}];
qcvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]],Qc[[j]]*Sc[[j]],Qc[[j]]^2,Sc[[j]]^2,Qc[[j]]^2*S[[j]],Q[[j]]*Sc[[j
   ]]^2},{j,npos-1}];
qcpvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]]},{j,npos-1}];
isoSimplify[expression_,isopairs_]:=Module[{isoSubs,expression2},
expression2=expression;
isoSubs=Flatten[Table[{exp=i+1-j,expc=j}->{If[exp-expc>0,exp-expc,0],If[
   expc-exp>0,expc-exp,0]},{i,5},{j,Range[i]}]];
Do[
expression2=CoefficientRules[expression2,isopairs[[i]]];
expression2=expression2/.isoSubs;
expression2=FromCoefficientRules[expression2,isopairs[[i]]];
,{i,Length[isopairs]}];
expression2]
(*Qmat=qcvect.qvect\[Transpose]*)
Qmat=Table[Outer[Times,qcvect[[j]],qvect[[j]]],{j,npos-1}];
Qmat=isoSimplify[Qmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
Qpmat=Table[Outer[Times,qcpvect[[j]],qpvect[[j]]],{j,npos-1}];
Qpmat=isoSimplify[Qpmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
(*QmatBIG={{Qmat,0},{0,-Qpmat}}*)
QmatBIG=Table[0,{npos-1},{19},{19}];
Do[
QmatBIG[[j,1;;12,1;;12]]=Qmat[[j]];
QmatBIG[[j,13;;19,13;;19]]=-Qpmat[[j]];
,{j,npos-1}];
eqns3=Table[\[ScriptP].QmatBIG[[j]].\[ScriptP]c,{j,npos-1}];
(*The final product is p, pc, and QmatBIG[[j]]*)
eqns4=Table[(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)
   *(mc+Qc[[j]]*nc+Sc[[j]]*oc))-c*(mc+Qc[[j]]*nc+Sc[[j]]*oc)*(f*fc-a*ac-(h
   +Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc[[j]]*lc)))*(ac*(h+Q[[j]]*k+S[[j
   ]]*l)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)*(mc+Qc[[j]]*nc+Sc[[j]]*oc))-cc*(
   m+Q[[j]]*n+S[[j]]*o)*(f*fc-a*ac-(h+Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc
   [[j]]*lc)))+(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*cc*(m+Q[[j]]*n+S[[j]]*o)-ac*(
   h+Q[[j]]*k+S[[j]]*l)*c*(mc+Qc[[j]]*nc+Sc[[j]]*oc))^2,{j,npos-1}];
Substitutions for SII, SIII, and WII
SIIsynthsubs={a->G-CC,
ac->Gc-CCc,
(*b->A-B+Q*(CC-A)-S*(DD-B),
bc->Ac-Bc+Qc*(CCc-Ac)-Sc*(DDc-Bc),*)
c->H-CC,
cc->Hc-CCc,
(*d->A-B+Q*(CC-A)-S*(F-B),
dc->Ac-Bc+Qc*(CCc-Ac)-Sc*(Fc-Bc),*)
f->G-DD,
fc->Gc-DDc,
g->H-F,
gc->Hc-Fc,
```

```
h->A-B,
hc->Ac-Bc,
k->CC-A,
kc->CCc-Ac,
l->-(DD-B),
lc->-(DDc-Bc),
m->A-B,
mc->Ac-Bc,
n->CC-A,
nc->CCc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
SIIIsynthsubs={a->G-DD,
ac->Gc-DDc,
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->H-DD,
cc->Hc-DDc,
(*d->A-B+Q*(DD-A)-S*(F-B),
dc->Ac-Bc+Qc*(DDc-Ac)-Sc*(Fc-Bc),*)
f->G-CC,
fc->Gc-CCc,
g->H-F,
gc->Hc-Fc,
h->A-CC,
hc->Ac-CCc,
k->DD-A,
kc->DDc-Ac,
l->0,
lc->0,
m->A-B,
mc->Ac-Bc,
n->DD-A,
nc->DDc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
WIIsynthsubs={a->-(G-CC),
ac->-(Gc-CCc),
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->-(H-CC),
cc->-(Hc-CCc),
(*d->B-CC+S*(F-B),
dc->Bc-CCc+Sc*(Fc-Bc),*)
f->(G-DD),
fc->(Gc-DDc),
g->(H-F),
gc->(Hc-Fc),
h->A-CC,
hc->Ac-CCc,
k->DD-A,
```

```
kc->DDc-Ac,
l->0,
lc->0,
m->B-CC,
mc->Bc-CCc,
n->0,
nc->0,
o->F-B,
oc->Fc-Bc};
Synthesis Equations (Truncated)
(*Synthesis Equations from general form*)
SIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIsynthsubs),{j,npos-1}];
SIIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIIsynthsubs),{j,npos-1}];
WIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.WIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.WIIsynthsubs),{j,npos-3}];
(*Synthesis equations from compact form*)
SIIeqns2=eqns4/.SIIsynthsubs;
SIIIeqns2=eqns4/.SIIIsynthsubs;
WIIeqns2=eqns4[[1;;8]]/.WIIsynthsubs;
Task Specification
(*Images here*)
functype="SIIaFunc";
(*Acceptable functype
"WIIFunc"
"SIIIaFunc"
"SIIIbFunc"
"SIIaFunc"
"SIIbFunc"*)
func[\[Phi]_]:=-0.02470723216917243`+0.06336903857340614` Cos[\[Phi
   ]]-0.04881051503139809` Cos[2 \[Phi]]+0.0191535205842668` Cos[3 \[Phi
   ]]-0.26280363530656836` Sin[\[Phi]]+0.07497345775585648` Sin[2 \[Phi
   ]]-0.018096624742974847` Sin[3 \[Phi]]
\[CapitalDelta]xyangles={(*1*){\[CapitalDelta]x->45*Degree,\[CapitalDelta]
   y->func[45*Degree]},
(*2*){\[CapitalDelta]x->90*Degree,\[CapitalDelta]y->func[90*Degree]},
(*3*){\[CapitalDelta]x->139*Degree,\[CapitalDelta]y->func[139*Degree]},
(*4*){\[CapitalDelta]x->170*Degree,\[CapitalDelta]y->func[170*Degree]},
(*5*){\[CapitalDelta]x->187*Degree,\[CapitalDelta]y->func[187*Degree]},
(*6*){\[CapitalDelta]x->203*Degree,\[CapitalDelta]y->func[203*Degree]},
(*7*){\[CapitalDelta]x->220*Degree,\[CapitalDelta]y->func[220*Degree]},
(*8*){\[CapitalDelta]x->252*Degree,\[CapitalDelta]y->func[252*Degree]},
(*9*){\[CapitalDelta]x->290*Degree,\[CapitalDelta]y->func[290*Degree]},
(*10*){\[CapitalDelta]x->325*Degree,\[CapitalDelta]y->func[325*Degree]}};
tasksubs\[Phi]\[Psi]=Which[functype=="WIIFunc",Map[{\[Phi]->\[CapitalDelta]
   x,\[Psi]->\[CapitalDelta]y}/.#&,\[CapitalDelta]xyangles],
functype=="SIIIaFunc",Map[{\[Phi]->\[CapitalDelta]x,\[Psi]->\[CapitalDelta]
   y}/.#&,\[CapitalDelta]xyangles],
functype=="SIIIbFunc",Map[{\[Psi]->\[CapitalDelta]x,\[Phi]->\[CapitalDelta]
   y}/.#&,\[CapitalDelta]xyangles],
```

```
functype=="SIIaFunc",Map[{\[Phi]->\[CapitalDelta]x,\[Psi]->\[CapitalDelta]
    y}/.#&,\[CapitalDelta]xyangles],
functype=="SIIbFunc",Map[{\[Psi]->\[CapitalDelta]x,\[Phi]->\[CapitalDelta]
    y}/.#&,\[CapitalDelta]xyangles],
True,Print["Error"];Abort[]];
tasksubsQS=MapIndexed[{Q[[#2[[1]]]]->Exp[\[Phi]*I],S[[#2[[1]]]]->Exp[\[Psi
    ]*I]}/.#&,tasksubs\[Phi]\[Psi]];
tasksubsQS=Flatten[tasksubsQS];
tasksubsQS=tasksubsQS~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.tasksubsQS);
inoutVars=Which[functype=="WIIFunc",{Q,S},functype=="SIIIaFunc",{Q,S},
    functype=="SIIIbFunc",{S,Q},functype=="SIIaFunc",{Q,S},functype=="
    SIIbFunc",{S,Q}];
points2graph={{0,0}}~Join~MapThread[{Mod[Arg[#1],2*\[Pi]],Arg[#2]}/.
    tasksubsQS&,inoutVars];
Show[ListPlot[points2graph,PlotStyle->Directive[Red,PointSize[Large]]],
Plot[func[x],{x,-\[Pi],2*\[Pi]+1}],PlotRange->{{-0,2*\[Pi]},{-.4,.3}}]
finalParameters1=Map[SetPrecision[{Re[#],Im[#]},300]&,Join[Q,S]/.
    tasksubsQS];
finalParameters2=finalParameters1;
finalParameters3=Map[ToString[InputForm[#,NumberMarks:>False]]&,
    finalParameters2,{2}];
finalParameters4=Map[StringReplace[#,"*^"~~a:(Shortest[___]~~NumberString)
    :>"e"<>ToString[a]]&,finalParameters3,{2}];
finalParameters5=Map[StringReplace[#,Shortest[("SetAccuracy[0,"~~__~~"]")
    ]->"0"]&,finalParameters4,{2}];
finalParameters6=Apply[#1<>"␣"<>#2&,finalParameters5,1];
finalParameters7=Prepend[finalParameters6,Length[finalParameters6]];
MatrixForm[finalParameters5]
TypeandTask={functype,tasksubsQS};
Task Specification Export (do not execute)
directoryname="C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
    Synthesis\\Function\\Stephenson␣II\\Homotopies\\11␣Position\\140728
    funcgenstephII3f\\Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="E:\\Dropbox\\Mathematica\\Six-bar␣Synthesis\\Function\\
    Stephenson␣II\\Homotopies\\11␣Position\\140728funcgenstephII3f\\
    Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
    Synthesis\\Function\\Stephenson␣III\\Homotopies\\11␣Position\\141029
    funcgenstephIII-2\\Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="E:\\Dropbox\\Mathematica\\Six-bar␣Synthesis\\Function\\
    Stephenson␣III\\Homotopies\\11␣Position\\141029funcgenstephIII-2\\
    Parameter␣Homotopies\\";
SetDirectory[directoryname];
foldername="150317funcgenSIIatsugewalker\\";
If[DirectoryQ[foldername],Print["Folder␣name␣already␣exists"]];
jobname="SIIatsuge";
CreateDirectory[foldername];
```

```
Export[foldername<>"final_parameters.txt",finalParameters7];
RenameFile[foldername<>"final_parameters.txt",foldername<>"
   final_parameters"];
CopyFile["..\\start_parameters",foldername<>"start_parameters"];
CopyFile["input_specific",foldername<>jobname<>"_specific"];
jobsh={"#!/bin/bash",
"#$␣-N␣"<>jobname,
"#$␣-q␣free64",
"#$␣-pe␣one-node-mpi␣64",
"#$␣-m␣beas",
"",
"module␣load␣openmpi-1.6.0/gcc-4.7.3",
"module␣load␣Bertini/1.4",
"",
"mpirun␣bertini␣"<>jobname<>"_specific␣../general_solutions␣>␣"<>jobname<>
   "_specific_out"};
Export[foldername<>jobname<>".txt",jobsh];
RenameFile[foldername<>jobname<>".txt",foldername<>jobname<>".sh"];
TypeandTask>>directoryname<>foldername<>"TypeandTask"
```

# D Mathematica code: Function Generation Analysis for Stephenson II

```
(*Type and Task*)
(*SetDirectory[NotebookDirectory[]];*)
{functype,tasksubsQS}=<<"TypeandTask";
(*Acceptable functype
"WIIFunc"
"SIIIaFunc"
"SIIIbFunc"
"SIIaFunc"
"SIIbFunc"*)
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
    i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
(*tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.params6);*)
(*Import Bertini Results for SIIFunc*)
(*If[FileExistsQ["ProgressReport.txt"],Interrupt[]]*)
begin=DateString[];
ProgressReport={begin};
AppendTo[ProgressReport,""];
Solns1=ReadList["nonsingular_solutions",Number];
Solns4=Partition[Rest[Solns1],2*10];
Solns1=.
Solns5=Partition[#,2]&/@Solns4;
Solns4=.
Solns6=Parallelize[Apply[#1+#2*I&,Solns5,{2}]];
Solns5=.
(*All solutions properly formatted*)
Solns7=Thread[{CC,CCc,DD,DDc,F,Fc,G,Gc,H,Hc}->#]&/@Solns6;
Solns6=.
```

```
Length[Solns7];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns7)␣imported␣and␣
   formatted:␣"<>ToString[Length[Solns7]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-2)*)
Solns8=Cases[Solns7,x_/;Round[{CCc,DDc,Fc,Gc,Hc}/.x,10^-2]==Round[{CC\[
   Conjugate],DD\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-2]];
Length[Solns8];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-8)*)
Solns9=Cases[Solns8,x_/;Round[{CCc,DDc,Fc,Gc,Hc}/.x,10^-8]==Round[{CC\[
   Conjugate],DD\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-8]];
Length[Solns9];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns9)␣that␣pertain␣to␣
   physical␣linkages:␣"<>ToString[Length[Solns9]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Pair solutions with their symmetric counterpart*)
temp=Solns9;
Solns10={};
While[Length[temp]>0,
pos=Position[temp,x_/;(Round[{CC,DD,F,G,H}/.x,10^-8]==Round[{CC,DD,F,G,H
   }/.First[temp],10^-8])||(Round[{CC,F,DD,H,G}/.x,10^-8]==Round[{CC,DD,F,
   G,H}/.First[temp],10^-8]),1,Heads->False];
AppendTo[Solns10,temp[[pos[[All,1]]]]];
temp=Delete[temp,pos];];
table10=Prepend[Sort[Tally[Length/@Solns10]],{"No.␣of␣redundancies","No.␣
   of␣soln␣groups"}];
MatrixForm[table10];
AppendTo[ProgressReport,"Singletons␣and␣solutions␣with␣their␣symmetric␣
   counterparts"];
AppendTo[ProgressReport,ToString[Grid[table10]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Take only one of each symmetric pair, or just the singleton*)
Solns10=Solns10[[All,1]];
Length[Solns10];
Solns10>>"Solns10"
(*Import from here*)
Solns10=<<"Solns10";
Length[Solns10];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns10)␣with␣one␣of␣each␣
   symmetric␣redundant␣pair␣removed:␣"<>ToString[Length[Solns10]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Gp=DD-G+CC;
Hpp=F-H+CC;
Hp=(DD-G)/(H-G)*(H-CC)+CC;
Gpp=(F-H)/(G-H)*(G-CC)+CC;
```

```
Fp=((DD-G)*(H-CC)-(F-H)*(G-CC))/(H-G)+CC;
ConstructCognates[Soln_]:=Module[{values,cognates},
values={{CC,DD,G,H,F},{CC,DD,Gp,Hp,Fp},{CC,Fp,Gpp,Hpp,F}}/.Soln;
cognates=Apply[{CC->#1,DD->#2,G->#3,H->#4,F->#5,CCc->#1\[Conjugate],DDc
    ->#2\[Conjugate],Gc->#3\[Conjugate],Hc->#4\[Conjugate],Fc->#5\[
    Conjugate]}&,values,2];
cognates]
(*Group solutions into cognate sets of 3*)
temp=Solns10;
Solns12={};
acc=10^-8;
While[Length[temp]>0,
pos=Position[temp,
x_/;(Round[{CC}/.x,acc]==Round[{CC}/.First[temp],acc])
&&(Round[{DD,G,H,F}/.x,acc]==Round[{DD,G,H,F}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{F,H,G,DD}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{DD,Gp,Hp,Fp}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{Fp,Hp,Gp,DD}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{Fp,Gpp,Hpp,F}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{F,Hpp,Gpp,Fp}/.First[temp],acc])
,1,Heads->False];
AppendTo[Solns12,temp[[pos[[All,1]]]]];
temp=Delete[temp,pos];];
table12=Prepend[Sort[Tally[Length/@Solns12]],{"No.␣of␣cognates","No.␣of␣
    soln␣groups"}];
MatrixForm[table12];
Length[Solns12];
AppendTo[ProgressReport,"Number␣of␣cognates␣groups␣(Solns12):␣"<>ToString[
    Length[Solns12]]];
AppendTo[ProgressReport,"Groups␣of␣Cognates"];
AppendTo[ProgressReport,ToString[Grid[table12]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*For groups of 1 and 2, add their missing cognates, place all solutions
    in a properly flattened list*)
Solns13Group1=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==1][[All
    ,1]]];
Solns13Group2=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==2][[All
    ,1]]];
Solns13Group3=Cases[Solns12,x_/;Length[x]==3];
Solns13Group4=Cases[Solns12,x_/;Length[x]>=4];
Solns13=Flatten[Join[Solns13Group1,Solns13Group2,Solns13Group3,
    Solns13Group4],1];
Length[Solns13];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns13)␣after␣constructed␣
    cognates␣were␣added:␣"<>ToString[Length[Solns13]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a maximum on the size of link length*)
maxlinklength=5;
Solns14=Cases[Solns13,x_/;Map[Norm[#/.{A->0,B->1}/.x]<maxlinklength&,{CC-A
```

```
    ,DD-B,F-B,F-DD,G-CC,H-CC,H-G,G-DD,H-F}]==Table[True,{9}]];
Length[Solns14];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns14)␣after␣long␣link␣
    length␣linkages␣were␣removed:␣"<>ToString[Length[Solns14]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a minimum on the size of link length*)
minlinklength=.05;
Solns15=Cases[Solns14,x_/;Map[Norm[#/.{A->0,B->1}/.x]>minlinklength&,{CC-A
    ,DD-B,F-B,F-DD,G-CC,H-CC,H-G,G-DD,H-F}]==Table[True,{9}]];
Length[Solns15];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns15)␣after␣small␣link␣
    length␣linkages␣were␣removed:␣"<>ToString[Length[Solns15]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
    branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
    *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
    DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
    CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
```

```
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
    *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
    RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
    SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
    -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
    SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
    Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
    branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
    ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
    coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
    figures in my dissertation*)
(*outvar -this should be a list of variables to appear in the output
    configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
    (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
    invar is a valid parameter*)
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
    *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
    Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
    SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
    ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
```

```
    branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
    ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
    into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
    log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
    NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
    *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
    position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
    All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
    /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]]
]];
```

```
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
    removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
    sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
    ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
    chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
    ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
```

```
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
    branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
    tasksubsQS]
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
    Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
    \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
    Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
    [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
    ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
    ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
    ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
    tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
    checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
    taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
    configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
    the complex planes of individual variables for determining whether or
    not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
    ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
    each checkvar in each configuration as a percent of that variables
    range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
```

```
   between two configurations for each checkvar as a percent of that
   variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
   nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
   variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
   ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
   check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
   ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
   tolRectsPercent],,Print["Error"];Abort[]];
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
   ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
   indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
   )*)
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
```

```
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
    matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
    ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
    ]],"error"]
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
    error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
    ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
    ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
    WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
    SIIFunc2SIIPath","SIIFunc2SIIIPath"};
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
```

```
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalD],\[ScriptCapitalG
   ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD],\[
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
```

```
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalA],\[ScriptCapitalC
   ],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
   ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
```

```
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH],\[
    ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[ScriptCapitalP
    ]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[ScriptCapitalD],\[
    ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalD],\[ScriptCapitalG
    ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
    ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
    inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
    line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
```

```
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
    ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
    Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
    ->500];
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Analysis Execution*)
Solns16=Which[
functype=="WIIFunc",Map[Join[{CC->0,CCc->0,G->1,Gc->1,H->1,Hc->1},#]&,
    Solns15],
functype=="SIIIaFunc",Map[Join[{A->0,Ac->0,DD->1,DDc->1},#]&,Solns15],
functype=="SIIIbFunc",Map[Join[{A->0,Ac->0,DD->1,DDc->1},#]&,Solns15],
functype=="SIIaFunc",Map[Join[{A->0,Ac->0,B->1,Bc->1},#]&,Solns15],
functype=="SIIbFunc",Map[Join[{A->0,Ac->0,B->1,Bc->1},#]&,Solns15],
True,Print["Error"];Abort[]];
Which[
functype=="WIIFunc",inputAngles={QQ};outputAngles={QQ,RR,SS};type="WII",
functype=="SIIIaFunc",inputAngles={QQ};outputAngles={QQ,SS,UU};type="SIII"
    ,
functype=="SIIIbFunc",inputAngles={SS};outputAngles={QQ,SS,UU};type="SIII"
    ,
functype=="SIIaFunc",inputAngles={QQ};outputAngles={QQ,RR,SS};type="SII",
functype=="SIIbFunc",inputAngles={SS};outputAngles={QQ,RR,SS};type="SII",
True,Print["Error"];Abort[]];
ctr1=0;
NumToRun1=Length[Solns16]*Length[inputAngles];
```

```
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Solution␣to␣forward␣kinematics␣equations"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun1]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr1];
SetSharedVariable[ctrfloor1];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   Script0]\[Script0]\[ScriptR]1];
FwdKinWithCtr[type_,dim_,invar_,outvar_]:=Module[{branches},
branches=Quiet[TimeConstrained[FwdKin[type,dim,invar,outvar],15*60,"Timed␣
   Out"],{Det::mindet,Infinity::indet,Inverse::luc,Inverse::sing,Power::
   infy,Divide::infy}];
If[branches=="Timed␣Out",Print["Time␣out␣near␣",ctr1]];
ctr1++;
ctrfloor1=Floor[ctr1,10];
If[ctrfloor1!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[Script0
   ]\[Script0]\[ScriptR]1,ProgressReport[[-1]]=ToString[ctrfloor1]<>"␣/␣"
   <>ToString[NumToRun1];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[Script0]\[Script0]\[
   ScriptR]1=ctrfloor1;
branches]
(*Print["counter ",Dynamic[ctr1],"/",NumToRun1]
ProgressIndicator[Dynamic[ctr1],{0,NumToRun1}]*)
branches=ParallelTable[FwdKinWithCtr[type,dim,invar,outputAngles],{dim,
   Solns16},{invar,inputAngles}];
branches=Map[Flatten[#,1]&,branches];
(*branches=Parallelize[MapThread[AddTracePoint[pathtype,#1,#2]&,{Solns16,
   branchesNoTracePoint}]];*)
ProgressReport[[-1]]=ToString[ctr1]<>"␣/␣"<>ToString[NumToRun1];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
branches>>"branches"
ctr2=0;
NumToRun2=Total[Length/@branches];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Determination␣if␣task␣positions␣are␣included␣in␣
   branches"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun2]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr2];
SetSharedVariable[ctrfloor2];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   Script0]\[Script0]\[ScriptR]2];
TaskCheckWithCtr[branch_,checkvar_,checktask_,\[ScriptT]\[Script0]\[
   ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT]_:"default",\[ScriptT]\[Script0]\[ScriptL]\[ScriptCapitalR]\[
   ScriptE]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[
   ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{
   taskconfigsUnion},
```

```
taskconfigsUnion=TaskCheck[branch,checkvar,checktask,\[ScriptT]\[ScriptO
   ]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT],\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[
   ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]];
ctr2++;
ctrfloor2=Floor[ctr2,100];
If[ctrfloor2!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
   ]\[ScriptO]\[ScriptR]2,ProgressReport[[-1]]=ToString[ctrfloor2]<>"␣/␣"
   <>ToString[NumToRun2];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
   ScriptR]2=ctrfloor2;
taskconfigsUnion]
(*Print["counter ",Dynamic[ctr2],"/",NumToRun2]
ProgressIndicator[Dynamic[ctr2],{0,NumToRun2}]*)
checkvar={QQ,SS};
checktask={{1}~Join~Q/.tasksubsQS,{1}~Join~S/.tasksubsQS};
checktol=10^-2;
analyses=ParallelMap[TaskCheckWithCtr[#,checkvar,checktask,{checktol,
   checktol},{checktol,checktol}]&,branches,{2}];
ProgressReport[[-1]]=ToString[ctr2]<>"␣/␣"<>ToString[NumToRun2];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
designs=MapThread[{#1,#2,#3}&,{Solns16,branches,analyses}];
(*designs indexed by
-dim
-branches
        -branch1
                -config1
                -config2
                -etc
        -branch2
        -etc
-analysis*)
designs>>"designs"
DeleteFile["branches"]
designsreduced=ReduceDesign[#,11]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length11=Length[designsreduced];
If[length11>0,designsreduced>>"designsreduced11"]
designsreduced=ReduceDesign[#,10]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length10=Length[designsreduced];
If[length10>0,designsreduced>>"designsreduced10"]
designsreduced=ReduceDesign[#,9]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length9=Length[designsreduced];
If[length9>0,designsreduced>>"designsreduced9"]
designsreduced=ReduceDesign[#,8]&/@designs;
```

```
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length8=Length[designsreduced];
If[length8>0,designsreduced>>"designsreduced8"]
designsreduced=ReduceDesign[#,7]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length7=Length[designsreduced];
If[length7>0,designsreduced>>"designsreduced7"]
designsreduced=ReduceDesign[#,6]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length6=Length[designsreduced];
If[length6>0,designsreduced>>"designsreduced6"]
resultstable={{"","No. of linkages"},{"11 position",length11},{"10 
    position",length10},{"9 position",length9},{"8 position",length8},{"7 
    position",length7},{"6 position",length6}};
AppendTo[ProgressReport,ToString[Grid[resultstable]]];
AppendTo[ProgressReport,""];
AppendTo[ProgressReport,"Analysis complete!  See designsreduced* files for
    the linkage solutions."];
AppendTo[ProgressReport,""];
end=DateString[];
AppendTo[ProgressReport,end];
AppendTo[ProgressReport,""];
computationtime=DateDifference[begin,end,"Hour"][[1]];
AppendTo[ProgressReport,"The analysis computation took "<>ToString[
    computationtime]<>" hours."];
Export["ProgressReport.txt",ProgressReport];
(*designsreduced=<<"designsreduced7";
Length[designsreduced]*)
(*plotrange={{-1,3},{-1,2}};
funcplotrange={{-1,2*\[Pi]},{-.6,.1}};
pivotsize=.08;*)
(*Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
    Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
    ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Grid[{
{Draw[functype,designsreduced[[des,1]],designsreduced[[des,2,br,config]]],
CreateFunctionPlot[designsreduced[[des,2,br]],config,functype]},
{Column[Cases[designsreduced[[des,1]],x_/;MemberQ[{CC,DD,F,G,H},x[[1]]]]],
CreateAnalysisTable[designsreduced[[des,3]]]}
},Alignment->Left],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
{config,1,Length[designsreduced[[des,2,br]]],1}]]*)
```

# E Mathematica code: Function Generation Analysis for Stephenson III

```
(*Type and Task*)
(*SetDirectory[NotebookDirectory[]];*)
{functype,tasksubsQS}=<<"TypeandTask";
(*Acceptable functype
"WIIFunc"
"SIIIaFunc"
"SIIIbFunc"
"SIIaFunc"
"SIIbFunc"*)
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
    i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
(*tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.params6);*)
(*Import Bertini Results for SIIIFunc*)
(*If[FileExistsQ["ProgressReport.txt"],Interrupt[]]*)
begin=DateString[];
ProgressReport={begin};
AppendTo[ProgressReport,""];
Solns1=ReadList["nonsingular_solutions",Number];
Solns4=Partition[Rest[Solns1],2*18];
(*Solns1=.*)
Solns5=Partition[#,2]&/@Solns4;
(*Solns4=.*)
Solns6=Parallelize[Apply[#1+#2*I&,Solns5,{2}]];
(*Solns5=.*)
(*All solutions properly formatted*)
Solns7=Thread[{CC,CCc,G,Gc,r1,rc1,B,Bc,F,Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4
    }->#]&/@Solns6;
```

```
(*Solns6=.*)
Length[Solns7];
793020
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns7)␣imported␣and␣
   formatted:␣"<>ToString[Length[Solns7]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-2)*)
Solns8=Cases[Solns7,x_/;Round[{Bc,CCc,Fc,Gc,Hc}/.x,10^-2]==Round[{B\[
   Conjugate],CC\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-2]];
Length[Solns8];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-8)*)
Solns9=Cases[Solns8,x_/;Round[{Bc,CCc,Fc,Gc,Hc}/.x,10^-8]==Round[{B\[
   Conjugate],CC\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-8]];
Length[Solns9];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns9)␣that␣pertain␣to␣
   physical␣linkages:␣"<>ToString[Length[Solns9]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Solns10=Map[Cases[#,x_/;MemberQ[{r1,rc1,r2,rc2,r3,rc3,r4,rc4},x[[1]]]==
   False]&,Solns9];
ConstructCognates[Soln_]:=Module[{Gcogn,Hcogn,Bcogn,Fcogn,r1cogn,r2cogn,
   r3cogn,r4cogn,cognate},
Gcogn=DD-G+CC/.DD->1/.Soln;
Hcogn=(DD-G)/(H-G)*(H-CC)+CC/.DD->1/.Soln;
Bcogn=(DD-G)/(H-G)*(B-CC)+CC/.DD->1/.Soln;
Fcogn=(DD-G)/(H-G)*(F-CC)+CC/.DD->1/.Soln;
(*r1cogn=-(Gcogn-1)*(CCc/.Soln);
r2cogn=-(Hcogn-1)*Bcogn\[Conjugate];
r3cogn=-(Hcogn-1)*(Fcogn\[Conjugate]-Bcogn\[Conjugate]);
r4cogn=Bcogn*(Fcogn\[Conjugate]-Bcogn\[Conjugate]);*)
cognate={B->Bcogn,CC->(CC/.Soln),F->Fcogn,G->Gcogn,H->Hcogn,Bc->Bcogn\[
   Conjugate],CCc->(CC/.Soln)\[Conjugate],Fc->Fcogn\[Conjugate],Gc->Gcogn
   \[Conjugate],Hc->Hcogn\[Conjugate](*,r1->r1cogn,r2->r2cogn,r3->r3cogn,
   r4->r4cogn,rc1->r1cogn\[Conjugate],rc2->r2cogn\[Conjugate],rc3->r3cogn
   \[Conjugate],rc4->r4cogn\[Conjugate]*)};
{Soln,cognate}]
(*Group solutions into cognate sets of 2*)
temp=Solns10;
Solns12={};
acc=10^-8;
While[Length[temp]>0,
pos=Position[temp,x_/;(Round[{CC}/.x,acc]==Round[{CC}/.First[temp],acc])
&&(Round[{B,F,G,H}/.x,acc]==Round[{B,F,G,H}/.ConstructCognates[First[temp
   ]][[1]],acc]
||Round[{B,F,G,H}/.x,acc]==Round[{B,F,G,H}/.ConstructCognates[First[temp
   ]][[2]],acc])
,{1},Heads->False];
AppendTo[Solns12,temp[[pos[[All,1]]]]];
```

```
temp=Delete[temp,pos];];
table12=Prepend[Sort[Tally[Length/@Solns12]],{"No. of cognates","No. of
    soln groups"}];
MatrixForm[table12];
Length[Solns12];
AppendTo[ProgressReport,"Number of cognates groups (Solns12): "<>ToString[
    Length[Solns12]]];
AppendTo[ProgressReport,"Groups of Cognates"];
AppendTo[ProgressReport,ToString[Grid[table12]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*For groups of 1, add their missing cognates, place all solutions in a
    properly flattened list*)
Solns13Group1=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==1][[All
    ,1]]];
Solns13Group2=Cases[Solns12,x_/;Length[x]==2];
Solns13Group3=Cases[Solns12,x_/;Length[x]>=3];
Solns13=Flatten[Join[Solns13Group1,Solns13Group2,Solns13Group3],1];
Length[Solns13];
AppendTo[ProgressReport,"Number of solutions (Solns13) after constructed
    cognates were added: "<>ToString[Length[Solns13]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Solns13>>"Solns13"
(*Solns13=<<"Solns13";*)
(*Transform linkages so A=0 and B=1*)
Solns14=Map[Thread[{A,B,CC,DD,F,G,H,Ac,Bc,CCc,DDc,Fc,Gc,Hc}->(Join[{0,B,CC
    ,1,F,G,H}/B,{0,Bc,CCc,1,Fc,Gc,Hc}/Bc]/.#)]&,Solns13];
(*Place a maximum on the size of link length*)
maxlinklength=5;
Solns15=Cases[Solns14,x_/;Map[Norm[#/.x]<maxlinklength&,{CC-A,CC-B,DD-A,F-
    B,G-CC,G-DD,H-DD,H-G,H-F}]==Table[True,{9}]];
Length[Solns15];
AppendTo[ProgressReport,"Number of solutions (Solns15) after long link
    length linkages were removed: "<>ToString[Length[Solns15]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a minimum on the size of link length*)
minlinklength=.05;
Solns16=Cases[Solns15,x_/;Map[Norm[#/.x]>minlinklength&,{CC-A,CC-B,DD-A,F-
    B,G-CC,G-DD,H-DD,H-G,H-F}]==Table[True,{9}]];
Length[Solns16];
AppendTo[ProgressReport,"Number of solutions (Solns16) after small link
    length linkages were removed: "<>ToString[Length[Solns16]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
```

```
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
   branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
   branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
   CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
   *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
   DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
   CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
   CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
   *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
   RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
   SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
   -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
   SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
   Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
   branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
   ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
   coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
   figures in my dissertation*)
```

```
(*outvar -this should be a list of variables to appear in the output
    configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
    (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
    invar is a valid parameter*)
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
    *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
    Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
    SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
    ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
    ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
    into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
    log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
    NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
    *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
```

```
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
    position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
    All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
    /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]
]];
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
```

```
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
    removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
    sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
    ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
    chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
    ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
    branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
    tasksubsQS]
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
    Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
```

```
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
   \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
   Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
   [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
   ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
   ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
   ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
   tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
   checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
   taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
   configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
   the complex planes of individual variables for determining whether or
   not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
   ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
   each checkvar in each configuration as a percent of that variables
   range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
   between two configurations for each checkvar as a percent of that
   variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
   nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
   variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
   ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
   check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
   ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
```

```
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
   tolRectsPercent],,Print["Error"];Abort[]];
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
   ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
   indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
   )*)
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
   matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
   ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
   ]],"error"]
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
   error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
```

```
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
    ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
    ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
    WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
    SIIFunc2SIIPath","SIIFunc2SIIIPath"};
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalD],\[ScriptCapitalG
    ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
```

```
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
    ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
    ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD],\[
    ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
    ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalA],\[ScriptCapitalC
    ],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
    ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
    ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
    ScriptCapitalG],\[ScriptCapitalH]}}],
```

```
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
   ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH],\[
   ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[ScriptCapitalP
   ]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
```

```
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalD],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
   ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
   ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
   inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
   line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
   ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
   Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
   ->500];
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
```

```
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Analysis Execution*)
Which[
functype=="WIIFunc",inputAngles={QQ};outputAngles={QQ,RR,SS};type="WII",
functype=="SIIIaFunc",inputAngles={QQ};outputAngles={QQ,SS,UU};type="SIII"
    ,
functype=="SIIIbFunc",inputAngles={SS};outputAngles={QQ,SS,UU};type="SIII"
    ,
functype=="SIIaFunc",inputAngles={QQ};outputAngles={QQ,RR,SS};type="SII",
functype=="SIIbFunc",inputAngles={SS};outputAngles={QQ,RR,SS};type="SII",
True,Print["Error"];Abort[]];
ctr1=0;
NumToRun1=Length[Solns16]*Length[inputAngles];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Solution␣to␣forward␣kinematics␣equations"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun1]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr1];
SetSharedVariable[ctrfloor1];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
    ScriptO]\[ScriptO]\[ScriptR]1];
FwdKinWithCtr[type_,dim_,invar_,outvar_]:=Module[{branches},
branches=Quiet[TimeConstrained[FwdKin[type,dim,invar,outvar],15*60,"Timed␣
    Out"],{Det::mindet,Infinity::indet,Inverse::luc,Inverse::sing,Power::
    infy,Divide::infy}];
If[branches=="Timed␣Out",Print["Time␣out␣near␣",ctr1]];
ctr1++;
ctrfloor1=Floor[ctr1,10];
If[ctrfloor1!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
    ]\[ScriptO]\[ScriptR]1,ProgressReport[[-1]]=ToString[ctrfloor1]<>"␣/␣"
    <>ToString[NumToRun1];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
    ScriptR]1=ctrfloor1;
branches]
(*Print["counter ",Dynamic[ctr1],"/",NumToRun1]
ProgressIndicator[Dynamic[ctr1],{0,NumToRun1}]*)
branches=ParallelTable[FwdKinWithCtr[type,dim,invar,outputAngles],{dim,
    Solns16},{invar,inputAngles}];
branches=Map[Flatten[#,1]&,branches];
(*branches=Parallelize[MapThread[AddTracePoint[pathtype,#1,#2]&,{Solns16,
    branchesNoTracePoint}]];*)
```

```
ProgressReport[[-1]]=ToString[ctr1]<>"␣/␣"<>ToString[NumToRun1];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
branches>>"branches"
ctr2=0;
NumToRun2=Total[Length/@branches];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Determination␣if␣task␣positions␣are␣included␣in␣
   branches"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun2]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr2];
SetSharedVariable[ctrfloor2];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   ScriptO]\[ScriptO]\[ScriptR]2];
TaskCheckWithCtr[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[
   ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT]_:"default",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[
   ScriptE]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[
   ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{
   taskconfigsUnion},
taskconfigsUnion=TaskCheck[branch,checkvar,checktask,\[ScriptT]\[ScriptO
   ]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT],\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[
   ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]];
ctr2++;
ctrfloor2=Floor[ctr2,100];
If[ctrfloor2!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
   ]\[ScriptO]\[ScriptR]2,ProgressReport[[-1]]=ToString[ctrfloor2]<>"␣/␣"
   <>ToString[NumToRun2];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
   ScriptR]2=ctrfloor2;
taskconfigsUnion]
(*Print["counter ",Dynamic[ctr2],"/",NumToRun2]
ProgressIndicator[Dynamic[ctr2],{0,NumToRun2}]*)
checkvar={QQ,SS};
checktask={{1}~Join~Q/.tasksubsQS,{1}~Join~S/.tasksubsQS};
checktol=10^-2;
analyses=ParallelMap[TaskCheckWithCtr[#,checkvar,checktask,{checktol,
   checktol},{checktol,checktol}]&,branches,{2}];
ProgressReport[[-1]]=ToString[ctr2]<>"␣/␣"<>ToString[NumToRun2];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
designs=MapThread[{#1,#2,#3}&,{Solns16,branches,analyses}];
(*designs indexed by
-dim
```

```
-branches
        -branch1
                -config1
                -config2
                -etc
        -branch2
        -etc
-analysis*)
designs>>"designs"
DeleteFile["branches"]
designsreduced=ReduceDesign[#,11]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length11=Length[designsreduced];
If[length11>0,designsreduced>>"designsreduced11"]
designsreduced=ReduceDesign[#,10]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length10=Length[designsreduced];
If[length10>0,designsreduced>>"designsreduced10"]
designsreduced=ReduceDesign[#,9]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length9=Length[designsreduced];
If[length9>0,designsreduced>>"designsreduced9"]
designsreduced=ReduceDesign[#,8]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length8=Length[designsreduced];
If[length8>0,designsreduced>>"designsreduced8"]
designsreduced=ReduceDesign[#,7]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length7=Length[designsreduced];
If[length7>0,designsreduced>>"designsreduced7"]
designsreduced=ReduceDesign[#,6]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length6=Length[designsreduced];
If[length6>0,designsreduced>>"designsreduced6"]
resultstable={{"","No.␣of␣linkages"},{"11␣position",length11},{"10␣
   position",length10},{"9␣position",length9},{"8␣position",length8},{"7␣
   position",length7},{"6␣position",length6}};
AppendTo[ProgressReport,ToString[Grid[resultstable]]];
AppendTo[ProgressReport,""];
AppendTo[ProgressReport,"Analysis␣complete!␣␣See␣designsreduced*␣files␣for
   ␣the␣linkage␣solutions."];
AppendTo[ProgressReport,""];
end=DateString[];
AppendTo[ProgressReport,end];
AppendTo[ProgressReport,""];
computationtime=DateDifference[begin,end,"Hour"][[1]];
AppendTo[ProgressReport,"The␣analysis␣computation␣took␣"<>ToString[
   computationtime]<>"␣hours."];
Export["ProgressReport.txt",ProgressReport];
(*designsreduced=<<"designsreduced6";
Length[designsreduced]*)
```

```
(*plotrange={{-1,2},{-1,1}};
funcplotrange={{-1,2*\[Pi]},{-2,2}};
pivotsize=.05;*)
(*Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
   Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
   ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Grid[{
{Draw[functype,designsreduced[[des,1]],designsreduced[[des,2,br,config]]],
CreateFunctionPlot[designsreduced[[des,2,br]],config,functype]},
{Column[Cases[designsreduced[[des,1]],x_/;MemberQ[{CC,DD,F,G,H},x[[1]]]]],
CreateAnalysisTable[designsreduced[[des,3]]]}
},Alignment->Left],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
{config,1,Length[designsreduced[[des,2,br]]],1}]]*)
```

# F Mathematica code: Function Generation Viewer

```
(*Type and Task*)
SetDirectory[NotebookDirectory[]];
{functype,tasksubsQS}=<<"TypeandTask";
(*Acceptable functype
"WIIFunc"
"SIIIaFunc"
"SIIIbFunc"
"SIIaFunc"
"SIIbFunc"*)
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
    i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
(*tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.params6);*)
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
```

```
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
    branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
    *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
    DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
    CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
    *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
    RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
    SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
    -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
    SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
    Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
    branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
    ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
    coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
    figures in my dissertation*)
(*outvar -this should be a list of variables to appear in the output
    configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
    (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
    invar is a valid parameter*)
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
    *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
    Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
```

```
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
   SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
   ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
   branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
   ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
   into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
   log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
   NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
   *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
   position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
   All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
```

```
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
    /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]
]];
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
```

```
   removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
   sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
   ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
   chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
   ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
   branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
   tasksubsQS]
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
   Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
   \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
   Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
   [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
   ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
   ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
   ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
```

```
    tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
    checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
    taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
    configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
    the complex planes of individual variables for determining whether or
    not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
    ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
    each checkvar in each configuration as a percent of that variables
    range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
    between two configurations for each checkvar as a percent of that
    variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
    nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
    variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
    ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
    ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
    check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
    ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
    ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
    tolRectsPercent],,Print["Error"];Abort[]];
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
    ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
    indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
    )*)
```

```
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
   matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
   ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
   ]],"error"]
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
   error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
   ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
   ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
   WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
   SIIFunc2SIIPath","SIIFunc2SIIIPath"};
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
```

```
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{\[ScriptCapitalC],\[
    ScriptCapitalG],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{\[ScriptCapitalD],\[
    ScriptCapitalG],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalC],\[
    ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD
    ],\[ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]}};
```

```
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalB],\[
    ScriptCapitalD],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD
    ],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalA],\[
    ScriptCapitalC],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD
    ],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
    ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
```

```
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalB],\[
    ScriptCapitalD],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],EdgeForm[Black],Polygon[{{\[ScriptCapitalC],\[
    ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH
    ],\[ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
    ScriptCapitalP]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[
    ScriptCapitalD],\[ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalD],\[
    ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH
    ],\[ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
```

```
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
   ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
   inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
   line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
   ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
   Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
   ->500];
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Solution Viewer*)
designsreduced=<<"designsreduced8";
```

```
Length[designsreduced]
303
plotrange={{-1,3},{-1,2}};
funcplotrange={{-1,2*\[Pi]},{-.6,.6}};
pivotsize=.08;
Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
   Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
   ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Grid[{
{Draw[functype,designsreduced[[des,1]],designsreduced[[des,2,br,config]]],
CreateFunctionPlot[designsreduced[[des,2,br]],config,functype]},
{Column[Cases[designsreduced[[des,1]],x_/;MemberQ[{CC,DD,F,G,H},x[[1]]]]],
CreateAnalysisTable[designsreduced[[des,3]]]}
},Alignment->Left],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
{config,1,Length[designsreduced[[des,2,br]]],1}]]
```

# G Mathematica code: Path Generation Specification

```
General Synthesis Equations
(*Synthesis equations take this form*)
eqns1=(a*bc*(g*gc-c*cc-d*dc)-c*dc*(f*fc-a*ac-b*bc))*(ac*b*(g*gc-c*cc-d*dc)
    -cc*d*(f*fc-a*ac-b*bc))+(a*bc*cc*d-ac*b*c*dc)^2;
(*Here is another way to write those equations with dot products*)
eqns2={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c}.{bc,-dc,-bc*d*dc,b*bc*dc}*{b,-d,-b
    *d*dc,b*bc*d}.{ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc}-{a*cc,ac*c}.{bc*d,-
    b*dc}*{b*dc,-bc*d}.{ac*c,a*cc};
Expand[eqns1]==Expand[eqns2]
True
(*b=Bmat.q*)
Bmat={{h,k,l,0,0,0,0,0,0,0,0,0},{-m,-n,-o,0,0,0,0,0,0,0,0,0},
{-m*(k*nc+l*oc)-h*(m*mc+n*nc+o*oc),-n*(h*mc+l*oc)-k*(m*mc+n*nc+o*oc),-o*(h
    *mc+k*nc)-l*(m*mc+n*nc+o*oc),-h*m*nc,-h*m*oc,-oc*(h*n+k*m),-nc*(h*o+l*m
    ),-mc*(k*o+l*n),-k*n*mc,-l*o*mc,-k*n*oc,-l*o*nc},
{h*(kc*n+lc*o)+m*(h*hc+k*kc+l*lc),k*(hc*m+lc*o)+n*(h*hc+k*kc+l*lc),l*(hc*m
    +kc*n)+o*(h*hc+k*kc+l*lc),h*m*kc,h*m*lc,lc*(h*n+k*m),kc*(h*o+l*m),hc*(k
    *o+l*n),k*n*hc,l*o*hc,k*n*lc,l*o*kc}};
Bcmat={{hc,kc,lc,0,0,0,0,0,0,0,0,0},{-mc,-nc,-oc,0,0,0,0,0,0,0,0,0},
{-mc*(kc*n+lc*o)-hc*(m*mc+n*nc+o*oc),-nc*(hc*m+lc*o)-kc*(m*mc+n*nc+o*oc),-
    oc*(hc*m+kc*n)-lc*(m*mc+n*nc+o*oc),-hc*mc*n,-hc*mc*o,-o*(hc*nc+kc*mc),-
    n*(hc*oc+lc*mc),-m*(kc*oc+lc*nc),-kc*nc*m,-lc*oc*m,-kc*nc*o,-lc*oc*n},
{hc*(k*nc+l*oc)+mc*(h*hc+k*kc+l*lc),kc*(h*mc+l*oc)+nc*(h*hc+k*kc+l*lc),lc
    *(h*mc+k*nc)+oc*(h*hc+k*kc+l*lc),hc*mc*k,hc*mc*l,l*(hc*nc+kc*mc),k*(hc*
    oc+lc*mc),h*(kc*oc+lc*nc),kc*nc*h,lc*oc*h,kc*nc*l,lc*oc*k}};
(*d=Dmat.q'*)
Dmat={{h*mc+k*nc+l*oc,k*mc,l*mc,h*nc,h*oc,k*oc,l*nc},{-(hc*m+kc*n+lc*o),-
    hc*n,-hc*o,-kc*m,-lc*m,-lc*n,-kc*o}};
Dcmat={{hc*m+kc*n+lc*o,kc*m,lc*m,hc*n,hc*o,kc*o,lc*n},{-(h*mc+k*nc+l*oc),-
    h*nc,-h*oc,-k*mc,-l*mc,-l*nc,-k*oc}};
avect={a*(g*gc-c*cc),c*(f*fc-a*ac),a,c};
aCvect={ac*(g*gc-c*cc),cc*(f*fc-a*ac),ac,cc};
cvect={a*cc,ac*c};
cCvect={ac*c,a*cc};
(*BDmat={{Bmat,0},{0,Dmat}}*)
BDmat=Table[0,{6},{19}];
BDmat[[1;;4,1;;12]]=Bmat;
BDmat[[5;;6,13;;19]]=Dmat;
BcDcmat=Table[0,{6},{19}];
BcDcmat[[1;;4,1;;12]]=Bcmat;
BcDcmat[[5;;6,13;;19]]=Dcmat;
(*acvect={a,c}*)
acvect=Table[,{6}];
acvect[[1;;4]]=avect;
acvect[[5;;6]]=cvect;
aCcCvect=Table[,{6}];
aCcCvect[[1;;4]]=aCvect;
aCcCvect[[5;;6]]=cCvect;
\[ScriptP]=Transpose[BcDcmat].acvect;
```

```
\[ScriptP]c=Transpose[BDmat].aCcCvect;
\[ScriptP]={hc*(a*\[Xi]+c*(k*nc+l*oc))-mc*(c*\[Beta]+a*(kc*n+lc*o)),
kc*(a*\[Xi]+c*(h*mc+l*oc))-nc*(c*\[Beta]+a*(hc*m+lc*o)),
lc*(a*\[Xi]+c*(h*mc+k*nc))-oc*(c*\[Beta]+a*(hc*m+kc*n)),
hc*mc*(c*k-a*n),
hc*mc*(c*l-a*o),
(kc*mc+hc*nc)*(c*l-a*o),
(lc*mc+hc*oc)*(c*k-a*n),
(lc*nc+kc*oc)*(c*h-a*m),
kc*nc*(c*h-a*m),
lc*oc*(c*h-a*m),
kc*nc*(c*l-a*o),
lc*oc*(c*k-a*n),
a*cc*(hc*m+kc*n+lc*o)-ac*c*(h*mc+k*nc+l*oc),
a*cc*kc*m-ac*c*h*nc,
a*cc*lc*m-ac*c*h*oc,
-ac*c*k*mc+a*cc*hc*n,
-ac*c*l*mc+a*cc*hc*o,
a*cc*kc*o-ac*c*l*nc,
-ac*c*k*oc+a*cc*lc*n};
\[ScriptP]c={h*(ac*\[Xi]+cc*(kc*n+lc*o))-m*(cc*\[Beta]+ac*(k*nc+l*oc)),
k*(ac*\[Xi]+cc*(hc*m+lc*o))-n*(cc*\[Beta]+ac*(h*mc+l*oc)),
l*(ac*\[Xi]+cc*(hc*m+kc*n))-o*(cc*\[Beta]+ac*(h*mc+k*nc)),
h*m*(cc*kc-ac*nc),
h*m*(cc*lc-ac*oc),
(k*m+h*n)*(cc*lc-ac*oc),
(l*m+h*o)*(cc*kc-ac*nc),
(l*n+k*o)*(cc*hc-ac*mc),
k*n*(cc*hc-ac*mc),
l*o*(cc*hc-ac*mc),
k*n*(cc*lc-ac*oc),
l*o*(cc*kc-ac*nc),
ac*c*(h*mc+k*nc+l*oc)-a*cc*(hc*m+kc*n+lc*o),
ac*c*k*mc-a*cc*hc*n,
ac*c*l*mc-a*cc*hc*o,
-a*cc*kc*m+ac*c*h*nc,
-a*cc*lc*m+ac*c*h*oc,
ac*c*k*oc-a*cc*lc*n,
-a*cc*kc*o+ac*c*l*nc};
\[Beta]\[Xi]subs={\[Beta]->f*fc-a*ac-h*hc-k*kc-l*lc,\[Xi]->g*gc-c*cc-m*mc-
   n*nc-o*oc};
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
   i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
qvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]],Q[[j]]*S[[j]],Q[[j]]^2,S[[j]]^2,Q[[j]]^2*Sc[[j]],Qc[[j]]*S[[j]]^2},{
```

```
   j,npos-1}];
qpvect=Table[{1,Q[[j]],S[[j]],Qc[[j]],Sc[[j]],Q[[j]]*Sc[[j]],Qc[[j]]*S[[j
   ]]},{j,npos-1}];
qcvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]],Qc[[j]]*Sc[[j]],Qc[[j]]^2,Sc[[j]]^2,Qc[[j]]^2*S[[j]],Q[[j]]*Sc[[j
   ]]^2},{j,npos-1}];
qcpvect=Table[{1,Qc[[j]],Sc[[j]],Q[[j]],S[[j]],Qc[[j]]*S[[j]],Q[[j]]*Sc[[j
   ]]},{j,npos-1}];
isoSimplify[expression_,isopairs_]:=Module[{isoSubs,expression2},
expression2=expression;
isoSubs=Flatten[Table[{exp=i+1-j,expc=j}->{If[exp-expc>0,exp-expc,0],If[
   expc-exp>0,expc-exp,0]},{i,5},{j,Range[i]}]];
Do[
expression2=CoefficientRules[expression2,isopairs[[i]]];
expression2=expression2/.isoSubs;
expression2=FromCoefficientRules[expression2,isopairs[[i]]];
,{i,Length[isopairs]}];
expression2]
(*Qmat=qcvect.qvect\[Transpose]*)
Qmat=Table[Outer[Times,qcvect[[j]],qvect[[j]]],{j,npos-1}];
Qmat=isoSimplify[Qmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
Qpmat=Table[Outer[Times,qcpvect[[j]],qpvect[[j]]],{j,npos-1}];
Qpmat=isoSimplify[Qpmat,Riffle[Thread[{Q,Qc}],Thread[{S,Sc}]]];
(*QmatBIG={{Qmat,0},{0,-Qpmat}}*)
QmatBIG=Table[0,{npos-1},{19},{19}];
Do[
QmatBIG[[j,1;;12,1;;12]]=Qmat[[j]];
QmatBIG[[j,13;;19,13;;19]]=-Qpmat[[j]];
,{j,npos-1}];
eqns3=Table[\[ScriptP].QmatBIG[[j]].\[ScriptP]c,{j,npos-1}];
(*The final product is p, pc, and QmatBIG[[j]]*)
eqns4=Table[(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)
   *(mc+Qc[[j]]*nc+Sc[[j]]*oc))-c*(mc+Qc[[j]]*nc+Sc[[j]]*oc)*(f*fc-a*ac-(h
   +Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc[[j]]*lc)))*(ac*(h+Q[[j]]*k+S[[j
   ]]*l)*(g*gc-c*cc-(m+Q[[j]]*n+S[[j]]*o)*(mc+Qc[[j]]*nc+Sc[[j]]*oc))-cc*(
   m+Q[[j]]*n+S[[j]]*o)*(f*fc-a*ac-(h+Q[[j]]*k+S[[j]]*l)*(hc+Qc[[j]]*kc+Sc
   [[j]]*lc)))+(a*(hc+Qc[[j]]*kc+Sc[[j]]*lc)*cc*(m+Q[[j]]*n+S[[j]]*o)-ac*(
   h+Q[[j]]*k+S[[j]]*l)*c*(mc+Qc[[j]]*nc+Sc[[j]]*oc))^2,{j,npos-1}];
Substitutions for SII, SIII, and WII
SIIsynthsubs={a->G-CC,
ac->Gc-CCc,
(*b->A-B+Q*(CC-A)-S*(DD-B),
bc->Ac-Bc+Qc*(CCc-Ac)-Sc*(DDc-Bc),*)
c->H-CC,
cc->Hc-CCc,
(*d->A-B+Q*(CC-A)-S*(F-B),
dc->Ac-Bc+Qc*(CCc-Ac)-Sc*(Fc-Bc),*)
f->G-DD,
fc->Gc-DDc,
g->H-F,
gc->Hc-Fc,
```

```
h->A-B,
hc->Ac-Bc,
k->CC-A,
kc->CCc-Ac,
l->-(DD-B),
lc->-(DDc-Bc),
m->A-B,
mc->Ac-Bc,
n->CC-A,
nc->CCc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
SIIIsynthsubs={a->G-DD,
ac->Gc-DDc,
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->H-DD,
cc->Hc-DDc,
(*d->A-B+Q*(DD-A)-S*(F-B),
dc->Ac-Bc+Qc*(DDc-Ac)-Sc*(Fc-Bc),*)
f->G-CC,
fc->Gc-CCc,
g->H-F,
gc->Hc-Fc,
h->A-CC,
hc->Ac-CCc,
k->DD-A,
kc->DDc-Ac,
l->0,
lc->0,
m->A-B,
mc->Ac-Bc,
n->DD-A,
nc->DDc-Ac,
o->-(F-B),
oc->-(Fc-Bc)};
WIIsynthsubs={a->-(G-CC),
ac->-(Gc-CCc),
(*b->A-CC+Q*(DD-A),
bc->Ac-CCc+Qc*(DDc-Ac),*)
c->-(H-CC),
cc->-(Hc-CCc),
(*d->B-CC+S*(F-B),
dc->Bc-CCc+Sc*(Fc-Bc),*)
f->(G-DD),
fc->(Gc-DDc),
g->(H-F),
gc->(Hc-Fc),
h->A-CC,
hc->Ac-CCc,
k->DD-A,
```

```
kc->DDc-Ac,
l->0,
lc->0,
m->B-CC,
mc->Bc-CCc,
n->0,
nc->0,
o->F-B,
oc->Fc-Bc};
Synthesis Equations (Truncated)
(*Synthesis Equations from general form*)
SIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIsynthsubs),{j,npos-1}];
SIIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.SIIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.SIIIsynthsubs),{j,npos-1}];
WIIeqns1=Table[(\[ScriptP]/.\[Beta]\[Xi]subs/.WIIsynthsubs).QmatBIG[[j
   ]].(\[ScriptP]c/.\[Beta]\[Xi]subs/.WIIsynthsubs),{j,npos-3}];
(*Synthesis equations from compact form*)
SIIeqns2=eqns4/.SIIsynthsubs;
SIIIeqns2=eqns4/.SIIIsynthsubs;
WIIeqns2=eqns4[[1;;8]]/.WIIsynthsubs;
Task Specification
(*Images here*)
pathtype="SIIIFunc2SIPath";
(*Acceptable pathtype
"WIIFunc2WIPath"
"SIIIFunc2SIPath"
"SIIIFunc2SIIPath"
"SIIFunc2SIIPath"
"SIIFunc2SIIIPath"*)
InvKin2R[K_,l1_,l2_,P_]:=Module[{Kx,Ky,Px,Py,\[Nu]\[Nu],\[Zeta]\[Zeta]},
(*The output is the 2 configurations of the 2R chain*)
(*{Ax,Ay}={Re[A],Im[A]};
{px,py}={Re[p],Im[p]};*)
{Kx,Ky}=K;
{Px,Py}=P;
\[Nu]\[Nu]=Map[ArcTan[Kx-Px,Ky-Py]+#*ArcCos[((Kx-Px)^2+(Ky-Py)^2+l1^2-l2
   ^2)/(-2*l1*Sqrt[(Kx-Px)^2+(Ky-Py)^2])]&,{+1,-1}];
\[Zeta]\[Zeta]=Map[ArcTan[Px-Kx-l1*Cos[#],Py-Ky-l1*Sin[#]]-#&,\[Nu]\[Nu]];
(*MapThread[{V->Exp[I*#1],Z->Exp[I*#2]}&,{\[Nu]\[Nu],\[Zeta]\[Zeta]}]*)
MapThread[{\[Nu]->#1,\[Zeta]->#2}&,{\[Nu]\[Nu],\[Zeta]\[Zeta]}]]
InvKin2RAlt[\[ScriptCapitalK]_,l1_,l2_,\[ScriptCapitalP]_]:=Module[{K,P,Z,
   V},
{K,P}=Apply[#1+#2*I&,{\[ScriptCapitalK],\[ScriptCapitalP]},1];
Z=Map[1/(2*l1*l2)*((P-K)*(P\[Conjugate]-K\[Conjugate])-l1^2-l2^2+#*Sqrt[((
   P-K)*(P\[Conjugate]-K\[Conjugate])-l1^2-l2^2)^2-4*l1^2*l2^2])
   &,{+1,-1}];
V=Map[(P-K)/(l1+#*l2)&,Z];
MapThread[{\[Nu]->Arg[#1],\[Zeta]->Arg[#2]}&,{V,Z}]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
```

```
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Kpt={0,0};
l1=2;
l2=2.5;
taskpoints={(*1*){-1.7,-4},
(*2*){-1.2,-4},
(*3*){-.6,-4},
(*4*){0,-4},
(*5*){.6,-4},
(*6*){1.2,-4},
(*7*){1.7,-4},
(*8*){1.7,-3.5},
(*9*){.5,-3.3},
(*10*){-.5,-3.3},
(*11*){-1.7,-3.5}};
\[Nu]\[Zeta]angles=Map[InvKin2R[Kpt,l1,l2,#]&,taskpoints];*)
(*A*)
Kpt={-0,0};
l1=2;
l2=2.5;
taskpoints={(*1*){-1.75,-4},
(*2*){-1.2,-4},
(*3*){-.6,-4},
(*4*){0,-4},
(*5*){.6,-4},
(*6*){1.2,-4},
(*7*){1.75,-4},
(*8*){1.4,-3.7},
(*9*){.3,-3.3},
(*10*){-.9,-3.1},
(*11*){-1.6,-3.4}};
\[Nu]\[Zeta]angles=Map[InvKin2RAlt[Kpt,l1,l2,#]&,taskpoints];
(*(*B*)
Kpt={-0,0};
l1=2;
l2=2.5;
taskpoints={(*1*){-1.75,-4},
```

```
(*2*){-1.2,-4},
(*3*){-.6,-4},
(*4*){0,-4},
(*5*){.6,-4},
(*6*){1.2,-4},
(*7*){1.75,-4},
(*8*){1.6,-3.6},
(*9*){.6,-3.4},
(*10*){-.6,-3.4},
(*11*){-1.6,-3.6}};
\[Nu]\[Zeta]angles=Map[InvKin2R[Kpt,l1,l2,#]&,taskpoints];*)
(*(*C*)
Kpt={-0,0};
l1=2;
l2=2.5;
taskpoints={(*1*){-1.75,-4},
(*2*){-1.125,-4},
(*3*){-.5,-4},
(*4*){.125,-4},
(*5*){.75,-4},
(*6*){1.375,-4},
(*7*){2,-4},
(*8*){1,-2.7},
(*9*){-.6,-2},
(*10*){-2.4,-1.9},
(*11*){-2.3,-3}};
\[Nu]\[Zeta]angles=Map[InvKin2R[Kpt,l1,l2,#]&,taskpoints];*)
Manipulate[
\[ScriptCapitalK]=Kpt;
\[ScriptCapitalL]Up=Kpt+l1*{Cos[\[Nu]],Sin[\[Nu]]}/.\[Nu]\[Zeta]angles[[
    pos,1]];
\[ScriptCapitalL]Down=Kpt+l1*{Cos[\[Nu]],Sin[\[Nu]]}/.\[Nu]\[Zeta]angles[[
    pos,2]];
\[ScriptCapitalP]=taskpoints[[pos]];
mainPlot=Graphics[{
Thick,Red,Line[{\[ScriptCapitalK],\[ScriptCapitalL]Up,\[ScriptCapitalP]}],
    Blue,Line[{\[ScriptCapitalK],\[ScriptCapitalL]Down,\[ScriptCapitalP]}],
GPivot[\[ScriptCapitalK],.05],MPivot[\[ScriptCapitalL]Up,.05],MPivot[\[
    ScriptCapitalL]Down,.05],
Black,PointSize[Large],Point[taskpoints]
},Axes->True,PlotRange->{{-4,4},{-4.5,1}},ImageSize->400];
\[Nu]\[Zeta]func1=Map[{Mod[\[Nu],2*\[Pi],-\[Pi]*3/2],Mod[\[Zeta],2*\[Pi
    ],-\[Pi]]}/.#&,\[Nu]\[Zeta]angles[[All,1]]];
\[Nu]\[Zeta]func2=Map[{Mod[\[Nu],2*\[Pi],-\[Pi]*3/2],Mod[\[Zeta],2*\[Pi
    ],-\[Pi]]}/.#&,\[Nu]\[Zeta]angles[[All,2]]];
secondaryPlot=ListPlot[{\[Nu]\[Zeta]func1,\[Nu]\[Zeta]func2,{\[Nu]\[Zeta]
    func1[[pos]],\[Nu]\[Zeta]func2[[pos]]}},PlotMarkers->{},PlotStyle->{
    Directive[Red,PointSize[Medium]],Directive[Blue,PointSize[Medium]],
    Directive[Green,PointSize[Large]]},ImageSize->400,AxesLabel->{\[Nu],\[
    Zeta]},Joined->{True,True,False}];
Row[{mainPlot,secondaryPlot},"␣␣␣"]
```

```
,{pos,1,npos,1}]
P=Table[ToExpression["P"<>ToString[j]],{j,0,npos-1}];
tasksubsP=MapThread[#1->{1,I}.#2&,{P,taskpoints}];
(*Choose elbow up or elbow down of 2R here.  Choose 1 for Up (red) and 2
   for Down (blue)*)
UpOrDown=2;
Chain2Rsubs={K->{1,I}.Kpt,L->{1,I}.(Kpt+l1*{Cos[\[Nu]],Sin[\[Nu]]}),First[
   tasksubsP]}/.\[Nu]\[Zeta]angles[[1,UpOrDown]];
\[CapitalDelta]\[Nu]\[Zeta]angles=Map[{\[CapitalDelta]\[Nu]->(\[Nu]/.#)
   -(\[Nu]/.\[Nu]\[Zeta]angles[[1,UpOrDown]]),\[CapitalDelta]\[Zeta]->(\[
   Zeta]/.#)-(\[Zeta]/.\[Nu]\[Zeta]angles[[1,UpOrDown]])}&,\[Nu]\[Zeta]
   angles[[2;;-1,UpOrDown]]];
(*For custom elbow up/down specification*)
\[CapitalDelta]\[Nu]\[Zeta]angles=Map[{\[CapitalDelta]\[Nu]->(\[Nu]/.#)
   -(\[Nu]/.\[Nu]\[Zeta]angles[[1,UpOrDown]]),\[CapitalDelta]\[Zeta]->(\[
   Zeta]/.#)-(\[Zeta]/.\[Nu]\[Zeta]angles[[1,UpOrDown]])}&,
Join[\[Nu]\[Zeta]angles[[2;;7,2]],\[Nu]\[Zeta]angles[[8;;11,1]]]];
(*Choose what kind of path generator you will be creating here from the
   above choices*)
tasksubs\[Phi]\[Psi]=Which[pathtype=="WIIFunc2WIPath",Map[{\[Psi]->-\[
   CapitalDelta]\[Nu],\[Phi]->\[CapitalDelta]\[Zeta]}/.#&,\[CapitalDelta
   ]\[Nu]\[Zeta]angles],
pathtype=="SIIIFunc2SIPath",Map[{\[Phi]->-\[CapitalDelta]\[Nu],\[Psi]->\[
   CapitalDelta]\[Zeta]}/.#&,\[CapitalDelta]\[Nu]\[Zeta]angles],
pathtype=="SIIIFunc2SIIPath",Map[{\[Psi]->-\[CapitalDelta]\[Nu],\[Phi]->\[
   CapitalDelta]\[Zeta]}/.#&,\[CapitalDelta]\[Nu]\[Zeta]angles],
pathtype=="SIIFunc2SIIPath",Map[{\[Phi]->-\[CapitalDelta]\[Nu],\[Psi]->\[
   CapitalDelta]\[Zeta]}/.#&,\[CapitalDelta]\[Nu]\[Zeta]angles],
pathtype=="SIIFunc2SIIIPath",Map[{\[Psi]->-\[CapitalDelta]\[Nu],\[Phi]->\[
   CapitalDelta]\[Zeta]}/.#&,\[CapitalDelta]\[Nu]\[Zeta]angles],
True,Print["Error"];Abort[]];
tasksubsQS=MapIndexed[{Q[[#2[[1]]]]->Exp[\[Phi]*I],S[[#2[[1]]]]->Exp[\[Psi
   ]*I]}/.#&,tasksubs\[Phi]\[Psi]];
tasksubsQS=Flatten[tasksubsQS];
tasksubsQS=tasksubsQS~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
   Conjugate]]]/.tasksubsQS);
points2graph={{0,0}}~Join~MapThread[{Arg[#1],Arg[#2]}/.tasksubsQS&,{Q,S}];
ListLinePlot[points2graph,PlotMarkers->{},PlotStyle->Directive[Red,
   PointSize[Large]]]
finalParameters1=Map[SetPrecision[{Re[#],Im[#]},300]&,Join[Q,S]/.
   tasksubsQS];
finalParameters2=finalParameters1;
finalParameters3=Map[ToString[InputForm[#,NumberMarks:>False]]&,
   finalParameters2,{2}];
finalParameters4=Map[StringReplace[#,"*^"~~a:(Shortest[___]~~NumberString)
   :>"e"<>ToString[a]]&,finalParameters3,{2}];
finalParameters5=Map[StringReplace[#,Shortest[("SetAccuracy[0,"~~__~~"]")
   ]->"0"]&,finalParameters4,{2}];
finalParameters6=Apply[#1<>"␣"<>#2&,finalParameters5,1];
finalParameters7=Prepend[finalParameters6,Length[finalParameters6]];
MatrixForm[finalParameters5]
```

```
TypeandTask={pathtype,tasksubsP,Chain2Rsubs};
Task Specification Export (do not execute)
directoryname="C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
   Synthesis\\Function\\Stephenson␣II\\Homotopies\\11␣Position\\140728
   funcgenstephII3f\\Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="E:\\Dropbox\\Mathematica\\Six-bar␣Synthesis\\Function\\
   Stephenson␣II\\Homotopies\\11␣Position\\140728funcgenstephII3f\\
   Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="C:\\Users\\kinematica\\Dropbox\\Mathematica\\Six-bar␣
   Synthesis\\Function\\Stephenson␣III\\Homotopies\\11␣Position\\141029
   funcgenstephIII-2\\Parameter␣Homotopies\\";
SetDirectory[directoryname];
directoryname="E:\\Dropbox\\Mathematica\\Six-bar␣Synthesis\\Function\\
   Stephenson␣III\\Homotopies\\11␣Position\\141029funcgenstephIII-2\\
   Parameter␣Homotopies\\";
SetDirectory[directoryname];
foldername="150318pathSIII2SIwalkerA\\";
If[DirectoryQ[foldername],Print["Folder␣name␣already␣exists"]];
jobname="SIII2SIA";
CreateDirectory[foldername];
Export[foldername<>"final_parameters.txt",finalParameters7];
RenameFile[foldername<>"final_parameters.txt",foldername<>"
   final_parameters"];
CopyFile["..\\start_parameters",foldername<>"start_parameters"];
CopyFile["input_specific",foldername<>jobname<>"_specific"];
jobsh={"#!/bin/bash",
"#$␣-N␣"<>jobname,
"#$␣-q␣free64",
"#$␣-pe␣one-node-mpi␣64",
"#$␣-m␣beas",
"",
"module␣load␣openmpi-1.6.0/gcc-4.7.3",
"module␣load␣Bertini/1.4",
"",
"mpirun␣bertini␣"<>jobname<>"_specific␣../general_solutions␣>␣"<>jobname<>
   "_specific_out"};
Export[foldername<>jobname<>".txt",jobsh];
RenameFile[foldername<>jobname<>".txt",foldername<>jobname<>".sh"];
TypeandTask>>directoryname<>foldername<>"TypeandTask"
```

# H Mathematica code: Path Generation Analysis for Stephenson II Inversions

```
(*Type and Task*)
(*SetDirectory[NotebookDirectory[]];*)
{pathtype,tasksubsP,Chain2Rsubs}=<<"TypeandTask";
(*Acceptable pathtype
"WIIFunc2WIPath"
"SIIIFunc2SIPath"
"SIIIFunc2SIIPath"
"SIIFunc2SIIPath"
"SIIFunc2SIIIPath"*)
taskpoints=Map[{Re[#[[2]]],Im[#[[2]]]}&,tasksubsP];
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
   i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
   Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
   Conjugate]]]/.params6);
(*Import Bertini Results for SIIFunc*)
(*If[FileExistsQ["ProgressReport.txt"],Interrupt[]]*)
begin=DateString[];
ProgressReport={begin};
AppendTo[ProgressReport,""];
Solns1=ReadList["nonsingular_solutions",Number];
Solns4=Partition[Rest[Solns1],2*10];
Solns1=.
Solns5=Partition[#,2]&/@Solns4;
Solns4=.
Solns6=Parallelize[Apply[#1+#2*I&,Solns5,{2}]];
Solns5=.
(*All solutions properly formatted*)
Solns7=Thread[{CC,CCc,DD,DDc,F,Fc,G,Gc,H,Hc}->#]&/@Solns6;
```

```
Solns6=.
Length[Solns7];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns7)␣imported␣and␣
    formatted:␣"<>ToString[Length[Solns7]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-2)*)
Solns8=Cases[Solns7,x_/;Round[{CCc,DDc,Fc,Gc,Hc}/.x,10^-2]==Round[{CC\[
    Conjugate],DD\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
    ,10^-2]];
Length[Solns8];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-8)*)
Solns9=Cases[Solns8,x_/;Round[{CCc,DDc,Fc,Gc,Hc}/.x,10^-8]==Round[{CC\[
    Conjugate],DD\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
    ,10^-8]];
Length[Solns9];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns9)␣that␣pertain␣to␣
    physical␣linkages:␣"<>ToString[Length[Solns9]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Pair solutions with their symmetric counterpart*)
temp=Solns9;
Solns10={};
While[Length[temp]>0,
pos=Position[temp,x_/;(Round[{CC,DD,F,G,H}/.x,10^-8]==Round[{CC,DD,F,G,H
    }/.First[temp],10^-8])||(Round[{CC,F,DD,H,G}/.x,10^-8]==Round[{CC,DD,F,
    G,H}/.First[temp],10^-8]),1,Heads->False];
AppendTo[Solns10,temp[[pos[[All,1]]]]];
temp=Delete[temp,pos];];
table10=Prepend[Sort[Tally[Length/@Solns10]],{"No.␣of␣redundancies","No.␣
    of␣soln␣groups"}];
MatrixForm[table10];
AppendTo[ProgressReport,"Singletons␣and␣solutions␣with␣their␣symmetric␣
    counterparts"];
AppendTo[ProgressReport,ToString[Grid[table10]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Take only one of each symmetric pair, or just the singleton*)
Solns10=Solns10[[All,1]];
Length[Solns10];
Solns10>>"Solns10"
(*Import from here*)
Solns10=<<"Solns10";
Length[Solns10];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns10)␣with␣one␣of␣each␣
    symmetric␣redundant␣pair␣removed:␣"<>ToString[Length[Solns10]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Gp=DD-G+CC;
Hpp=F-H+CC;
Hp=(DD-G)/(H-G)*(H-CC)+CC;
```

```
Gpp=(F-H)/(G-H)*(G-CC)+CC;
Fp=((DD-G)*(H-CC)-(F-H)*(G-CC))/(H-G)+CC;
ConstructCognates[Soln_]:=Module[{values,cognates},
values={{CC,DD,G,H,F},{CC,DD,Gp,Hp,Fp},{CC,Fp,Gpp,Hpp,F}}/.Soln;
cognates=Apply[{CC->#1,DD->#2,G->#3,H->#4,F->#5,CCc->#1\[Conjugate],DDc
   ->#2\[Conjugate],Gc->#3\[Conjugate],Hc->#4\[Conjugate],Fc->#5\[
   Conjugate]}&,values,2];
cognates]
(*Group solutions into cognate sets of 3*)
temp=Solns10;
Solns12={};
acc=10^-8;
While[Length[temp]>0,
pos=Position[temp,
x_/;(Round[{CC}/.x,acc]==Round[{CC}/.First[temp],acc])
&&(Round[{DD,G,H,F}/.x,acc]==Round[{DD,G,H,F}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{F,H,G,DD}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{DD,Gp,Hp,Fp}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{Fp,Hp,Gp,DD}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{Fp,Gpp,Hpp,F}/.First[temp],acc]
||Round[{DD,G,H,F}/.x,acc]==Round[{F,Hpp,Gpp,Fp}/.First[temp],acc])
,1,Heads->False];
AppendTo[Solns12,temp[[pos[[All,1]]]]];
temp=Delete[temp,pos];];
table12=Prepend[Sort[Tally[Length/@Solns12]],{"No.␣of␣cognates","No.␣of␣
   soln␣groups"}];
MatrixForm[table12];
Length[Solns12];
AppendTo[ProgressReport,"Number␣of␣cognates␣groups␣(Solns12):␣"<>ToString[
   Length[Solns12]]];
AppendTo[ProgressReport,"Groups␣of␣Cognates"];
AppendTo[ProgressReport,ToString[Grid[table12]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*For groups of 1 and 2, add their missing cognates, place all solutions
   in a properly flattened list*)
Solns13Group1=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==1][[All
   ,1]]];
Solns13Group2=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==2][[All
   ,1]]];
Solns13Group3=Cases[Solns12,x_/;Length[x]==3];
Solns13Group4=Cases[Solns12,x_/;Length[x]>=4];
Solns13=Flatten[Join[Solns13Group1,Solns13Group2,Solns13Group3,
   Solns13Group4],1];
Length[Solns13];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns13)␣after␣constructed␣
   cognates␣were␣added:␣"<>ToString[Length[Solns13]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a maximum on the size of link length*)
maxlinklength=5;
```

```
Solns14=Cases[Solns13,x_/;Map[Norm[#/.{A->0,B->1}/.x]<maxlinklength&,{CC-A
    ,DD-B,F-B,F-DD,G-CC,H-CC,H-G,G-DD,H-F}]==Table[True,{9}]];
Length[Solns14];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns14)␣after␣long␣link␣
    length␣linkages␣were␣removed:␣"<>ToString[Length[Solns14]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a minimum on the size of link length*)
minlinklength=.05;
Solns15=Cases[Solns14,x_/;Map[Norm[#/.{A->0,B->1}/.x]>minlinklength&,{CC-A
    ,DD-B,F-B,F-DD,G-CC,H-CC,H-G,G-DD,H-F}]==Table[True,{9}]];
Length[Solns15];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns15)␣after␣small␣link␣
    length␣linkages␣were␣removed:␣"<>ToString[Length[Solns15]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Path Generator Conversion Functions*)
SegmentStretchRotateTranslate[{A_,B_},{Ap_,Bp_},p_]:=(Bp-Ap)/(B-A)*(p-A)+
    Ap
(*(A,B) are endpoints of initial line segment
(Ap,Bp) are endpoints of final line segment
p is fixed frame coordinates on initial line segment
Output is fixed frame coordinates on final line segment*)
WIIFunc2WIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,pathgendim
    ,pathgendimc},
DIM=dim~Join~{CC->0,G->1,H->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
    chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][DD]/.DIM),
CC->(\[ScriptCapitalT][CC]/.DIM),
DD->(\[ScriptCapitalT][G]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][H]/.DIM),
```

```
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
    chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][CC]/.DIM),
CC->L/.chaindim,
DD->(\[ScriptCapitalT][G]/.DIM),
F->(\[ScriptCapitalT][H]/.DIM),
G->(\[ScriptCapitalT][DD]/.DIM),
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][DD]/.DIM),
DD->(\[ScriptCapitalT][H]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
```

```
H->(\[ScriptCapitalT][CC]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
AddTracePoint[type_,dim_,branchesforsinglelinkage_]:=Module[{j,vars,
    Pexpression,brancheswithP},
If[MemberQ[{"WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
    SIIFunc2SIIPath","SIIFunc2SIIIPath"},type],,Print["Error"];Abort[]];
(*Beginning is in order to accommodate linkages that "Timed Out"*)
j=1;While[j<=Length[branchesforsinglelinkage],
If[Length[branchesforsinglelinkage[[j]]]!=0,
vars=branchesforsinglelinkage[[j,1,All,1]];Break[],j++]];
If[j>Length[branchesforsinglelinkage],Goto["end"]];
Goto[type];
(*****)Label["WIIFunc2WIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
If[Length[{QQ,SS,TT}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(F-A)+TT*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(H-F)+RR*(P0-H)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
If[Length[{QQ,TT,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)/.dim;
Goto["end"];
(*****)
Label["end"];
brancheswithP=Map[Append[#,PP->Pexpression/.#]&,branchesforsinglelinkage
    ,{2}]]
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
```

```
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
   branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
   branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
   CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
   *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
   DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
   CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
   CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
   *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
   RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
   SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
   -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
   SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
   Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
   branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
   ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
   coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
   figures in my dissertation*)
(*outvar -this should be a list of variables to appear in the output
   configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
   (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
   invar is a valid parameter*)
```

```
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
    *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
    Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
    SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
    ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
    ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
    into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
    log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
    NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
    *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
    position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
```

```
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
    All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
    /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]
]];
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
```

```
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
    removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
    sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
    ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
    chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
    ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
    branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
    tasksubsQS]
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
    Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
    \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
    Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
    [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
```

```
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
   ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
   ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
   ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
   tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
   checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
   taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
   configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
   the complex planes of individual variables for determining whether or
   not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
   ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
   each checkvar in each configuration as a percent of that variables
   range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
   between two configurations for each checkvar as a percent of that
   variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
   nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
   variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
   ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
   check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
   ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
   tolRectsPercent],,Print["Error"];Abort[]];
```

```
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
   ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
   indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
   )*)
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
   matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
   ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
   ]],"error"]
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
   error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
   ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
   ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
   WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
   SIIFunc2SIIPath","SIIFunc2SIIIPath"};
```

```
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
   ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,DD->1};
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.{A->0,DD->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.{A->0,DD->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.{A->0,DD->1}/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalD],\[ScriptCapitalG
   ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
```

```
   ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD],\[
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalA],\[ScriptCapitalC
   ],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
```

```
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
   ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH],\[
   ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[ScriptCapitalP
   ]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalD],\[ScriptCapitalG
```

```
   ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
   ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
   ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
   inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
   line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
   ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
   Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
   ->500];
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
```

```
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Analysis Execution*)
Solns16=Which[
pathtype=="WIIFunc2WIPath",WIIFunc2WIPath[#,Chain2Rsubs]&/@Solns15,
pathtype=="SIIIFunc2SIPath",SIIIFunc2SIPath[#,Chain2Rsubs]&/@Solns15,
pathtype=="SIIIFunc2SIIPath",SIIIFunc2SIIPath[#,Chain2Rsubs]&/@Solns15,
pathtype=="SIIFunc2SIIPath",SIIFunc2SIIPath[#,Chain2Rsubs]&/@Solns15,
pathtype=="SIIFunc2SIIIPath",SIIFunc2SIIIPath[#,Chain2Rsubs]&/@Solns15,
True,Print["Error"];Abort[]];
Which[
pathtype=="WIIFunc2WIPath",inputAngles={QQ(*,SS*)};outputAngles={RR,SS,UU
    };type="WI",
pathtype=="SIIIFunc2SIPath",inputAngles={(*QQ,*)SS};outputAngles={QQ,SS,TT
    };type="SI",
pathtype=="SIIIFunc2SIIPath",inputAngles={QQ(*,SS*)};outputAngles={RR,SS,
    UU};type="SII",
pathtype=="SIIFunc2SIIPath",inputAngles={(*QQ,*)SS};outputAngles={RR,SS,UU
    };type="SII",
pathtype=="SIIFunc2SIIIPath",inputAngles={QQ,RR(*,SS*)};outputAngles={QQ,
    TT,UU};type="SIII",
True,Print["Error"];Abort[]];
ctr1=0;
NumToRun1=Length[Solns16]*Length[inputAngles];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Solution␣to␣forward␣kinematics␣equations"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun1]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr1];
SetSharedVariable[ctrfloor1];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
    ScriptO]\[ScriptO]\[ScriptR]1];
FwdKinWithCtr[type_,dim_,invar_,outvar_]:=Module[{branches},
branches=Quiet[TimeConstrained[FwdKin[type,dim,invar,outvar],15*60,"Timed␣
    Out"],{Det::mindet,Infinity::indet,Inverse::luc,Inverse::sing,Power::
    infy,Divide::infy}];
If[branches=="Timed␣Out",Print["Time␣out␣near␣",ctr1]];
ctr1++;
ctrfloor1=Floor[ctr1,10];
If[ctrfloor1!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
    ]\[ScriptO]\[ScriptR]1,ProgressReport[[-1]]=ToString[ctrfloor1]<>"␣/␣"
    <>ToString[NumToRun1];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
    ScriptR]1=ctrfloor1;
branches]
(*Print["counter ",Dynamic[ctr1],"/",NumToRun1]
ProgressIndicator[Dynamic[ctr1],{0,NumToRun1}]*)
branchesNoTracePoint=ParallelTable[FwdKinWithCtr[type,dim,invar,
```

```
   outputAngles],{dim,Solns16},{invar,inputAngles}];
branchesNoTracePoint=Map[Flatten[#,1]&,branchesNoTracePoint];
branches=Parallelize[MapThread[AddTracePoint[pathtype,#1,#2]&,{Solns16,
   branchesNoTracePoint}]];
ProgressReport[[-1]]=ToString[ctr1]<>"␣/␣"<>ToString[NumToRun1];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
branches>>"branches"
ctr2=0;
NumToRun2=Total[Length/@branches];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Determination␣if␣task␣positions␣are␣included␣in␣
   branches"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun2]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr2];
SetSharedVariable[ctrfloor2];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   ScriptO]\[ScriptO]\[ScriptR]2];
TaskCheckWithCtr[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[
   ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT]_:"default",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[
   ScriptE]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[
   ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{
   taskconfigsUnion},
taskconfigsUnion=TaskCheck[branch,checkvar,checktask,\[ScriptT]\[ScriptO
   ]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT],\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[
   ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]];
ctr2++;
ctrfloor2=Floor[ctr2,100];
If[ctrfloor2!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
   ]\[ScriptO]\[ScriptR]2,ProgressReport[[-1]]=ToString[ctrfloor2]<>"␣/␣"
   <>ToString[NumToRun2];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
   ScriptR]2=ctrfloor2;
taskconfigsUnion]
(*Print["counter ",Dynamic[ctr2],"/",NumToRun2]
ProgressIndicator[Dynamic[ctr2],{0,NumToRun2}]*)
checkvar={PP};
checktask={tasksubsP[[All,2]]};
checktol=.02;
analyses=ParallelMap[TaskCheckWithCtr[#,checkvar,checktask,{checktol},{
   checktol}]&,branches,{2}];
ProgressReport[[-1]]=ToString[ctr2]<>"␣/␣"<>ToString[NumToRun2];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
```

```
Export["ProgressReport.txt",ProgressReport];
designs=MapThread[{#1,#2,#3}&,{Solns16,branches,analyses}];
(*designs indexed by
-dim
-branches
        -branch1
                -config1
                -config2
                -etc
        -branch2
        -etc
-analysis*)
designs>>"designs"
DeleteFile["branches"]
designsreduced=ReduceDesign[#,11]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length11=Length[designsreduced];
If[length11>0,designsreduced>>"designsreduced11"]
designsreduced=ReduceDesign[#,10]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length10=Length[designsreduced];
If[length10>0,designsreduced>>"designsreduced10"]
designsreduced=ReduceDesign[#,9]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length9=Length[designsreduced];
If[length9>0,designsreduced>>"designsreduced9"]
designsreduced=ReduceDesign[#,8]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length8=Length[designsreduced];
If[length8>0,designsreduced>>"designsreduced8"]
designsreduced=ReduceDesign[#,7]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length7=Length[designsreduced];
If[length7>0,designsreduced>>"designsreduced7"]
designsreduced=ReduceDesign[#,6]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length6=Length[designsreduced];
If[length6>0,designsreduced>>"designsreduced6"]
resultstable={{"","No.␣of␣linkages"},{"11␣position",length11},{"10␣
   position",length10},{"9␣position",length9},{"8␣position",length8},{"7␣
   position",length7},{"6␣position",length6}};
AppendTo[ProgressReport,ToString[Grid[resultstable]]];
AppendTo[ProgressReport,""];
AppendTo[ProgressReport,"Analysis␣complete!␣␣See␣designsreduced*␣files␣for
   ␣the␣linkage␣solutions."];
AppendTo[ProgressReport,""];
end=DateString[];
AppendTo[ProgressReport,end];
AppendTo[ProgressReport,""];
computationtime=DateDifference[begin,end,"Hour"][[1]];
AppendTo[ProgressReport,"The␣analysis␣computation␣took␣"<>ToString[
```

```
   computationtime]<>"␣hours."];
Export["ProgressReport.txt",ProgressReport];
(*designsreduced=<<"designsreduced6";
Length[designsreduced]*)
(*plotrange={{-60,60},{-100,40}};
pivotsize=.08;*)
(*Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
   Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
   ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Row[{
Show[Draw[pathtype,designsreduced[[des,1]],designsreduced[[des,2,br,config
   ]]],DrawTrace[designsreduced[[des,2,br]]]],
Column[{"Quantity of branches"~~Length[designsreduced[[des,2]]],
CreateAnalysisTable[designsreduced[[des,3]]],"",
Column[Cases[designsreduced[[1,1]],x_/;MemberQ[{A,B,CC,DD,F,G,H,P0},x
   [[1]]]]]}]
},"   "],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
{config,1,Length[designsreduced[[des,2,br]]],1}]]*)
```

# I Mathematica code: Path Generation Analysis for Stephenson III Inversions

```
(*Type and Task*)
(*SetDirectory[NotebookDirectory[]];*)
{pathtype,tasksubsP,Chain2Rsubs}=<<"TypeandTask";
(*Acceptable pathtype
"WIIFunc2WIPath"
"SIIIFunc2SIPath"
"SIIIFunc2SIIPath"
"SIIFunc2SIIPath"
"SIIFunc2SIIIPath"*)
taskpoints=Map[{Re[#[[2]]],Im[#[[2]]]}&,tasksubsP];
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
   i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
   Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
   Conjugate]]]/.params6);
(*Import Bertini Results for SIIIFunc*)
(*If[FileExistsQ["ProgressReport.txt"],Interrupt[]]*)
begin=DateString[];
ProgressReport={begin};
AppendTo[ProgressReport,""];
Solns1=ReadList["nonsingular_solutions",Number];
Solns4=Partition[Rest[Solns1],2*18];
(*Solns1=.*)
Solns5=Partition[#,2]&/@Solns4;
(*Solns4=.*)
Solns6=Parallelize[Apply[#1+#2*I&,Solns5,{2}]];
(*Solns5=.*)
(*All solutions properly formatted*)
Solns7=Thread[{CC,CCc,G,Gc,r1,rc1,B,Bc,F,Fc,H,Hc,r2,rc2,r3,rc3,r4,rc4
```

```
   }->#]&/@Solns6;
(*Solns6=.*)
Length[Solns7];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns7)␣imported␣and␣
   formatted:␣"<>ToString[Length[Solns7]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-2)*)
Solns8=Cases[Solns7,x_/;Round[{Bc,CCc,Fc,Gc,Hc}/.x,10^-2]==Round[{B\[
   Conjugate],CC\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-2]];
Length[Solns8];
(*Unknowns and their conjugates are truely conjugate (to accuracy 10^-8)*)
Solns9=Cases[Solns8,x_/;Round[{Bc,CCc,Fc,Gc,Hc}/.x,10^-8]==Round[{B\[
   Conjugate],CC\[Conjugate],F\[Conjugate],G\[Conjugate],H\[Conjugate]}/.x
   ,10^-8]];
Length[Solns9];
AppendTo[ProgressReport,"Number␣of␣solutions␣(Solns9)␣that␣pertain␣to␣
   physical␣linkages:␣"<>ToString[Length[Solns9]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Solns10=Map[Cases[#,x_/;MemberQ[{r1,rc1,r2,rc2,r3,rc3,r4,rc4},x[[1]]]==
   False]&,Solns9];
ConstructCognates[Soln_]:=Module[{Gcogn,Hcogn,Bcogn,Fcogn,r1cogn,r2cogn,
   r3cogn,r4cogn,cognate},
Gcogn=DD-G+CC/.DD->1/.Soln;
Hcogn=(DD-G)/(H-G)*(H-CC)+CC/.DD->1/.Soln;
Bcogn=(DD-G)/(H-G)*(B-CC)+CC/.DD->1/.Soln;
Fcogn=(DD-G)/(H-G)*(F-CC)+CC/.DD->1/.Soln;
(*r1cogn=-(Gcogn-1)*(CCc/.Soln);
r2cogn=-(Hcogn-1)*Bcogn\[Conjugate];
r3cogn=-(Hcogn-1)*(Fcogn\[Conjugate]-Bcogn\[Conjugate]);
r4cogn=Bcogn*(Fcogn\[Conjugate]-Bcogn\[Conjugate]);*)
cognate={B->Bcogn,CC->(CC/.Soln),F->Fcogn,G->Gcogn,H->Hcogn,Bc->Bcogn\[
   Conjugate],CCc->(CC/.Soln)\[Conjugate],Fc->Fcogn\[Conjugate],Gc->Gcogn
   \[Conjugate],Hc->Hcogn\[Conjugate](*,r1->r1cogn,r2->r2cogn,r3->r3cogn,
   r4->r4cogn,rc1->r1cogn\[Conjugate],rc2->r2cogn\[Conjugate],rc3->r3cogn
   \[Conjugate],rc4->r4cogn\[Conjugate]*)};
{Soln,cognate}]
(*Group solutions into cognate sets of 2*)
temp=Solns10;
Solns12={};
acc=10^-8;
While[Length[temp]>0,
pos=Position[temp,x_/;(Round[{CC}/.x,acc]==Round[{CC}/.First[temp],acc])
&&(Round[{B,F,G,H}/.x,acc]==Round[{B,F,G,H}/.ConstructCognates[First[temp
   ]][[1]],acc]
||Round[{B,F,G,H}/.x,acc]==Round[{B,F,G,H}/.ConstructCognates[First[temp
   ]][[2]],acc])
,{1},Heads->False];
AppendTo[Solns12,temp[[pos[[All,1]]]]];
```

```
temp=Delete[temp,pos];];
table12=Prepend[Sort[Tally[Length/@Solns12]],{"No. of cognates","No. of
    soln groups"}];
MatrixForm[table12];
Length[Solns12];
AppendTo[ProgressReport,"Number of cognates groups (Solns12): "<>ToString[
    Length[Solns12]]];
AppendTo[ProgressReport,"Groups of Cognates"];
AppendTo[ProgressReport,ToString[Grid[table12]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*For groups of 1, add their missing cognates, place all solutions in a
    properly flattened list*)
Solns13Group1=Map[ConstructCognates,Cases[Solns12,x_/;Length[x]==1][[All
    ,1]]];
Solns13Group2=Cases[Solns12,x_/;Length[x]==2];
Solns13Group3=Cases[Solns12,x_/;Length[x]>=3];
Solns13=Flatten[Join[Solns13Group1,Solns13Group2,Solns13Group3],1];
Length[Solns13];
AppendTo[ProgressReport,"Number of solutions (Solns13) after constructed
    cognates were added: "<>ToString[Length[Solns13]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
Solns13>>"Solns13"
(*Solns13=<<"Solns13";*)
(*Transform linkages so A=0 and B=1*)
Solns14=Map[Thread[{A,B,CC,DD,F,G,H,Ac,Bc,CCc,DDc,Fc,Gc,Hc}->(Join[{0,B,CC
    ,1,F,G,H}/B,{0,Bc,CCc,1,Fc,Gc,Hc}/Bc]/.#)]&,Solns13];
(*Place a maximum on the size of link length*)
maxlinklength=5;
Solns15=Cases[Solns14,x_/;Map[Norm[#/.x]<maxlinklength&,{CC-A,CC-B,DD-A,F-
    B,G-CC,G-DD,H-DD,H-G,H-F}]==Table[True,{9}]];
Length[Solns15];
AppendTo[ProgressReport,"Number of solutions (Solns15) after long link
    length linkages were removed: "<>ToString[Length[Solns15]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Place a minimum on the size of link length*)
minlinklength=.05;
Solns16=Cases[Solns15,x_/;Map[Norm[#/.x]>minlinklength&,{CC-A,CC-B,DD-A,F-
    B,G-CC,G-DD,H-DD,H-G,H-F}]==Table[True,{9}]];
Length[Solns16];
AppendTo[ProgressReport,"Number of solutions (Solns16) after small link
    length linkages were removed: "<>ToString[Length[Solns16]]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
(*Path Generator Conversion Functions*)
SegmentStretchRotateTranslate[{A_,B_},{Ap_,Bp_},p_]:=(Bp-Ap)/(B-A)*(p-A)+
    Ap
(*(A,B) are endpoints of initial line segment
(Ap,Bp) are endpoints of final line segment
```

```
p is fixed frame coordinates on initial line segment
Output is fixed frame coordinates on final line segment*)
WIIFunc2WIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,pathgendim
    ,pathgendimc},
DIM=dim~Join~{CC->0,G->1,H->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
    chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][DD]/.DIM),
CC->(\[ScriptCapitalT][CC]/.DIM),
DD->(\[ScriptCapitalT][G]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][H]/.DIM),
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
    Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
    pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
    DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
```

```
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
   chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][CC]/.DIM),
CC->L/.chaindim,
DD->(\[ScriptCapitalT][G]/.DIM),
F->(\[ScriptCapitalT][H]/.DIM),
G->(\[ScriptCapitalT][DD]/.DIM),
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
   chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][DD]/.DIM),
DD->(\[ScriptCapitalT][H]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][CC]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
AddTracePoint[type_,dim_,branchesforsinglelinkage_]:=Module[{j,vars,
   Pexpression,brancheswithP},
If[MemberQ[{"WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
   SIIFunc2SIIPath","SIIFunc2SIIIPath"},type],,Print["Error"];Abort[]];
(*Beginning is in order to accommodate linkages that "Timed Out"*)
j=1;While[j<=Length[branchesforsinglelinkage],
If[Length[branchesforsinglelinkage[[j]]]!=0,
vars=branchesforsinglelinkage[[j,1,All,1]];Break[],j++]];
If[j>Length[branchesforsinglelinkage],Goto["end"]];
Goto[type];
(*****)Label["WIIFunc2WIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
```

```
If[Length[{QQ,SS,TT}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(F-A)+TT*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(H-F)+RR*(P0-H)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
If[Length[{QQ,TT,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)/.dim;
Goto["end"];
(*****)
Label["end"];
brancheswithP=Map[Append[#,PP->Pexpression/.#]&,branchesforsinglelinkage
    ,{2}]]
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
    branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
    *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
```

```
    DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
    CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
    *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
    RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
    SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
    -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
    SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
    Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
    branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
    ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
    coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
    figures in my dissertation*)
(*outvar -this should be a list of variables to appear in the output
    configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
    (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
    invar is a valid parameter*)
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
    *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
    Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
    SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
    ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
```

```
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
   branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
   ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
   into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
   log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
   NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
   *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
   position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
   All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
   /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
```

```
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]]
]];
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
    removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
    sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
    ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
```

```
    chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
    ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
    branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
    tasksubsQS]
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
    Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
    \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
    Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
    [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
    ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
    ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
    ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
    tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
    checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
    taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
    configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
    the complex planes of individual variables for determining whether or
    not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
    ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
    each checkvar in each configuration as a percent of that variables
```

```
   range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
   ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
   ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
   between two configurations for each checkvar as a percent of that
   variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
   nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
   variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
   ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
   check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
   ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
   tolRectsPercent],,Print["Error"];Abort[]];
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
   ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
   indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
   )*)
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
```

```
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
   matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
   ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
   ]],"error"]
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
   error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
   ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
   ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
   WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
   SIIFunc2SIIPath","SIIFunc2SIIIPath"};
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
   ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
```

```
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{\[ScriptCapitalD],\[ScriptCapitalG
   ],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD],\[
```

```
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalA],\[ScriptCapitalC
   ],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
   ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[ScriptCapitalP
   ]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
```

```
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],Polygon[{{\[ScriptCapitalC],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH],\[
   ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[ScriptCapitalP
   ]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[ScriptCapitalD],\[
   ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],Polygon[{{\[ScriptCapitalD],\[ScriptCapitalG
   ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
   ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
   ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
   inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
```

```
   line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
   ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
   Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
   ->500];
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Analysis Execution*)
Solns17=Which[
pathtype=="WIIFunc2WIPath",WIIFunc2WIPath[#,Chain2Rsubs]&/@Solns16,
pathtype=="SIIIFunc2SIPath",SIIIFunc2SIPath[#,Chain2Rsubs]&/@Solns16,
pathtype=="SIIIFunc2SIIPath",SIIIFunc2SIIPath[#,Chain2Rsubs]&/@Solns16,
pathtype=="SIIFunc2SIIPath",SIIFunc2SIIPath[#,Chain2Rsubs]&/@Solns16,
pathtype=="SIIFunc2SIIIPath",SIIFunc2SIIIPath[#,Chain2Rsubs]&/@Solns16,
True,Print["Error"];Abort[]];
Which[
pathtype=="WIIFunc2WIPath",inputAngles={QQ(*,SS*)};outputAngles={RR,SS,UU
   };type="WI",
pathtype=="SIIIFunc2SIPath",inputAngles={(*QQ,*)SS};outputAngles={QQ,SS,TT
   };type="SI",
pathtype=="SIIIFunc2SIIPath",inputAngles={QQ(*,SS*)};outputAngles={RR,SS,
   UU};type="SII",
pathtype=="SIIFunc2SIIPath",inputAngles={(*QQ,*)SS};outputAngles={RR,SS,UU
```

```
   };type="SII",
pathtype=="SIIFunc2SIIIPath",inputAngles={QQ,RR(*,SS*)};outputAngles={QQ,
   TT,UU};type="SIII",
True,Print["Error"];Abort[]];
ctr1=0;
NumToRun1=Length[Solns17]*Length[inputAngles];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Solution␣to␣forward␣kinematics␣equations"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun1]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr1];
SetSharedVariable[ctrfloor1];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   Script0]\[Script0]\[ScriptR]1];
FwdKinWithCtr[type_,dim_,invar_,outvar_]:=Module[{branches},
branches=Quiet[TimeConstrained[FwdKin[type,dim,invar,outvar],15*60,"Timed␣
   Out"],{Det::mindet,Infinity::indet,Inverse::luc,Inverse::sing,Power::
   infy,Divide::infy}];
If[branches=="Timed␣Out",Print["Time␣out␣near␣",ctr1]];
ctr1++;
ctrfloor1=Floor[ctr1,10];
If[ctrfloor1!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[Script0
   ]\[Script0]\[ScriptR]1,ProgressReport[[-1]]=ToString[ctrfloor1]<>"␣/␣"
   <>ToString[NumToRun1];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[Script0]\[Script0]\[
   ScriptR]1=ctrfloor1;
branches]
(*Print["counter ",Dynamic[ctr1],"/",NumToRun1]
ProgressIndicator[Dynamic[ctr1],{0,NumToRun1}]*)
branchesNoTracePoint=ParallelTable[FwdKinWithCtr[type,dim,invar,
   outputAngles],{dim,Solns17},{invar,inputAngles}];
branchesNoTracePoint=Map[Flatten[#,1]&,branchesNoTracePoint];
branches=Parallelize[MapThread[AddTracePoint[pathtype,#1,#2]&,{Solns17,
   branchesNoTracePoint}]];
ProgressReport[[-1]]=ToString[ctr1]<>"␣/␣"<>ToString[NumToRun1];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
branches>>"branches"
ctr2=0;
NumToRun2=Total[Length/@branches];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,"Determination␣if␣task␣positions␣are␣included␣in␣
   branches"];
AppendTo[ProgressReport,"0␣/␣"<>ToString[NumToRun2]];
Export["ProgressReport.txt",ProgressReport];
SetSharedVariable[ctr2];
SetSharedVariable[ctrfloor2];
SetSharedVariable[\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[
   Script0]\[Script0]\[ScriptR]2];
TaskCheckWithCtr[branch_,checkvar_,checktask_,\[ScriptT]\[Script0]\[
```

```
   ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT]_:"default",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[
   ScriptE]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[
   ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{
   taskconfigsUnion},
taskconfigsUnion=TaskCheck[branch,checkvar,checktask,\[ScriptT]\[ScriptO
   ]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[
   ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[
   ScriptT],\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[
   ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]];
ctr2++;
ctrfloor2=Floor[ctr2,100];
If[ctrfloor2!=\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO
   ]\[ScriptO]\[ScriptR]2,ProgressReport[[-1]]=ToString[ctrfloor2]<>"␣/␣"
   <>ToString[NumToRun2];Export["ProgressReport.txt",ProgressReport]];
\[ScriptC]\[ScriptT]\[ScriptR]\[ScriptF]\[ScriptL]\[ScriptO]\[ScriptO]\[
   ScriptR]2=ctrfloor2;
taskconfigsUnion]
(*Print["counter ",Dynamic[ctr2],"/",NumToRun2]
ProgressIndicator[Dynamic[ctr2],{0,NumToRun2}]*)
checkvar={PP};
checktask={tasksubsP[[All,2]]};
checktol=.02;
analyses=ParallelMap[TaskCheckWithCtr[#,checkvar,checktask,{checktol},{
   checktol}]&,branches,{2}];
ProgressReport[[-1]]=ToString[ctr2]<>"␣/␣"<>ToString[NumToRun2];
AppendTo[ProgressReport,DateString[]];
AppendTo[ProgressReport,""];
Export["ProgressReport.txt",ProgressReport];
designs=MapThread[{#1,#2,#3}&,{Solns17,branches,analyses}];
(*designs indexed by
-dim
-branches
        -branch1
                -config1
                -config2
                -etc
        -branch2
        -etc
-analysis*)
designs>>"designs"
DeleteFile["branches"]
designsreduced=ReduceDesign[#,11]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length11=Length[designsreduced];
If[length11>0,designsreduced>>"designsreduced11"]
designsreduced=ReduceDesign[#,10]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length10=Length[designsreduced];
```

```
If[length10>0,designsreduced>>"designsreduced10"]
designsreduced=ReduceDesign[#,9]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length9=Length[designsreduced];
If[length9>0,designsreduced>>"designsreduced9"]
designsreduced=ReduceDesign[#,8]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length8=Length[designsreduced];
If[length8>0,designsreduced>>"designsreduced8"]
designsreduced=ReduceDesign[#,7]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length7=Length[designsreduced];
If[length7>0,designsreduced>>"designsreduced7"]
designsreduced=ReduceDesign[#,6]&/@designs;
designsreduced=DeleteCases[designsreduced,x_/;Length[x[[2]]]==0];
length6=Length[designsreduced];
If[length6>0,designsreduced>>"designsreduced6"]
resultstable={{"","No. of linkages"},{"11 position",length11},{"10 
   position",length10},{"9 position",length9},{"8 position",length8},{"7 
   position",length7},{"6 position",length6}};
AppendTo[ProgressReport,ToString[Grid[resultstable]]];
AppendTo[ProgressReport,""];
AppendTo[ProgressReport,"Analysis complete!  See designsreduced* files for
   the linkage solutions."];
AppendTo[ProgressReport,""];
end=DateString[];
AppendTo[ProgressReport,end];
AppendTo[ProgressReport,""];
computationtime=DateDifference[begin,end,"Hour"][[1]];
AppendTo[ProgressReport,"The analysis computation took "<>ToString[
   computationtime]<>" hours."];
Export["ProgressReport.txt",ProgressReport];
(*designsreduced=<<"designsreduced6";
Length[designsreduced]*)
(*plotrange={{-60,60},{-100,40}};
pivotsize=.08;*)
(*Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
   Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
   ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Row[{
Show[Draw[pathtype,designsreduced[[des,1]],designsreduced[[des,2,br,config
   ]]],DrawTrace[designsreduced[[des,2,br]]]],
Column[{"Quantity of branches"~~Length[designsreduced[[des,2]]],
CreateAnalysisTable[designsreduced[[des,3]]],"",
Column[Cases[designsreduced[[des,1]],x_/;MemberQ[{A,B,CC,DD,F,G,H,P0},x
   [[1]]]]]}]
},"   "],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
```

```
{config,1,Length[designsreduced[[des,2,br]]],1}]]*)
```

# J Mathematica code: Path Generation Viewer

```
(*Type and Task*)
SetDirectory[NotebookDirectory[]];
{pathtype,tasksubsP,Chain2Rsubs}=<<"TypeandTask";
(*Acceptable pathtype
"WIIFunc2WIPath"
"SIIIFunc2SIPath"
"SIIIFunc2SIIPath"
"SIIFunc2SIIPath"
"SIIFunc2SIIIPath"*)
taskpoints=Map[{Re[#[[2]]],Im[#[[2]]]}&,tasksubsP];
(*Generate a bunch of symbols*)
npos=11;
SymbolLists[symbols_,npos_]:=Module[{symsindexed},
symsindexed=Table[base<>ToString[j],{base,ToString/@symbols},{j,1,npos}];
Do[ToExpression[ToString[symbols[[i]]]<>"="<>ToString[symsindexed[[i]]]],{
    i,Length[symbols]}]]
Clear[Q,Qc,S,Sc]
SymbolLists[{Q,Qc,S,Sc},npos-1]
(*Params1=ReadList["final_parameters",Number];
Params2=Partition[Rest[Params1],2];
Params3=Apply[#1+#2*I&,Params2,1];
Params4=Thread[Join[Q,S]->Params3];
tasksubsQS=Params4~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.Params4);*)
params1=Import["final_parameters","Text"];
params2=StringSplit[params1];
params3=ToExpression[StringReplace[params2,"e"->"*10^"]];
params4=Partition[Rest[params3],2];
params5=Apply[#1+#2*I&,params4,1];
params6=Thread[Join[Q,S]->params5];
tasksubsQS=params6~Join~(Join[Thread[Qc->Q\[Conjugate]],Thread[Sc->S\[
    Conjugate]]]/.params6);
(*Path Generator Conversion Functions*)
SegmentStretchRotateTranslate[{A_,B_},{Ap_,Bp_},p_]:=(Bp-Ap)/(B-A)*(p-A)+
    Ap
(*(A,B) are endpoints of initial line segment
(Ap,Bp) are endpoints of final line segment
p is fixed frame coordinates on initial line segment
Output is fixed frame coordinates on final line segment*)
WIIFunc2WIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,pathgendim
    ,pathgendimc},
DIM=dim~Join~{CC->0,G->1,H->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
    chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
```

```
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
   chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][DD]/.DIM),
CC->(\[ScriptCapitalT][CC]/.DIM),
DD->(\[ScriptCapitalT][G]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][H]/.DIM),
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,DD->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
   chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][H]/.DIM),
DD->(\[ScriptCapitalT][CC]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][DD]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{K,L}/.
   chaindim,p];
pathgendim={A->K/.chaindim,
B->(\[ScriptCapitalT][CC]/.DIM),
CC->L/.chaindim,
DD->(\[ScriptCapitalT][G]/.DIM),
F->(\[ScriptCapitalT][H]/.DIM),
```

```
G->(\[ScriptCapitalT][DD]/.DIM),
H->(\[ScriptCapitalT][F]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
SIIFunc2SIIIPath[dim_,chaindim_]:=Module[{DIM,\[ScriptCapitalT],p,
   pathgendim,pathgendimc},
DIM=dim~Join~{A->0,B->1};
\[ScriptCapitalT][p_]:=SegmentStretchRotateTranslate[{A,B}/.DIM,{L,K}/.
   chaindim,p];
pathgendim={A->(\[ScriptCapitalT][F]/.DIM),
B->K/.chaindim,
CC->(\[ScriptCapitalT][DD]/.DIM),
DD->(\[ScriptCapitalT][H]/.DIM),
F->L/.chaindim,
G->(\[ScriptCapitalT][G]/.DIM),
H->(\[ScriptCapitalT][CC]/.DIM),
P0->(P0/.chaindim)};
pathgendimc={Ac->A\[Conjugate],Bc->B\[Conjugate],CCc->CC\[Conjugate],DDc->
   DD\[Conjugate],Fc->F\[Conjugate],Gc->G\[Conjugate],Hc->H\[Conjugate],
   Pc0->P0\[Conjugate]}/.pathgendim;
Join[pathgendim,pathgendimc]]
AddTracePoint[type_,dim_,branchesforsinglelinkage_]:=Module[{vars,
   Pexpression,brancheswithP},
If[MemberQ[{"WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
   SIIFunc2SIIPath","SIIFunc2SIIIPath"},type],,Print["Error"];Abort[]];
vars=branchesforsinglelinkage[[1,1,All,1]];
Goto[type];
(*****)Label["WIIFunc2WIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
If[Length[{QQ,SS,TT}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(F-A)+TT*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(P0-F)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
If[Length[{RR,SS,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=B+SS*(F-B)+UU*(H-F)+RR*(P0-H)/.dim;
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
If[Length[{QQ,TT,UU}\[Intersection]vars]==3,,Print["Error"];Abort[]];
Pexpression=A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)/.dim;
Goto["end"];
(*****)
```

```
Label["end"];
brancheswithP=Map[Append[#,PP->Pexpression/.#]&,branchesforsinglelinkage
    ,{2}]]
(*Forward Kinematics*)
FourbarFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(
    DDc-CCc)-SSc*(DDc-Bc),RR*RRc-1,SS*SSc-1};
FourbarFwdKin[dim_]:=Module[{vars,varsc,FKeqns,res,input,FKeqnsEval,J,
    FKsolns,branches1,branches2,branches3,branches4,branches6},
vars={RR,SS};
varsc={RRc,SSc};
FKeqns=FourbarFKeqns/.QQc->1/QQ;
res=360;
input=QQ->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars]&,input];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIiso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#]]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
    branches4=Delete[branches4,i]]
];
branches6=Map[{QQ->(QQ/.#),RR->(RR/.#),SS->(SS/.#),DetJ->(DetJ/.#)}&,(*
    branches5*)(*branches4*)branches3,{2}];
branches6]
(*Forward Kinematics Loop equations for each type of linkage*)
WIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(CC-A)+RR*(G-CC)+TT*(H-G)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc
    *(Gc-CCc)+TTc*(Hc-Gc)-SSc*(Fc-Bc)-UUc*(Hc-Fc)};
WIIFKeqns={A-CC+QQ*(DD-A)+TT*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+TTc*(Gc-
    DDc)-RRc*(Gc-CCc),
B-CC+SS*(F-B)+UU*(H-F)-RR*(H-CC),Bc-CCc+SSc*(Fc-Bc)+UUc*(Hc-Fc)-RRc*(Hc-
    CCc)};
SIFKeqns={A-B+QQ*(CC-A)+RR*(DD-CC)-SS*(DD-B),Ac-Bc+QQc*(CCc-Ac)+RRc*(DDc-
    CCc)-SSc*(DDc-Bc),
A-B+QQ*(F-A)+TT*(H-F)-SS*(G-B)-UU*(H-G),Ac-Bc+QQc*(Fc-Ac)+TTc*(Hc-Fc)-SSc
    *(Gc-Bc)-UUc*(Hc-Gc)};
SIIFKeqns={A-B+QQ*(CC-A)+RR*(G-CC)-SS*(DD-B)-TT*(G-DD),Ac-Bc+QQc*(CCc-Ac)+
    RRc*(Gc-CCc)-SSc*(DDc-Bc)-TTc*(Gc-DDc),
A-B+QQ*(CC-A)+RR*(H-CC)-SS*(F-B)-UU*(H-F),Ac-Bc+QQc*(CCc-Ac)+RRc*(Hc-CCc)-
    SSc*(Fc-Bc)-UUc*(Hc-Fc)};
SIIIFKeqns={A-CC+QQ*(DD-A)+UU*(G-DD)-RR*(G-CC),Ac-CCc+QQc*(DDc-Ac)+UUc*(Gc
    -DDc)-RRc*(Gc-CCc),
A-B+QQ*(DD-A)+UU*(H-DD)-SS*(F-B)-TT*(H-F),Ac-Bc+QQc*(DDc-Ac)+UUc*(Hc-DDc)-
    SSc*(Fc-Bc)-TTc*(Hc-Fc)};
FwdKin[type_,dim_,invar_,outvar_]:=Module[{vars,varsc,Neqns,inputSub,
```

```
   Loopeqns,FKeqns,res,input,FKeqnsEval,J,FKsolns,branches1,branches2,
   branches3,branches4,colsets,rowsets,MinorJMatrices,JMinors,branches5},
(*type -the type of linkage being analyzed should be "WI","WII","SI","SII
   ","SIII"*)
(*dim -the dimensions as lists of substitutions.  The dimensions are
   coordinates of each pivot in the first position in the global frame*)
(*invar -the input variable should be QQ, RR, SS, TT, UU according to the
   figures in my dissertation*)
(*outvar -this should be a list of variables to appear in the output
   configuration.  The list can only include QQ, RR, SS, TT, UU*)
If[MemberQ[{"WI","WII","SI","SII","SIII"},type],,Print["Error"];Abort[]];
   (*Make sure the linkage type is specified correctly*)
If[MemberQ[{QQ,RR,SS,TT,UU},invar],,Print["Error"];Abort[]];(*Make sure
   invar is a valid parameter*)
If[Depth[outvar]==2,,Print["Error"];Abort[]];(*Make sure outvar is a list
   *)
If[Length[{QQ,RR,SS,TT,UU}\[Union]outvar]==5,,Print["Error"];Abort[]];(*
   Make sure outvar contains valid parameters*)
vars=DeleteCases[{QQ,RR,SS,TT,UU},invar];
varsc=ToExpression[ToString[#]<>"c"]&/@vars;
Neqns=Thread[vars*varsc-1];
inputSub=ToExpression[ToString[invar]<>"c"]->1/invar;
Loopeqns=Which[type=="WI",WIFKeqns,type=="WII",WIIFKeqns,type=="SI",
   SIFKeqns,type=="SII",SIIFKeqns,type=="SIII",SIIIFKeqns];
FKeqns=Join[Loopeqns/.inputSub,Neqns];
res=360;
input=invar->#&/@Exp[I*Range[0,2*\[Pi],2*\[Pi]/(res-1)]];
vars=Riffle[vars,varsc];
FKeqnsEval=FKeqns/.dim;
J=D[FKeqnsEval,{vars}];
FKsolns=Map[NSolve[FKeqnsEval/.#,vars,Method->"EndomorphicMatrix"]&,input
   ];
branches1=Sorting[dim,FKeqns,input,FKsolns];
branches2=Flatten[remIso/@branches1,1];
branches3=Map[Append[#,DetJ->Chop[Det[J/.#],10^-3]]&,branches2,{2}];
branches4=Map[Split[#,Sign[DetJ/.#1]==Sign[DetJ/.#2]&]&,branches3];
branches4=Flatten[branches4,1];
For[i=Length[branches4],i>=1,i=i-1,
If[Total[Count[{#},Join[{___},branches4[[i]],{___}]]&/@branches4]>1,
   branches4=Delete[branches4,i]]
];
branches5=Map[Cases[#,x_/;MemberQ[outvar,x[[1]]]||x[[1]]==DetJ]&,branches4
   ,{2}];
branches5]
(*A general module for sorting the solutions to the forward kinematics
   into trajectories*)
Sorting[dim_,FKeqns_,input_,FKsolns_]:=Module[{x,y,yV,F,J,br,bran,Its,tol,
   log,i,j,k,ycur,ToBeAdded,Added,nmatch,NotAdded,ToBeJoined,pairs,
   NotToBeJoined,Pre,Post,NewSequences,BranchSequences,branches},
(*dim- includes linkage's dimensions as substitutions*)
(*FKeqns- the fwd kin eqns written symbolically*)
```

```
(*input- the input variable written as substitutions*)
(*FKsolns- fwd kin solutions indexed by position, solution*)
x=input;(*input values indexed by position*)
y=FKsolns;(*output values indexed by position*)
yV=FKsolns[[1,1,All,1]];(*a vector of the output symbols*)
F=FKeqns/.dim;(*the fwd kin eqns with the input and outputs left symbolic
    *)
J=D[F,{yV}];(*Jacobian of F with the input and outputs left symbolic*)
br={{x[[1]],#}}&/@y[[1]];(*the active branches: indices are branch,
    position, (1-input,2-output) *)
bran={};(*the completed branches: same indices*)
Its=5;(*Newton iterations*)
tol=10^-3;(*tolerance when comparing numbers*)
log={};
For[i=2,i<=Length[x],i++,(*i is the current position*)
ycur=br[[All,-1,2]];
Do[ycur=(yV/.x[[i]]/.#)-Inverse[J/.x[[i]]/.#].(F/.x[[i]]/.#)&/@ycur;
ycur=Thread[yV->#]&/@ycur,{Its}];
ToBeAdded=Position[Round[y[[i,All,All,2]],tol],Round[#,tol]]&/@ycur[[All,
    All,2]];
ToBeAdded=ToBeAdded[[All,All,1]];
Added={};
For[j=Length[br],j>=1,j=j-1,(*j is the current branch*)
nmatch=Length[ToBeAdded[[j]]];
Which[nmatch==1,
AppendTo[br[[j]],{x[[i]],y[[i,ToBeAdded[[j,1]]]]}];
AppendTo[Added,ToBeAdded[[j,1]]],
nmatch==0,
AppendTo[bran,br[[j]]];
br=Delete[br,j];
AppendTo[log,"Path␣ended␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~ToString[Det[J
    /.x[[i]]/.ycur[[j]]]]~~"."],
nmatch>1,
Do[br=Insert[br,br[[j]],j],{nmatch-1}];
Do[AppendTo[br[[j-1+k]],{x[[i]],y[[i,ToBeAdded[[j,k]]]]}],{k,nmatch}];
Added=Join[Added,ToBeAdded[[j]]];
AppendTo[log,"Paths␣may␣be␣splitting␣at␣"~~x[[i]]~~"␣where␣Det[J]␣=␣"~~
    ToString[Det[J/.x[[i]]/.ycur[[j]]]]
]];
NotAdded=Complement[Range[Length[y[[i]]]],Added];
If[Length[NotAdded]>0,AppendTo[log,"At␣"~~x[[i]]~~",␣there␣was␣"~~ToString
    [Length[NotAdded]]~~"␣new␣branch(es)␣added."]];
Do[AppendTo[br,{{x[[i]],y[[i,k]]}}],{k,NotAdded}];
];
bran=Join[bran,br];
bran=Map[Flatten,bran,{2}];
(*End of main sorting*)
ToBeJoined={};(*To contain pairs of bran indices that have matching first
    and last configurations*)
pairs=Subsets[Range[Length[bran]],{2}];(*All combinations of 2 branch
    indices*)
```

```
Apply[If[bran[[#1,-1]]==bran[[#2,1]],AppendTo[ToBeJoined,{#1,#2}]]&,pairs
    ,1];
Apply[If[bran[[#2,-1]]==bran[[#1,1]],AppendTo[ToBeJoined,{#2,#1}]]&,pairs
    ,1];
NotToBeJoined=Complement[Range[Length[bran]],Flatten[ToBeJoined]];(*bran
    indices that do not have a matching first or last configuration*)
(*The objective of the following is to extend index pairs into long index
    chains*)
For[j=1,j<=Length[bran],j++,
Pre=Position[ToBeJoined,{___,j}];(*Position of index chains ending with j
    *)
Post=Position[ToBeJoined,{j,___}];(*Position of index chains beginning
    with j*)
NewSequences=Flatten[Table[
If[ii!=jj,ToBeJoined[[First[ii]]]~Join~Rest[ToBeJoined[[First[jj]]]],Null]
,{ii,Pre},{jj,Post}],1];(*The new index chains to be created*)
NewSequences=DeleteCases[NewSequences,Null];
If[Length[NewSequences]>0,ToBeJoined=Delete[ToBeJoined,Pre~Join~Post]];(*
    Remove seperate unjoined chains*)
ToBeJoined=ToBeJoined~Join~NewSequences(*Add new joined chains*)];
(*Cyclic chains will have the same first & last element which the last is
    removed here*)
ToBeJoined=Map[If[Last[#]==First[#],Most[#],#]&,ToBeJoined];
(*Join the index chains with singletons*)
BranchSequences=Join[ToBeJoined,{#}&/@NotToBeJoined];
(*Use index sequences to piece together new branches*)
(*branches=Map[Flatten[bran[[#]],1]&,BranchSequences];*)
(*Prime the branches list by associating the first index of each branch
    sequence into the list*)
branches=bran[[First/@BranchSequences]];
(*Complete each branch sequence with the Rest command*)
Do[branches[[j]]=Join[branches[[j]],Rest[bran[[BranchSequences[[j,k
    ]]]]]],{j,Length[BranchSequences]},{k,2,Length[BranchSequences[[j]]]}];
branches]
(*Remove "imaginary" positions from a branch.  Output gives contiguous
    chains of real positions.*)
remIiso[br_]:=Module[{bran,branch,branches},
bran=Map[If[Round[{Norm[QQ],Norm[RR],Norm[SS],Norm[TT],Norm[UU
    ]}/.#,10^-5]=={1,1,1,1,1},#,{}]&,br];
branch=Split[bran,#1!={}&&#2!={}&];
branches=DeleteCases[branch,{{}}]]
(*Analysis Functions*)
AnalysisTool[branches_]:=Module[{branchgraphpoints,colors},
branchgraphpoints=Map[{Cto\[Theta][QQ],Re[SS],Im[SS]}/.#&,branches,{2}];
colors={Red,Green,Blue,Cyan,Magenta,Brown,Orange,Pink,Purple};
Graphics3D[{
Opacity[.1],Cylinder[{{-\[Pi],0,0},{\[Pi],0,0}},1],Opacity[1],
MapIndexed[{colors[[Mod[First[#2],Length[colors],1]]],Point[#1]}&,
    branchgraphpoints],
PointSize[Large],Point[MapThread[{Cto\[Theta][#1],Re[#2],Im[#2]}&,{Q,S}]/.
    tasksubsQS]
```

```
},Axes->True,PlotRange->{{-\[Pi],\[Pi]},{-2,2},{-2,2}},RotationAction->"
    Clip",ImageSize->750]]
RectangleTest[a_,b_,w_,p_]:=Module[{Conditions},
(*all coordinates written as complex numbers*)
(*a,b -endpoints of a line segment along the length of the rectangle*)
(*w -width of the rectangle*)
(*p -point to test*)
(*If[b-a==0,Conditions={False,False};Goto["end"]];*)
Conditions={0<Chop[N[(p-a)*(b\[Conjugate]-a\[Conjugate])+(p\[Conjugate]-a
    \[Conjugate])*(b-a)]]<Chop[N[2*(b-a)*(b\[Conjugate]-a\[Conjugate])]],
Chop[N[-Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]<Chop[N[I*((b-a)*(p\[
    Conjugate]-a\[Conjugate])-(b\[Conjugate]-a\[Conjugate])*(p-a))]]<Chop[N
    [Sqrt[(b-a)*(b\[Conjugate]-a\[Conjugate])]*w]]};
(*Label["end"];*)
Conditions=={True,True}]
TaskCheck[branch_,checkvar_,checktask_,\[ScriptT]\[ScriptO]\[ScriptL]\[
    ScriptCapitalC]\[ScriptI]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP
    ]\[ScriptE]\[ScriptR]\[ScriptC]\[ScriptE]\[ScriptN]\[ScriptT]_:"default
    ",\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT]_:"default"]:=Module[{nvar,ntask,nbranch,
    tolBase,tolCircsPercent,tolRectsPercent,tolCircs,tolRects,configs,
    checkCircs,checkRects,taskconfigsCircs,taskconfigsRects,
    taskconfigsUnion},
(*branch -a single branch from a single mechanism (a list of
    configurations)*)
(*checkvar -a list of variables of the branch to be checked*)
(*checktask -lists of task values that corresponds to the checkvar list*)
(*The next two optional arguments define shapes (circles & rectangles) in
    the complex planes of individual variables for determining whether or
    not those variables' trajectories pass through task position values*)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI]\[ScriptR]\[
    ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the diameter of circles around
    each checkvar in each configuration as a percent of that variables
    range.  It is indexed by checkvar. *)
(*\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE]\[ScriptC]\[
    ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[ScriptC]\[
    ScriptE]\[ScriptN]\[ScriptT] -defines the width of rectangles formed
    between two configurations for each checkvar as a percent of that
    variables range.  It is indexed by checkvar. *)
nvar=Length[checkvar];
ntask=Length[First[checktask]];
nbranch=Length[branch];
(*Indexed by variables to check*)
tolBase=Table[Max[Apply[Norm[#2-#1]&,Subsets[checktask[[m]],{2}],1]],{m,
    nvar}];
(*Percentage that defines diameter of circles at each point, indexed by
    variables to check*)
tolCircsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalC]\[ScriptI
    ]\[ScriptR]\[ScriptC]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
```

```
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolCircsPercent=="default",tolCircsPercent=Table[.02,{nvar}]];
(*Percentage that defines width of rectangles, indexed by variables to
   check*)
tolRectsPercent=\[ScriptT]\[ScriptO]\[ScriptL]\[ScriptCapitalR]\[ScriptE
   ]\[ScriptC]\[ScriptT]\[ScriptS]\[ScriptCapitalP]\[ScriptE]\[ScriptR]\[
   ScriptC]\[ScriptE]\[ScriptN]\[ScriptT];
If[tolRectsPercent=="default",tolRectsPercent=Table[.02,{nvar}]];
(*Radii of circles at each point, indexed by variables to check*)
tolCircs=tolCircsPercent/2*tolBase;
(*Width of rectangles, indexed by variables to check*)
tolRects=tolRectsPercent*tolBase;
If[nvar==Length[checktask]==Length[tolCircsPercent]==Length[
   tolRectsPercent],,Print["Error"];Abort[]];
If[Length/@checktask==Table[ntask,{nvar}],,Print["checktask␣members␣do␣not
   ␣all␣have␣the␣same␣length"];Abort[]];
(*configs lists values of each checked variable not as substitutions,
   indexed as (var,config)*)
configs=Table[var/.branch[[n]],{var,checkvar},{n,nbranch}];
(*checkCircs is a table of True/False indexed by (var,task position,config
   )*)
checkCircs=Table[Norm[tp-configs[[m,n]]]<=tolCircs[[m]]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch}];
(*checkRects is a table of True/False indexed by (var,task position,config
   )*)
checkRects=Table[RectangleTest[configs[[m,n]],configs[[m,n+1]],tolRects[[m
   ]],tp]
,{m,nvar},{tp,checktask[[m]]},{n,nbranch-1}];
(*taskconfigs is indexed by task position*)
taskconfigsCircs=Table[Position[Transpose[checkCircs[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsRects=Table[Position[Transpose[checkRects[[All,j]]],Table[True
   ,{nvar}]][[All,1]]
,{j,ntask}];
taskconfigsUnion=MapThread[Union[#1,#2]&,{taskconfigsCircs,
   taskconfigsRects}];
taskconfigsUnion(*A list indexed by task position that shows at which
   point in the branch that task position appears*)]
ReduceDesign[design_,nmatch_]:=Module[{dim,branches,analysis,
   matchpositions},
dim=design[[1]];
branches=design[[2]];
analysis=design[[3]];
matchpositions=Position[analysis,x_/;Count[x,Except[{}]]==nmatch,1][[All
   ,1]];
{dim,branches[[matchpositions]],analysis}]
(*Animation Functions*)
(*Map from a complex number to an angle*)
Cto\[Theta][Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,ArcTan[Re[Cnum],Im[Cnum
   ]],"error"]
```

```
(*Another map from a complex number to an angle*)
Cto\[Theta]2[Cnum_]:=If[Round[Norm[Cnum],10^-9]==1,Chop[-I*Log[Cnum]],"
    error"]
(*Map from a complex number to a 2D vector*)
CtoXY[Cnum_]:={Re[Cnum],Im[Cnum]}
plotrange={{-25,25},{-8,40}};
funcplotrange={{-1,2*\[Pi]},{-3,3}};
pivotsize=.1;
Draw[type_,dim_,config_]:=Module[{AcceptableTypes,\[ScriptCapitalA],\[
    ScriptCapitalB],\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF
    ],\[ScriptCapitalG],\[ScriptCapitalH],\[ScriptCapitalP],primitives},
AcceptableTypes={"WIIFunc","SIIIaFunc","SIIIbFunc","SIIaFunc","SIIbFunc","
    WIMot","WIIFunc2WIPath","SIIIFunc2SIPath","SIIIFunc2SIIPath","
    SIIFunc2SIIPath","SIIFunc2SIIIPath"};
If[MemberQ[AcceptableTypes,type],,Print["Error"];Abort[]];
Goto[type];
(*****)Label["WIIFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.{CC->0,G->1,H->1};
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[CC+RR*(G-CC)]/.{CC->0,G->1,H->1}/.config;
\[ScriptCapitalH]=CtoXY[CC+RR*(H-CC)]/.{CC->0,G->1,H->1}/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{\[ScriptCapitalC],\[
    ScriptCapitalG],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalB],\[ScriptCapitalF]},{\[ScriptCapitalD],\[ScriptCapitalG
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
    ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIIaFunc"];
(*****)Label["SIIIbFunc"];
(*Output needs to be QQ, SS, UU*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,DD->1};
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.{A->0,DD->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.{A->0,DD->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.{A->0,DD->1}/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{\[ScriptCapitalD],\[
    ScriptCapitalG],\[ScriptCapitalH]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
    ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
    ]},{\[ScriptCapitalF],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
    ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
```

```
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["SIIaFunc"];
(*****)Label["SIIbFunc"];
(*Output needs to be QQ, RR, SS*)
\[ScriptCapitalA]=CtoXY[A]/.{A->0,B->1};
\[ScriptCapitalB]=CtoXY[B]/.{A->0,B->1};
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(CC-A)+RR*(G-CC)]/.{A->0,B->1}/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(CC-A)+RR*(H-CC)]/.{A->0,B->1}/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalC],\[
   ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalB],\[ScriptCapitalD
   ],\[ScriptCapitalF]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]}};
Goto["end"];
(*****)Label["WIMot"];
Goto["end"];
(*****)Label["WIIFunc2WIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(DD-B)+RR*(CC-DD)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(DD-B)+RR*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalB],\[
   ScriptCapitalD],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalD
   ],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
   ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
   ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[A+QQ*(CC-A)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(F-A)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(G-B)]/.dim/.config;
```

```
\[ScriptCapitalH]=CtoXY[A+QQ*(F-A)+TT*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(F-A)+TT*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalA],\[
    ScriptCapitalC],\[ScriptCapitalF]},{\[ScriptCapitalB],\[ScriptCapitalD
    ],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalC],\[ScriptCapitalD]},{\[
    ScriptCapitalG],\[ScriptCapitalH]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(P0-F)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalB],\[
    ScriptCapitalD],\[ScriptCapitalF]},{\[ScriptCapitalC],\[ScriptCapitalG
    ],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH],\[
    ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
    pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
    ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(CC-H)]/.dim/.config;
\[ScriptCapitalD]=CtoXY[B+SS*(DD-B)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[B+SS*(F-B)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(G-H)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[B+SS*(F-B)+UU*(H-F)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[B+SS*(F-B)+UU*(H-F)+RR*(P0-H)]/.dim/.config;
primitives={GrayLevel[.5,1],EdgeForm[Black],Polygon[{{\[ScriptCapitalC],\[
    ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalG],\[ScriptCapitalH
    ],\[ScriptCapitalP]},{\[ScriptCapitalC],\[ScriptCapitalG],\[
    ScriptCapitalP]}}],GrayLevel[0,.5],Polygon[{\[ScriptCapitalB],\[
    ScriptCapitalD],\[ScriptCapitalF]}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalC]},{\[
    ScriptCapitalD],\[ScriptCapitalG]},{\[ScriptCapitalF],\[ScriptCapitalH
    ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB]},MPivot[#,
```

```
   pivotsize]&/@{\[ScriptCapitalC],\[ScriptCapitalD],\[ScriptCapitalF],\[
   ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)Label["SIIFunc2SIIIPath"];
\[ScriptCapitalA]=CtoXY[A]/.dim;
\[ScriptCapitalB]=CtoXY[B]/.dim;
\[ScriptCapitalC]=CtoXY[CC]/.dim;
\[ScriptCapitalD]=CtoXY[A+QQ*(DD-A)]/.dim/.config;
\[ScriptCapitalF]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(F-H)]/.dim/.config;
\[ScriptCapitalG]=CtoXY[A+QQ*(DD-A)+UU*(G-DD)]/.dim/.config;
\[ScriptCapitalH]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)]/.dim/.config;
\[ScriptCapitalP]=CtoXY[A+QQ*(DD-A)+UU*(H-DD)+TT*(P0-H)]/.dim/.config;
primitives={GrayLevel[0,.5],EdgeForm[Black],Polygon[{{\[ScriptCapitalD],\[
   ScriptCapitalG],\[ScriptCapitalH]},{\[ScriptCapitalF],\[ScriptCapitalH
   ],\[ScriptCapitalP]}}],
GrayLevel[0,1],Thick,Line[{{\[ScriptCapitalA],\[ScriptCapitalD]},{\[
   ScriptCapitalC],\[ScriptCapitalG]},{\[ScriptCapitalB],\[ScriptCapitalF
   ]}}],
GPivot[#,pivotsize]&/@{\[ScriptCapitalA],\[ScriptCapitalB],\[
   ScriptCapitalC]},MPivot[#,pivotsize]&/@{\[ScriptCapitalD],\[
   ScriptCapitalF],\[ScriptCapitalG],\[ScriptCapitalH]},
PointSize[Large],Point[\[ScriptCapitalP]],Point[taskpoints]};
Goto["end"];
(*****)
Label["end"];
Graphics[primitives,Axes->True,PlotRange->Dynamic[plotrange],ImageSize
   ->500]]
DrawTrace[branch_]:=Graphics[{Point[Map[CtoXY[PP]/.#&,branch]]}]
CreateAnalysisTable[analysis_]:=Module[{analysis2,analysis3},
analysis2=MapIndexed[Prepend[#1,"br␣"<>ToString[First[#2]]]&,analysis];
analysis3=Prepend[analysis2,Join[{"task␣no."},Range[0,npos-1]]];
MatrixForm[analysis3]]
CreateFunctionPlot[branch_,configindex_,functype_]:=Module[{inoutFunc,
   inoutTask,pointsFunc,pointsTask,pointsFuncMod,pointsTaskMod,function,
   line},
Which[
functype=="WIIFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
functype=="SIIaFunc",inoutFunc={QQ,SS};inoutTask={Q,S},
functype=="SIIbFunc",inoutFunc={SS,QQ};inoutTask={S,Q},
True,Print["Error"];Abort[]];
pointsFunc=Map[Arg/@inoutFunc/.#&,branch];
pointsTask=Arg/@Thread[inoutTask]/.tasksubsQS;
PrependTo[pointsTask,{0,0}];
{pointsFuncMod,pointsTaskMod}=Apply[{Mod[#1,2*\[Pi]],Mod[#2,2*\[Pi],-\[Pi
   ]]}&,{pointsFunc,pointsTask},{2}];
function=ListPlot[{pointsFuncMod,pointsTaskMod},PlotStyle->{Automatic,
   Directive[Red,PointSize[Large]]},PlotRange->funcplotrange,ImageSize
   ->500];
```

```
line=Graphics[{Line[Map[{pointsFuncMod[[configindex,1]],#}&,{-7,7}]]}];
Show[function,line]]
GPivot[pt_,sc_:1,rot_:0]:=Module[{r,l,h,t,g},
(*x,y -location of pivot*)
(*sc -optional scale factor*)
(*rot -optional rotation*)
r=1.5/2;(*radius of pivot*)
l=3.4;(*length of base*)
h=.3;(*height of base*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={t,Circle[pt+{-2*r,0},r,{0,-Pi/2}],Circle[pt+{2*r,0},r,{-Pi/2,-Pi}],
EdgeForm[t],White,Disk[pt,#]&/@{r,r*2/3},
Rectangle[pt+{-l/2,-r-h},pt+{l/2,-r},RoundingRadius->.005]};
Rotate[Scale[g,sc,pt],rot,pt]]
MPivot[pt_,sc_:1]:=Module[{r,t,g},
r=1.2/2;(*radius of pivot*)
t=AbsoluteThickness[1.5];(*line thickness*)
g={EdgeForm[t],White,Disk[pt,#]&/@{r,r*.625}};
Scale[g,sc,pt]]
(*Solution Viewer*)
designsreduced=<<"designsreduced6";
Length[designsreduced]
plotrange={{-2,2},{-1,1.5}};
pivotsize=.08;
Module[{\[ScriptD]\[ScriptE]\[ScriptS]=1,\[ScriptB]\[ScriptR]=1},
   Manipulate[
If[\[ScriptD]\[ScriptE]\[ScriptS]!=des,br=1;config=1];\[ScriptD]\[ScriptE
   ]\[ScriptS]=des;
If[\[ScriptB]\[ScriptR]!=br,config=1];\[ScriptB]\[ScriptR]=br;
Row[{
Show[Draw[pathtype,designsreduced[[des,1]],designsreduced[[des,2,br,config
   ]]],DrawTrace[designsreduced[[des,2,br]]]],
Column[{"Quantity␣of␣branches"~~Length[designsreduced[[des,2]]],
CreateAnalysisTable[designsreduced[[des,3]]],"",
Column[Cases[designsreduced[[des,1]],x_/;MemberQ[{A,B,CC,DD,F,G,H,P0},x
   [[1]]]]]}]
},"␣␣␣"],
{des,1,Length[designsreduced],1},
{br,1,Length[designsreduced[[des,2]]],1},
{config,1,Length[designsreduced[[des,2,br]]],1}]]
```