A Unified Framework for Motion and Path Synthesis of Planar and Spherical mechanisms

A Dissertation Proposal presented

by

Shashank Sharma

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

Stony Brook University

April 2019

Stony Brook University
The Graduate School

Shashank Sharma

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby approve
this dissertation proposal.

Professor Anurag Purwar, Dissertation Advisor
Department of Mechanical Engineering

Professor Qiaode Jeffrey Ge, Chairperson of Dissertation Committee
Department of Mechanical Engineering

Professor Nilanjan Chakraborty, Committee Member
Department of Mechanical Engineering

Professor Jahangir Rastegar, Committee Member
Department of Mechanical Engineering

Professor Xianfeng David Gu, Committee Member (Outside)
Department of Computer Science

This dissertation proposal is accepted by the Graduate School

Richard Gerrig
Dean of the Graduate School

Abstract of the Dissertation Proposal

A Unified Framework for Motion and Path Synthesis of Planar and Spherical mechanisms

by

Shashank Sharma

Doctor of Philosophy

in

Mechanical Engineering

Stony Brook University

2019

This proposal presents work towards a Mechanism design framework which unifies the process on two fronts. First, on the problem specification front, Path and Motion synthesis constraints are handled collectively as a generalized Mixed synthesis problem. Second, on the solution side, simultaneous type and dimensional synthesis is carried out to give a designer increased variety of possible solutions. A Web-based application called MotionGen has also been made to carry out Path, Motion and Mixed synthesis for planar four-bar mechanisms.

Extension of proposed work to synthesis of more complex Spatial mech-

anisms will be investigated in future. Also, synthesis of planar and spherical mechanisms with additional links will be explored. A unified problem specification which includes path, motion and function constraints would also be looked into.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Firstly I would like to thank my parents, Mr. Shashi Kant and Mrs. Pammi Dixit for always being there for me through thick and thin. I am extremely grateful for the ethics and values instilled in me by them. Their unwavering trust in my abilities has motivated me to push my limits.

I am grateful to Professor Anurag Purwar, my advisor, for giving me this immense opportunity to work with him. He has always given me the freedom to shape my path and encouraged me to focus on research. His support and guidance have made this work possible.

I would like to express my thanks to my committee members, Professor Jeffrey Ge, Professor Nilanjan Chakraborty, Professor Jahangir Rastegar, and Professor David Gu, for taking their precious time in attending my presentation.

Chapter 1

Introduction

Maker culture is a new trend built upon DIY and hacker movement which encourages the creation of new devices and tinkering with the old. It spans electronics, robotics, 3-D printing, metalworking, and woodworking. The rise of the maker culture is closely associated with the rise of hackerspaces, Fab Labs and other "makerspaces" where like-minded individuals share ideas, skills, and tools. University campuses have especially been a hotbed for makerspaces. With the rise of cheap 3-D printing technologies and low-cost sensors, actuators, and micro-controllers, manufacturing tools have become accessible to many.

Thus, with the recent maker-movement and democratization of manufacturing capability, the creation of new devices is within everyone's grasp. However, there is still one missing piece. There is a lack of machine design tools for mechanism synthesis which can be used by creators. Unfortunately, design theory for even the simplest of mechanism i.e. a four bar is too complex for the uninitiated. This chasm needs to be filled for real innovation to thrive.

During concept generation, a designer breaks down a task into multiple constraints and criteria. Mechanisms satisfying the constraints are deemed feasible and the best mechanism is chosen based on the specified criteria. The type of constraints a designer can impose for a mechanism design problem are extremely diverse. The constrains could include the motion of a moving link, path traced by a point on a moving link, relationship between the input and output links, position of grounded joints, lengths of moving links, etc.

Mechanism synthesis problems have been conventionally posed as motion, path or function synthesis. Each of these formulations enable a designer to

impose only a single kind of constraint on the solution mechanism. However, complex practical situations require multiple types of constraints to fully define a design problem. One of the aims of this dissertation is to unify the three conventional synthesis approaches into a single framework. This would enable designers to solve a more general class of problems and enhance the capabilities of existing methodologies.

Another aspect of the mechanism design process is its type and dimensional synthesis. Type synthesis is selection of joint types (revolute, prismatic, etc.) for a concept mechanism while dimensional synthesis is calculating its link dimensions. Most of the existing methodologies carry out type and dimensional synthesis separately due to which the concept generation process depend considerably on the designer's experience. Methodologies which simultaneously carry out type and dimensional synthesis are capable of generating more diverse concept mechanisms. Subsequently, the focus of this dissertation is on such a framework which can offer enhanced choice and flexibility to designers.

Finally, an actual web-based application called MotionGen2.0 is created to implement the proposed algorithm which can be used by creators. User interaction and productivity have been kept as the centerpiece. The application is scalable by design and accessible to everyone through the web and mobile app. With the availability of MotionGen2.0, professionals and students will now be able to unleash their imagination. Empowering the right people using the right tools is the motivation for this work.

Thus, this dissertation works towards achieving a unified design process as shown in Fig. 1.1. This framework unifies problem specification with the ability to handle a diverse variety of design constraints. Also, problem solution is unified by cassying out simultaneous type and dimensional synthesis for planar, spherical and spatial mechanisms.

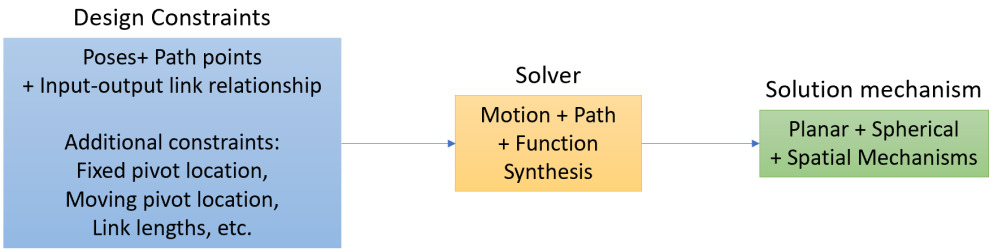


Figure 1.1: A unified mechanism synthesis methodology

Chapter 2

Mixed Synthesis

2.1 Introduction

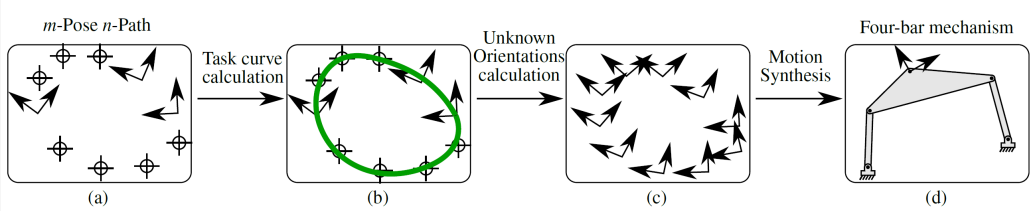


Figure 2.1: An Overview of our Approach to the Alt-Burmester Problems: (a) specify m-pose, n-path points; (b) a task curve is fit through the $m+n$ path points using Fourier series; (c) use the harmonic content of the path data to find the *missing* orientations at the n-path points; and (d) finally, compute both type and dimensions of planar four-bar linkages.

Conventionally, mechanism synthesis problems have been categorized and studied independently as path, motion, and function synthesis problems [1]. Path synthesis problems specify only path-point coordinates (x_i, y_i), while motion synthesis problems specify pose constraints (x_i, y_i, ζ_i), where (x_i, y_i) are the coordinates of the path-points or the origin of a moving frame attached to a given pose, while ζ_i is the orientation of the moving frame. In function synthesis, only input-output angle pairs (θ_i, ψ_i) are specified. Unfortunately, most of the real world problems do not conform to such a rigid categorization – many practical problems provide a mixture of path, motion and

function synthesis requirements. However, a synthesis approach which seamlessly incorporates all the three conventional synthesis problems has been elusive. As a result, machine designers often have to compromise on design specifications. In this chapter, the focus is on the synthesis of planar four-bar mechanisms for a hybrid of path and motion synthesis problems. This problem formulation, which consolidates both path and orientation data, has been termed as mixed synthesis in this chapter.

Murray's group termed the combined path and motion problems as the Alt-Burmester problems [2] named after Alt's [3] and Burmester's [4] work on path and motion generation, respectively. Brake et al. [5] discuss the dimensionality of solution sets for a variety of path-point and pose combinations. However, a finite number of solutions exist only for a subset of possible m-pose, n-path point synthesis problem. For example, there exist finite solutions for nine path-points and for five poses independently. We define such problems to be fully constrained problems. For a lesser number of path-points or poses, usually an infinite number of solutions are obtained. Subsequently, the authors explore only fully-constrained or under-constrained problem sets where up to nine constraints can be used to find four-bar mechanism parameters. This ignores the vast majority of over-constrained problems in m-pose, n-path point mixed synthesis family of problems, where exact solutions are not possible and only approximate, albeit useful solutions, can still be obtained. This is reflective of real-world design problems, which usually impose a large number of often challenging constraints.

A graphical approach has been presented by Zimmerman [6] to solve the mixed path, motion and function problem using sketching tools built in modern Computer-Aided Design (CAD) softwares. The proposed methodology can conveniently solve under-constrained and fully-constrained mixed synthesis problems and generate four-bar mechanisms. However, this methodology is unable to solve generalized m pose, n path-point synthesis problems, which may be over-constrained. This study does state the possibility of including prismatic joints in the synthesized mechanism. However, it is not the focus of study and details on synthesizing mechanical dyads with at least one prismatic joint are not included.

Motion synthesis turns out to be a mathematically less complex problem than path synthesis as each dyad can be generated independently effectively halving the number of unknowns. Typically, path synthesis problems involve solving a nonlinear system of equations. We have recently presented a generalized framework for solving motion synthesis problems using an efficient

algorithm that involves solving a linear system of equations using singular value decomposition [7–11]. The algorithm produces multiple solutions and can compute both the type and dimensions of the four-bar mechanisms. The algorithm produces results in real-time and is thus amenable to its implementation in interactive computational design tools [7].

In this chapter, we are presenting an approach to solve the Alt-Burmester problem by reducing it to a pure motion synthesis problem so that the aforementioned algorithm can be leveraged. In a planar four-bar linkage, the path of a coupler point is inextricably tied to the orientation of the coupler. This coupling can be revealed by analyzing and relating the harmonic content of the path and orientation data. First, an analytical relationship between the orientation- and path-data is obtained using the harmonic breakdown of the loop closure equation. Then, this relation is used to reformulate the mixed synthesis problem into a motion synthesis problem by attaching compatible orientations to input path points and consequently turning them into poses. The Fourier approximation based analytical approach proposed in this chapter can handle almost all possible variations of path points or poses. Once, the problem has been converted into a pure motion synthesis problem, we re-purpose our algebraic fitting approach in [10, 11] to solve for four-bar linkages. Figure 5.1 provides an overview of this approach.

We note that here mixed synthesis does *not* refer to the mixed exact-approximate path or motion synthesis, wherein we have a set of precision and approximate constraints. Our definition of *mixed* refers to a mixture of path-point and pose constraints.

This chapter's original contributions are in 1) the formulation of a Fourier descriptor based closed-form relationship between coupler orientations and path, 2) the novel use of this relationship to solve the generalized m-pose, n-path mixed synthesis problem, and 3) the incorporation of task-driven algebraic fitting based motion synthesis within the mixed synthesis algorithm for synthesis.

Rest of the chapter is organized as follows. Section 2.2 calculates a new path-orientation formulation from existing four-bar loop closure Fourier decomposition. Section 2.3 discusses the use of path-orientation relationship to reformulate mixed synthesis into motion synthesis problem. Section 2.4 reviews algebraic fitting based motion synthesis algorithm. Section 2.5 proposes a new algorithm to solve mixed synthesis problem and finally in section 5.4, we present a few examples to demonstrate the efficacy of the proposed approach.

2.2 Fourier Descriptors based relations

Use of Fourier descriptors is abundant in the domain of mechanism synthesis. It has been used for planar four-bar mechanism synthesis using optimization routines [12–16], atlas-based search algorithms [17,18], and machine learning approach [19]. Fourier descriptors have also been used to synthesize spherical [20] and spatial mechanism [21]. A class of single degree of freedom open-loop mechanisms termed as planar coupled serial chain mechanisms [22, 23] have also been generated with the help of Fourier descriptors.

In this section, we are interested in exploring the relationship between the coupler path and coupler orientation to establish a closed-form relationship between them. This would give us a framework for dealing with both pose and path constraints simultaneously. Path and motion synthesis formulations, which use Fourier decomposition of four-bar closure equation [14–16] are used as a starting point here. In [16], Li et al. presented a decomposition of the design space of four-bar mechanisms by using Fourier descriptors in the context of planar motion approximation. Harmonic decomposition of four-bar loop closure equation has been analyzed to independently fit rotational and translational Fourier descriptors and synthesize motion.

A four-bar mechanism is represented by its design parameters x_0, y_0, l_1, l_2, l_3, l_4, r, θ_1, and α as displayed in Fig. 2.2. These parameters are constant for a given four-bar mechanism. Coupler angle λ represents the varying orientation of coupler link with respect to fixed link at any given instant. Point P is the location of the coupler point in the global frame. which is also a variable. Coupler orientation ζ refers to the orientation of a moving frame attached to the coupler point, while δ is the constant angle at which moving frame is attached to coupler with respect to the coupler link line AB. All of these design parameters are unknown before a mechanism has been synthesized. Our goal is to find an explicit closed-form relationship between coupler path and orientation which forms the heart of our mixed synthesis algorithm.

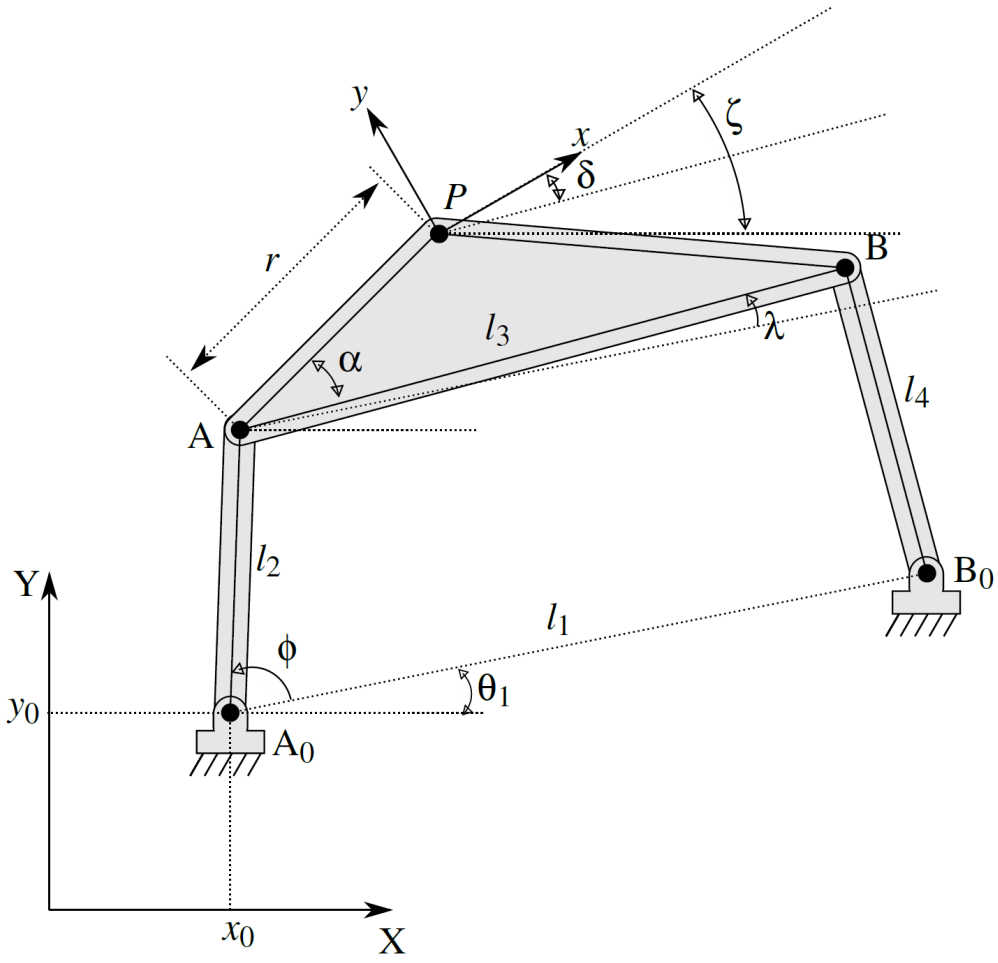


Figure 2.2: Visualization of parameters describing a four-bar mechanism

2.2.1 Coupler angle

The Fourier series representation of the coupler angle λ for a four bar mechanism is given as

$$e^{j\lambda} = \sum_{k=-\infty}^{\infty} C_k e^{jk\phi} = \sum_{k=-\infty}^{\infty} C_k e^{jk\omega t} e^{jk\phi_0}, \tag{2.1}$$

where C_k are the harmonic descriptors of coupler angle, ϕ the crank angle, ϕ_0 the initial crank angle, and ω is the constant angular speed of the input link.

2.2.2 Coupler path

The analytical equation which defines the path of coupler point P for a four bar mechanism is given by

$$P = A_0 + l_2 e^{j\theta_1} e^{j\phi} + r e^{j\alpha} e^{j\theta_1} e^{j\lambda}, \tag{2.2}$$

where A_0 is the complex form of the position of input link fixed pivot, l_2 is the length of input link, θ_1 is the angle of fixed link, and r and α are the coupler parameters. Being a periodic function, it can also be represented as a Fourier series:

$$\mathbf{P} = \sum_{k=-\infty}^{\infty} P_k e^{jk\omega t}. \tag{2.3}$$

Substituting (2.1) into (2.2) and then equating resulting (2.2) and (2.3), we get harmonic descriptors P_k for the path as following

$$\begin{aligned} P_0 &= C_0 r e^{j(\alpha+\theta_1)} + (jy_0 + x_0); \qquad k = 0, && (2.4)\\ P_1 &= C_1 e^{j\phi_0} r e^{j(\alpha+\theta_1)} + l_2 e^{j\theta_1} e^{j\phi_0}; \qquad k = 1, && (2.5)\\ P_k &= C_k e^{jk\phi_0} r e^{j(\alpha+\theta_1)}; \qquad k \neq 0, 1. && (2.6) \end{aligned}$$

2.2.3 Coupler orientation

The orientation (ζ) at the coupler point for a four-bar mechanism can be defined as

$$\zeta = \delta + \lambda + \theta_1 = \arg(e^{j(\delta+\lambda+\theta_1)}), \tag{2.7}$$

where δ is the fixed angle at which moving frame is attached to coupler with respect to $\theta_1 + \lambda$. As λ varies periodically while δ and θ_1 remain constant, the orientation can be decomposed harmonically as

$$e^{j(\delta+\lambda+\theta_1)} = \sum_{k=-\infty}^{\infty} C_k^* e^{jk\omega t}, \tag{2.8}$$

where C_k^* are the harmonic descriptors for orientation and obtained as

$$C_k^* = C_k e^{j(\delta+\theta_1)} e^{jk\phi_0} \tag{2.9}$$

by substituting for $e^{j\lambda}$ from Eq. (2.1) in Eq. (2.8).

2.2.4 Path-orientation relation

With the above relations, it is now possible to find explicit closed form relations between the Fourier descriptors of coupler path and coupler orientation data. Using Eqns. (2.4), (2.5), (2.6), and (2.9), relationship between the harmonic descriptors of path (P_k) and orientation (C_k^*) is found to be

$$C_0^* = (P_0 + z_2)z_1, \tag{2.10}$$
$$C_1^* = (P_1 + z_3)z_1, \tag{2.11}$$
$$C_k^* = P_k z_1, \tag{2.12}$$

where

$$z_1 = \frac{e^{j(\delta-\alpha)}}{r}, \tag{2.13}$$
$$z_2 = -(x_0 + jy_0), \tag{2.14}$$
$$z_3 = -(l_2 e^{j\theta_1} e^{j\phi_0}). \tag{2.15}$$

Using the above relationship, the orientation at coupler point can be defined exclusively using path harmonic descriptors as follows

$$e^{j\zeta(t)} = z_1 \left(z_2 + z_3 e^{j\omega t} + \sum_{k=-\infty}^{\infty} P_k e^{jk\omega t} \right). \tag{2.16}$$

Subsequently, using Eq. (2.16) for n path points, the system of equation describing orientation at each path point turns out to be

$$\begin{bmatrix} e^{j\zeta_1} \\ e^{j\zeta_2} \\ \vdots \\ e^{j\zeta_n} \end{bmatrix} = \begin{bmatrix} 1 & e^{j\omega t_1} & \sum_{k=p}^{-p} P_k e^{jk\omega t_1} \\ 1 & e^{j\omega t_2} & \sum_{k=p}^{-p} P_k e^{jk\omega t_2} \\ \vdots & \vdots & \vdots \\ 1 & e^{j\omega t_n} & \sum_{k=p}^{-p} P_k e^{jk\omega t_n} \end{bmatrix} \begin{bmatrix} z_1 z_2 \\ z_1 z_3 \\ z_1 \end{bmatrix}. \tag{2.17}$$

Thus, the orientations at different points of a four-bar coupler path are dependent on path descriptors and three complex variables z_1, z_2, and z_3, which are termed as Mixed Synthesis Parameters (MSP). The MSP are dependent on four-bar mechanism design parameters according to Eqns. (2.13), (2.14), and (2.15). The Eq. (2.17) is the key to the mixed synthesis formulation. It will help us find orientation information for path points as discussed in the next section.

2.3 Calculating unknown orientations

The aim of this section is to reformulate m-pose, n-path point mixed synthesis problems into an $m+n$- pose motion synthesis problems. To enable that, generation of orientation data for n path points and converting them to n poses is required. Eq. (2.17) will be used to accomplish this objective.

For a m-pose, n-path synthesis problem, a task path described by a trigonometric polynomial curve with an open interval is calculated and represented as

$$z(t) = \sum_{k=-p}^{p} T_k e^{ik\omega t} \qquad \forall \quad t \in [0, t_{max}], t_{max} < 1, \tag{2.18}$$

where $z(t) = x(t) + iy(t)$ denotes the point coordinates in complex form at time t, k are the frequency indices, T_k are the task curve Fourier descriptors, ω is the angular velocity of crank, and $[0, t_{max}]$ is the time interval over which the curve is defined. The T_k can be calculated by least square minimization of

$$\Delta = \sum_{i=1}^{n} \left\| z(t_i) - \sum_{k=-p}^{p} T_k e^{ik\omega t_i} \right\|^2, \tag{2.19}$$

where Δ is the fitting error measure and $z(t_i)$ are the complex-valued point data at time t_i. Analytically solving the minimization problem gives a linear system of equation as follows

$$\Omega \mathbb{X} = \mathbb{Y}, \tag{2.20}$$

where

$$\mathbb{X} = [\ldots, \underset{m\rightarrow}{T_m}, \ldots]^T, \tag{2.21}$$

$$\Omega = \begin{bmatrix} & \cdots & \\ \vdots & \sum_{i=0}^{n} e^{i(k-m)\theta_i} & \vdots \\ & \cdots & \\ & \underset{k\rightarrow}{} & \end{bmatrix} \downarrow m, \tag{2.22}$$

$$\mathbb{Y} = [\ldots, \underset{m\rightarrow}{\sum_{i=0}^{n} z(t_i) e^{-im\theta_i}}, \ldots]^T. \tag{2.23}$$

Here, k and m vary from $-p$ to p which denote the column and row index of an element in the matrix. Thus, $\mathbb{X}$ and $\mathbb{Y}$ are m-dimensional vectors while Ω is a $m \times n$ dimensional matrix. LU decomposition can be used to solve the above system. More details can be found in the work done by Wu et al. [14]. In [24], we have proposed a method to calculate optimal time parametrization for task curve for Fourier descriptor fitting of the path data. In our implementation, task curves are represented using up to eleven descriptors i.e. $p \in [-5, 5]$. If $m + n < 11$, we use a lesser number of descriptors to generate a unique task curve.

The reasoning behind using a task curve with low higher order harmonic content is supported in literature [16, 25], which says that the magnitude of high harmonics for coupler path of a four-bar mechanism has an insignificant impact. Thus, the fitted task path is a good prospective four-bar coupler curve and the task curve descriptors T_k can be equated to coupler path descriptors P_k.

The intention now is to find the MSP i.e. $\{z_1, z_2, z_3\}$ using available orientation data and subsequently generate unknown orientations. We define the system of equation given by Eq. (2.17) as fully constrained if all the MSP can be calculated exactly. For a fully-constrained MSP computation problem, three poses are required to calculate the MSP directly from Eq. (2.17).

Physically, this condition makes perfect sense as the user might know orientations at the initial position, final position and an additional intermediate location while a sequence of path points might be given in addition.

For under-constrained MSP computation problem, there are only one or two poses given. As a result, additional constraints are required to uniquely calculate the MSP. The MSP are dependent on four-bar mechanism parameters according to Eqns. (2.13), (2.14), and (2.15). These equations can be used to generate additional constraints which are called Mixed Constraints (MIC). The three possible MIC are

1. Specify coupler parameters i.e. $\{r, \alpha, \delta\} \rightarrow z_1$
2. Specify actuating fixed pivot i.e. $\{x_0, y_0\} \rightarrow z_2$
3. Specify scale of input link, orientation of fixed pivot line, and initial angle i.e. $\{l_2, \theta_1, \phi_0\} \rightarrow z_3$

Thus, if two poses are input by the user, one MIC is required to fully define the system of equations in Eq. (2.17). If only one pose is specified by user, two MIC are required to solve the problem. Two pose problem is fairly common when only the first and last orientations are important, such as in pick-and-place operations. The MIC also mirror practical user-specified constraints, such as selection of the location of the fixed pivot where an actuator might be situated. In another case, there might be a restriction on coupler link dimensions. Thus, the MIC represent a set of practical design constraints.

It is important to note that a pure path synthesis problem cannot be restructured into a motion synthesis problem without fully defining all three MSP. However, constraining all MSP simultaneously makes the synthesis less useful as most of the mechanism parameters are then fixed.

For over-constrained MSP computation problems, the number of poses specified is more than three. In this case, a least square solution to Eq. (2.17) can be calculated using complex Singular Value Decomposition (SVD). Real SVD solvers, which are more easily available, can also be used by reducing the complex system of equation in Eq. (2.17) into an equivalent real system of equation in accordance with [26]. The K_1 formulation presented in [26] has been used in our implementation. According to the formulation, a complex system of equation

$$(A + iB)(x + iy) = b + ic \tag{2.24}$$

can be written as a real system of equation

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{2.25}$$

Finding least square solution to this equivalent real system of equations gives the solution to original complex problem and values of MSP can easily be calculated in over-constrained cases.

Once the values of MSP z_1, z_2, z_3 are calculated using m poses, orientations at n path points can be found out by simple matrix multiplication using the system of equation in Eq. (2.17). As a result, n path points and m poses are converted to $m + n$ poses. The motion synthesis algorithm can now be used to calculate dyads. A review of algebraic fitting based motion synthesis algorithm is discussed in next section.

2.4 Motion synthesis algorithm

Now that the mixed synthesis problem has been reformulated as motion synthesis problem, solution mechanisms can be achieved by calculating the dyads. Algebraic fitting based motion synthesis algorithm [7–10] has been used in our implementation. In this approach, a planar four-bar linkage is split open in two dyads and each dyad is computed independently thus reducing significant computational burden. Moreover, this approach enables us to carry out simultaneous type and dimensional synthesis of four-bar linkages, i.e. it takes into consideration the possibility of both revolute and prismatic joints. Another benefit of the approach is its fast and efficient computation.

First, using kinematic mapping [27], each of the user-defined pose $\{x, y, \zeta\}$ is mapped to quaternion space defined by a four-dimensional vector $\mathbf{Z} = \{Z_1, Z_2, Z_3, Z_4\}$ called planar quaternions [28]. This space is also termed as the Image Space of planar kinematics [27]. This mapping is defined by

$$Z_1 = \frac{1}{2}(x \cos\frac{\zeta}{2} + y \sin\frac{\zeta}{2}), \tag{2.26}$$

$$Z_2 = \frac{1}{2}(-x \sin\frac{\zeta}{2} + y \cos\frac{\zeta}{2}), \tag{2.27}$$

$$Z_3 = \sin\frac{\zeta}{2}, \tag{2.28}$$

$$Z_4 = \cos\frac{\zeta}{2}. \tag{2.29}$$

The geometric constraints of all types of dyads can be represented by a single algebraic equation as following

$$\begin{aligned} q_1(Z_1^2 + Z_2^2) + q_2(Z_1Z_3 - Z_2Z_4) + q_3(Z_2Z_3 + Z_1Z_4) \\ +q_4(Z_1Z_3 + Z_2Z_4) + q_5(Z_2Z_3 - Z_1Z_4) + q_6Z_3Z_4 \\ +q_7(Z_3^2 - Z_4^2) + q_8(Z_3^2 + Z_4^2) = 0, \end{aligned} \tag{2.30}$$

where $q_i (i = 1, 2, \cdot, 8)$ are the homogeneous coefficients of the manifold surface represented by the above equation. In [10], we call this as a generalized (G-) manifold, which is capable of representing all types of mechanical dyads. For every pose, one such linear equation with unknowns as q_i is obtained. Assembling all the G-manifold equations for all the poses results in the following over-constrained homogeneous linear system on equation

$$Aq = 0, \tag{2.31}$$

where

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \cdots & \cdots & \cdots & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & \cdots & \cdots & \cdots & A_{28} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & A_{n4} & \cdots & \cdots & \cdots & A_{n8} \end{bmatrix}, \tag{2.32}$$

and

$$q = \begin{bmatrix} q_1 & q_2 & \cdots & q_8 \end{bmatrix}^T \tag{2.33}$$

The elements of each row of the matrix A are given as

$$A_{i1} = Z_{i1}^2 + Z_{i2}^2, \tag{2.34}$$
$$A_{i2} = Z_{i1}Z_{i3} - Z_{i2}Z_{i4}, \tag{2.35}$$
$$A_{i3} = Z_{i2}Z_{i3} + Z_{i1}Z_{i4}, \tag{2.36}$$
$$A_{i4} = Z_{i1}Z_{i3} + Z_{i2}Z_{i4}, \tag{2.37}$$
$$A_{i5} = Z_{i2}Z_{i3} - Z_{i1}Z_{i4}, \tag{2.38}$$
$$A_{i6} = Z_{i3}Z_{i4}, \tag{2.39}$$
$$A_{i7} = Z_{i3}^2 - Z_{i4}^2, \tag{2.40}$$
$$A_{i8} = Z_{i3}^2 + Z_{i4}^2, \tag{2.41}$$

where i is the pose index ranging from $i = (1, 2, \cdots, n)$. The least square solution to this homogeneous system of equation can be found out using

the singular value decomposition of the coefficient matrix A [29]. The right singular vectors corresponding to the smallest singular values are candidate solutions for the minimization problem. The subspace spanned by the three smallest singular value right-vectors represents a family of possible dyad solutions. However, for these dyads to make physical sense, the following extra constraints are required to be satisfied

$$\begin{aligned} q_1 q_6 + q_2 q_5 - q_3 q_4 &= 0 \\ 2q_1 q_7 - q_2 q_4 - q_3 q_5 &= 0 \end{aligned} \tag{2.42}$$

An analytical solution to the above reduces to a quartic equation, which can give zero, two, or four real dyad solutions. Complex solutions do not represent a physical dyad. Combining any of the two dyads results in a four-bar mechanism. For further details, see [10]. As a result, the path synthesis problem is solved and prospective solutions are generated. Using this motion synthesis algorithm also enables us to simultaneously carry out type and dimensional synthesis. However, that is not the focus of this work.

The above methodology for motion computation works for five or more poses when the system of equation is fully-constrained or over-constrained. To handle under-constrained cases, additional Motion Synthesis constraints (MOC) also called geometric constraints outlined in [8] are used. We note that the constraints being discussed here are the ones required to define the motion computation problem, which are different from the constraints discussed earlier in the context of MSP computation. We ensure that the context would make it clear which constraints are being discussed.

2.5 Unified synthesis algorithm

A unified synthesis algorithm to solve the Alt-Burmester problems has been summarized in Algorithm 3. It states that when the synthesis problem has zero poses, the Fourier descriptor based path synthesis algorithm as described by Wu et al. [14] is used. For all the other cases, mixed synthesis approach using Eq. (2.17) can be used to solve for four-bar mechanisms. It must be noted that except for the case where there are no poses, the synthesis calculates both type and dimensions.

A key advantage of the methodology outlined is that it can handle motion, path and mixed synthesis problems seamlessly. Various permutations of

Algorithm 1: Algorithm for Unified Motion, Path and Mixed Synthesis

Input: Path points and Poses

if *n(Pose)=0* **then**
  Calculate T_k using Eq.(3.2)
  Calculate the four-bar mechanism by solving a minimization problem in a four dimensional subspace as described by Wu et al. [14]
else
  Calculate T_k using Eq.(3.2)
  Calculate MSP using Eq.(2.17)
  Calculate the singular vectors using Eq.(2.31)
  Calculate the dyads using Eq.(2.42)
end

Output: Synthesized mechanism

$(0, 1, \cdots, m)$ poses and $(0, 1, \cdots, n)$ path point problems are presented in Table 2.1. The legends in the table are MOC = Motion Synthesis constraint [8], MIC = Mixed Synthesis Constraint, FD = Fully Defined, and X=trivial or undefined. The * refers to conditions where a Fourier task curve with just four points needs to be fitted and would have unsymmetrical descriptors.

Table 2.1: Various Possibilities for Unified Motion, Path, and Mixed Synthesis Problem

		Path Points					
		0	1	2	3	4	n
Poses	0	X	X	X	X	FD*	FD
	1	X	X	X	2 MIC*	2 MIC	2 MIC
	2	X	X	1 MIC*	1 MIC	1 MIC	1 MIC
	3	2 MOC	1 MOC*	FD	FD	FD	FD
	4	1 MOC	FD	FD	FD	FD	FD
	5	FD	FD	FD	FD	FD	FD
	m	FD	FD	FD	FD	FD	FD

Motion Synthesis constraints can be used to specify the position of fixed or moving pivots using line or point constraints or any other compatible geometric constraint; see [8] for details. Mixed Synthesis Constraints, described

earlier, involve constraints on actuating pivot, coupler dimensions, and other mechanism parameters. Fully Defined entails that no extra constraints are needed to exactly or least square solve the mixed synthesis Eq. (2.17). If either one of the MSP computation problem or Motion synthesis problem is under-constrained, the mixed synthesis problem is defined to be under-constrained. These under-constrained cases in Table 2.1 require additional constraints to be solved. In the table, zero-pose (or, pure-path synthesis) problems are solved using Wu et al. [14]. In that case, when four or more path-points are specified, the problem is fully defined and is non-trivial; however for $n = 4$, we can only calculate unsymmetrical descriptors. When only one pose is given, then two MIC are required to calculate all the MSP; with two poses, one MIC is required; and for three poses, no additional MIC are needed. However, for three poses, at least two path points need to be specified to obtain five poses needed for the motion synthesis algorithm. Burmester [4] showed that one needs five poses to solve a motion generation problem uniquely. In [7], we have extended Burmester problem to show that one can specify not only five poses, but a combination of pose and other geometric constraints to have unique mechanism design solutions. Therefore, for one path-point with three poses, we need one MOC and for zero path point, we need two MOC to get a total of five constraints. However, zero path-point problem reduces to a pure motion generation problem. We will illustrate some of these permutations and combinations in the examples next.

2.6 Examples

In this section, we present some examples to illustrate the effectiveness of the proposed algorithm. First example aims to validate the approach by extracting path-points and poses from a known mechanism. Second example solves mixed synthesis problem with fully-constrained MSP computation involving three poses and five path points. Third and fourth examples deal with under-constraint MSP computation and motion synthesis cases and require additional mixed- and motion-constraints, respectively. Demonstrating valid results from each of these cases proves the robustness of proposed algorithm. It also demonstrates the flexibility of the algorithm and its ability to incorporate various constraints. In the Figures 2.3, 4.2, 4.3, 2.6, 4.4 and 2.8, the blue and red curves denote the coupler curve in two possible assembly modes.

2.6.1 Example 1: Reverse Engineering a Mechanism

To validate the proposed mixed synthesis algorithm, points and poses from a known planar four-bar mechanism are taken and then our algorithm is used to synthesize mechanisms. Ideally, we should get the exact same mechanism. However, a similar mechanism is also acceptable since approximations occur at various steps – from task curve generation to algebraic fitting of the pose data.

A sample mechanism displayed in Fig. 2.3 is used to generate seven path-points and five poses. The mechanism has been defined using the position of its fixed pivots, moving pivots and coupler coordinates as shown in Table 2.2.

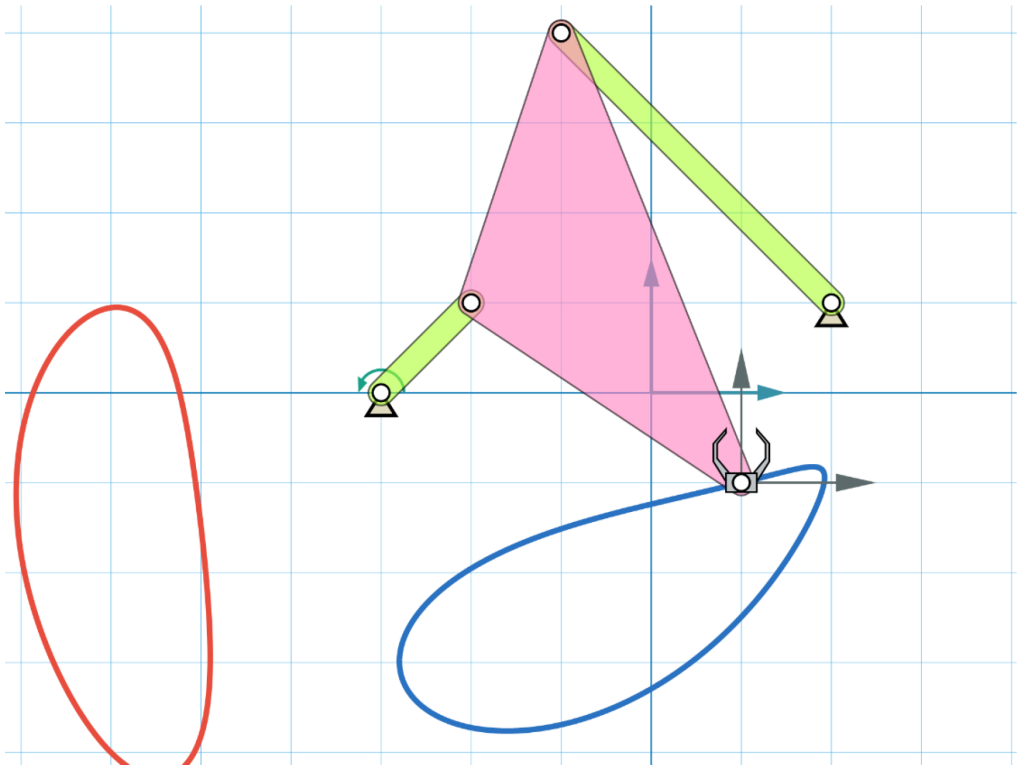


Figure 2.3: Example 1: Known target mechanism

The arbitrarily sampled poses and path points are listed in Table 2.3. This is a fully defined problem and all the MIC can be computed without requiring any additional information. These constraints are used as input to mixed

Table 2.2: Example 1: Sample mechanism design parameters as shown in Fig. 2.3

Point	X	Y
Input link fixed pivot	−3.0	0.0
Input link moving pivot	−2.0	1.0
Output link fixed pivot	2.0	1.0
Output link moving pivot	−1.0	4.0
Coupler point	1.0	−1.0

synthesis algorithm. Four solution dyads are output as listed in Table 2.4. This also allows us to reverse-engineer a known mechanism since there are six planar four-bars that can satisfy the given constraints. Figure 4.2 shows a four-bar obtained by assembling dyads 1 and 4. It is observed that the mechanism generated is very similar to the original mechanism. This approximate result is due to the best-fitted low harmonic task curve following the original coupler curve closely but not exactly. The average path error measured by calculating the deviation of input points from final path using the Euclidean distance is 0.0694 units for the displayed configuration. The maximum angular deviation among all the given poses is for the pose 3 as 0.0736 rad.

Table 2.3: Example 1: Input data

No.	Type of Data	x	y	ζ (rad)
1	Point	0.350	−1.160	
2	Point	−0.410	−1.340	
3	Pose	−1.585	−1.737	5.853
4	Point	−2.110	−2.030	
5	Point	−2.800	−2.970	
6	Pose	−2.216	−3.665	5.896
7	Point	−0.420	−3.500	
8	Point	0.910	−2.580	
9	Pose	1.520	−1.832	0.351
10	Pose	1.912	−1.036	0.385
11	Point	1.560	−0.860	
12	Pose	1.000	−1.000	0.000

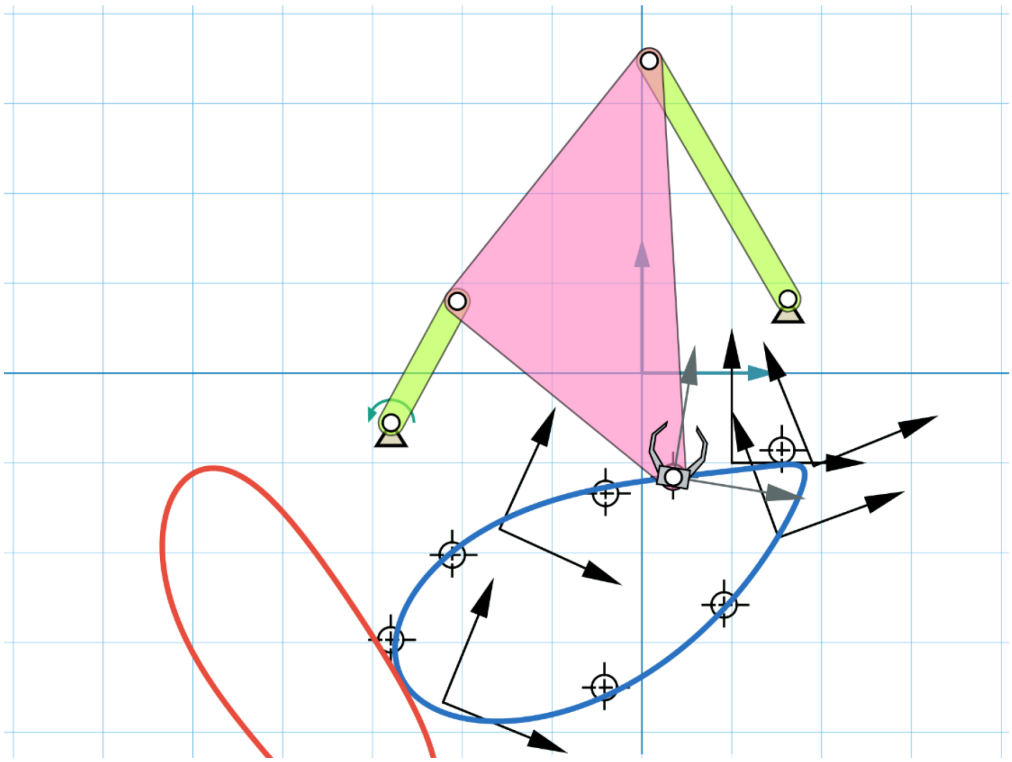


Figure 2.4: Example 1: Mechanism generated using mixed synthesis algorithm

Table 2.4: Example 1: Output dyad data

Dyad	Fixed Pivot	Moving Pivot	Coupler Point
1	$-2.793, -0.554$	$-2.056, 0.799$	$0.350, -1.160$
2	$-0.172, 6.978$	$1.340, 8.356$	$0.350, -1.160$
3	$-18.181, 11.280$	$-4.226, 6.257$	$0.350, -1.160$
4	$1.625, 0.824$	$0.081, 3.477$	$0.350, -1.160$

In this example, the input data consisted of five poses and seven path points. Greater than three input poses over-constraints the MSP computation due to which the path-orientation relationship is satisfied using SVD. Thus, this example demonstrates mixed synthesis with over-constrained MSP computation.

2.6.2 Example 2: Mixed synthesis with fully-constrained MSP computation

In this example, the input data consists of three poses and five path points which fully constrains the MSP computation problem. The data input to mixed synthesis algorithm is given in Table 2.5. The two dyads generated as output have been shown in Table 2.6. The final mechanism has been displayed in Fig 4.3. It can be observed that a good match has been established with the constraints. Average path error is 0.0226 units while the maximum angular deviation for poses is 0.0106 rad for second pose. Note that in this case, the path-orientation relationship has an exact solution, i.e. MSP are uniquely determined using SVD.

Table 2.5: Example 2: Input data

No.	Type of Data	x	y	ζ (rad)
1	Pose	-5.263	1.441	0.161
2	Point	-3.810	1.690	
3	Point	-2.890	1.590	
4	Point	-2.010	1.120	
5	Pose	-1.416	0.789	5.919
6	Point	-0.200	0.490	
7	Point	1.040	0.600	
8	Pose	2.206	1.203	0.405

Table 2.6: Example 2: Output dyad data

Dyad	Fixed Pivot	Moving Pivot	Coupler Point
1	$0.771, 3.424$	$-2.927, 0.266$	$-5.263, 1.441$
2	$-3.931, -2.151$	$-6.160, 6.975$	$-5.263, 1.441$

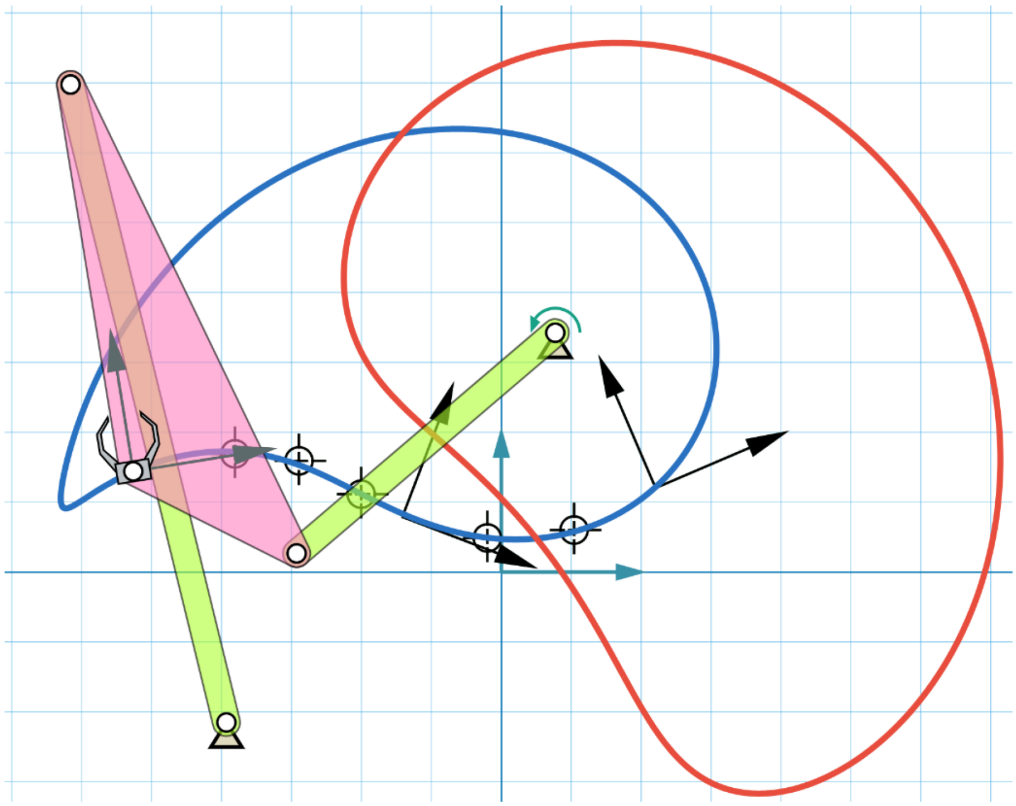


Figure 2.5: Example 2: Mixed synthesis with fully-constrained MSP computation for three poses and five path points

One of the major advantages of mixed synthesis is the additional flexibility it imparts to users while specifying inputs and generating good solutions. Using a pure motion synthesis algorithm, the user would have to input all the data as poses even if the problem demanded otherwise. This would lead to an over-constrained motion problem, which when solved using existing kinematic mapping based algebraic fitting approach [7–10] usually produces poor solutions. A comparable motion synthesis problem for the same path points, but with orientations also given is displayed in Fig 2.6. It can be observed that the solution provides a poor fit to the given constraints. This happens because the orientation provided are not compatible with the motion of coupler of the planar-four linkages.

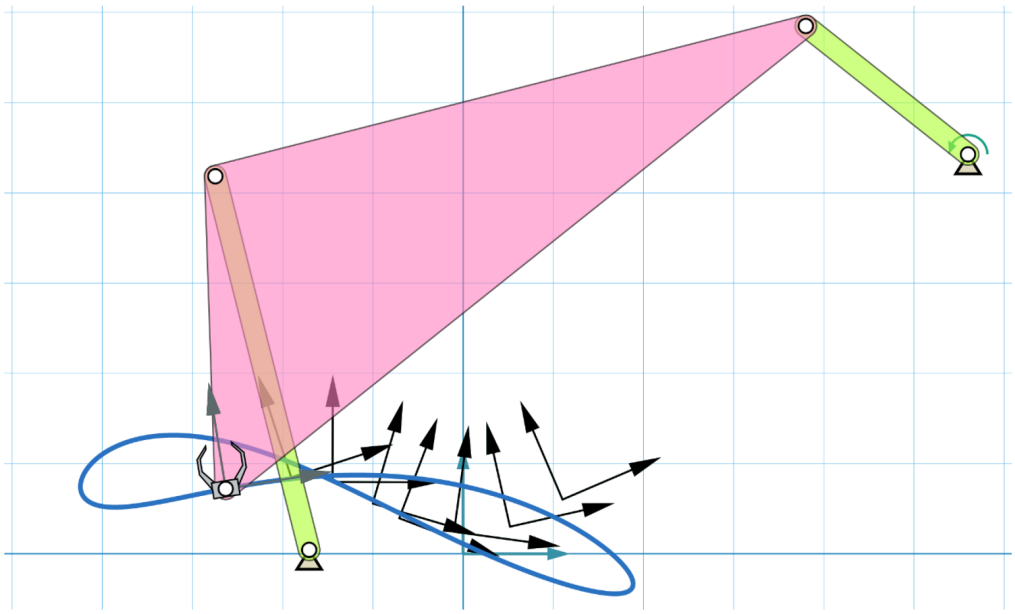


Figure 2.6: Example 2: Over-constrained motion synthesis for eight poses produces a poor solution.

2.6.3 Example 3: Mixed synthesis with under-constrained MSP computation using mixed constraints

This example shows mixed synthesis with under-constrained MSP computation problem where two poses and four path points are specified in the in-

put. Lesser than three input poses makes MSP computation problem under-constrained. To solve for MSP, an additional mixed constraint is required which could specify any of z_1, z_2, z_3. The constraint data input to mixed synthesis algorithm is shown in Table 2.7. A MIC is used to specify z_2 by defining the preferred location of a fixed joint at point $(1, 3)$. The four dyads generated as output are shown in Table 2.8. One of the final mechanisms is displayed in Fig 4.4 using dyads 3 and 4. It can be observed that the generated mechanism closely satisfies path and mixed constraints. The average path error is 0.0948 units while the maximum angular deviation for poses is 0.0389 rad for the second pose in the displayed mechanism. Note that in this case, the path-orientation relationship has an infinite solutions and the use of MIC restricts the solution space to a unique solution to the MSP.

Table 2.7: Example 3: Input data

No.	Type of Data	x	y	ζ (rad)
1	Pose	4.962	-0.514	0.134
2	Point	3.850	-1.480	
3	Point	1.920	-0.740	
4	Point	0.850	0.760	
5	Point	3.360	1.650	
6	Pose	4.900	1.178	0.510

Table 2.8: Example 3: Output dyad data

Dyad	Fixed Pivot	Moving Pivot	Coupler Point
1	$8.705, 10.738$	$8.705, 10.738$	$4.962, -0.514$
2	$-7.961, 8.082$	$-3.450, 6.381$	$4.962, -0.514$
3	$0.973, 3.179$	$1.004, -0.295$	$4.962, -0.514$
4	$4.748, 0.792$	$6.635, -0.004$	$4.962, -0.514$

2.6.4 Example 4: Mixed synthesis with under-constrained motion synthesis using motion constraints

This example shows mixed synthesis with under-constrained motion synthesis problem where three poses and one path point is specified in the input.

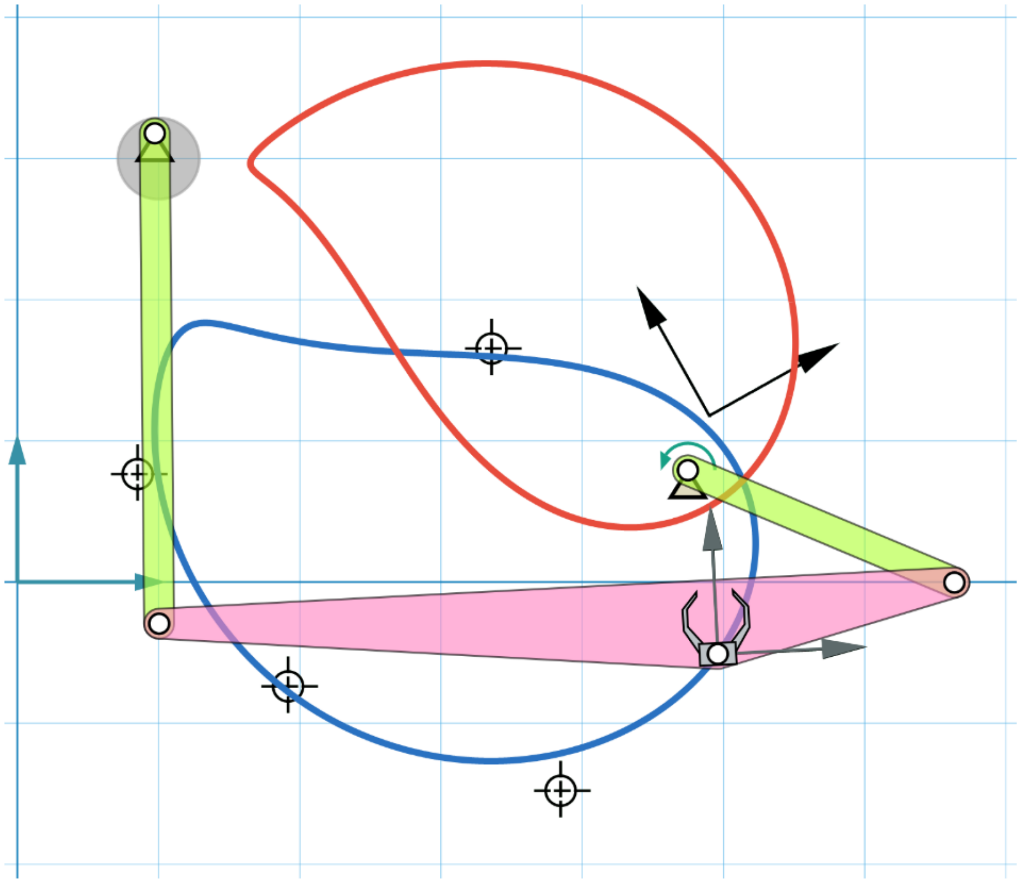


Figure 2.7: Example 3: Under-constrained mixed synthesis for two poses and four path points using additional mixed constraint

Motion synthesis is under-constraint because the total pose and path constraints are just four. Even though three poses specified can be used to calculate the MSP, an additional motion constraint is required to solve the motion synthesis problem. The data input to mixed synthesis algorithm is given in Table 2.9. A line constraint is used as MOC in the example presented. The line segment is defined by its end points $(-4, 1)$ and $(1, 4)$. The four dyads generated as output are shown in Table 2.10. One of the final mechanisms is displayed in Fig 2.8 using dyads 1 and 2. It can be observed that the generated mechanism closely satisfies path constraints. The average path error is 0.0026 units while the maximum angular deviation for poses is 0.0005 rad for second pose in the displayed mechanism. Also, both the fixed pivots fall on the line constraint specified. Thus, the synthesis problem is successfully solved. Note that in this case, it is not the path-orientation relationship that is under-defined but the algebraic fitting algorithm which requires at-least five poses to be fully defined.

Table 2.9: Example 4: Input data

No.	Type of Data	x	y	ζ (rad)
1	Pose	-2.018	-1.391	0.146
2	Pose	0.288	-1.115	0.287
3	Point	1.700	0.360	
4	Pose	2.895	1.253	1.487

Table 2.10: Example 4: Output dyad data

Dyad	Fixed Pivot	Moving Pivot	Coupler Point
1	$0.647, 3.788$	$0.058, 2.651$	$-2.018, -1.391$
2	$-1.181, 2.692$	$-2.002, 1.708$	$-2.018, -1.391$
3	$-3.322, 1.407$	$-4.111, 3.061$	$-2.018, -1.391$
4	$-3.322, 1.407$	$-4.111, 3.061$	$-2.018, -1.391$

2.7 Conclusion

In this chapter, we have presented a generalized m pose, n path-point mixed synthesis approach for four-bar mechanisms. Original contributions of this

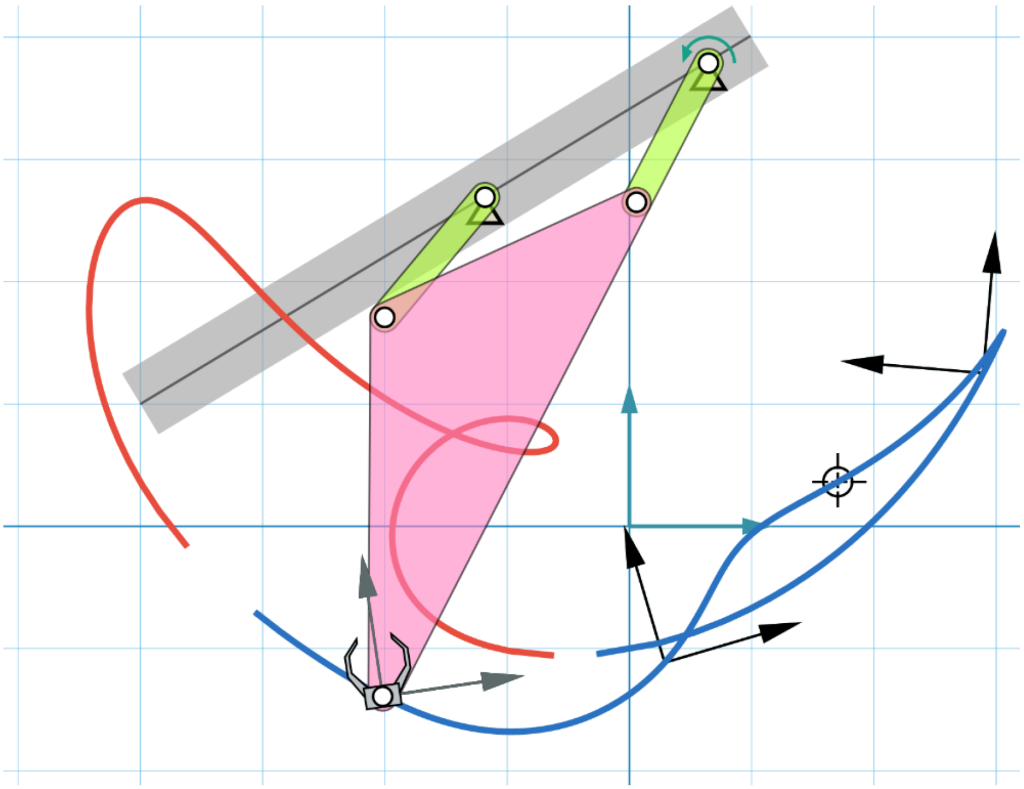


Figure 2.8: Example 4: Under-constrained mixed synthesis for three poses and one path points using additional motion constraint

chapter include the closed-form relationship between coupler orientation and coupler path and exploiting this relationship to present a novel framework for solving the mixed synthesis problem. Another novel feature is the use of task-driven motion synthesis algorithm within the framework to keep the computation cost at minimum and perform simultaneous type and dimensional synthesis. A few examples were presented to demonstrate the effectiveness of the approach.

Chapter 3

Path Synthesis

3.1 Introduction

This chapter is concerned with the path generation problem, wherein a path is usually given as a sequence of discrete points in R^2 and the goal is to find dimensions of a planar four-bar mechanism such that a point on the coupler of the mechanism traces the given points as close as possible [30]. Optimization-based techniques attempt to minimize an objective function and find mechanisms, which best approximate a curve generated by using Fourier series [31]. This curve, called the task curve requires point data on the prescribed path *and* associated time parameter values. Discrete Fourier transform (DFT) has been used to calculate the Fourier coefficients or Fourier descriptors (FDs) of the task curve from the prescribed path. FDs have been frequently used in computational shape analysis to create the objective function [13,15,19,25,32–34]. This chapter focuses on solving the path generation problem using a FD based technique.

McGarva and Mullineux [32] studied the inherent dependency of FDs of a task curve on time parametrization and concluded that different parametrization leads to different FDs. In previous studies, this limitation has been ignored and the parametrization has been assumed to be uniform. This results in a task curve which might be sub-optimal for use in mechanism synthesis.

Fig. 3.1 demonstrates an instance, where using a non-uniform parametrization yields a better mechanism than a uniform parametrization. A test mechanism is taken and ten arbitrary points are sampled on its coupler curve to generate the input data. Once the task curve is calculated, a four-bar cou-

pler curve is fitted to synthesize a mechanism for each case. Comparing the resulting mechanisms with the input shows that non-uniform parametrization fits the points more accurately and better matches the first mechanism. Although, in Computer Aided Design (CAD), finding parametrization for a given sequence of points in the context of curve interpolation is a common problem, here we have the additional burden of ensuring that the parametrization is compatible with the properties of the coupler curves of planar four-bar mechanisms. In the example presented later on, we will show that the optimized task curve matches the harmonic contents of the coupler curve better than an arbitrary choice of parameters. Finding this optimum non-uniform parametrization serves as the motivation of this chapter.

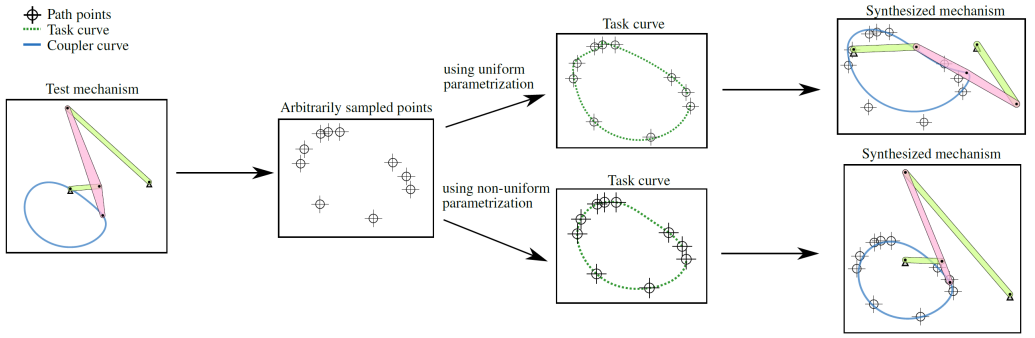


Figure 3.1: Path generation of two four-bar mechanisms; one using uniform parametrization while the other using optimal non-uniform parametrization

Recently, Li et al. [35] proposed an approach to eliminate time dependency using arc length parametrization. It is well-known in CAD community that finding an explicit closed-form expression for arc-length parametrization is impossible. As a result, they numerically *guess* the non-uniform parametrization using the distance between input points. Geometrically, the formulated parametrization is close to being time-independent. But this approach forces the coupler point to move at a constant speed along the task curves and eliminates a host of other possible four-bar mechanisms.

In this chapter, a novel methodology which calculates the optimal parametrization for a sequence of points has been proposed. For a four-bar coupler curve, the magnitude of its higher order harmonics has been observed to be negligible by Freudenstein [25] and Li et al. [16]. This property is used to search for an optimal parametrization to generate a task curve with low magnitude

higher order harmonics. Coupler speed criteria for enhanced control over the task curve has also been incorporated. A new cost function is proposed that combines the cost of fitting to the given data points, the cost of low magnitude higher order harmonic, and the penalty for larger speed ratios. Nelder-Mead optimization is used to compute two critical state space parameters that minimize the cost function and provide optimized time parameters. Thereafter, we use the algorithm presented by Wu et al. [33] to synthesize a four-bar mechanism from the optimal task path. This algorithm fits task curve FDs to coupler curve FDs using a four-dimensional search space instead of the conventional ten-dimensional search space.

The rest of this chapter is organized as follows. Section 3.2 reviews an existing FD based path generation approach, which supplements the proposed algorithm for four-bar mechanism synthesis. Section 3.3 introduces a family of non-uniform parametrization and a methodology for finding the optimal one among them. Section 3.4 presents an example, which illustrates the effectiveness of the proposed approach.

3.2 Review of Fourier descriptor based path generation

This section provides a brief overview of an existing FD based path generation algorithm proposed by Wu et al. [33], which is used in conjunction with the improved task curve generation method as discussed in Sec. 3.3 for four-bar synthesis.

In this method, input path points are used to calculate a task curve described by a trigonometric polynomial curve with an open interval and is represented as

$$z(t) = \sum_{k=-p}^{p} T_k e^{ik\omega_o t} \qquad \forall \quad t \in [0, t_{max}], \tag{3.1}$$

where $z(t) = x(t) + iy(t)$ denotes the point coordinates in complex form at time t, k are the frequency indices, T_k are the FDs, ω_o is the angular velocity of crank, and $[0, t_{max}]$ is the time interval over which the curve is defined. It has been shown that a task curve with $p = 5$ captures the four-bar coupler path very accurately and is deemed sufficient for practical implementation [16]. Taking $\omega_0 = 2\pi$ and $t_{max} \in (0, 1]$ represents the possible

closed and open task paths. The FDs are calculated by solving the least square fitting problem with the objective function as

$$\Delta = \sum_{i=1}^{n} \left\| z(t_i) - \sum_{k=-p}^{p} T_k e^{ik\omega_o t_i} \right\|^2, \tag{3.2}$$

where n is the number of input points. If $n < (2p+1)$, then the highest $2\left\lceil \frac{2p+1-n}{2} \right\rceil$ harmonics are specified as zero to find a unique solution. Here, $\lceil ... \rceil$ represents the ceiling function. Finding the domain $[0, t_{max}]$ for the open task path is a one-dimensional optimization problem for minimum error measure defined in Eq. (3.2). Once a time parametrization is chosen or assumed, the task curve can be calculated.

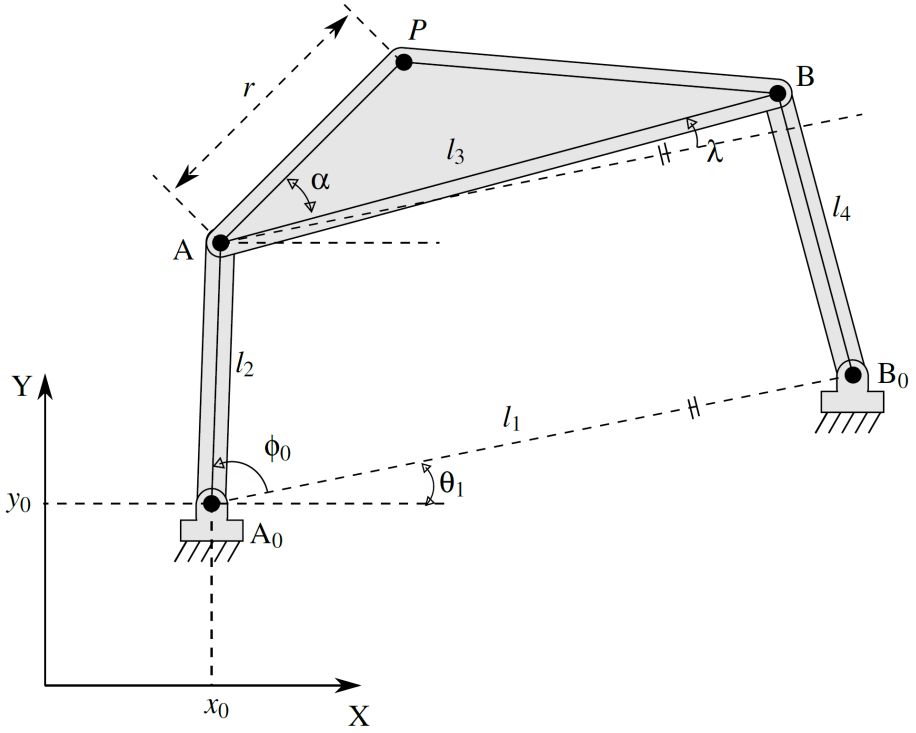


Figure 3.2: A planar four-bar mechanism showing dimensional parameters

A four-bar mechanism can be represented by the design parameters x_0, y_0, l_1, l_2, l_3, l_4, r, α, θ_1, and initial crank angle ϕ_0 as displayed in Fig. 3.2. The analytic equation of coupler point P can be given as

$$P = A_0 + l_2 e^{i\theta_1} e^{i\phi} + r e^{i\alpha} e^{i\theta_1} e^{i\lambda}, \tag{3.3}$$

where λ is the coupler angle as shown in Fig. 3.2 and A_0 is the complex form of the fixed pivot given by (x_0, y_0). A closed form expression for λ using loop-closure criteria is available in literature [33], and can be approximated using FDs as

$$e^{i\lambda} = \sum_{k=-p}^{p} C_k e^{ik\phi}, \tag{3.4}$$

where C_k are coupler angle FDs whose value depends upon the link lengths l_1, l_2, l_3 and l_4, and $\phi = \phi_0 + \omega_0 t$.

The coupler path can also be expressed as a Fourier series given in Eq. (3.5) where P_k are coupler path FDs.

$$P = \sum_{k=-p}^{p} P_k e^{ik\omega t}. \tag{3.5}$$

By substituting Eq (3.4) into Eq. (3.3) and collecting coefficients of $e^{ik\omega t}$ to form Eq (3.5), we can express the P_k as

$$P_0 = re^{i\alpha}e^{i\theta_1}C_0 + A_0, \tag{3.6}$$

$$P_1 = re^{i\alpha}e^{i\theta_1}C_1e^{i\phi_0} + l_2e^{i(\theta_1+\phi_0)}, \text{and} \tag{3.7}$$

$$P_k = re^{i\alpha}e^{i\theta_1}C_ke^{ik\phi_0}|_{k\neq 0,1}. \tag{3.8}$$

The coupler path can now be fitted to the task curve to calculate the four-bar design parameters using Eq. (3.1) and Eq. (3.5). Equating T_k to P_k leads to a system of equations with ten unknowns given as following

$$S = \left\{ l_2, \frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, x_0, y_0, \theta_1, \phi_0, \mathbb{C}, \mathbb{S} \right\}, \tag{3.9}$$

where $\mathbb{C} = r\cos(\alpha + \theta_1)$ and $\mathbb{S} = r\sin(\alpha + \theta_1)$. Equation (3.8) depends on six design variables $\left\{\frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, \phi_0, \mathbb{C}, \mathbb{S}\right\}$, while the remaining four variables $\{l_2, x_0, y_0, \theta_1\}$ exist independently in Eq. (3.6) and (3.7). Wu et al. [33] show that the six-dimensional design space in Eq (3.8) can be further reduced to a four-dimensional space of $\left\{\frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, \phi_0\right\}$ by analytically minimizing the following objective function

$$I = \sum_{k\neq 0,1} |C_k r e^{i(\alpha+\theta_1+k\phi_0)} - T_k|^2. \tag{3.10}$$

This objective function is obtained by summing the squared difference of P_k given in Eq. (3.8) and T_k.

The direct search method has been used by Wu et al. [33] to solve this optimization problem. In summary, matching task path FDs to coupler path FDs using a four-dimensional search space forms the core of this approach.

3.3 Optimum Parametrization

The task curve calculation is inherently associated with the time parametrization used. However, there are an infinite ways to select a non-uniform parametrization. To make the problem tractable and facilitate the selection of a single non-uniform parametrization, a chord-length based parametrization scheme for n-points sequence is defined as

$$t_1 = 0, \tag{3.11}$$

$$t_k = t_{max} \left(\frac{\sum_{i=2}^{k} |z_i - z_{i-1}|^{\alpha}}{\sum_{i=2}^{n} |z_i - z_{i-1}|^{\alpha}} \right). \tag{3.12}$$

Here, t_k represents the time parameter associated with the k^{th} path point, t_{max} is the interval domain as defined in Eq. (3.1), z_i represents the coordinates of i^{th} point and $\alpha \in \mathbb{R}$ is the parametrization control variable (PCV). Varying the control variable α generates multiple parametrizations. In this scheme, when $\alpha = 0$ and $\alpha = 1$, we obtain uniform and arc length parametrizations, respectively. Physically, uniform parametrization results in a task curve where the coupler takes equal amount of time to pass through each target point. Similarly, the arc length parametrization approximates constant speed motion of the coupler. By varying α, one can generate different parametrizations for the calculation of the task curves, which in turn, could provide a range of mechanism design solutions and also facilitates selection of an optimal parametrization, leading to mechanisms that provide better fit with the input data.

To measure the quality of the task curve, we define a cost function as following

$$C_t = C_f + C_h + C_s \tag{3.13}$$

where C_f is the cost attributed to FD fitting error, C_h is the cost due to higher order harmonic content, and C_s is the cost due to enforced speed criteria on the task curve.

If the path (z_i), PCV (α), and time domain (t_{max}) are known, the FDs of an open task curve can be mean square fitted as shown in Eq. 3.2. The square-root of the residual in the fitting process is normalized to define C_f in the cost function and is given as

$$C_f = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left\| z_i - \sum_{k=-p}^{p} T_k e^{ik\omega_o t_i}\right\|^2}. \tag{3.14}$$

Here, n represents the total path points, z_i are the path coordinates, T_k are the task curve FDs and t_i is the time parameter attached to i^{th} point. This term ensures that the selected parametrization and domain accurately represent the original point data.

The second term in the cost function, namely harmonic cost, is motivated by the observation from Li et al. [16] that the magnitude of higher order harmonics are negligible for four-bars. As a result, task curves whose higher order harmonics have minimal magnitude would be better prospective curves for the path synthesis process. To enforce this harmonic criterion, a weighted harmonic magnitude based metric is defined as

$$C_h = \frac{1}{2p+1}\sum_{k=-p}^{p}(|k|+1)^{\beta}\left\|T_k\right\|. \tag{3.15}$$

Here, p is the maximum number of harmonics being considered, $\beta \geq 1$ is a constant and T_k are the task curve FDs. The term $(|k|+1)^{\beta}$ adds larger weight to the higher order harmonics.

In practical scenarios, large coupler speed changes can lead to large induced forces on the links which could compromise their rigidity and render the kinematic analysis useless. In contrast, a uniform speed motion of the coupler can also be undesirable in some instances, such as when designing a quick-return mechanism. Thus, enforcing a speed-based criteria over the task curve gives users more control in design. To select a task curve with desirable speed properties, speed cost (C_s) is defined using quadratic penalty function as follows

$$C_s = w\left[max(0, S_{r,min} - S_r)^2 + max(0, S_r - S_{r,max})^2\right], \tag{3.16}$$

where S_r represents ratio of task curve's maximum to minimum speed. $S_{r,min}$ and $S_{r,max}$ are the minimum and maximum speed ratio bounds enforced by

the designer and w is the penalty weight imposed. In case there are no speed restrictions, $S_{r,min}$ and $S_{r,max}$ can be set to 1 and infinity, respectively. $w = 10^3$ has been taken in the implementation. The expression for task curve speed can be calculated by differentiating Eq. (3.1) which gives

$$s(t) = \left\| \sum_{k=-p}^{p} ik\omega_o T_k e^{ik\omega_o t} \right\|. \tag{3.17}$$

Maximum and minimum speeds can be calculated numerically by sampling a large number of points over the task curve.

Thus, an optimal task curve can be calculated by minimizing the total cost (C_t) as given in Eq. (3.13). The search space is two dimensional with α and t_{max} as the state variables. Thereafter, the optimal chord-length based time parametrization can be calculated using Eq. (3.11) and Eq. (3.12). Nelder-Mead optimization has been used for searching the state space. With the task curve known, a four-bar mechanism can be generated as discussed in Sec. 3.2. The complete Fourier based path synthesis algorithm using optimum parametrization has been summarized in the Algorithm 2.

Algorithm 2: Path generation using Fourier descriptor based approach with optimal parametrization

Input: Set of path points

1 Search for optimum α and t_{max} by minimizing C_t given in Eq. (3.13).

2 Calculate $\left\{\frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, \phi_0, \mathbb{C}, \mathbb{S}\right\}$ by minimizing I given in Eq. (3.10)

3 Calculate $\{l_2, x_0, y_0, \theta_1\}$ using Eq. (3.6) and Eq. (3.7) to synthesize a four-bar mechanism.

Output: Four-bar design parameters

3.4 Example

This example demonstrates the improvement made using the proposed methodology. A 12-point trajectory is taken as the input and is given in Table 3.1. In this example, we use $\omega_o = 2\pi$ rad/s (Eq. 3.2) and $\beta = 2$ (Eq. 3.15).

First, we use uniform parametrization to create a task curve. The task curve is calculated to have $t_{max} = 0.8770$ and FDs as given in Table 3.2. The

Table 3.1: Input point data

No.	Coordinate (x, y)	No.	Coordinate (x, y)	No.	Coordinate (x, y)
1	$0.000, -1.000$	5	$-2.866, -2.118$	9	$0.246, -2.135$
2	$-0.550, -0.942$	6	$-2.608, -2.488$	10	$0.876, -1.615$
3	$-1.696, -1.018$	7	$-2.098, -2.720$	11	$0.986, -1.329$
4	$-2.821, -1.715$	8	$-0.546, -2.551$	12	$0.593, -1.123$

Table 3.2: Task curve Fourier Descriptors

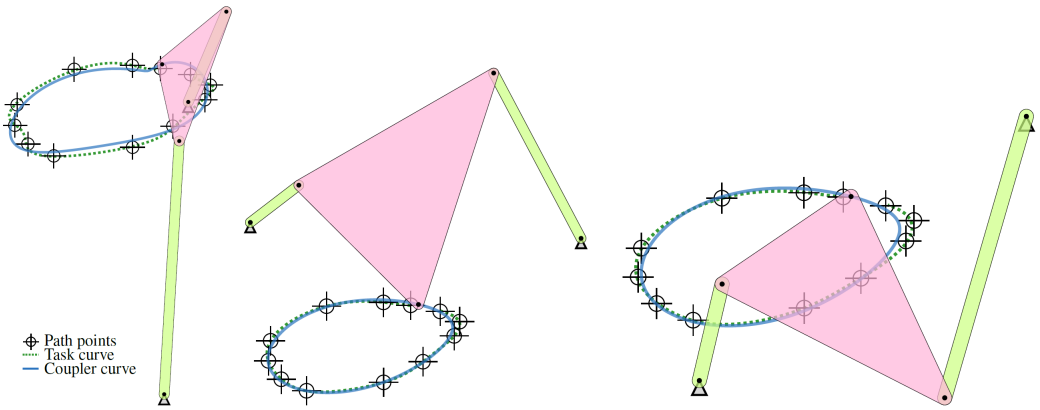
	Parametrization		
Descriptor	Uniform	Optimal	Optimal with speed criteria
-5	$0.035 - 0.051i$	$-0.001 + 0.013i$	$-0.003 + 0.017i$
-4	$-0.021 + 0.033i$	$-0.034 - 0.013i$	$-0.011 - 0.007i$
-3	$0.008 + 0.037i$	$-0.022 - 0.014i$	$-0.021 - 0.016i$
-2	$-0.015 + 0.086i$	$-0.009 - 0.018i$	$-0.003 - 0.036i$
-1	$0.420 - 0.455i$	$0.412 - 0.290i$	$0.378 - 0.261i$
0	$-0.822 - 1.696i$	$-0.960 - 1.763i$	$-0.970 - 1.764i$
1	$0.352 + 1.334i$	$0.715 + 1.167i$	$0.719 + 1.148i$
2	$0.074 - 0.204i$	$-0.005 - 0.072i$	$-0.015 - 0.065i$
3	$0.054 - 0.051i$	$-0.053 - 0.008i$	$-0.072 + 0.013i$
4	$-0.051 - 0.077i$	$-0.045 + 0.002i$	$-0.021 - 0.002i$
5	$-0.019 + 0.042i$	$-0.002 - 0.010i$	$0.003 - 0.032i$

Table 3.3: Coupler curve Fourier Descriptors

	Parametrization		
Descriptor	Uniform	Optimal	Optimal with speed criteria
-5	$0.011 + 0.006i$	$-0.001 + 0.000i$	$-0.001 + 0.000i$
-4	$0.008 + 0.019i$	$0.000 + 0.001i$	$-0.001 + 0.000i$
-3	$0.049 - 0.019i$	$-0.006 - 0.011i$	$-0.007 - 0.014i$
-2	$0.005 + 0.104i$	$-0.005 - 0.020i$	$-0.001 - 0.031i$
-1	$0.430 - 0.453i$	$0.412 - 0.291i$	$0.376 - 0.262i$
0	$-0.822 - 1.696i$	$-0.960 - 1.763i$	$-0.970 - 1.764i$
1	$0.352 + 1.334i$	$0.715 + 1.167i$	$0.719 + 1.148i$
2	$0.055 - 0.170i$	$0.004 - 0.074i$	$-0.006 - 0.077i$
3	$0.000 - 0.034i$	$-0.006 + 0.006i$	$-0.004 + 0.007i$
4	$-0.006 - 0.025i$	$0.003 - 0.000i$	$0.003 - 0.001i$
5	$-0.005 - 0.012i$	$-0.001 - 0.000i$	-0.001 - 0.000i

Table 3.4: Synthesized mechanism parameters

	Parametrization		
Variable	Uniform	Optimal	Optimal with speed criteria
l_1	10.020	9.271	10.140
l_2	1.921	2.085	1.464
l_3	6.719	6.520	3.994
l_4	5.423	5.160	8.139
x_0	-6.464	-5.560	11.951
y_0	1.659	0.931	-2.386
θ_1	-0.346	-0.191	0.859
r	12.191	11.519	10.194
α	0.119	0.001	0.661
ϕ_0	0.774	0.764	3.218



(a) Uniform parametrization (b) Optimal parametrization (c) Optimal parametrization with speed criteria

Figure 3.3: Synthesized solutions using different parametrizations

task curve fitting error, as defined in Eq (3.2), is found to be $\Delta = 0.3509$. From this task curve, a mechanism is synthesized to find the four-bar design parameters. Coupler curve FDs and computed mechanism parameters are given in Table 3.3 and Table 3.4, respectively. Coupler curve fitting error as defined in Eq. (3.10), is calculated to be $I = 0.0228$. The synthesized mechanism can be viewed in Fig. 3.3a. Next, mechanism synthesis is accomplished using optimal parametrization. The task curve is calculated to have $t_{max} = 0.9336$, $\alpha = 0.5823$ and FDs as given in Table 3.2. The task curve fitting error is observed to be $\Delta = 0.0591$, which is better than the previous case. Subsequently, a mechanism is synthesized and the coupler curve FDs and the solution mechanism parameters are given in Table 3.3 and Table 3.4, respectively. The coupler curve fitting error is found to be $I = 0.0067$, which is also less than the uniform parametrization case. The synthesized mechanism can be viewed in Fig. 3.3b. While using non-uniform parametrization enables the reduction of fitting error Δ, this example demonstrates that a task curve with low magnitude higher order harmonics decreases I and leads to a better task-coupler curve matching. Fig. 3.4 displays the comparison of weighted FDs, given as $T_{w,k} = (|k| + 1)^2 \|T_k\|$, for uniform and optimal parametrization. These figures shows that for optimal parametrization case, the magnitude of the higher order harmonics is less compared to the uniform case for both the task and the coupler curves.

Finally, mechanism synthesis involving speed criteria is carried out. Speed ratio for the task curve calculated using uniform parametrization is observed

to be 8.22. Let us assume that the user desires to constrain S_r, such that $1 \leq S_r \leq 2$. After applying the speed criteria, the task curve is calculated to have $t_{max} = 0.9234$, $\alpha = 0.7885$ and FDs as given in Table 3.2. The task curve fitting error is observed to be $\Delta = 0.1813$. The S_r of the generated task curve is 2. Comparison of task curve speeds has been done in Fig 3.5. It can be observed that the new task curve has reduced the speed ratio. Synthesized mechanism design parameters are given in Table 3.4. The coupler curve fitting error is observed to be $I = 0.0444$ and the solution is displayed in Fig. 3.3c.

3.5 Conclusion

In this chapter, a non-uniform parametrization scheme has been proposed for the task curve calculation from a given sequence of path points. A novel methodology to find the optimal parametrization based on fitting accuracy, the harmonic properties of four-bar coupler path, and user imposed speed criteria has been demonstrated. Synthesis of a more accurate four-bar mechanism for path generation has been shown using an example. The proposed approach improves upon the existing FD based path generation algorithm for mechanism synthesis.

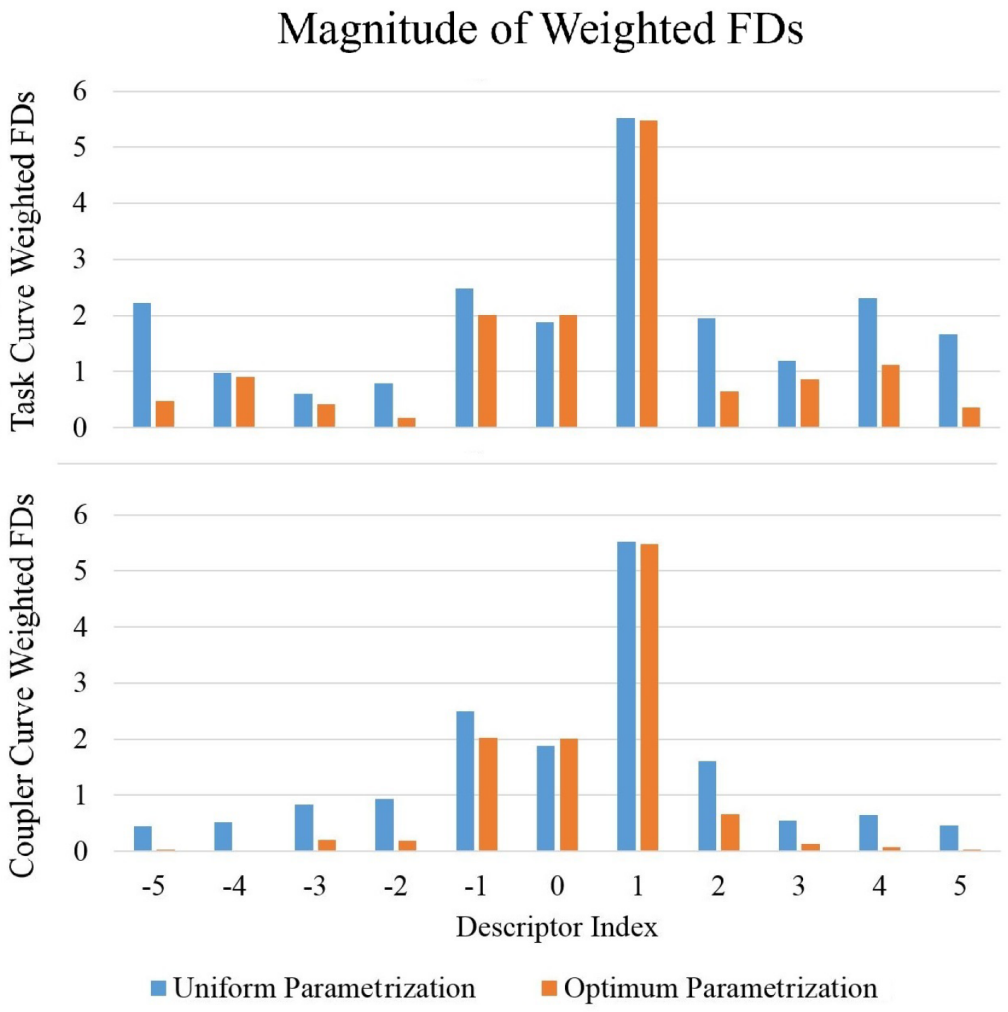


Figure 3.4: Comparison of task curve and coupler curve weighted FDs for uniform and optimal parametrization

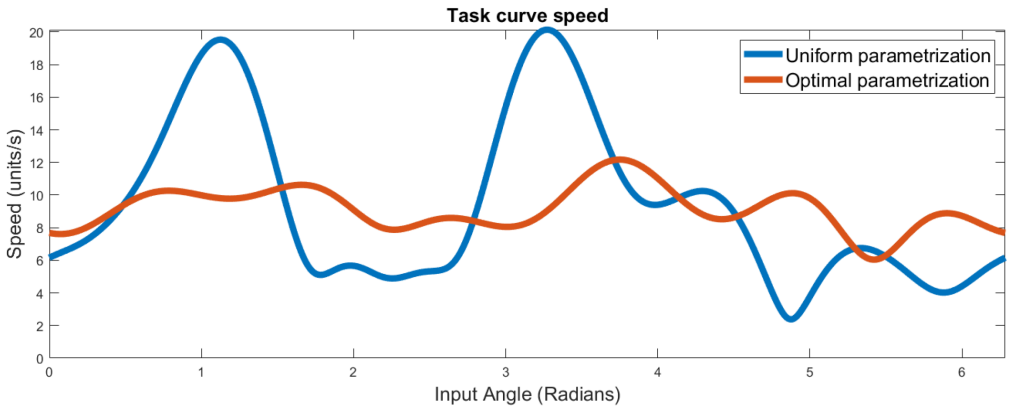


Figure 3.5: Comparison of task curve speeds obtained with and without speed criteria

Chapter 4

Motion Synthesis

4.1 Introduction

A spherical four-bar mechanism is constructed by rigidly joining the floating link of two spherical dyads [36]. These dyads can be RR, RP, PR or PP depending on their constituent joints being revolute or prismatic in nature [37]. Each link of a spherical linkage is constrained to rotate about the same fixed point in space [27]. In this paper, we focus on the motion generation problem which can be described as designing a four-bar spherical mechanism whose floating link passes through a sequence of prescribed poses [38].

Motion synthesis of spherical mechanism for a discrete number of poses has been well studied. Bodduluri and McCarthy [39] carry out finite position synthesis by minimizing the normal distance in the image space. Lin [40] use homotopy methods to generate spherical four-bar mechanisms for motion and path generation. Ge and Larochelle [41] use NURBS to synthesize spherical mechanisms. Ruth and McCarthy [42] describe SphinxPC, a computer-aided design software system for spherical four-bar linkage synthesis. Brunnthaler et al. [43] use kinematic mapping to synthesize spherical four-bars. Zhuang et al. [44] have used an adaptive genetic algorithm to synthesize spherical four-bars. Li et al. [45] solve a more general n-discrete pose problem using kinematic mapping. However, these studies have restricted themselves to spherical RRRR four-bar mechanisms and not considered spherical prismatic joints. Also, the design scenarios where a constraint on pivot locations exist have not been studied.

In this paper, we present a unified type and dimensional synthesis ap-

proach for motion generation which incorporates fixed and moving pivot constraints. Also, the designer has the flexibility to specify the nature of each position to be exact or approximate. The algorithm uses kinematic mapping [46] to create generalized manifold expressions in the image space [10, 47, 48]. A three-step process is adopted to apply the exact, approximate and bilinear synthesis constraints to enable efficient computation. We show that for each manifold solution, four RR, two RP, two PR, and one PP solution dyad exists. Thus, the proposed framework takes into consideration the importance of each position while giving the designer greater control over pivot locations. The algorithm's ability to synthesize multiple types of mechanisms in real-time makes it an extremely practical tool.

The major intellectual contributions of this paper can be summarized as 1) proposing a real-time motion generation algorithm that handles a set of i-approximate, j-exact pose constraints; 2) incorporates location of fixed or moving pivots as additional constraints available to designer and 3) accomplishes type synthesis and generate spherical RR, RP, PR and PP dyads.

The remaining paper is organized as follows. Section 4.2 discusses the representation of spherical positions in image space. Section 4.3 demonstrates the calculation of manifold equations for RR, RP, PR, and PP spherical dyads. Section 4.4 derives the algebraic equation representing each constraint a designer can enforce. Finally, Section 4.5 presents the algorithm to solve the motion synthesis problem and Section 4.6 uses the algorithm to solve sample test-cases.

4.2 Kinematic Mapping of Spherical Displacement

A spherical pose can be defined using the Euler-Rodrigues parameters which involves rotation axis and angle. The axis is represented by a unit vector $s = (s_x, s_y, s_z)$ and rotation angle θ. The displacement can be transformed to a unit quaternion $q = (q_1, q_2, q_3, q_4)$ as

$$q_1 = s_x sin(\theta/2), \quad q_2 = s_y sin(\theta/2), \tag{4.1}$$

$$q_3 = s_z sin(\theta/2), \quad q_4 = cos(\theta/2) \tag{4.2}$$

The components (s_x, s_y, s_z) represent a point in a projective three-space called image space of spherical displacements. Thus, a spherical displacement

can be defined as a homogeneous transformation of point $a = (a_1, a_2, a_3, a_4)$ or plane $l = (l_1, l_2, l_3, l_4)$ from moving frame M to fixed frame F as follows

$$A = [R]a, \quad L = [\bar{R}]l \tag{4.3}$$

$$R = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_4q_3) & 2(q_1q_3 + q_4q_2) & 0 \\ 2(q_1q_2 + q_4q_3) & q_4^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_4q_1) & 0 \\ 2(q_1q_3 - q_4q_2) & 2(q_2q_3 + q_4q_1) & q_4^2 - q_1^2 - q_2^2 + q_3^2 & 0 \\ 0 & 0 & 0 & q_4^2 + q_1^2 + q_2^2 + q_3^2 \end{bmatrix} \tag{4.4}$$

$$[\bar{R}] = [R] \tag{4.5}$$

where $q_4^2 + q_1^2 + q_2^2 + q_3^2 = 1$, $A = (A_1, A_2, A_3, A_4)$ and $L = (L_1, L_2, L_3, L_4)$ represent a point and plane in F respectively. Note that for spherical displacements, $[\bar{R}] = [R]^{-T}$ is same as $[R]$ due to which plane and point coordinates transformations are exactly same.

4.3 Constructing the Generalized (G-) Constraint manifold

In this section, the kinematic constraints of four types of spherical dyads i.e. RR, RP, PR, PP are explored. A point A or plane L on the coupler of four-bar spherical mechanism can be geometrically constrained depending on the dyad type. The types of spherical binary links are shown in Fig 4.1.

For a spherical RR dyad, its coupler point $B = (B_1, B_2, B_3, B_4)$ is constrained on a sphere whose radius and center are given by homogeneous coordinates $A = (A_0, A_1, A_2, A_3, A_4)$. This constraint can be given as

$$C_{S,RR} : 2A_1B_1 + 2A_2B_2 + 2A_3B_3 + A_0B_4 = A_4\left(\frac{B_1^2 + B_2^2 + B_3^2}{B_4}\right), \tag{4.6}$$

$$A_4^2r^2 - A_0A_4 = A_1^2 + A_2^2 + A_3^2, \tag{4.7}$$

where r is the radius of sphere formed by RR dyad.

For a spherical PR dyad, a fixed plane $L = (L_1, L_2, L_3, L_4)$ and a point on coupler $A = (A_1, A_2, A_3, A_4)$ are constrained to stay at a fixed perpendicular distance d. Since the plane passes through the origin, $L_4 = 0$ and the geometric constraint is given as

$$C_{S,PR} : A_1L_1 + A_2L_2 + A_3L_3 = dA_4\sqrt{L_1^2 + L_2^2 + L_3^2}. \tag{4.8}$$

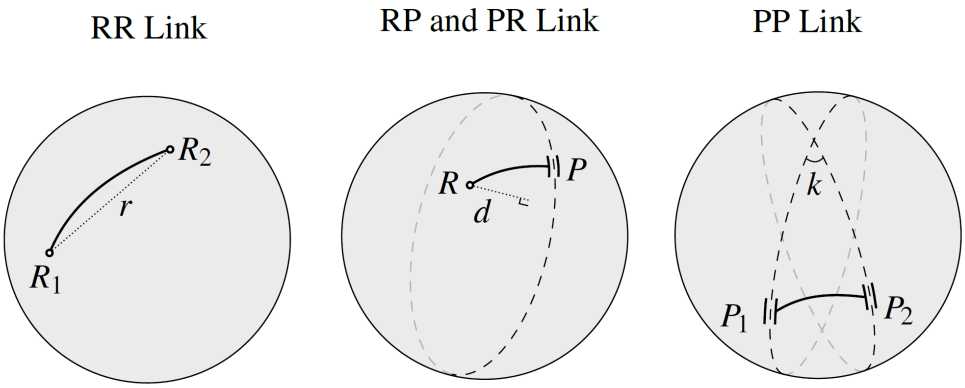


Figure 4.1: Types of Spherical binary links

For a spherical RP dyad, a fixed point $A = (A_1, A_2, A_3, A_4)$ and a plane on coupler $L = (L_1, L_2, L_3, L_4)$ are constrained to stay at a fixed perpendicular distance d. Again, the plane L passes through the origin and the geometric constraint is given by the same equation

$$C_{S,RP} : A_1L_1 + A_2L_2 + A_3L_3 = dA_4\sqrt{L_1^2 + L_2^2 + L_3^2}. \tag{4.9}$$

Finally, for a spherical RP dyad, a fixed plane $L = (L_1, L_2, L_3, L_4)$ and a plane on coupler $M = (M_1, M_2, M_3, M_4)$ are constrained to stay at a fixed angle whose cosine is given as k and the constraint is given as

$$C_{S,PP} : L_1M_1 + L_2M_2 + L_3M_3 = k\sqrt{L_1^2 + L_2^2 + L_3^2}\sqrt{M_1^2 + M_2^2 + M_3^2}. \tag{4.10}$$

To find the G-manifold equation, the fixed frame coupler point or line coordinates in geometric constraint equations are substituted with moving coordinates according to transformation relationships in Eq (4.3). For an RR dyad, the constraint equation becomes

$$\begin{aligned} &P_0(q_4^2 + q_1^2 + q_2^2 + q_3^2) + P_1(q_4^2 + q_1^2 - q_2^2 - q_3^2) \\ &+P_2(2q_1q_2 - 2q_4q_3) + P_3(2q_1q_3 + 2q_4q_2) \\ &+P_4(2q_1q_2 + 2q_4q_3) + P_5(q_4^2 - q_1^2 + q_2^2 - q_3^2) \\ &+P_6(2q_2q_3 - 2q_4q_1) + P_7(2q_1q_3 - 2q_4q_2) \\ &+P_8(2q_2q_3 + 2q_4q_1) + P_9(q_4^2 - q_1^2 - q_2^2 + q_3^2) = 0 \end{aligned} \tag{4.11}$$

where the coefficients $(P_0, P_1, ...P_9)$ are given as

$$\begin{aligned} P_0 &= \frac{A_0 b_4 - A_4(b_1^2 + b_2^2 + b_3^2)}{2b_4}, \\ P_1 = A_1 b_1, \quad P_2 &= A_1 b_2, \quad P_3 = A_1 b_3, \\ P_4 = A_2 b_1, \quad P_5 &= A_2 b_2, \quad P_6 = A_2 b_3, \\ P_7 = A_3 b_1, \quad P_8 &= A_3 b_2, \quad P_9 = A_3 b_3 \end{aligned} \tag{4.12}$$

Here $A = (A_1, A_2, A_3, A_4)$ are the fixed point coordinates in F while $b = (b_1, b_2, b_3, b_4)$ are the moving point coordinates in M.

In a similar manner, the constraint equation for a PR dyad can be calculated. PR dyad is defined by fixed plane coordinates $P = (P_1, P_2, P_3, 0)$ in F and moving point coordinates $a = (a_1, a_2, a_3, a_4)$ in M. The quadric is of the same form as Eq (4.19) with the coefficients as

$$\begin{aligned} P_0 &= -da_4\sqrt{L_1^2 + L_2^2 + L_3^2}, \\ P_1 = L_1 a_1, \quad P_2 &= L_1 a_2, \quad L_3 = L_1 a_3, \\ P_4 = L_2 a_1, \quad P_5 &= L_2 a_2, \quad L_6 = L_2 a_3, \\ P_7 = L_3 a_1, \quad P_8 &= L_3 a_2, \quad L_9 = L_3 a_3 \end{aligned} \tag{4.13}$$

A RP dyad is defined by fixed point coordinates $A = (A_1, A_2, A_3, A_4)$ in F and moving plane coordinates $l = (l_1, l_2, l_3, 0)$ in M. The quadric is of the same form as Eq (4.19) with the coefficients as

$$\begin{aligned} P_0 &= -dA_4\sqrt{l_1^2 + l_2^2 + l_3^2}, \\ P_1 = A_1 l_1, \quad P_2 &= A_1 l_2, \quad P_3 = A_1 l_3, \\ P_4 = A_2 l_1, \quad P_5 &= A_2 l_2, \quad P_6 = A_2 l_3, \\ P_7 = A_3 l_1, \quad P_8 &= A_3 l_2, \quad P_9 = A_3 l_3 \end{aligned} \tag{4.14}$$

A PP dyad represented by fixed plane coordinate $L = (L_1, L_2, L_3, 0)$ in F and a moving plane coordinates $m = (m_1, m_2, m_3, 0)$ in M, constraint equation is of the same form as Eq (4.19) with coefficients

$$\begin{aligned} P_0 &= -k\sqrt{L_1^2 + L_2^2 + L_3^2}\sqrt{m_1^2 + m_2^2 + m_3^2}, \\ P_1 = L_1 m_1, \quad P_2 &= L_1 m_2, \quad P_3 = L_1 m_3, \\ P_4 = L_2 m_1, \quad P_5 &= L_2 m_2, \quad P_6 = L_2 m_3, \\ P_7 = L_3 m_1, \quad P_8 &= L_3 m_2, \quad P_9 = L_3 m_3 \end{aligned} \tag{4.15}$$

As can be observed, $(P_1 ... P_9)$ are have the same expression consisting of fixed and moving point or plane coordinates. This means that for spherical mechanisms, a G-manifold can simultaneously represent an RR, RP, PR or RR dyad. Thus, each solution manifold can represent any of the dyads.

From the above relationships, it can be observed that the 10 homogeneous coefficients P_i are not independent. They satisfy a bilinear constraint as follows

$$\frac{P_4}{P_1} = \frac{P_5}{P_2} = \frac{P_6}{P_3} = \frac{A_2}{A_1} = \lambda_1 \tag{4.16}$$

$$\frac{P_7}{P_1} = \frac{P_8}{P_2} = \frac{P_9}{P_3} = \frac{A_3}{A_1} = \lambda_2 \tag{4.17}$$

If $p = (p_1, p_2, p_3)$, then the bilinear constraints can be compactly represented as $\lambda_1 p = (p_4, p_5, p_6)$ and $\lambda_2 p = (p_7, p_8, p_9)$.

4.4 Constraints

A general motion synthesis problem can involve pivot location constraints in addition to pose constraints. These pivot constraints can be represented as a point or plane constraint. The poses or pivot constraints can be satisfied in an exact or approximate manner, depending on their importance to the mechanism designer. The algebraic form of each of these constraints is discussed below.

4.4.1 Pose Constraint

A pose constraint can be represented algebraically as

$$\begin{bmatrix} A_0 & A_1 & \cdots & A_9 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_9 \end{bmatrix} = 0 \tag{4.18}$$

where P_j are the homogeneous dyad coordinates and A_j are the homogenous coefficients given by

$$\begin{aligned} A_0 &= (q_4^2 + q_1^2 + q_2^2 + q_3^2) \quad A_1 = (q_4^2 + q_1^2 - q_2^2 - q_3^2) \\ A_2 &= (2q_1q_2 - 2q_4q_3) \quad A_3 = (2q_1q_3 + 2q_4q_2) \\ A_4 &= (2q_1q_2 + 2q_4q_3) \quad A_5 = (q_4^2 - q_1^2 + q_2^2 - q_3^2) \\ A_6 &= (2q_2q_3 - 2q_4q_1) \quad A_7 = (2q_1q_3 - 2q_4q_2) \\ A_8 &= (2q_2q_3 + 2q_4q_1) \quad A_9 = (q_4^2 - q_1^2 - q_2^2 + q_3^2) \end{aligned} \tag{4.19}$$

where (q_1, q_2, q_34, q_4) are the quaternion representation.

4.4.2 Point Pivot constraint

The fixed or moving pivot of a spherical mechanism can be constrained to a point if its position is specified by the designer. For a fixed pivot represented as $(F_1, F_2, F_3, 1)$, the constraint equations involving homogenous dyad coordinates P_i are

$$P_1 - F_1 = 0 \tag{4.20}$$
$$P_4 - F_2 = 0 \tag{4.21}$$
$$P_7 - F_3 = 0 \tag{4.22}$$

Similarly for a moving pivot specified by $(m_1, m_2, m_3, 1)$ in the moving frame, the algebraic constraint involving homogenous dyad coordinates P_i can be given as

$$P_1 - m_1 = 0 \tag{4.23}$$
$$P_2 - m_2 = 0 \tag{4.24}$$
$$P_3 - m_3 = 0 \tag{4.25}$$

4.4.3 Plane Pivot constraint

A design problem may require the fixed or moving pivot of the synthesized dyad to be constrained to a plane specified by the designer. To constrain a fixed pivot to a plane $(L_1', L_2', L_3', 0)$, a fixed pivot will satisfy

$$L_1'P_1 + L_2'P_4 + L_3'P_7 = 0 \tag{4.26}$$

Similarly, to constrain the moving pivot to a plane, the algebraic constraint involving homogenous dyad coordinates P_i will be

$$L_1' P_1 + L_2' P_2 + L_3' P_3 = 0 \tag{4.27}$$

4.5 Motion Synthesis Algorithm

Once the user has specified all the pose and pivot constraints and their nature(approximate or exact), the motion synthesis algorithm can be invoked. It is carried out in three steps. First, the approximate constraints are applied. To the resultant solution space, the exact constraints are applied. Finally, the bilinear constraints are imposed on the solution space to calculate the synthesized dyads.

The procedure to handle a i-approximate j-exact constraint motion synthesis is explained in detail below. It is well known that a unique solution exists for five pose problem. Thus, the number of exact constraints is restricted to $j \leq 5$ since a solution doesn't exist otherwise.

4.5.1 Applying Approximate constraints

For a set of i-approximate constraints including both pose and pivot constraints, a set of i linear equations can be obtained as follows

$$\begin{bmatrix} A_{1,0} & A_{1,1} & \cdots & A_{1,9} \\ A_{2,0} & A_{2,1} & \cdots & A_{2,9} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i,0} & A_{i,1} & \cdots & A_{i,9} \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_9 \end{bmatrix} = [A_1]P = 0 \tag{4.28}$$

where $[A_1]$ is the coefficient matrix computed using the approximate constraint specified by designer. This system of equation is sovled using Singular Value Decomposition(SVD) and the approximate solution space is given by right singular vectors $(u_1, u_2, \cdots, u_{j+5})$. Thus, the solution homogeneous dyad vector P is a linear combination of singular vectors and can be represented as

$$P = \sum_{i=1}^{j+5} \alpha_i u_i = \begin{bmatrix} u_1 & u_2 & \cdots & u_{j+5} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{j+5} \end{bmatrix} = [U]\alpha. \tag{4.29}$$

If no approximate constraints are specified by mechanism designer, the solution space $[U]$ can be represented as a 10×10 identity matrix i.e. no restriction is imposed on the solution space.

4.5.2 Applying Exact Constraints

Once the approximate constraints are applied, the next step is to impose the exact constraints. The j-exact constraints can be represented as a system of equations given as

$$\begin{bmatrix} A_{1,0} & A_{1,1} & \cdots & A_{1,9} \\ A_{2,0} & A_{2,1} & \cdots & A_{2,9} \\ \vdots & \vdots & \ddots & \vdots \\ A_{j,0} & A_{j,1} & \cdots & A_{j,9} \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_9 \end{bmatrix} = [A_2]P = [A_2][U]\alpha = 0 \tag{4.30}$$

where $[A_2]$ is the coefficient matrix is determined by the exact pose and pivot locations specified by the designer. The solution can computed by performing SVD of $[A_2][U]$ and determining the five-dimensional null-space corresponding to singular values of negligible magnitude, represented by five right singular vectors $(v_1, v_2, \cdots, v_5)$. Thus, the solution vector α can be represented as linear combination of singular vectors given by

$$\alpha = \sum_{i=1}^{5} \beta_i v_i = \begin{bmatrix} v_1 & v_2 & \cdots & v_5 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_5 \end{bmatrix} = [V]\beta. \tag{4.31}$$

As a result, the solution space constrained to satisfy both approximate and exact constraints can be represented as

$$P = [U][V]\beta \tag{4.32}$$

If the design problem does not involve any exact constraints, the solution space $[V]$ can be represented as a 5×5 identity matrix i.e. no additional restriction is imposed on the solution space.

The resultant solution-space $[U][V]$ is five-dimensional because there exist four bilinear constraint conditions which need to be imposed to generate unique valid homogenous dyad vectors.

4.5.3 Applying Bilinear Constraints

The solution space is linear combination of five null-space basis vectors v_i given as

$$P = \sum_{i=1}^{5} \alpha_i v_i = \begin{bmatrix} v_1 & \cdots & v_5 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_5 \end{bmatrix} = \mathbf{V}\alpha \tag{4.33}$$

However, since $p_1, \cdots, p_9$ are not independent, additional bilinear constraints given in eq (4.16) need to be applied. Accordingly, this system of equation can be broken into three smaller systems as follows

$$p^T = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} v_{1,1} & \cdots & v_{5,1} \\ v_{1,2} & \cdots & v_{5,2} \\ v_{1,3} & \cdots & v_{5,3} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_5 \end{bmatrix} = W_1\alpha \tag{4.34}$$

$$\lambda_1 p^T = \begin{bmatrix} p_4 \\ p_5 \\ p_6 \end{bmatrix} = \begin{bmatrix} v_{1,4} & \cdots & v_{5,4} \\ v_{1,5} & \cdots & v_{5,5} \\ v_{1,6} & \cdots & v_{5,6} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_5 \end{bmatrix} = W_2\alpha \tag{4.35}$$

$$\lambda_2 p^T = \begin{bmatrix} p_7 \\ p_8 \\ p_9 \end{bmatrix} = \begin{bmatrix} v_{1,7} & \cdots & v_{5,7} \\ v_{1,8} & \cdots & v_{5,8} \\ v_{1,9} & \cdots & v_{5,9} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_5 \end{bmatrix} = W_3\alpha \tag{4.36}$$

$$\tag{4.37}$$

These equations can be combined as

$$W_2\alpha = \lambda_1 W_1\alpha \tag{4.38}$$

$$W_3\alpha = \lambda_2 W_1\alpha \tag{4.39}$$

This generalized eigenvalue problem can be written as

$$\begin{bmatrix} W_2 - \lambda_1 W_1 \\ W_3 - \lambda_2 W_1 \end{bmatrix}_{6x5} \alpha_{5x1} = 0 \tag{4.40}$$

For this system of equations to be compatible and have a non-trivial solution, the determinant of two resulting 5×5 matrices need to be zero. Once the eigenvalues (λ_1, λ_2) are known, they can be substituted back in Eq. (4.40) to

find the eigenvector $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$. Finally, the solution manifolds can be calculated using Eq. (4.33).

Once the homogeneous dyad vectors are calculated, the fixed and moving pivot coordinates can be calculated using the following inverse relationships

$$F_1 : F_2 : F_3 = P_1 : P_4 : P_7 \tag{4.41}$$

$$m_1 : m_2 : m_3 = P_1 : P_2 : P_3 \tag{4.42}$$

If $F = (P_1, P_4, P_7)$ and $m = (P_1, P_2, P_3)$, there exist four solutions namely $(F, m), (-F, m), (F, -m), (-F, -m)$.

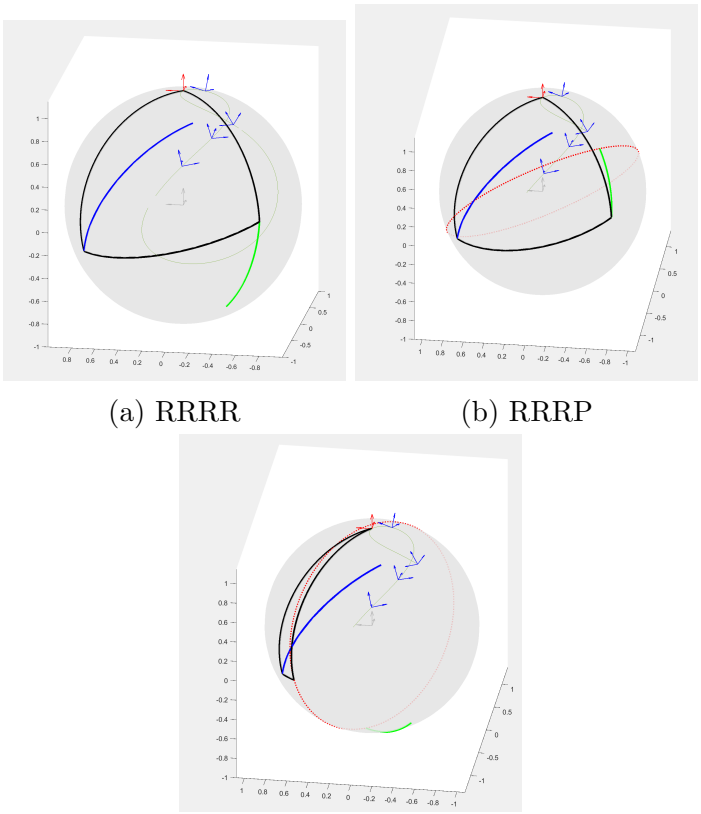
Also, F_4 and M_4 end up being the free coordinates. Thus, the coordinates $F = (F_1, F_2, F_3, F_4)$ and $m = (m_1, m_2, m_3, m_4)$ define both point or a plane. When $(F_4 = 1, m_4 = 1)$ the solution is RR dyad, $(F_4 = 0, m_4 = 1)$ results in a PR dyad, $(F_4 = 1, m_4 = 0)$ results in a RP dyad and $(F_4 = 0, m_4 = 0)$ results in a PP dyad. As a result, each solution manifold can be used to generate each type of dyad. No other values of homogenous coordinate are possible due to the unit circle restriction i.e. $F_1^2 + F_2^+ F_3^2 = 1$ and $m_1^2 + m_2^+ m_3^2 = 1$.

Since point coordinates P and $-P$ represent two different points while plane coordinates represent the same plane, there exist four RR, two RP, two PR and one PP solution spherical dyad for each manifold.

4.6 Examples

4.6.1 Synthesis involving Five-exact pose constraints

In this example, a four-bar spherical mechanism is synthesized which satisfies five-exact pose constraints. The input data has been taken from Yufeng et al and is displayed in the Table 4.1. Using our algorithm, we are able to generate all six possible dyads as demonstrated in Table 4.2. As demonstrated in section each solution dyad can be realized as an RR, RP, PR, PP dyad. A four-bar mechanism has been synthesized using the first and third generated dyads. Fig 4.2 shows three possible RRRR, RRRP, and RRPR solution mechanisms.



(a) RRRR (b) RRRP

(c) RRPR

Figure 4.2: Example of synthesized four-bar mechanisms for five-exact pose problem.

Table 4.1: Input Pose Data for example 1

No.	Rotation Axis	Rotation Angle	Type
1	$1, 1, 1$	0	Exact
2	$0.361087, -0.534155, 0.764391$	0.972627	Exact
3	$-0.387198, -0.302257, 0.871044$	-0.406367	Exact
4	$-0.456816, 0.490272, 0.742261$	-1.510039	Exact
5	$-0.512158, 0.281847, 0.811331$	-1.266222	Exact

Table 4.2: Synthesized four-bar dyads for example 1

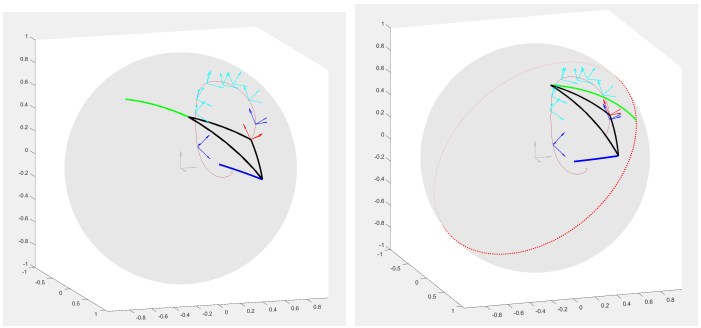
No.	Fixed Pivot	Moving Pivot
1	$-0.4807, -0.1572, 0.8627$	$-0.6078, 0.7496, -0.2621$
2	$0.5113, 0.0705, -0.8565$	$0.5913, -0.7837, 0.1905$
3	$-0.5380, -0.4477, -0.7142$	$-0.6566, -0.7504, 0.0762$
4	$-0.5151, 0.3890, -0.7638$	$-0.9855, -0.0013, -0.1694$
5	$-0.4376, -0.2138, -0.8734$	$-0.6930, -0.6461, -0.3197$
6	$0.3585, 0.1982, 0.9123$	$0.6228, 0.6417, 0.4476$

4.6.2 Synthesis involving Two-exact pose and Six-approximate pose constraints

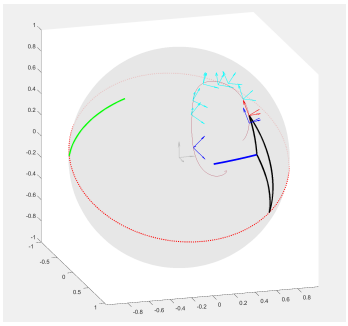
In this example, a mechanism designer wants to satisfy exactly first and last poses while he wants the intermediate poses to be approximately satisfied. The input data specified by the designer is displayed in the Table 4.3. Using our algorithm, we are able to generate four possible dyads as demonstrated in Table 4.4. Fig 4.2 shows three possible RRRR, RRRP and RRPR solution mechanisms generated using the third and fourth dyad.

4.6.3 Synthesis involving Two-exact pose, three-approximate pose, and exact fixed-pivot plane constraint

In this example, the design problem involves two-exact and three-approximate poses. Additionally, the designer wants to constrain the fixed pivots of the four-bar linkage onto a plane passing through origin whose normal vector is $(0, 0, 1)$. The pose constraints are displayed in Table 4.5. Using our algorithm, we are able to generate three valid dyads displayed in Table 4.6. Fig 4.4 shows one RRRR mechanisms generated using the second and third



(a) RRRR (b) RRRP



(c) RRPR

Figure 4.3: Example of synthesized four-bar mechanisms for two-exact, six-approximate pose problem.

Table 4.3: Input Pose Data for example 2

No.	Rotation Axis	Rotation Angle	Type
1	$0.2728, 0.4839, 0.8315$	2.2402	Exact
2	$0.1338, 0.4625, 0.8764$	1.9451	Approximate
3	$0.1168, 0.3688, 0.9221$	1.8389	Approximate
4	$0.1535, 0.3910, 0.9075$	1.5936	Approximate
5	$0.2134, 0.4104, 0.8866$	1.3136	Approximate
6	$0.3515, 0.5735, 0.7400$	1.1435	Approximate
7	$0.3894, 0.7045, 0.5933$	1.1202	Approximate
8	$0.4002, 0.8245, 0.4002$	1.3216	Exact

Table 4.4: Synthesized four-bar dyads for example 2

No.	Fixed Pivot	Moving Pivot
1	$-0.4807, -0.1572, 0.8627$	$-0.6078, 0.7496, -0.2621$
2	$0.5113, 0.0705, -0.8565$	$0.5913, -0.7837, 0.1905$
3	$-0.5380, -0.4477, -0.7142$	$-0.6566, -0.7504, 0.0762$
4	$-0.5151, 0.3890, -0.7638$	$-0.9855, -0.0013, -0.1694$

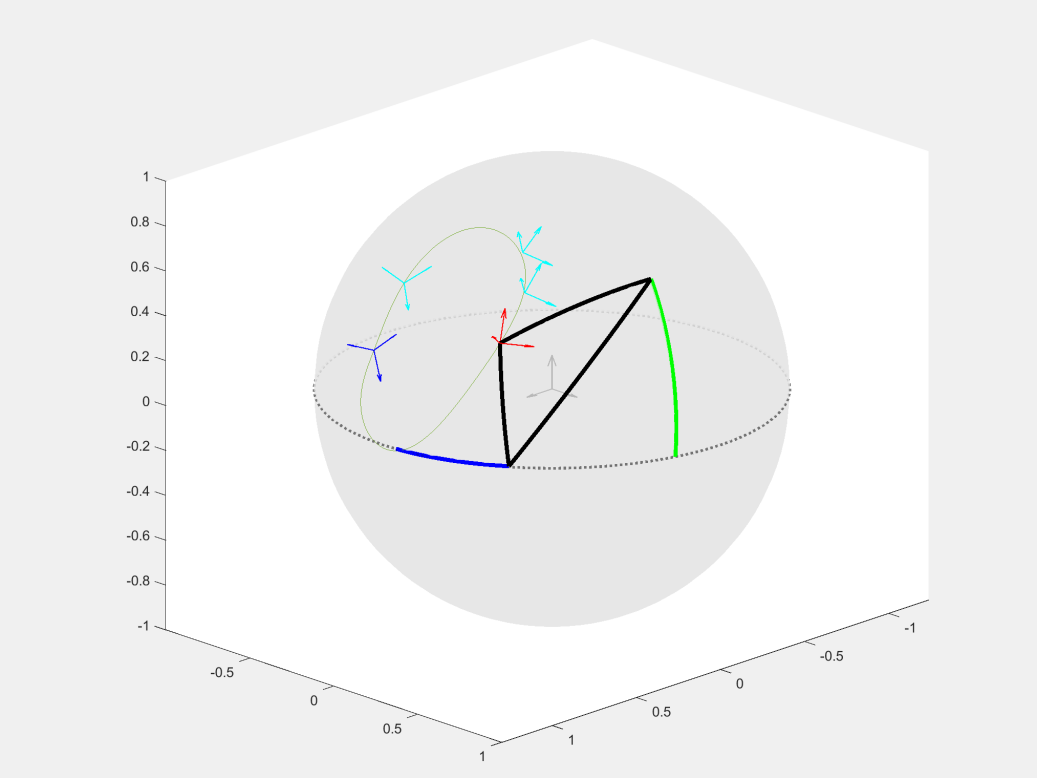
dyad.

Table 4.5: Input Pose Data for example 3

No.	Rotation Axis	Rotation Angle	Type
1	$0.2728, 0.4839, 0.8315$	2.2402	Exact
2	$0.1338, 0.4625, 0.8764$	1.9451	Approximate
3	$0.1168, 0.3688, 0.9221$	1.8389	Approximate
4	$0.3894, 0.7045, 0.5933$	1.1202	Approximate
5	$0.4002, 0.8245, 0.4002$	1.3216	Exact

Table 4.6: Synthesized four-bar dyads for example 2

No.	Fixed Pivot	Moving Pivot
1	$-0.9976, -0.0696, 0.0000$	$-0.8233, -0.5675, -0.0031$
2	$-0.2381, -0.9712, -0.0000$	$-0.1186, -0.7073, -0.6969$
3	$-0.0801, 0.9968, -0.0000$	$0.7144, -0.1874, 0.6742$



(a) RRRR

Figure 4.4: Example of synthesized four-bar mechanisms for two-exact, three-approximate pose problem with a plane constraint.

4.7 Conclusion

In this paper, we have presented a motion synthesis algorithm which synthesizes spherical four-bar mechanisms. The algorithm is able to handle pivot location constraints and enforce the relative importance of pose constraints to the design problem. The solutions mechanisms include both revolute and prismatic joints and thus gives the designer an additional choice.

Chapter 5

Simulation of Planar and Spherical Mechanism

5.1 Introduction

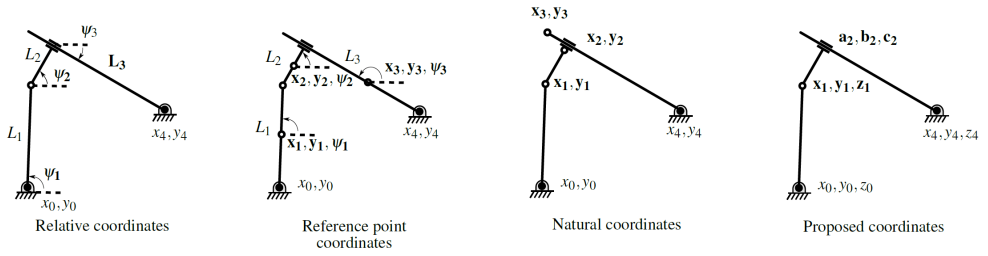


Figure 5.1: Different types of mechanism representations

Kinematic simulation of a mechanism requires calculation of the position and orientation of all of its constituent links during its entire range of motion. Simulation methodologies can be broadly classified into three categories: graphical, analytical and numerical [49]. The graphical analysis method is based on dyadic decomposition, i.e., identification of four-bar loops in mechanisms [50]. Although this approach is prominently used in simulation packages like Linkages [51] and PMKS [52], its limitations are well known [53]; e.g., they are unable to handle complex mechanisms like a double butterfly mechanism. Analytical methods involve solving a loop closure constraint-based system of non-linear equations [54]. Most analytical methods use the Polynomial continuation method [55, 56], elimination method or

Grobner bases [57] to solve the simulation problem. These methods are able to find all the possible assembly configurations of a given mechanism. However, they are not general in nature since the motion equations need to be derived for each type of mechanism separately. As a result, these approaches cannot be used to simulate n-bar planar or spherical mechanisms without manually deriving equations on a case by case basis.

Numerical simulation methods are iterative in nature and can handle extremely complex mechanism [58, 59]. They use numerical methods like the Newton-Rhapson method to solve the system of non-linear equations for one solution only instead of all possible ones [60]. These methods accept the mechanism joints and link information as inputs. Subsequently, the algorithm repeatedly solves the finite displacement problem, i.e., the input link is iteratively moved with finite displacement and consequently, positions of remaining links are calculated. As a result, the entire range of motion for the specified mechanism is calculated.

Hernández and Petuya have worked on a *geometrical-iterative method* which performs better than Newton-Rhapson method [61]. However, the approach is limited to n-bar planar mechanism with revolute joints only. Radhakrishnan and Campbell [62] have created a computational tool for planar mechanism, which carries out position analysis of planar mechanisms using a geometrically iterative algorithm. However, due to the use of dyadic decomposition, it shares the limitations of graphical methods and is limited to the planar mechanisms.

Commercial CAD software like Autodesk Inventor, Solidworks, ADAMS, etc. solve differential algebraic equations numerically to provide multi-body simulation capability [63–66]. However, their use is more prominent during detail design stage rather than the conceptual design stage. Creation of feature-based assembly of planar and spherical mechanisms and initializing constraints on these systems is a nontrivial task. Changing the type of joints or the number of links for a mechanism is also more involved than carrying out the same operation on a purely kinematic simulators like PMKS. Additionally, their solvers model the motion problem as a set of coupled differential and algebraic equations. This type of model is more suited for dynamic simulations rather than kinematic simulations which involves purely algebraic constraints. Also, the algebraic equations for commercial softwares are modeled using reference point representation which leads to more number of constraints when compared to other representations. Thus, use of these softwares for concept design is not ideal. SAM and GIM are two more pack-

ages which supports n-bar simulation for planar linkages with both revolute and prismatic joints [67, 68]. Furlong et al. [69] have demonstrated a virtual reality environment for simulating spherical four-bar mechanisms and in the academic domain, SPHINX, ISIS and OSIRIS are softwares which enable the analysis and synthesis of spherical mechanisms [70–73].

However, currently there are no approaches which unify n-bar planar and spherical mechanism analysis and can be demonstrated to more complex linkage systems. The proposed algorithm hopes to bridge this gap and augment the capability of pure kinematic design systems like MotionGen [7].

Selection of an apt mechanism representation and constraints is important as it has a profound effect on algorithm's simplicity and efficiency. Conventionally, a multi-body system has been specified using multiple representations namely: relative coordinates, reference point coordinates, and natural coordinates [58]. Relative coordinates are based on parameters specifying one link relative to another; reference point coordinates are based on specifying absolute position of each link independently; while the natural coordinates are based on each link being specified by two points. Using relative coordinates enables a scalable representation while reference point coordinates tend to be more computationally efficient. Natural coordinates provide a compromise between the two approaches in terms of simplicity and efficiency. Most commercial softwares use reference point coordinate representation which usually leads to maximum number of constraint equations and subsequently high computation time.

Figure 5.1 shows an RRPR (R: Revolute, P: Prismatic) four-bar mechanism and its specification using different representations. For the relative coordinate representation, there are three unknown coordinates i.e. ψ_1, ψ_2, L_3. For the reference point coordinate representation, the mechanism has nine unknown variables i.e. location and orientation of each link x_i, y_i, ψ_i. Similarly, for the natural coordinate representation, there are six unknown variables namely $x_1, y_1, x_2, y_2, x_3, y_3$. Since the four-bar mechanisms are a single degree of freedom mechanisms, each of the representation requires two, eight and five constraint equations, respectively to fully define the motion. In this chapter, we will seek to derive unified constraint equations for all types of planar and spherical linkages consisting of both revolute and prismatic joints. We use homogeneous coordinates to write geometric constraints on points, lines, and planes. For example, our representation for the shown RRPR mechanism will require using six unknown point and line coordinates i.e. $x_1, y_1, z_1, a_2, b_2, c_2$. Such a representation would keep the number of un-

known variables smaller while also enabling construction of simple geometric constraint equations.

In this chapter, we propose a novel simulation approach where planar-and spherical-mechanisms are represented as a collection of geometric constraints spanned by points, lines, and planes. The approach can handle any complex mechanism involving both revolute and prismatic joints. It is scalable in nature and can be used to analyze both spatial and spherical mechanism. The geometric mechanism representation enables the design of unified constraint equations which are easily programmed. As a result, a simple generalized real-time framework for mechanism simulation is achieved.

Our previous work on mechanism synthesis problems includes the creation of geometric constraint equations for four-bar mechanisms with revolute or prismatic joints [10,45,48,74,75]. In [76], we have demonstrated an algebraic fitting based approach in the space of planar quaternions to simulate planar four-bar linkages. However, that approach does not scale for more complex planar or spherical linkages. In this chapter, we show that by using homogenous coordinates, we can derive unified geometric constraint equations for both planar and spherical linkages, which simplifies the simulation without resorting to calculations for individual types of mechanisms. The rigidity constraints imposed by the links are modeled as simple geometric constraints using points, lines and planes. Once the mechanism is specified, the solver proceeds with iteratively perturbing the input and solving the constraints for other links. To the best of authors' knowledge, this work is the first attempt at using a point-line-plane mechanism representation and presenting unified geometric constraints for simulation.

The major intellectual contributions of this chapter can be summarized as 1) presenting generalized constraint equations for planar and spherical mechanisms using point-line-plane representation and 2) enabling real-time simulation of n-bar planar or spherical mechanisms.

Rest of the chapter is organized as follows. Section 5.2 discusses the representation and constraints required to describe the motion of a general planar and spherical mechanism. Section 5.3 demonstrates the algorithm required to simulate a mechanism using the iterative numerical approach. Finally, in Section 5.4, we present a few examples to demonstrate the use of proposed algorithm.

5.2 Mechanism representation and constraints

Planar mechanisms can be uniquely specified using their joint and link data. A joint can be prismatic or revolute, which naturally associates with points and lines, respectively. We use homogeneous coordinates to represent both points and lines. Thus, a point P is given by homogeneous coordinates (x, y, z) whose affine coordinates are given as $(\frac{x}{z}, \frac{y}{z})$, while a line L is also represented using homogeneous coordinates (a, b, c), where equation of the line passing through the point P in the projective plane is given by $ax + by + cz = 0$. Depending on the constraint being expressed, this line can be fixed or floating in the plane. A link can be represented by a subset of joints. The link can be binary, ternary or n-ary depending on the number of joints it contains. An example six-bar planar mechanism is displayed in Fig. 5.2. Its joints are represented as points and lines while its links are defined as a group of joints as shown in Table 5.1.

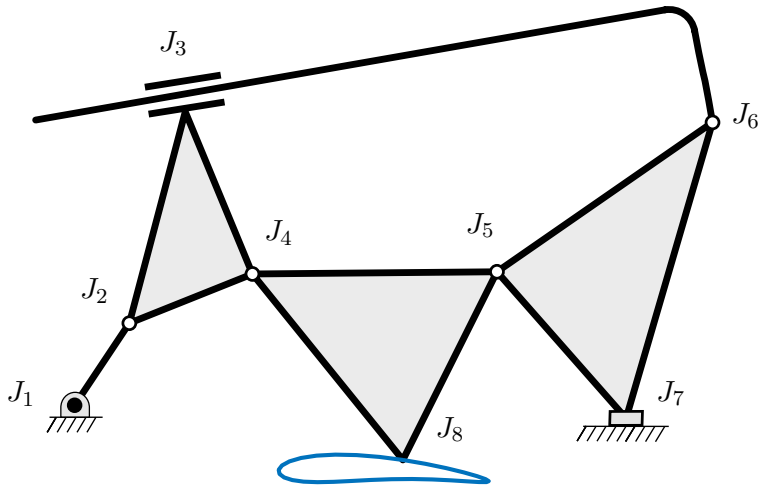


Figure 5.2: Planar Stephenson II six-bar linkage

Similarly, spherical mechanisms can also consist of revolute and prismatic joints. A spherical prismatic joint constrains the link movement along a circular arc instead of a line. We represent a spherical revolute joint as a point P in terms of its homogeneous coordinates (x, y, z, w) with respect to the center of the unit sphere such that its Affine coordinates are $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$. A spherical prismatic joint is defined as a plane $Pl : (a, b, c, 0)$ passing through the center of the sphere and is given by the equation $ax + by + cz = 0$. The

Table 5.1: Joint and Link data for Stephenson II linkage using Affine Coordinates

Joint	Type	Coordinates
$J_{1,input}$	Revolute	0, -1
J_2	Revolute	1, .5
J_3	Prismatic	-0.17, 0.98, -4.28
J_4	Revolute	3.25, 1.4
J_5	Revolute	7.72, 1.44
J_6	Revolute	11.66, 4.17
J_7	Prismatic	0, 1, 1.24
J_8	Coupler point	6, -2

Link	Constituent joints
L_1	J_1, J_2
L_2	J_2, J_3, J_4
L_3	J_3, J_6
L_4	J_4, J_5, J_8
L_5	J_5, J_6, J_7
$L_{6,ground}$	J_1, J_7

intersection of the plane and the unit sphere defines the great circle along which the constituent links are constrained to move for a spherical prismatic joint. Similar to the lines for planar mechanisms, this plane can be fixed or moving depending on the geometric constraint being expressed. An example RRPR spherical mechanism is displayed in Fig. 5.3. Its joints are represented as points and planes while its links are defined as a group of joints as shown in Table 5.2.

Table 5.2: Joint and Link data for Spherical RRPR linkage using Affine Coordinates; the coordinates are given wrt the fixed coordinate frame located at the center of the reference sphere.

Joint	Type	Coordinates
$J_{1,input}$	Revolute	0.94, 0.24, 0.24
J_2	Revolute	0.80, 0.27, 0.53
J_3	Prismatic	0.68, -0.68, 0.26
J_4	Revolute	-0.38, 0.76, 0.53
J_5	Coupler point	0.50, -0.21, 0.84

Link	Constituent joints
L_1	J_1, J_2
L_2	J_2, J_3, J_5
L_3	J_3, J_4
$L_{4,ground}$	J_1, J_4

During the motion, a mechanism is subjected to a set of constraints imposed by the rigidity of its links. Thus, to simulate a mechanism, these constraint equations need to be formulated. For planar and spherical mechanisms, modeling three constraint equations are sufficient for simulation. We propose a unique constraint equation for each of the binary links RR, RP, PR, and PP. But, we will see that all of these constraints can be expressed in a single equation. Any link with more than two joints can easily be reduced to an equivalent collection of binary links. For example, a ternary link can be treated as three binary links. Thus, these constraints can successfully be

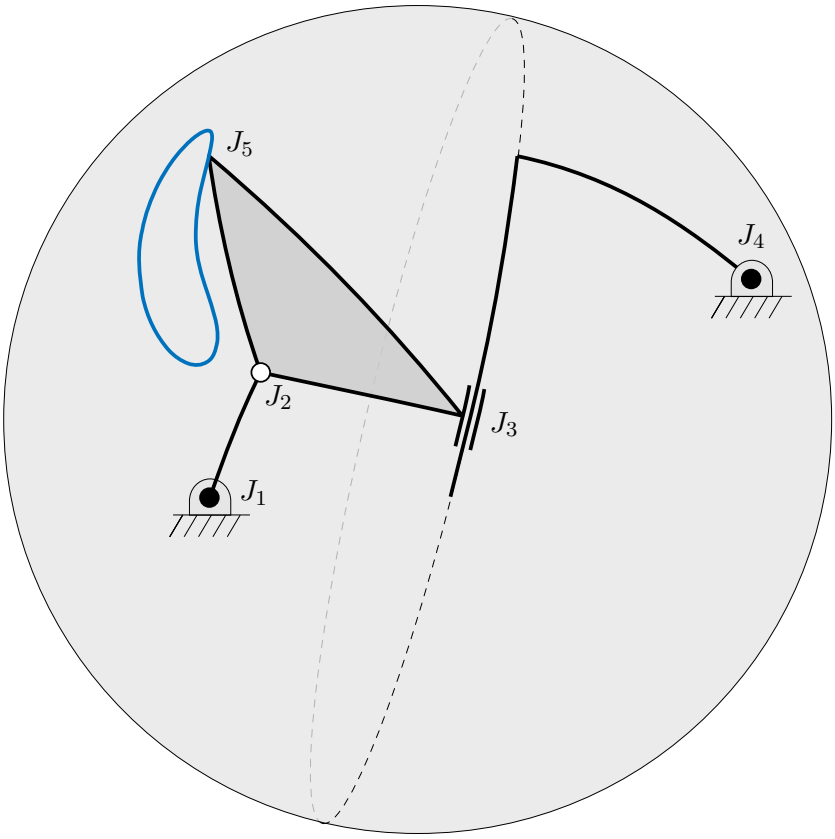


Figure 5.3: Spherical RRPR four-bar linkage

used to enforce the rigidity of any link in a general mechanism. Figure 5.4 and Fig. 5.5 show different planar and spherical binary links, which are building blocks for any planar and spherical mechanism. The RR link imposes the constraint that the distance between two points remains constant; an RP or PR link imposes the constraint that the distance between a point and a line (planar case) or a point and a plane (spherical case) is constant; and for the PP link, the angle between two lines (planar case) or two planes (spherical case) remains constant. RP and PR links are inversions of each other and are expressed by the same constraint. For planar/spherical cases, RP link has a fixed point and a floating line/plane, while PR link has a fixed line/plane and a floating point.

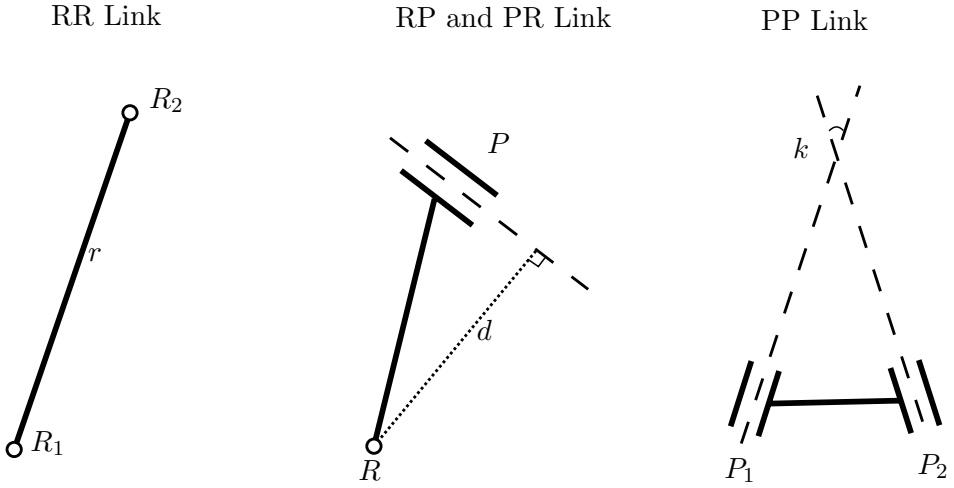


Figure 5.4: Types of binary planar links

The first general constraint enforces the rigidity of a spherical binary link with two revolute joints represented by two homogenous point coordinates of the fixed point (a_1, a_2, a_3, a_4) and floating point (b_1, b_2, b_3, b_4), where a_4 and b_4 are homogenizing factors. The constraint equation is given as

$$C_{S,RR} : 2a_1b_1 + 2a_2b_2 + 2a_3b_3 + a_0b_4 = a_4\left(\frac{b_1^2 + b_2^2 + b_3^2}{b_4}\right), \tag{5.1}$$

where a_0 is given as

$$a_0 = a_4r^2 - \frac{a_1^2 + a_2^2 + a_3^2}{a_4}. \tag{5.2}$$

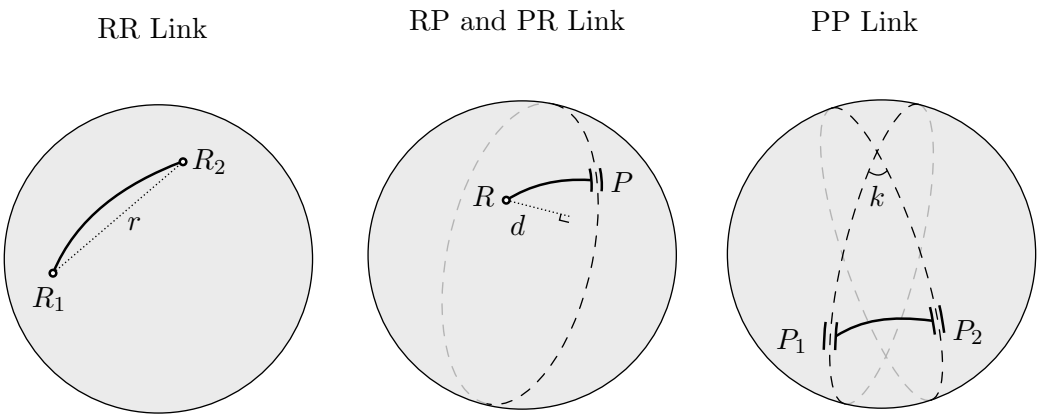


Figure 5.5: Types of binary spherical links

Here, r is the radius of the sphere formed by the RR link with the center given by (a_1, a_2, a_3, a_4). When the z-coordinate is set to zero, the constraint equation degenerates into the one for a planar RR link. The constraint equation for a planar RR link represented by points (a_1, a_2, a_4) and (b_1, b_2, b_4) is thus given as

$$C_{P,RR} : 2a_1b_1 + 2a_2b_2 + a_0b_4 = a_4 \left(\frac{b_1^2 + b_2^2}{b_4} \right), \tag{5.3}$$

where a_0 is given as

$$a_0 = a_4 r^2 - \frac{a_1^2 + a_2^2}{a_4}. \tag{5.4}$$

The second general constraint enforces the rigidity of a binary link with one prismatic and one revolute joint represented by a homogeneous point and a plane given by (a_1, a_2, a_3, a_4) and (L_1, L_2, L_3, L_4), respectively. The general constraint equation for a spherical RP link is given as

$$C_{RP} : a_1L_1 + a_2L_2 + a_3L_3 + a_4L_4 = da_4\sqrt{L_1^2 + L_2^2 + L_3^2}, \tag{5.5}$$

where d is the signed perpendicular distance between the revolute joint and prismatic joint. For spherical linkages, the prismatic plane always passes through the origin, i.e. $L_4 = 0$. Thus, the spherical RP link constraint equation reduces to

$$C_{S,RP} : a_1L_1 + a_2L_2 + a_3L_3 = da_4\sqrt{L_1^2 + L_2^2 + L_3^2}. \tag{5.6}$$

When the z-coordinates is set to zero, the general constraint equation degenerates into a planar case. Thus, constraint equation for a planar RP or PR link represented by a point (a_1, a_2, a_4) and a line (L_1, L_2, L_4) is given as

$$C_{P,RP} : a_1L_1 + a_2L_2 + a_4L_4 = da_4\sqrt{L_1^2 + L_2^2}. \tag{5.7}$$

When the perpendicular distance d becomes zero, the equation describes a line passing through a point, i.e. constraint equation of PR or RP links. This is usually the case with the PR links where the moving joint point is constrained to be on the fixed line of the prismatic joint and in case of the RP link where the moving line is constrained to pass through the point of the fixed joint.

Finally, the third constraint enforces the rigidity of a spherical binary link with two prismatic joints represented as $(L_1, L_2, L_3, 0)$ and $(M_1, M_2, M_3, 0)$. The constraint equation is given as

$$C_{S,PP} : L_1M_1 + L_2M_2 + L_3M_3 = k\sqrt{L_1^2 + L_2^2 + L_3^2}\sqrt{M_1^2 + M_2^2 + M_3^2}, \tag{5.8}$$

where k represents to the cosine of angle between two prismatic joints. Similarly, for planar PP binary link represented by two lines (L_1, L_2, L_4) and (M_1, M_2, M_4), the constraint equation degenerates to

$$C_{P,PP} : L_1M_1 + L_2M_2 = k\sqrt{L_1^2 + L_2^2}\sqrt{M_1^2 + M_2^2}. \tag{5.9}$$

When the two prismatic joints on a binary link are defined as two parallel lines, a degree of freedom is added to the mechanism. This situation is impractical and will not been considered further in this chapter.

It can be seen that the Eqs. 5.3, 5.6, 5.7, 5.8, 5.9 are all degenerate case of the Eq. 5.1. In the projective plane for the planar geometric constraints, the lines and points are dual to each other; thus, their meanings can be interchanged without changing the underlying structure of the equations. In the projective three-space for the spherical constraints, the points and planes are dual to each other and thus their meanings can be interchanged. Thus, the Eq. 5.1 is the single equation that unifies all the geometric constraints associated with all types of links for both planar and spherical mechanisms. This facilitates creation of the following metrics for computation:

1. Distance between two points in space;

2. Perpendicular distance between a point and a plane;

3. Angle between two planes

For links with prismatic joints, the line or plane coordinates are homogenous in nature, i.e. multiplying a non-zero scalar λ to prismatic coordinates (L_1, L_2, L_3) does not change the coordinates. Thus, the magnitude of this vector can be fixed to unity without losing generality and another constraint can be written as

$$C_P : \quad L_1^2 + L_2^2 + L_3^2 - 1 = 0. \tag{5.10}$$

For spherical mechanisms, an additional geometric constraint is imposed on the joints due to the spherical nature of the motion. It is assumed that all the revolute joints move on the unit sphere which leads to the constraint

$$C_{S,R} : \quad a_1^2 + a_2^2 + a_3^2 - 1 = 0, \tag{5.11}$$

where (a_1, a_2, a_3) are the coordinates of any revolute joint on a spherical mechanism. Thus, the rigidity constraints described in Eqs. (5.1), (5.3), (5.6), (5.7), (5.8), (5.9), (5.10) and (5.11) are sufficient to uniquely determine the unknown coordinates of a n-bar planar or spherical mechanism. This concludes our discussion on representation and constraints for a generalized planar or spherical mechanism.

5.3 Solving Constraint Equations

In this section, we discuss the algorithmic steps required to solve the kinematic simulation problem. The general approach is to iteratively perturb the input links by a finite displacement and find the new position of the mechanism.

5.3.1 Input link perturbation

The simulation process involves iteratively perturbing the input link by a finite displacement. Depending on the actuating joint being revolute or prismatic, the displacement could be translation or rotational in nature. In this chapter, we restrict ourselves to consider actuation at the fixed joints. The relations governing the motion of input link are derived in this subsection.

For a perturbed RR link with the actuating fixed joint (x_1, y_1) and moving joint (x_2, y_2), the new coordinates of moving revolute joint can be given as

$$\begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix} = [\mathbf{T}]^{-\mathbf{1}}[\mathbf{R}][\mathbf{T}] \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}, \tag{5.12}$$

where

$$[\mathbf{R_x}] = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{and} \quad [\mathbf{T}] = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.13}$$

In the above equation, (X_2, Y_2) represent the moving joint after perturbation and θ is the angle through which the input link is perturbed.

For a perturbed RP link with the actuating fixed joint (x_1, y_1) and moving joint (a, b, c), the new coordinates of moving line representing the prismatic joint can be given as

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = ([\mathbf{T}]^{-\mathbf{1}}[\mathbf{R}][\mathbf{T}])^{-\mathbf{T}} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{5.14}$$

where (A, B, C) are the moving line coordinates after perturbation, and $\mathbf{T}$ and $\mathbf{R_x}$ are the translation and rotation matrices as described in Eqs. (5.13).

For a planar mechanism with the actuation being at prismatic joint, input link perturbation causes translation of other joints on the input link. For a perturbed PR link with the actuating joint (a, b, c) and moving joint (x, y), the new coordinates of translating revolute joint can be given as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{b}{\sqrt{a^2+b^2}}d \\ \frac{-a}{\sqrt{a^2+b^2}}d \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.15}$$

where (X, Y) are the moving joint coordinates after perturbation and d is the distance through which the prismatic joint is moved along the fixed line. It can be seen that in Eq. (5.15) the actuating line coordinate c doesn't effect the new position of moving joint coordinates as new position only depends on direction cosines.

For a perturbed PP link with the actuating fixed joint (a_1, b_1, c_1) and moving joint (a_2, b_2, c_2), the new coordinates of translating prismatic joint can be given as

$$\begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{a_1 b_2 - a_2 b_1}{\sqrt{a_1^2 + b_1^2}} d \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{5.16}$$

where (A_2, B_2, C_2) are the moving prismatic joint coordinates after perturbation and d is the distance through which the input link has been perturbed.

Similarly, relationships determining the values of perturbed joints for spherical mechanisms can also be calculated. For spherical mechanisms with a fixed revolute actuating joint, the moving joints rotate around the axis passing through the actuation joint and the centre of sphere. The transformation matrix which rotates spherical link around an axis passing through the centre of sphere $(0, 0, 0)$ and an arbitrary point on surface of the sphere (l, m, n) is given by

$$[\mathbf{R}]_{(l,m,n)} = [\mathbf{R_x}]^{-1}[\mathbf{R_y}]^{-1}[\mathbf{R_z}][\mathbf{R_y}][\mathbf{R_x}] \tag{5.17}$$

$$[\mathbf{R_x}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{n}{\sqrt{m^2+n^2}} & \frac{-m}{\sqrt{m^2+n^2}} \\ 0 & \frac{m}{\sqrt{m^2+n^2}} & \frac{n}{\sqrt{m^2+n^2}} \end{bmatrix} \tag{5.18}$$

$$[\mathbf{R_y}] = \begin{bmatrix} \sqrt{m^2+n^2} & 0 & -l \\ 0 & 1 & 0 \\ l & 0 & \sqrt{m^2+n^2} \end{bmatrix} \tag{5.19}$$

$$[\mathbf{R_z}] = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

where $\mathbf{R_x}$, $\mathbf{R_y}$, $\mathbf{R_z}$ are the rotation matrix around x,y and z axis and θ is the angle by which the link is rotated around the axis.

Using Eq. (5.17), the new coordinates of the moving joints of a perturbed spherical RR link can be given as

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = [\mathbf{R}]_{(x_1,y_1,z_1)} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{5.21}$$

where (x_1, y_1, z_1) are the fixed joint coordinates, (x_2, y_2, z_2) are the moving

joint coordinates before perturbation and (X_2, Y_2, Z_2) are the moving joint coordinates after perturbation.

For a spherical RP link with a fixed revolute joint, the coordinates of moving prismatic joint can be given as

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = [\mathbf{R}]_{(x,y,z)} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5.22}$$

where (x, y, z) are the fixed joint coordinates, (a, b, c) are the moving joint coordinates before perturbation, (A, B, C) are the moving joint coordinates after perturbation, and as described above.

When the actuation joint is prismatic in nature, the moving joints translate on the intersection of a parallel plane and the unit sphere. This motion can also be characterized as rotation around an axis which passes through the centre of the sphere and 'pole' of the prismatic joint. The poles of a great circle are defined as intersection of two circles perpendicular to the initial circle. If a spherical prismatic joint is defined as a plane (a, b, c), its pole coordinates are also given as (a, b, c). Thus, for a spherical RP link with fixed prismatic joint, the coordinates of moving revolute joint can be given as rotation i.e.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [\mathbf{R}]_{(a,b,c)} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.23}$$

where (a, b, c) are the fixed prismatic joint coordinates, (x, y, z) are the moving revolute joint coordinates before perturbation and (X, Y, Z) are the moving revolute joint coordinates after perturbation.

For a spherical PP link with a fixed prismatic joint, the coordinates of a moving prismatic joint can be given as

$$\begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = [\mathbf{R}]_{(a_1,b_1,c_1)} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{5.24}$$

where (a_1, b_1, c_1) are the coordinates of fixed prismatic joint, (a_2, b_2, c_2) are moving prismatic joint coordinates before perturbation and (A_2, B_2, C_2) are moving prismatic joint coordinates after perturbation.

With these expressions, we can successfully calculate the location of input link after imparting it a discrete perturbation. The next step is to find the

coordinates of all the other unknown joint coordinates which are compatible with the rigidity constraints imposed on the mechanism during simulation.

5.3.2 Numerical nonlinear system of equation solving

For any multi-body system, the position problem is always based on solving a system of constraint equations. This set of equations can be represented as

$$\mathbf{\Phi}(\mathbf{q}) = 0 \tag{5.25}$$

where $\mathbf{q}$ is the state vector which consists of all the unknown coordinates. The well-known Newton-Rhapson method can be used to solve this non-linear system of equation. It is featured by quadratic convergence in the neighborhood of the solution. Since the input link is perturbed by a small finite displacement, the previous state of mechanism serves as a good initial approximation. The number of constraint equations should be equal to or greater than the number of unknowns for this approach to work. For planar and spherical mechanisms, it is always possible to satisfy this criterion using the constraints outlined in section.

The iterative algorithm followed can be defined as

$$\mathbf{q}_{i+1} = \mathbf{q}_i - [\boldsymbol{J}^{-1}(\mathbf{q}_i)]\mathbf{\Phi}(\mathbf{q}_i) \tag{5.26}$$

where $\mathbf{q}_i$ is the state vector at i^{th} iteration, $\mathbf{\Phi}(\mathbf{q}_i)$ is the vector of residuals at $\mathbf{q} = \mathbf{q}_i$, and $[\boldsymbol{J}^{-1}(\mathbf{q}_i)]$ is the inverse of Jacobian matrix evaluated at $\mathbf{q} = \mathbf{q}_i$. The Jacobian matrix is of the following form

$$[\boldsymbol{J}(\mathbf{q})] = \begin{bmatrix} \frac{\partial \phi_1}{\partial q_1} & \frac{\partial \phi_1}{\partial q_2} & \cdots & \frac{\partial \phi_1}{\partial q_n} \\ \frac{\partial \phi_2}{\partial q_1} & \frac{\partial \phi_2}{\partial q_2} & \cdots & \frac{\partial \phi_2}{\partial q_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \phi_m}{\partial q_1} & \frac{\partial \phi_m}{\partial q_2} & \cdots & \frac{\partial \phi_m}{\partial q_n} \end{bmatrix} \tag{5.27}$$

where m is the number of constraints and n is the number of unknown coordinates. Thus to calculate the Jacobian matrix, relations describing the first order partial derivatives of constraint equations are required.

The first order partial derivatives for spherical RR constraint given in

Eq. (5.1) can be given as follows

$$\frac{\partial C_{RR}}{\partial a_1} = 2(a_1 - b_1) \tag{5.28}$$

$$\frac{\partial C_{RR}}{\partial a_2} = 2(a_2 - b_2) \tag{5.29}$$

$$\frac{\partial C_{RR}}{\partial a_3} = 2(a_3 - b_3) \tag{5.30}$$

Here, the homogenous point coordinate a_0 and b_4 has been assumed as unity without loss in generality. For planar RR link, only $\frac{\partial C_{RR}}{\partial a_1}$ and $\frac{\partial C_{RR}}{\partial a_2}$ exists.

The first order partial derivatives for spherical RP constraint given in Eq. (5.6) can be given as follows

$$\frac{\partial C_{S,RP}}{\partial a_1} = L_1, \frac{\partial C_{S,RP}}{\partial a_2} = L_2, \frac{\partial C_{S,RP}}{\partial a_3} = L_3 \tag{5.31}$$

$$\frac{\partial C_{S,RP}}{\partial L_1} = a_1 - \frac{dL_1}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{5.32}$$

$$\frac{\partial C_{S,RP}}{\partial L_2} = a_2 - \frac{dL_2}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{5.33}$$

$$\frac{\partial C_{S,RP}}{\partial L_3} = a_3 - \frac{dL_3}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{5.34}$$

Here, the homogenous point coordinate a_0 has been assumed as unity without loss in generality. For planar RR link, $\frac{\partial C_{P,RP}}{\partial a_1}$ and $\frac{\partial C_{P,RP}}{\partial a_2}$ are same, $\frac{\partial C_{P,RP}}{\partial a_3}$ doesn't exist, $\frac{\partial C_{P,RP}}{\partial L_1}$ and $\frac{\partial C_{P,RP}}{\partial L_1}$ have $L_3 = 0$ and $\frac{\partial C_{P,RP}}{\partial L_4} = 1$

The first order partial derivatives for spherical PP constraint given in Eq. (5.8) can be given as follows

$$\frac{\partial C_{S,PP}}{\partial L_1} = M_1 - \frac{kL_1\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{5.35}$$

$$\frac{\partial C_{S,PP}}{\partial L_2} = M_2 - \frac{kL_2\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{5.36}$$

$$\frac{\partial C_{S,PP}}{\partial L_3} = M_3 - \frac{kL_3\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{5.37}$$

These equations degenerate to planar case when $L_3 = 0$ and $M_3 = 0$ and $\frac{\partial C_{P,PP}}{\partial L_4} = 0$

The first order partial derivatives for homogenous Prismatic joint constraint given in Eq. (5.10) can be given as follows

$$\frac{\partial C_P}{\partial a} = 2a, \quad \frac{\partial C_P}{\partial b} = 2b, \quad \frac{\partial C_P}{\partial c} = 2c \tag{5.38}$$

The first order partial derivatives for unit circle Revolute joint constraint given in Eq. (5.11) can be given as follows

$$\frac{\partial C_{S,R}}{\partial x} = 2x, \quad \frac{\partial C_{S,R}}{\partial y} = 2y, \quad \frac{\partial C_{S,R}}{\partial z} = 2z \tag{5.39}$$

To automate the calculation of residual vector $\mathbf{\Phi}(\mathbf{q}_i)$ and the Jacobian matrix $[\boldsymbol{J}(\mathbf{q}_i)]$, the constraints are handled in a sequential manner. While creating the residual vector in our implementation, first the rigidity constraints for each link are calculated and then the constraints for joints are calculated. Similarly, the Jacobian matrix is created in a column-first manner i.e. all the partial differential equations with respect to an unknown state variable are calculated before progressing to the next variable. The outlined method is just one way of calculating $\mathbf{\Phi}(\mathbf{q}_i)$ and $[\boldsymbol{J}(\mathbf{q}_i)]$ since their values are independent of the sequence adopted to calculate each element.

Thus, using the constraint equations and their first order partial derivatives, it is possible to solve iteratively for the solution using Newton-Rhapson method. The iterations are continued until a solution within desired accuracy is calculated.

For some input link perturbations, the Newton-Rhapson method might fail to converge even after many iterations. In these instances, there does not exist a mechanism state which fulfills all the constraint equations. As a result, this input perturbation is outside the possible limits of motion of mechanism.

Thus, by iteratively perturbing the input link and solving the constraints for other joint coordinates, we are able to simulate any general planar or spherical mechanism. Numerous techniques exist that can improve the convergence and efficiency of the Newton-Rhapson method. However, the basic method suffices to achieve real-time simulation. The complete algorithm has been described in Algo 3.

Algorithm 3: Algorithm for planar and spherical mechanism simulation

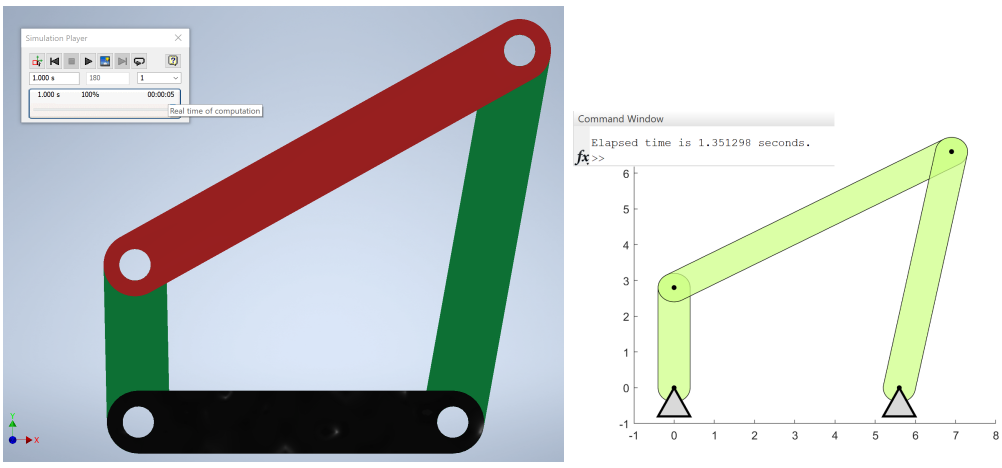
```
Input: Initial mechanism configuration
Calculate initial rigidity metric for each link
for range of input link motion do
    Perturb input link
    for max iterations do
        Calculate the constraint residual
        if residual ≤ ϵ then
            Solution found
            break
        end
        Calculate the Jacobian matrix
        Calculate predicted unknown joint coordinates
    end
    if solution found then
        Store as subsequent mechanism state
    else
        Motion limit reached
end
Animate the range of motion
Output: Mechanism simulation
```

5.4 Examples

This section presents sample examples to demonstrate the use of proposed algorithm for mechanism simulation. The simulation has been carried out in MATLAB on a PC running Core i5-7300 at 2.6GHz with 8gb RAM. The simulation is carried out within seconds for residual value of $1.0e-8$. Each closed-loop output curve is made up of 180 points while open-loop curves have less than 180. Each point corresponds to $\frac{2\pi}{180}$ radians or units input perturbations.

5.4.1 Speed comparison with a commercial software

To compare the speed of commercially available CAD systems and proposed algorithm, a planar four-bar crank rocker mechanism with revolute joints is modeled and simulated. Autodesk Inventor 2020 with educational license is used as reference commercial CAD system. The model which is run on both systems is displayed in Fig 5.6. Simulation is performed for 180 timesteps with one degree input link perturbation for each timestep. The simulation takes 5s to complete Inventor while it finishes in 1.4s when the proposed algorithm is used. Thus, even for a simple mechanism, there is a significant speed difference between the commercial solvers and proposed methodology.



(a) Autodesk Inventor (b) Proposed algorithm

Figure 5.6: Speed comparison for a planar crank rocker mechanism

5.4.2 Planar Stephenson-II linkage

A planar Stephenson-II six-bar linkage is simulated in this example. This linkage does not have a four-bar linkage and proves to be challenging to simulate using dyadic decomposition based approaches. However, our approach handles these non-dyadic mechanisms without issues.

The six-bar mechanism is displayed in Fig. 5.2 and its joint and link data is given in Table 5.1. The mechanism has J_1, J_7 as the fixed joints, J_2 as the perturbed joint and J_3, J_4, J_5, J_6, J_8 as the unknown joints defining the 11-dimensional state vector. The mechanism consists of ten rigidity constraint equations and one homogeneous coordinate equation for prismatic joint. The simulation algorithm successfully solves these constraints and plots the trajectory of the coupler point J_8 as shown in Fig. 5.2. The run-time of this simulation was 3.79s.

5.4.3 Planar Modified Theo Jansen linkage

In this example, a planar modified Theo Jansen linkage with one of its revolute joints replaced by a floating prismatic joints is simulated. The eight-bar mechanism is displayed in Fig. 5.7 and its joint and link data is given in Table 5.3. J_1, J_5 are the fixed joints, J_2 is the perturbed joint and the state vector consists coordinates of J_3, J_4, J_6, J_7, J_8. This results in a 11-dimensional state vector. Ten rigidity constraint equations for links and one homogeneous coordinate equation for prismatic joint are available for this mechanism. The simulation algorithm plots the trajectory of the coupler point J_8 as shown in Fig. 5.7. Note, the length of stride for this modified mechanism is larger than that of the conventional Theo Jansen mechanism which has revolute joints only. As a result, this mechanism is a prospective candidate for walking robots. The run-time of this simulation was 3.81s.

5.4.4 Spherical RRPR mechanism

This example presents the simulation of a spherical RRPR mechanism which is the spherical analog of the Whitworth quick-return mechanism. The four-bar mechanism is displayed in Fig. 5.3 and it's joint and link data is given in Table 5.2. The mechanism has J_1, J_4 as the fixed joints, J_2 as the perturbed joint and J_3, J_5 as the unknown joints defining the 6-dimensional state vector. The mechanism consists of four rigidity constraint equations, one unit sphere

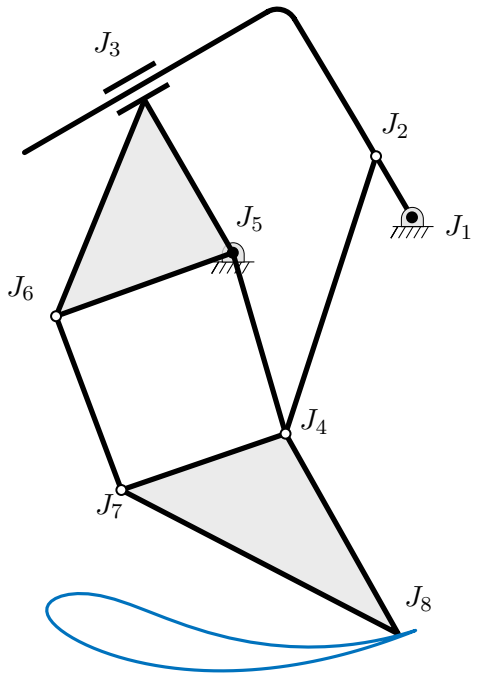


Figure 5.7: Planar modified Theo Jansen with floating prismatic joint

Table 5.3: Joint and Link data for Modified Theo Jansen linkage

Joint	Coordinates
$J_{1,input}$	2.77, 2.31
J_2	2.17, 3.33
J_3	-0.50, 0.87, -4.80
J_4	.66, -1.3
J_5	-.22, 1.72
J_6	-3.17, .66
J_7	-2.08, -2.24
J_8	2.54, -4.64

Link	Constituent joints
L_1	J_1, J_2
L_2	J_2, J_3
L_3	J_2, J_4
L_4	J_3, J_5, J_6
L_5	J_5, J_4
L_6	J_6, J_7
L_7	J_4, J_7, J_8
$L_{8,ground}$	J_1, J_5

equation for revolute joint and one homogeneous coordinate equation for prismatic joint. Once the simulation is completed, the trajectory of coupler point J_5 can be plotted as shown in Fig. 5.3. The run-time of this simulation was 1.49s.

5.4.5 Spherical Watt-I linkage

In this example, a spherical Watt-I six-bar linkage with prismatic input joint is simulated. Spherical Watt I type linkages have been used to design door hinges for spatial movement. The six-bar mechanism is shown in Fig. 5.8 and its link and joint data is given in Table 5.4. From the data, its known that J_1, J_6 are the fixed joints, J_2, J_3 are the perturbed joints and J_4, J_5, J_7, J_8 are the unknown joints representing the 12-dimensional state vector. The mechanism can be described using eight rigidity constraints for links and four unit circle constraints. Perturbing the input link along the input prismatic joint results in the motion of coupler point J_8 as shown in Fig. 5.8. The run-time of this simulation was 3.33s.

Table 5.4: Joint and Link data for Spherical RRPR linkage

Joint	Coordinates
$J_{1,input}$	0, 0, 1
J_2	0.93, 0, 0.37
J_3	0.85, -0.17, 0.51
J_4	0.70, 0.70, 0.14
J_5	0.73, 0.49, 0.49
J_6	0.81, 0.41, -0.41
J_7	0.48, -0.10, 0.87
J_8	0.49, 0.49, 0.73

Link	Constituent joints
L_1	J_1, J_2, J_3
L_2	J_2, J_4, J_5
L_3	J_4, J_6
L_4	J_3, J_7
L_5	J_5, J_7, J_8
$L_{6,ground}$	J_1, J_6

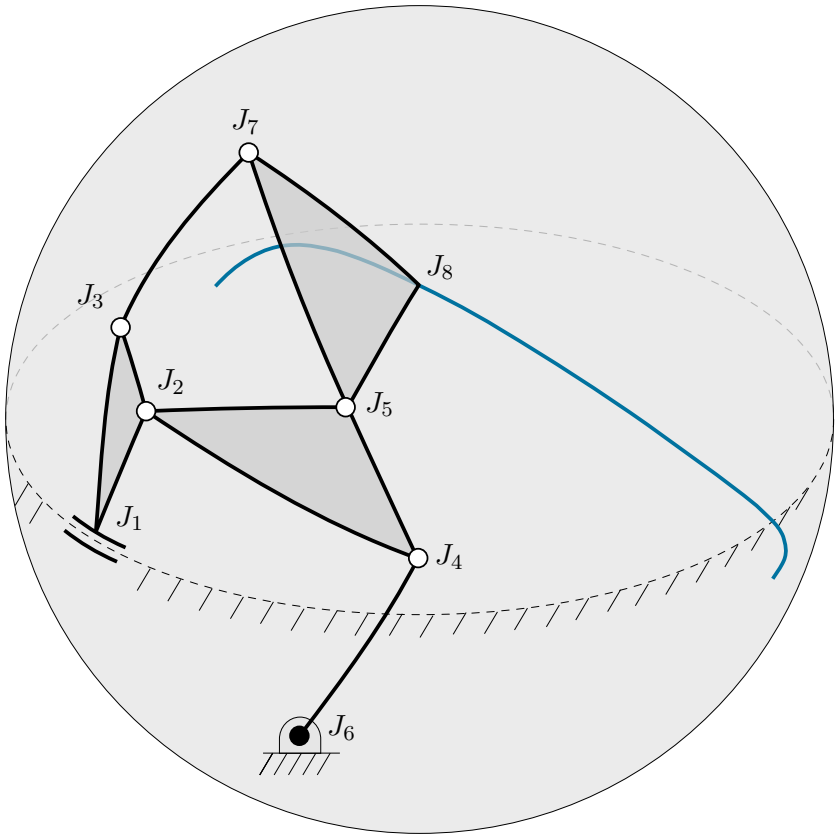


Figure 5.8: Spherical Watt I six-bar linkage

5.5 Conclusion

In this chapter, we have presented unified equations for motion simulation of planar and spherical $n-$bar mechanisms and an efficient algorithm for computation to enable real-time, interactive simulation. The approach is general and uses simple geometric primitives, such as point, line and planes to used to represent the constraints inherent in mechanisms.

Chapter 6

MotionGen 2.0

6.1 Software Review

Many existing kinematic design and synthesis softwares are being used by professionals and researchers. However, none of them seem to have widespread adoption. Some of the major ones have been discussed in this section. Special emphasis has been placed on determining the extent of ability, to which a software can simulate and synthesize mechanism.

6.1.1 Autodesk ForceEffect Motion

ForceEffect [77] is a mechanical design application developed by Autodesk Inc. It was available for iOS, Android, and Desktop platforms. It had two versions called Mechanical and Motion. Fig. 6.1 displays the dashboard of Autodesk ForceEffect Motion. Autodesk retired the app in 2016. It used to provide rich functionality for the simulation and analysis of multi-link mechanisms. However, it lacked any synthesis functionality.

Thus its abilities can be summarized as

- Analysis
    1. Static load analysis
    2. N-bar one degree of freedom mechanisms simulation (revolute and prismatic joints).
    3. Graphing the position, velocity, and accelerations.

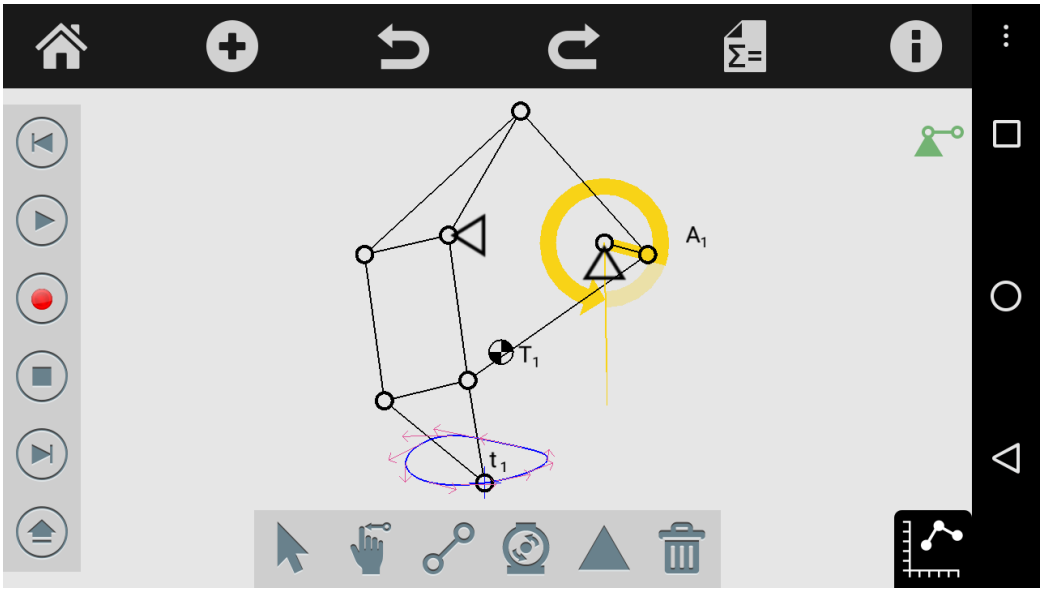


Figure 6.1: Autodesk ForceEffect Motion Dashboard

- Synthesis

    1. No synthesis ability.

6.1.2 Linkage

Linkage [51] is a Windows program developed by David Rector. Its focus is on quick prototyping of linkage mechanisms. It has an intuitive UI and feels like a well-built CAD software. Fig. 6.2 displays the dashboard of Linkage. The strength of Linkage lies in its ability to simulate n-bar mechanisms. However, due to the use of dyadic decomposition, it cannot be used to simulate mechanisms with more complex topologies like involvement of triads. As the purpose of software is a quick analysis of mechanisms, it lacks the ability to do path, motion or function synthesis.

Thus, its abilities can be summarized as

- Analysis

    1. N-bar multi-degree of freedom mechanisms simulation (revolute and prismatic joints). However, only dyadic decomposable mechanisms can be simulated.

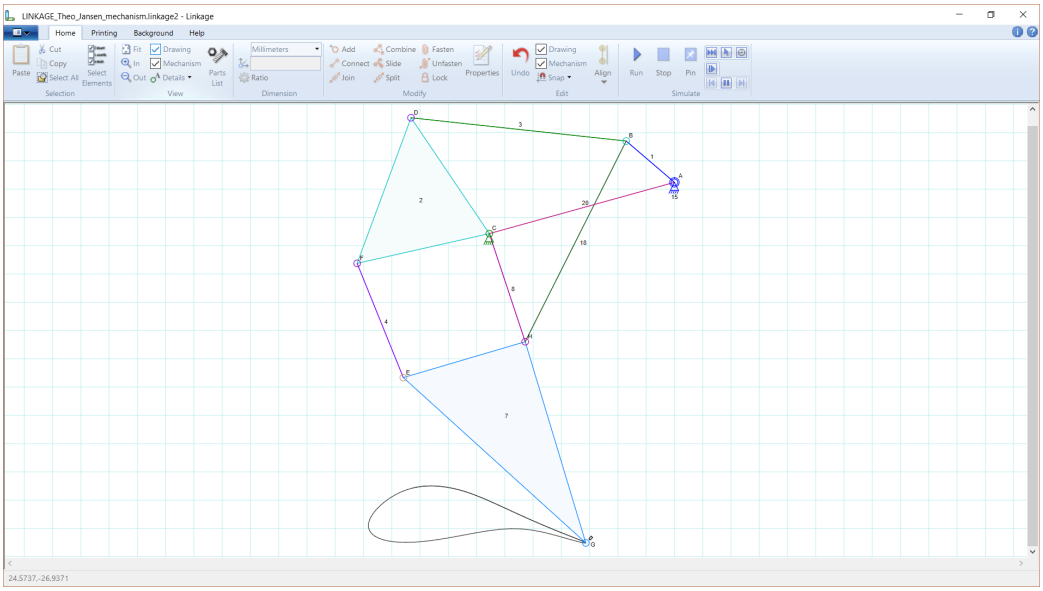


Figure 6.2: Linkage Dashboard

2. Can simulate gears and chains
3. Supports multiple simultaneous simulations

- Synthesis

1. No synthesis ability.

6.1.3 Planar Mechanism Kinematic Simulator (PMKS)

PMKS [52] is a web application developed by Design Engineering Lab at Oregon State University. Fig. 6.3 displays the dashboard of PMKS. Its purpose is to simulate one degree of freedom planar mechanisms with multiple rigid bodies. It can calculate quick and accurate results for the position, velocity, and acceleration. It can handle non-dyadic mechanisms using a novel approach. Synthesis of mechanism is not the focus of this application.

Thus, its abilities can be summarized as

- Analysis

1. N-bar one degree of freedom mechanisms simulation (revolute and prismatic joints).

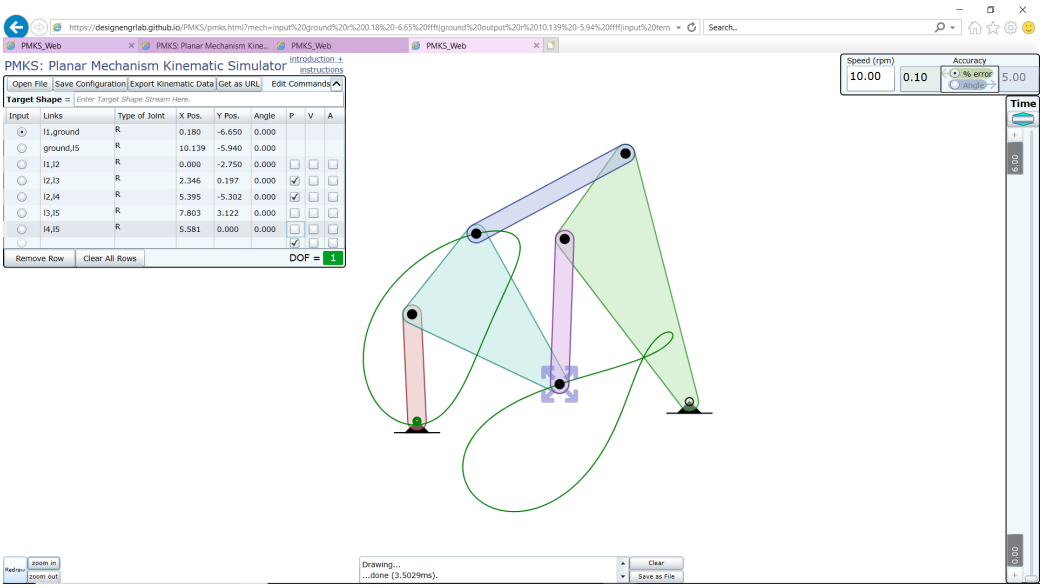


Figure 6.3: Planar Mechanism Kinematic Simulator Dashboard

2. Calculates position, velocity and acceleration data.

- Synthesis

1. No synthesis ability.

6.1.4 Synthesis and Analysis of Mechanisms (SAM)

SAM [67] is a feature-rich PC software developed by Artas Engineering. Fig. 6.4 displays the dashboard of SAM. Although its UI seems dated and nonintuitive, it boasts both mechanism analysis and synthesis capabilities. It has functionality to supports n-bar simulation for planar linkages with both Revolute and Prismatic joints. SAM offers a set of design wizards to synthesize four-bar mechanisms for motion and function generation. A wizard for path synthesis is not present.

The software does have the ability to uses Simplex Method or evolutionary algorithm for the local optimization. It lacks global optimization methods. Using these methods, path optimization of existing mechanism can possibly be modeled. However, it is up to the user to define the objective function.

Thus, features of SAM can be summarized as

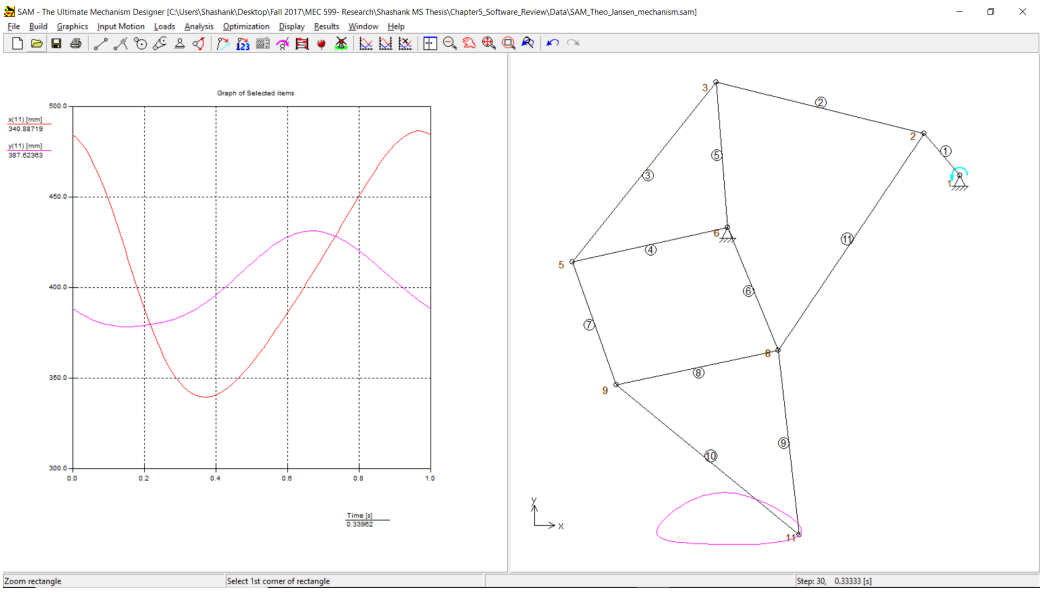


Figure 6.4: Synthesis and Analysis of Mechanisms Dashboard

- Analysis
  1. N-bar multi degree of freedom mechanisms simulation upto 10 simultaneous inputs (revolute and prismatic joints).
  2. Detailed graphing of position, velocity and acceleration data.
  3. Can simulate gears and chains
  4. Force analysis
- Synthesis
  1. Path synthesis possible but difficult.
  2. Motion synthesis of four-bar mechanism for 3 poses using graphical method.
  3. Function generation of four-bar mechanism for $\geq$ 3 input-output angle pair.
  4. Exact and approximate straight line mechanisms

6.1.5 GIM

GIM [68] has been developed by COMPMECH research group in the University of the Basque Country UPV/EHU. Fig. 6.5 displays the dashboard of GIM. Mechanisms with n-ary elements joined by revolute and prismatic pairs can be simulated in GIM, The position problem is solved iteratively using a numerical method.

Although the focus of GIM is analysis of mechanisms, it does have some synthesis capabilities. It uses graphical method to do 3,4,5 precision point synthesis. Motion synthesis for 3,4 pose is also possible in the software. Function generation for three input-output angles can be done. All of these synthesis options are very limited and doesn't give the user freedom to specify an arbitrary number of inputs. Also, it is not freely distributed which restricts its usage.

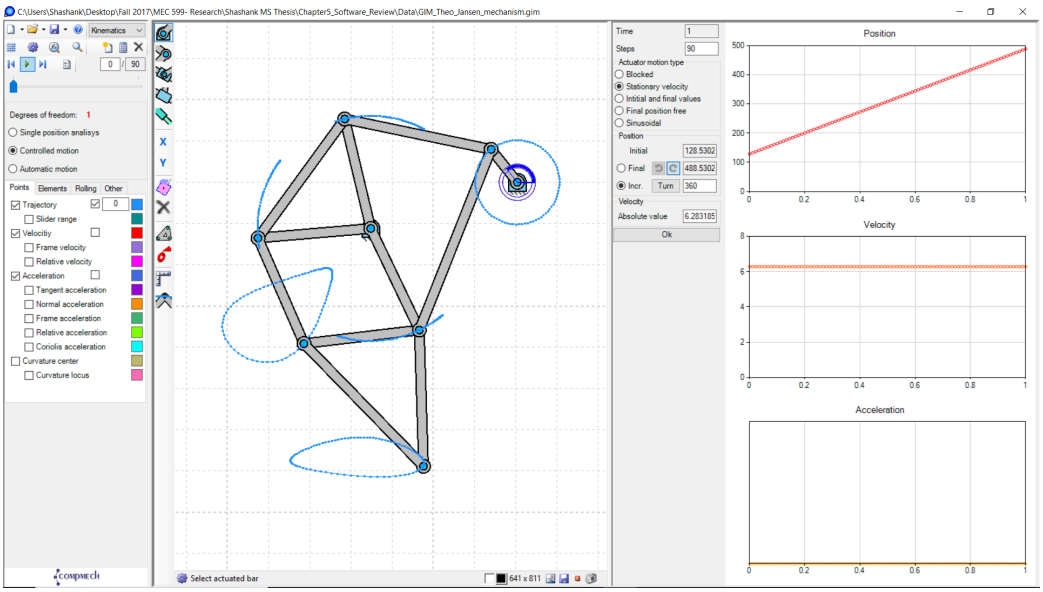


Figure 6.5: GIM Dashboard

- Analysis

  1. N-bar multi degree of freedom mechanisms simulation(revolute and prismatic joints).
  2. Verbose graphing of position, velocity and acceleration data.

3. Can simulate cams, gears and chains
4. Static analysis

- Synthesis
  1. Path synthesis of four bar mechanism for 3,4,5 precision points.
  2. Motion synthesis of four bar mechanism for 3,4 poses.
  3. Function generation of four bar mechanism for 3 input-output angle pair.

6.1.6 Limitations of existing software

After analyzing the most prevalent kinematic softwares, these are the common drawbacks observed

- None of the softwares is able to do n-point path synthesis for four-bar mechanisms.
- None of the softwares is available across all platforms i.e. on PC, Mac, and Mobile. Most of them require initial setup and installation.
- Majority of softwares have not been updated in a long time. They have old GUI's which is difficult to understand and use. This leads to a steep learning curve for newcomers.

To the best of authors knowledge, this thesis outlines the first attempt at n-point Path Synthesis using web-based implementation. Also, this is the first attempt to implement Generalized Mixed Synthesis.

6.2 Application Design Guiding Principles

This section outlines the design principles which have guided the course of application development and their immense importance. The goal is to create a kinematic analysis and syntheses software which is accessible everywhere, modular, scalable, and fast. It also needs to be easy to understand and use.

The software has been created upon MotionGen, a four-bar analysis and motion synthesis software. Henceforth, this newly developed software implementation of the previous MotionGen is referred to as MotionGen 2.0. Finer details comparing features of old software to new software has been discussed in next section. This section delves into the design philosophy incorporated.

6.2.1 Design Paradigm: MVC

The Model-View-Controller design pattern (MVC) is a global high-level pattern which acts upon the structure of software and basically separates the core logic from the rest of user interface. Using MVC design on an object-oriented programing language is especially beneficial to a codes structure. MVC self-enforces a structure on the codebase, making it more reusable and its communication channels better defined. As a result, the software is extremely adaptable to changes i.e. new functionality can easily be added or old obsolete functionality removed. Thus, MVC facilitates better maintenance and a robust program structure.

The MVC design pattern directs the programmer to group all objects into three types of objects: model, view, and controller. This categorization is done based on an object's role in software. Once the classification is done, MVC architecture has a set of well-defined guidelines which needs to be followed to standardize the exchange of data between these three individual types of object types. Thus, to follow the MVC framework, it is of paramount importance that the application designer chooses each system object to fall into one of three basic object types.

- **Model**: The model-objects is tasked with managing the core functionality of the application and stores all the data which defines the state of the application at any given point in time. It communicates with view objects who query it for information related to present system state. The controller object can instruct the model object to change its state based on user intent.

  For MotionGen2.0, the salient functionality is twofold, namely analysis and synthesis. Model objects containing analysis data include Links, Joints, and Linkages while the synthesis data is held in Points, Lines, Poses, and Path. Multiple instances of these model objects represent the state of workspace at any given point in time.

- **View**: The view-objects is tasked with managing the display or communication of information to the user. It communicates with model objects and gets the necessary data required by the user by querying state data. It then presents this information usually in an audio-visual manner.

  For MotionGen 2.0, HTML layered canvas acts as view objects on which

graphics are drawn. These graphics are dependent on the data obtained from the models representing system state.

- **Controller**: The controller sole purpose is to decipher the mouse, keyboard and touch inputs from the user. It then processes this information and instructs model and view objects to make appropriate changes. Thus, it provides the interface between the user and the application.

  In MotionGen 2.0, the controller is tasked with deciphering touch, mouse or keyboard input and deciding which action to carry out. This includes changes to model objects or view objects.

- **Passive model**: There are two types of MVC models, passive and active. In a passive model, only the controller objects can instruct the model object to change. However, in an active model, both view and controller objects have the ability to manipulate the model object.

  Passive model is being used as it is simpler and does not require an extra observer class to keep track of all changes happening to system state. Thus, when the controller modifies the model object, it handles the responsibility of instructing the view object to update itself accordingly. The model in this scenario is completely independent of the view and the controller, which means that there is no way for the model to report changes in its state.

Fig. 6.6 displays the flow of information through the software. Note that model objects don't push data themselves. It needs to be pulled by the view or pushed by the controller. As a result, the model has the capability to be a self-sufficient unit while view and controller are dependent on the model for information. This separation is what empowers the MVC approach and allow independent programming and testing of core logic from user interface.

6.2.2 Design Paradigm: RESTful web service

The Internet is the glue that binds diverse devices ranging from PC to mobile together. It is a neutral platform which is accessible to all. To reach out to a maximum number of users, it is imperative to embrace the web as the development platform.

RESTful web services are built to work best on the Web. REST was first introduced by Roy Fielding in year 2000 [78]. Representational State

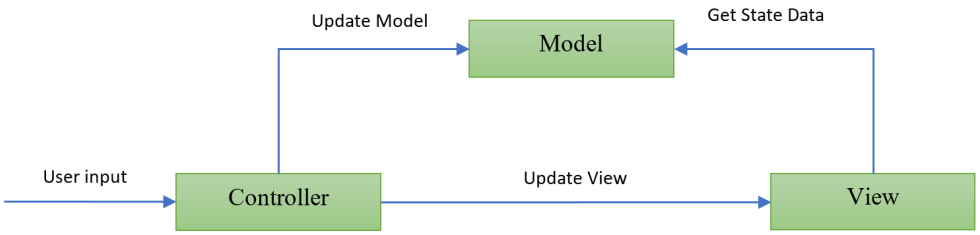


Figure 6.6: Passive MVC Architecture

Transfer (REST) is a web service architectural style which proposes certain guidelines to enhance the performance, scalability, and modifiability of the service on web.

RESTful directives focus on creation of a stateless service which can be used in a client/server framework easily. These stateless services offered through web are termed as web resources and they are accessible using unique Uniform Resource Identifiers (URIs). As a result, standardized interface to web resource can be achieved. The communication to these web resources is usually using HTTP which itself is a stateless protocol.

The set of rules which enable a web service to adopt the RESTful architectural style are as follows.

- **Addressability**: It is mandatory for each web resource exposed by the RESTful service to be identifiable on web. To accomplish this, URIs are employed as global resources addresses. Using these URIs, the client can make use of resources available with the web service.

- **Uniform interface**: The availability of web resources is established using a standardized stateless protocol like HTTP. Resources can subsequently be manipulated using a fixed set of four create, read, update, delete (CRUD) operations. For HTTP protocol, these turn out to be PUT, GET, POST, and DELETE requests.

- **Self-descriptive messages**: Requests to web resources need to be representation independent. This implies the server should be able to handle a variety of data formats, such as HTML, XML, plain text, JSON, etc.

- **Client-server architecture**: The motivation is to divide the application into manageable chunks. User interface concerns are separated from data storage and computation concerns. As a result, both the components can be developed independently of each other.

- **Stateless interaction**: Each interaction with the web service need to be self-sufficient and contain all the required data. Client-side information like the session state can be stored only on the client itself. Thus, the web service is unaware of any past or future interactions with client. If some information needs to be persisted, a database needs to be used and some kind of authentication system needs to be adopted.

- **Layered system**: The client connects to an intermediate web service which then handles the request and proxies it to relevant service. This operation happens automatically without the client knowing anything about it. This improves system scalability and stability. The intermediate server can also be used for load balancing.

Thus, creating RESTful web services helps transcend the boundaries imposed by platforms and devices. The client-server model enables dynamic updates to the web application. The client-server communication is standardized, making the application inherently scalable and modular.

In MotionGen2.0, resource identification is done by assigning different URLs to independent operations like path synthesis, motion synthesis, and mixed synthesis. Use of HTTP CRUD operations ascertains a standard interface. The server can handle requests containing XML, JSON or plain text files. Each query contains sufficient information for the server to process it independently enabling the server to be stateless. A multi-layer server architecture is established to handle static content requests, computation intensive requests and progress queries separately. This enhances the scalability of overall system. More details about MotionGen2.0's server back-end and client front-end have been presented in next chapter.

6.2.3 Stack selection: MEAN

A Web stack refers to a group of software which is used in tandem for the purpose of web development. Some of the main components of a minimal web stack are an operating system, web server, database server, and programming language. LAMP, which stands for Linux, Apache, MySQL, and PHP, is one

traditionally used and well-established web stack. However, lately, newer stacks have gained usage over LAMP and offer many new advantages. MEAN is one such web stack which is of interest.

MEAN consists of JavaScript-based tools — MongoDB [79] (database system), Express.js [80] (backend web framework), AngularJS [81] (front end frame work), and Node.js [82] (runtime environment)— used to develop web applications. MEAN stack consists of all the basic ingredients needed to create a RESTful web service.

MEAN stack has been chosen to develop MotionGen2.0. It offers numerous advantages over other web stacks. Most important advantages it has are

- It lets the developer write the entire code in JavaScript, be it back-end or front-end.
- It provides a more flexible noSQL database than conventional relational databases.
- All the MEAN stack technologies are open source and available for free.

Each of the technology in MEAN stack and its use in MotionGen2.0 has been discussed in a later section.

6.2.4 Mobile App: Apache Cordova

Mobile operating systems have conventionally been app driven instead of browser-based technologies. To target mobile users, one needs to create a mobile app catered to each mobile platform like ios, android, etc.

One way is to create native apps for each platform. This requires a different codebase for each platform in their programming language of choice. For example, applications in Android are developed using Java while those on iOS requires Objective-C or Swift. Maintaining these native apps is time intensive and redundant as updating a feature requires equal work on each platform. To get over this disadvantage, MotionGen2.0 for mobile has been used using Apache Cordova [83] framework.

Apache Cordova is an open-source mobile development framework which uses standard web technologies like HTML [84], CSS [85] and JavaScript [86] for cross-platform development. It effectively eliminates programming for a specific mobile platform and replaces it with standardized web development.

Apache Cordova provides native application wrappers which run web-pages in the form of webViews. WebView can be thought of as emulated browsers within a native app. Thus, when a Apache Cordova application is launched, a WebView is loaded within the app and the code is executed on it. This is a huge advantage as it permits the use of web-based front-end code as the apps source code. This standardizes the development process and features can simultaneously be pushed to both web and mobile.

Apache Cordova began as PhoneGap [87], a mobile development framework created by Nitobi, and was later acquired by Adobe. Adobe contributed PhoneGap to Apache [88] to be developed as open source under the name Cordova. Currently, Adobe PhoneGap is the commercial version of Apache Cordova which Adobe offers with additional features like desktop installer and online compiler. Although these features are great to have, issues have been observed with PhoneGap's buggy integrating with Cordova every release cycle. In order to maintain stability and reliability, MotionGen 2.0 has been developed on Cordova and not PhoneGap.

Since apps developed on Apache Cordova are implemented through webViews, they require additional standards-compliant APIs to access device specific functions such as sensors, device data, etc. These plugins convert JavaScript functions into native functions. MotionGen 2.0 uses plugins for file access, camera, email, file open, file copy, file sharing, PDF export and GIF export [89–98].

6.2.5 User experience: Response Time Limits

Quick reactiveness of application to user interaction is of utmost importance to conserve workflow. As a result, one of the most important usability metric for any website is its speed. The time it takes a web application to respond to user input is measured as the response time.

As Nielsen [99] discusses, speed affects the usability of a website in two ways. Firstly, a user has a limited working memory where information quickly degrades if engagement is not continuous. Secondly, waiting for the website to respond does not play well with a user's psychological desire to have control over the machine. This leads to a situation when user detaches from his workflow and loses his focus.

This high sensitivity of users work efficiency to website response time was first looked into by Miller [100] and further explored by Nielsen. They propose three response-time limits established based on their experiences

which are as follows.

- **0.1 second** is the response-time limit for having the user feel that the application is reacting instantaneously. Thus, to keep the user engaged, display of results is sufficient and no additional feedback is necessary.
- **1.0 second** is the response-time limit for the user's flow of thought to stay uninterrupted. Even though the user notices this reaction delay, no special feedback is usually necessary to keep him engaged. However, the user does lose the feeling of working directly with the software and becomes aware of each interaction.
- **10 seconds** is the response-time limit for keeping the user's attention focused on the task. A feedback is mandatory to communicate expected completion time, especially when response time is variable. This gives user the opportunity to engage with other tasks while waiting for the computer to finish

Thus, application response time is paramount to conserving a user's workflow. To implement path and mixed synthesis in MotionGen 2.0, optimization subroutines are used which are inherently time-consuming. Efforts have been made to minimize this response time with a least possible trade-off on accuracy of the calculations. Also, the progress of server, for a computationally intensive task, is communicated to the user in real-time to keep him informed and engaged.

6.2.6 User Interface: Responsive web design

Responsive web design (RWD) is an approach to web design which makes web pages adapt and render well on a variety of devices and window or screen sizes. As MotionGen 2.0 is targeted to work on a multitude of devices, it has been developed using the principles of RWD. To make sure MotionGen 2.0 looks and function at its best, UI has been tweaked appropriately for large screens (desktop), medium screens (tablets) and small screens (phones).

6.2.7 User Interface: Principles of display design

MotionGen 2.0's interface has been designed keeping user interaction as its focus. The interface is intended to provide intuitive controls to the user through

the design process. Christopher Wickens [101, 102] defined 13 principles of display design in his book "An Introduction to Human Factors Engineering". MotionGen 2.0's User Interface(UI) has been designed by utilizing some of these principles to create an effective display design. These principles can be divided into four categories as

- **Perceptual principles**- Clear display on a variety of screen sizes while balancing similarity and redundancy of interactions and features.
- **Mental model principles**- Impress upon user's intuition by icon, animation and feature design
- **Principles based on attention**- Keeping the clutter to a minimum and providing contextual feature access to user
- **Memory principles**- Pro-active aiding based on user intent while allowing consistency between other CAD softwares.

Application of these principles makes a huge impact on user productivity.

Having discussed the guiding principles of application architecture, it is now time to discusses the implementation and functionality details related to MotionGen2.0.

6.3 Functionality

The new MotionGen2.0 features a totally new code-base and web-based architecture compared to old MotionGen, which was focused exclusively on mobile devices. Enormous new functionality in both domains, mechanism analysis and synthesis, have been added in MotionGen2.0. It can now simulate one and multi-degrees of freedom planar n-bar linkages. It also has path and mixed four- bar synthesis capabilities in addition to motion synthesis. These additions make MotionGen2.0 a cutting-edge platform for mechanism designers accessible easily through the web.

The analysis capability of MotionGen2.0 include simulation of

- Multi degree of freedom n-bar mechanisms having revolute joints with sequential multi-inputs.
- Single degree of freedom n-bar mechanisms having revolute joints.

- Single degree of freedom 4-bar mechanisms having prismatic or revolute joints.
- Serial chain mechanisms.

The synthesis capability of MotionGen2.0 include

- Motion Synthesis ($\geq$5+point/line constraints, Tolerance based)
- Path Synthesis ($\geq$5pts)
- Mixed Synthesis ($\geq$3pose+$\geq$2path)
- Serial coupled chain mechanisms Synthesis ($\geq$5pts)

The focus of this document is synthesis and architecture related enhancements offered in MotionGen2.0. For more details regarding algorithms used to synthesize mechanisms, refer Chapter 3,2,5. A detailed study of analysis related additions in MotionGen2.0 is available in [103]

Other MotionGen2.0 backbone functionalities, which enable the user to interact with analysis and synthesis algorithms are as follows

- Platform - Web, Appstore
- User interaction - Touch, mouse
- Input - Text, XML
- Output - Text, XML, Pdf, Gif
- Tracing - image, camera
- Sample example problems
- Workspaces for multiple simultaneous workflows

The figure below summarizes the capability of MotionGen2.0 in a nice graphical form.

All the code-base for MotionGen2.0 resides on the Server (also referred as back-end). The server sends user-facing code (also referred as front-end) on request by the user through the browser or app. Synthesis calculations are handled by the server while analysis computation is done locally on user's device. This delicate load-balance enables real-time simulation with scalable synthesis computation power. Details about technologies used in back-end and front-end are discussed in next sections.

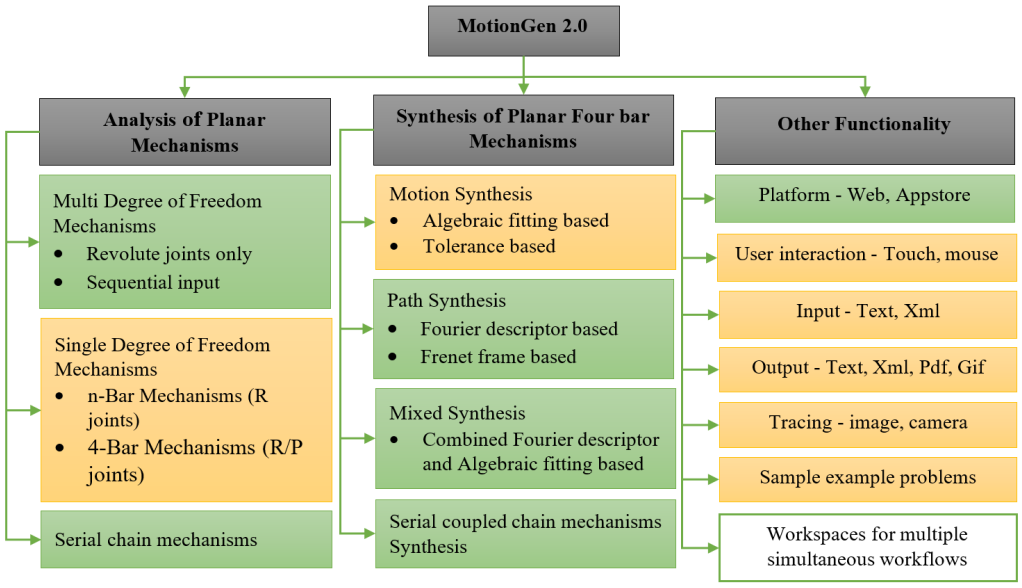


Figure 6.7: Functionality of MotionGen2.0 (Green: newly added, Yellow: updated, White: no change)

6.4 Back-end

6.4.1 Server Hardware

The hardware specification of server module are as follows

- Processor- 8x Intel Xeon CPU E5620 @ 2.40GHz
- Memory- 8161MB
- Storage- 2x ATA WDC WD1602ABKS-1 160GB, 1x ATA SAMSUNG HE161HJ 160GB

The storage devices have been setup in RAID configuration to mirror each other's data as a failsafe.

The software being used is Ubuntu Server 16.04.3 LTS which is an open source platform. It is the most used Linux server operating system because of its support and reliability.

6.4.2 Node.js

Node.js [82] is a server-side environment that allows Node developers to build servers and network applications with JavaScript for the first time. This means entire sites can be run on a unified JavaScript stack—both the client-side software and the server-side software.

It's an asynchronous, non-blocking, event-driven I/O system due to which, it can handle many requests concurrently. It's open source, cross-platform and actively developed. It is light-weight and efficient too.

MotionGen2.0 uses a two server setup. All client requests are handled by one server while the other is exclusively used for processing computationally complex and time-taking tasks.

The multi-threaded environment provided by multi-core CPU is being exploited in MotionGen2.0 using 'cluster' module of Node.js.

6.4.3 NPM

NPM [104] is the default package manager for Node.js platform. This centralized repo can be used to find the best framework and packages for the task at hand. Some of the packages being used in MotionGen2.0 are as follows

- **express** [105] is a web application framework that provides you with a simple API to build websites, web apps and backends.
- **http-proxy** [106] is a HTTP programmable proxying library that supports websockets.
- **socket.io** [107] is a framework which enables real-time bidirectional event-based communication between server and client.
- **socket.io-redis** [108] is a library which helps run multiple socket.io instances that can all broadcast and emit events to and from each other.
- **socket.io-emitter** [109] is a library which allows communication with socket.io servers easily from non-socket.io processes.
- **mongodb** [110] is the driver API which connects the server to Mongo daemon and database.

- **pm2** [111] is a production Runtime and Process Manager for Node.js apps with a built-in Load Balancer.
- **nodemon** [112] is a utility that will monitor for any changes in your source and automatically restart your server. Process manager helps to keep the application alive forever, restart on failure, reload without downtime and simplifies administrating.
- **fmin** [113] is a math library used to implement Nelder Mead optimization.
- **numeric** [114] is a math library used to compute SVD of a matrix. SVD for a complex system of equations is done by reducing it in higher dimensional real system [26].
- **mathjs** [115] is a well-documented math library used for matrix and complex number manipulation.
- **body-parser** [116] is a package which parses the body content with POST HTTP requests.
- **http-auth** [117] is a package which enables client-side authentication requirement
- **xmldom** [118] is a package which enables .xml related functionality
- **cordova** [119] is a framework which repackages the front-end into a mobile app.

6.4.4 Mongodb

MongoDB [79] is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas.

In MotionGen2.0, it is used to store server progress on computationally heavy tasks and store synthesis data to be used in future for machine learning purposes.

6.4.5 Back-end Architecture

The back-end adopts a multi layered architecture with various servers simultaneously running each dedicated to a specific task. Figure 6.8 visualizes how each of these components work in tandem to serve MotionGen2.0 from web.

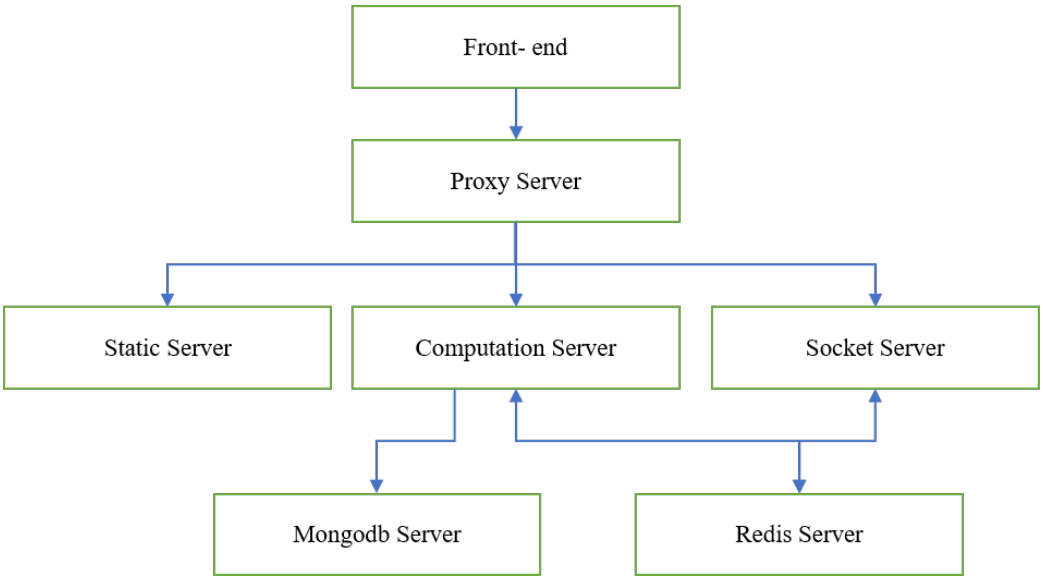


Figure 6.8: Back-end Architecture

The individual components of back-end architecture and their intended task are

- **Frontend** is the part of codebase served to the user which resides on their device. It is tasked with accessing server resources according to user requirements.
- **Proxy Server** is the server which handles all requests to the website and proxies them to the relevant route.
- **Static Server** is the server which hosts Frontend code and serves it to user device on request to website domain.
- **Computation Server** is the server which process all computationally tedious tasks including path, motion and mixed synthesis. It runs in a multi-threaded environment to take advantage of multi-core processing.

- **Socket Server** is the server which process websocket requests to enable realtime communication between server and clients.
- **Redis Server** is the server which acts as central communication hub between multithreaded cluster on Computation Server and single threaded Socket Server.
- **Mongodb Server** is the database being used to store the synthesis information being calculated on computation server.

6.5 Front-end

Front-end includes all the Static web content including HTML, CSS and js libraries which are served to the user. This is the part of code which actually resides on the user's device once it is transmitted from the server. Analysis computations are done on the front-end as the numeric complexity is lesser. This also subverts the movement of data to and fro from the server, making the process almost instantaneous and real-time.

Some of the libraries being used on the front end are as follows

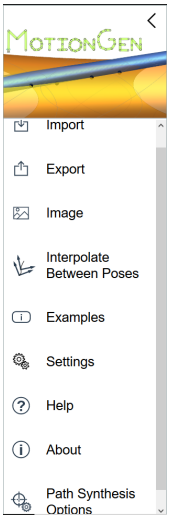
- **hammer.js** [120] is the library which enables simultaneous handling of touch and mouse based events on the front-end.
- **jquery.js** [121] is a library which simplifies how to traverse HTML documents, handle events, perform animations, and AJAX. All HTTP requests to the server are made using jquery.
- **Numeric.js** [114], **Math.js** [115], **Quartic.js** [122], **Decimal.js** [123] and **Complex.js** [124] act as the math engine and carry out all front-end calculations.

6.5.1 UI

The user interface for MotionGen2.0 has been designed to enhance user workflow. Contextual options have been added and less frequented options have been moved to side menu. Effectively, this has reduced display clutter drastically and the user can now focus exclusively on the task of designing or analyzing the mechanism. The core menus of MotionGen2.0 are as follows

- **Top menu** houses the contextual bar and undo, redo, delete buttons.
- **Options menu** consist of three-tier menu for Linkage control, Synthesis, and Analysis. Each tier can easily be accessed by selecting any of its other member buttons. Long pressing buttons can make extra options available.
- **Animation menu** is tasked with controlling the motion of mechanism drawn or synthesized.
- **Dyad menu** consists of a variety of dyads if multiple solutions for a synthesis problem exists.
- **Side menu** contains all the other advanced functionality, neatly accessible a click away. It has menus for import, export, image, examples, settings and path synthesis options.

Fig. 6.9a and Fig. 6.9b shows how each of the mentioned menus actually looks on MotionGen 2.0.



(a) Side Menu

(a) Top Menu / Information Bar

(b) Dyad Menu

(c) Options Menu

(d) Animation Menu

(b) Dashboard Menus

Figure 6.9: MotionGen2.0 Menus

6.5.2 Front-end Architecture

The front-end adopts an MVC architecture with clear demarcation between input, processing and output tasks. Figure 6.10 visualizes how each of these components work together to display MotionGen2.0 and make it work on users device.

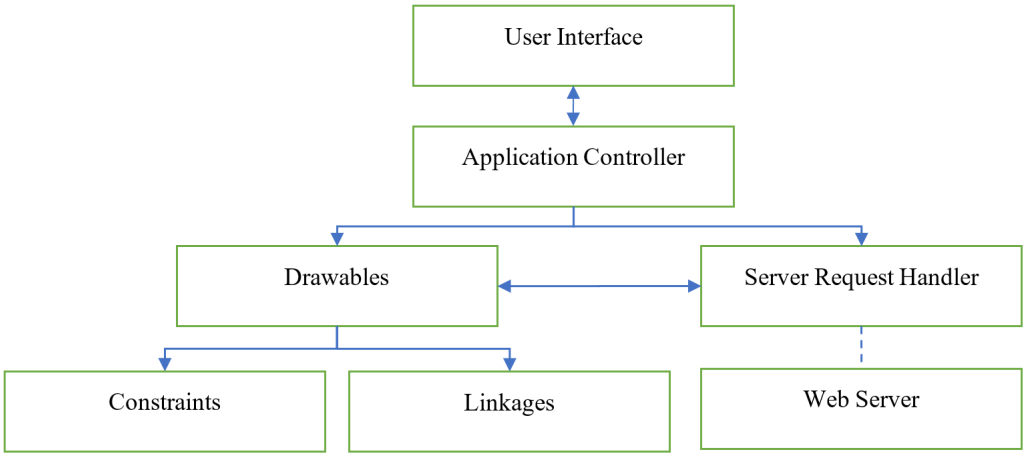


Figure 6.10: Front-end Architecture

The individual components of front-end and their function are

- **User Interface** consists of all the HTML content including the multi-layered canvas, menus, buttons, etc. Any input on the canvas layer triggers the Application Controller.
- **Application Controller** is the prominent communication hub of MotionGen 2.0. It listens for user input and responds by sending appropriate data to Drawables object or Server Request Handler object. It contains all event handlers and is the main entry point into the program.
- **Drawables** is an abstract class which stores the state of Client session at any given time. It responds to the Application controller and assigns information to Constraints or Linkage appropriately.

- **Server Request Handler** is an abstract class which deals with transfering computationally intensive tasks to web server. It works asynchronously with the server which is in contrast with the synchronous nature of most of the other operations in the application.
- **Constraints** is a class which consists of all the synthesis constraint info including Poses, Path points, Lines and Points. It assigns information based on the instructions that it receives from the Drawables class.
- **Linkages** is a class which consists of all the analysis data like the Links, Joints, Curves, Input Rpm's, etc. It also assigns information based on the instructions that it receives from the Drawables class.

6.6 Case study

This section presents a step by step process which the user can follow to synthesize his own mechanisms. Motion, Path, and Hybrid Synthesis workflow have been demonstrated.

6.6.1 Motion Synthesis workflow

Follow the following workflow to synthesize motion using >5 poses.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- Select 'Add Pose' button under Synthesis group in Options Menu.
- Tap on the screen to add a pose
- While adding, touch and drag to set the required orientation of the pose
- Repeat until the all poses have been inputted. Once 5 poses are inputted, MotionGen2.0 automatically synthesizes four bar in real-time.
- Update, delete or add poses as necessary according to the problem constraints.

Motion Synthesis is also possible using 3 or 4 poses. The user just needs to select additional line or point constraints to get the solution mechanism. The user has the ability to select is a mechanism should follow a subset of poses exactly and approximate others. An example of Motion Synthesis on MotionGen 2.0 is shown in Fig. 6.11.

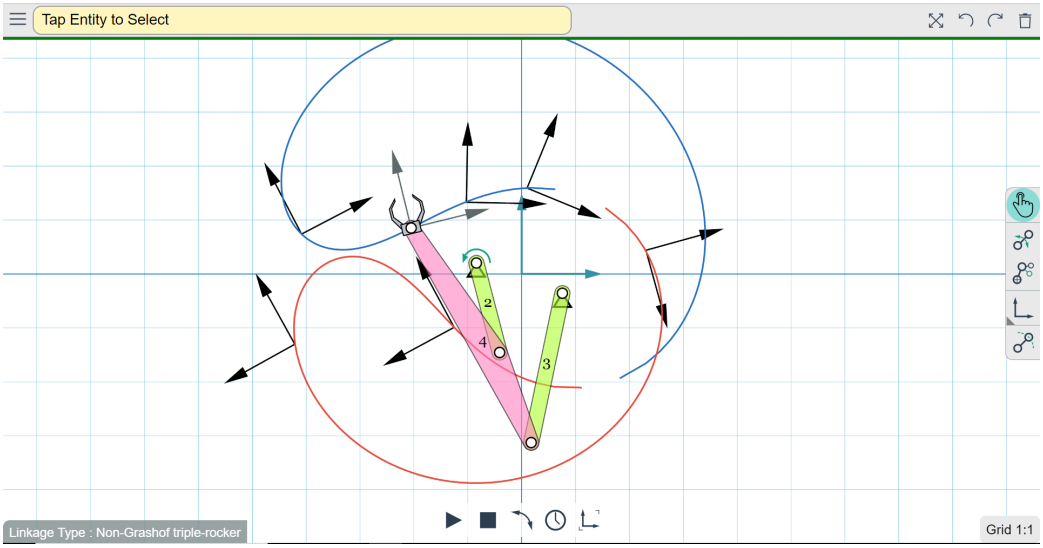


Figure 6.11: Instance of Motion Synthesis on MotionGen 2.0

6.6.2 Path Synthesis workflow

Follow the following workflow to synthesize path using >5 path points.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- Long press 'Add Pose' button under Synthesis group in Options Menu.
- When the 'Add Path point' button is displayed, select it
- Tap on the screen to add a pose
- While adding, touch and drag to set the path point to the exact required position.

- Repeat until all poses have been inputted. Once 5 poses are inputted, MotionGen2.0 automatically synthesizes a Single degree of freedom coupled mechanism. This is the default synthesis option selected
- Go into the side menu and access 'Path Synthesis Options'
- In the pop-up dialog box, select local optimization or global optimization for Fourier descriptor based synthesis. Select Fourier curve or B-spline curve for Frenet frame based path synthesis.
- NOTE: The path synthesis is not instantaneous and can take the server upto 10 seconds to process your request.
- Update, delete or add poses as necessary according to the problem constraints.

Other advanced options like harmonic count, subdivision recursion, and parameter control can be controlled from the 'Path Synthesis Options' popup box. These options grant the user enhanced flexibility and control over the synthesis process. An example of Path Synthesis on MotionGen 2.0 is shown in Fig. 6.12.

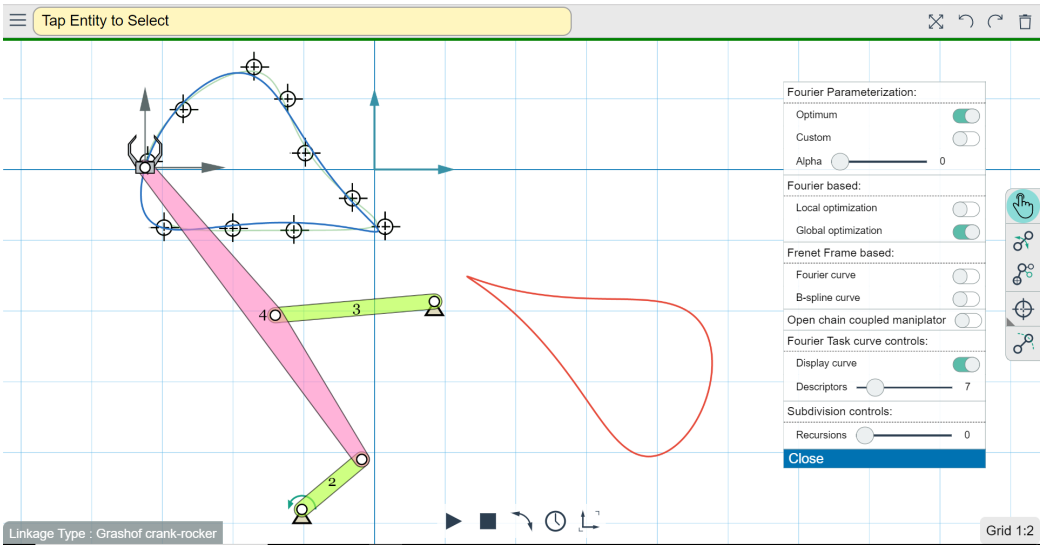


Figure 6.12: Instance of Path Synthesis on MotionGen 2.0

6.6.3 Mixed Synthesis workflow

Follow the following workflow to synthesize mechanism using >2 path points and >3poses.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- If you want to input a path point, Long press 'Add Pose' button under Synthesis group in Options Menu and then select 'Add Path point' button. Then tap on the workspace.
- Similarly 'Add Pose' can be selected
- Note: The order in which poses and paths are inputted decides the Fourier descriptors of task path.
- Repeat until all poses have been inputted. Once the minimum number of required constraints are inputted, the server will automatically compute the mechanism and display it onscreen.
- NOTE: The mixed synthesis is not instantaneous and can take the server upto 10 seconds to process your request.
- Update, delete or add poses or path points as necessary according to the problem constraints.

The server automatically uses the modified re-parametrization to find smoothest path curve through the inputted constraints. The user has the flexibility of order in which he inputs poses and path points. This is very similar to practical situations where the user knows just some intermediate orientations. An example of Mixed Synthesis on MotionGen 2.0 is shown in Fig. 6.13.

This completes the overview of MotionGen 2.0

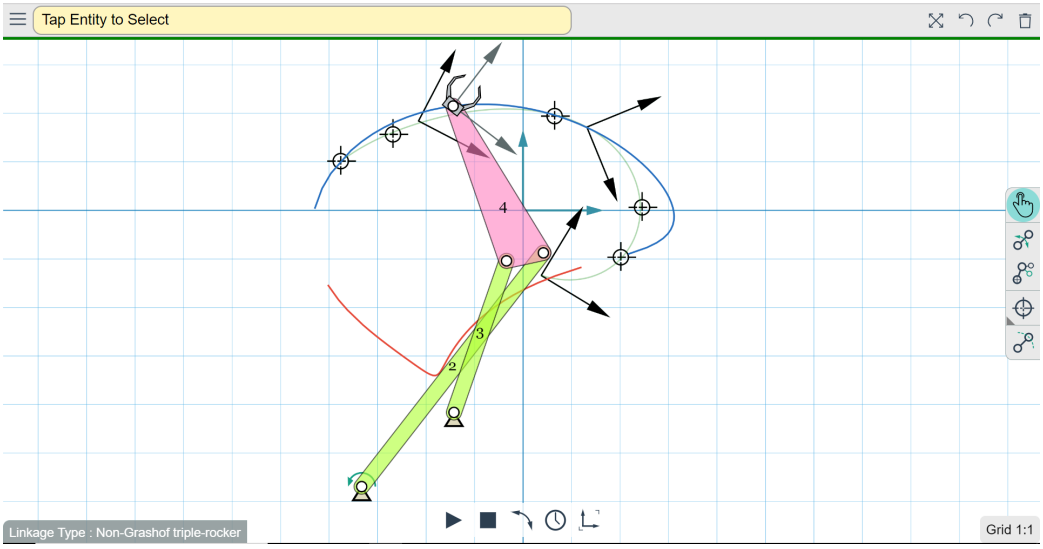


Figure 6.13: Instance of Mixed Synthesis on MotionGen 2.0

Chapter 7

Conclusion and Proposed Work

This thesis claims the following original contributions

- A Fourier based mixed synthesis approach which unifies path and motion synthesis for planar mechanisms.
- An non-uniform parameterization scheme to optimize Fourier parameters for planar path synthesis.
- A kinematic synthesis based approach for synthesis of spherical mechanisms for a motion synthesis problem.
- A point-line-plane representation for unified simulation of planar and spherical mechanisms.
- MotionGen2.0, a web-based application capable of performing motion, path, and mixed synthesis for planar mechanisms.

7.1 Proposed Work related to Mixed Synthesis

The path-orientation relationship derived for mixed synthesis can be used in pure over-constrained motion approximation problem to interactively guide the user towards a better mechanism. Calculation of best pose orientation would be possible and this flexibility would enhance the design process. Preliminary work on interactive tolerance-based motion synthesis approach has been done. Fig. 7.1 and Fig. 7.2 shows an example of seven pose motion

generation problem which uses the prediction model to generate a defect free mechanism.

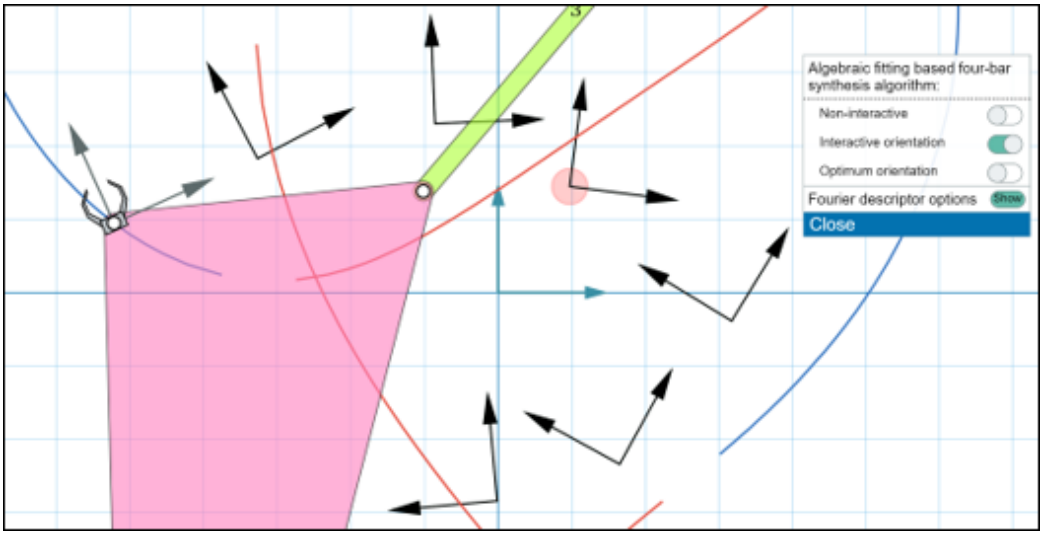


Figure 7.1: Prediction of pose whose orientation change would have maximum effect on solution quality

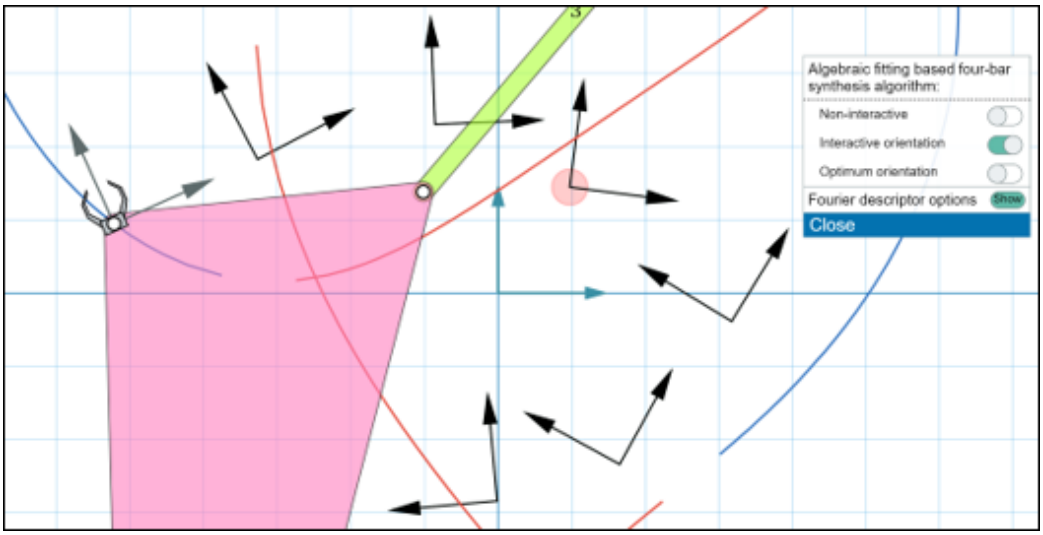


Figure 7.2: Prediction of new orientations for specified pose to generate better solution mechanism

Extension of the mixed synthesis framework to include function generation will also be investigated. This would unify the three conventional methodologies into a unified framework.

7.2 Proposed Work related to Path Synthesis

The existing Fourier based path synthesis algorithm is limited to a RRRR four-bar mechanism. Extending it to other types of four-bars like RRPR, RRRP or RRPP would be investigated.

7.3 Proposed Work related to Motion Synthesis

We would like to unify the motion synthesis framework for spherical and planar mechanisms. This would greatly improve the applicability of our algorithm and also bring together two fundamental synthesis problems.

Extension of proposed spherical synthesis framework to generate spherical six-bar mechanisms will also be investigated.

Finally, constraint based synthesis of spatial mechanisms will be investigated. There exist algorithms which incorporate sphere and plane constraints to generate SS,RRS, and RPS dyads. We would like to explore spatial line, circle and cylindrical constraints which would represent PS, RS, and CS dyads. This should increase the number of feasible solutions generated for a design problem.

7.4 Proposed Work related to Simulation

The proposed simulation framework is limited to inputs from the ground link only. Extension to actuation at any joint and also multiple inputs would make the algorithm powerful enough to handle all possible single and multi-DOF planar and spherical mechanisms.

Generalization of proposed constraint based framework to simulation of spatial mechanisms will also be investigated.

7.5 Proposed Work related to MotionGen 2.0

The current iteration of MotionGen is able to simulate n-bar mechanisms with revolute joints only. Implementing the proposed simulation algorithm will enable the software to handle the prismatic joints accurately.

Also, MotionGen does not support analysis or synthesis of spherical mechanisms at present. We would like to build these capabilities into the software too and will look into it.

Bibliography

[1] Erdman, A. G. and Sandor, G. N., 1997, Mechanism Design: Analysis and Synthesis, Prentice Hall, NJ, 3rd edition.

[2] Tong, Y., Myszka, D. H., and Murray, A. P., 2013, "Four-Bar Linkage Synthesis for a Combination of Motion and Path-Point Generation", 10.1115/DETC2013-12969.

[3] Alt, H., 1923, "Uber die Erzeugung gegebener ebener Kurven mit Hilfe des Gelenkvierecks", ZAMM, **3(1)**, pp. 13–19.

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

[5] Brake, D. A., Hauenstein, J. D., Murray, A. P., Myszka, D. H., and Wampler, C. W., 2016, "The Complete Solution of Alt–Burmester Synthesis Problems for Four-Bar Linkages", Journal of Mechanisms and Robotics, **8(4)**, pp. 041018–041018–8, 10.1115/1.4033251.

[6] Zimmerman, A. R. I., 2018, "Planar Linkage Synthesis for Mixed Motion, Path, and Function Generation Using Poles and Rotation Angles", ASME Journal of Mechanisms and Robotics, **10(2)**, pp. 025004–025004–8, 10.1115/1.4039064.

[7] Purwar, A., Deshpande, S., and Ge, Q. J., 2017, "MotionGen: Interactive Design and Editing of Planar Four-Bar Motions via a Unified Framework for Generating Pose- and Geometric-Constraints", ASME Journal of Mechanisms and Robotics, 10.1115/1.4035899.

[8] Deshpande, S. and Purwar, A., 2017, "A Task-driven Approach to Optimal Synthesis of Planar Four-bar Linkages for Extended

Burmester Problem", ASME Journal of Mechanisms and Robotics, 10.1115/1.4037801.

[9] Zhao, P., Li, X., Purwar, A., and Ge, Q. J., 2016, "A Task-Driven Unified Synthesis of Planar Four-Bar and Six-Bar Linkages With R- and P-Joints for Five-Position Realization", ASME Journal of Mechanisms and Robotics, **8(6)**, pp. 061003–061003–8, 10.1115/1.4033434.

[10] Ge, Q. J., Purwar, A., Zhao, P., and Deshpande, S., 2016, "A Task Driven Approach to Unified Synthesis of Planar Four-bar Linkages using Algebraic Fitting of a Pencil of G-manifolds", ASME Journal of Computing and Information Science in Engineering, 10.1115/1.4035528.

[11] Ge, Q. J., Zhao, P., Purwar, A., and Li, X., 2012, "A Novel Approach to Algebraic Fitting of a Pencil of Quadrics for Planar 4R Motion Synthesis", ASME Journal of Computing and Information Science in Engineering, **12**, pp. 041003–041003.

[12] Chu, J. and Cao, W.-q., 1993, "Synthesis of coupler curves of planar four-bar linkages through fast fourier transform", Chin. J. Mech. Eng, **29(5)**, pp. 117–122.

[13] Ullah, I. and Kota, S., 1997, "Optimal synthesis of mechanisms for path generation using Fourier descriptors and global search methods", Journal of Mechanical Design, **119(4)**, pp. 504–510.

[14] Wu, J., Ge, Q. J., Gao, F., and Guo, W. Z., 2011, "On the Extension of a Fourier Descriptor Based Method for Planar Four-Bar Linkage Synthesis for Generation of Open and Closed Paths", Journal of Mechanisms and Robotics-Transactions of the Asme, **3(3)**.

[15] Wu, J., Ge, Q. J., and Gao, F., 2009, "An Efficient Method for Synthesizing Crank-Rocker Mechanisms for Generating Low Harmonic Curves", ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 2009, dETC2009-87140.

[16] Li, X., Wu, J., and Ge, Q. J., 2016, "A Fourier Descriptor-Based Approach to Design Space Decomposition for Planar Motion Approxima-

tion", Journal of Mechanisms and Robotics, **8(6)**, pp. 064501–064501–5, 10.1115/1.4033528.

[17] Yue, C., Su, H.-J., and Ge, Q., 2011, "Path Generator via the Type-P Fourier Descriptor for Open Curves", Proceedings of 13th World Congress in Mechanism and Machine Science.

[18] Yue, C., Su, H.-J., and Ge, Q. J., 2012, "A hybrid computer-aided linkage design system for tracing open and closed planar curves", Computer-Aided Design.

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

[20] Chu, J. K. and Sun, J. W., 2010, "A New Approach to Dimension Synthesis of Spatial Four-Bar Linkage Through Numerical Atlas Method", Journal of Mechanisms and Robotics-Transactions of the Asme, **2(4)**.

[21] Chu, J. and Sun, J., 2010, "A new approach to dimension synthesis of spatial four-bar linkage through numerical atlas method", Journal of Mechanisms and Robotics, **2(4)**, p. 041004.

[22] Krovi, V., Ananthasuresh, G., and Kumar, V., 2002, "Kinematic and kinetostatic synthesis of planar coupled serial chain mechanisms", Journal of Mechanical Design, **124(2)**, pp. 301–312.

[23] Nie, X. and Krovi, V., 2005, "Fourier methods for kinematic synthesis of coupled serial chain mechanisms", Journal of Mechanical Design, **127(2)**, pp. 232–241.

[24] Sharma, S. and Purwar, A., "Optimal non-uniform parameterization scheme for fourier descriptor based path synthesis of four bar mechanisms", ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,, American Society of Mechanical Engineers, volume 2018, dETC2018-85567.

[25] Freudenstein, F., 1959, "Harmonic Analysis of Crank-and-Rocker Mechanisms With Application", ASME J. Appl. Mech, **26**, p. 673–675.

[26] Day, D. and Heroux, M. A., 2001, "Solving complex valued linear systems via equivalent real formulations", SIAM Journal on Scientific Computing, **23(2)**, p. 480–498.

[27] Bottema, O. and Roth, B., 1979, Theoretical Kinematics, Dover Publication Inc., New York.

[28] McCarthy, J. M., 1990, Introduction to Theoretical Kinematics, The MIT Press, Cambridge, MA.

[29] Golub, G. and Van Loan, C., 1996, Matrix Computations, Johns Hopkins Univ Press, Baltimore, MD.

[30] Erdman, A. G. and Sandor, G. N., 1991, Advanced Mechanism Design: Analysis and Synthesis, volume 2, Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.

[31] Nolle, H. and Hunt, K. H., 1971, "Optimum Synthesis of Planar Linkages to Generate Coupler Curves", Journal of Mechanisms, **6(3)**, p. 267.

[32] Mcgarva, J. and Mullineux, G., 1993, "Harmonic Representation of Closed Curves", Applied Mathematical Modelling, **17(4)**, pp. 213–218.

[33] Wu, J., Ge, Q. J., Gao, F., and Guo, W. Z., 2010, "On the extension of a fourier descriptor based method for four-bar linkage synthesis for generation of open and closed paths", 2010 ASME Mechanisms and Robotics Conference, dETC2010–29028.

[34] Li, X., Zhong, X., and Ge, Q., 2015, "Parametrization-Independent Non-Uniform Fourier Approach to Path Synthesis of Four-Bar Mechanism", Proceedings of the 14th IFToMM World Congress, pp. 440–448.

[35] Li, X. Y. and Chen, P., 2017, "A Parametrization-Invariant Fourier Approach to Planar Linkage Synthesis for Path Generation", Mathematical Problems in Engineering, **2017**.

[36] McCarthy, J. M. and Soh, G. S., 2010, Geometric design of linkages, volume 11, Springer.

[37] Chiang, C. H., 2000, Kinematics of spherical mechanisms, Krieger Pub., Malabar, FL, 00039067 C.H. Chiang. ill. ; 23 cm. Includes bibliographical references and index.

[38] Sandor, G. N. and Erdman, A. G., 1997, Advanced Mechanism Design: Analysis and Synthesis Vol. 2, Prentice-Hall, Englewood Cliffs, NJ.

[39] Bodduluri, R. M. C. and McCarthy, J. M., 1992, "Finite position synthesis using image curve of a spherical four-bar motion", ASME J. of Mechanical Design, **114(1)**.

[40] Lin, C.-C., 1998, "Complete solution of the five-position synthesis for spherical four-bar mechanisms", Journal of marine science and Technology, **6(1)**, pp. 17–27.

[41] Ge, Q. J. and Larochelle, P., 1999, "Algebraic motion approximation with NURBS motions and its application to spherical mechanism synthesis", ASME Journal of Mechanical Design, **121(4)**, pp. 529–532.

[42] Ruth, D. and McCarthy, J., 1999, "The design of spherical 4R linkages for four specified orientations", Mechanism and Machine Theory, **34(5)**, pp. 677 – 692, doi:https://doi.org/10.1016/S0094-114X(98)00048-2, URL http://www.sciencedirect.com/science/article/pii/S0094114X98000482.

[43] Brunnthaler, K., Schrocker, H., and Husty, M., 2006, Synthesis of spherical four-bar mechanisms using spherical kinematic mapping, Advances in Robot Kinematics, Springer, Netherlands.

[44] Zhuang, Y., Zhang, Y., and Duan, X., 2015, "Complete real solution of the five-orientation motion generation problem for a spherical four-bar linkage", Chinese Journal of Mechanical Engineering, **28(2)**, pp. 258–266, doi:10.3901/CJME.2015.0105.003, URL https://doi.org/10.3901/CJME.2015.0105.003.

[45] Li, X., Zhao, P., Purwar, A., and Ge, Q., 2018, "A Unified Approach to Exact and Approximate Motion Synthesis of Spherical Four-Bar Linkages Via Kinematic Mapping", ASME Journal of Mechanisms and Robotics, **10(1)**, p. 011003.

[46] Ravani, B. and Roth, B., 1983, "Motion Synthesis Using Kinematic Mappings", Journal of Mechanisms Transmissions and Automation in Design-Transactions of the Asme, **105(3)**, pp. 460–467.

[47] Zhao, P., Ge, X., Zi, B., and Ge, Q., 2016, "Planar linkage synthesis for mixed exact and approximated motion realization via kinematic mapping", Journal of Mechanisms and Robotics, **8(5)**, p. 051004.

[48] Deshpande, S. and Purwar, A., 2017, "A task-driven approach to optimal synthesis of planar four-bar linkages for extended burmester problem", ASME Journal of Mechanisms and Robotics, **9(6)**, p. 061005.

[49] Norton, R., 2011, Design of Machinery: An Introduction To The Synthesis and Analysis of Mechanisms and Machines, McGraw Hill, 5th edition.

[50] Erdman, A. G. and Sandor, G. N., 1991, Mechanism Design: Analysis and Synthesis, volume 1, Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.

[51] Rector, D., "Linkage", URL http://blog.rectorsquid.com/linkage-mechanism-designer-and-simulator/.

[52] Campbell, M., "Planar Mechanism Kinematic Simulator", URL http://design.engr.oregonstate.edu/pmksintro.html.

[53] Waldron, K. and Sreenivasan, S., 1996, "A study of the solvability of the position problem for multi-circuit mechanisms by way of example of the double butterfly linkage", Journal of Mechanical design, **118(3)**, pp. 390–395.

[54] Nielsen, J. and Roth, B., 1999, "On the Kinematic Analysis of Robotic Mechanisms", The International Journal of Robotics Research, **18(12)**, pp. 1147–1160, doi:10.1177/02783649922067771, URL https://doi.org/10.1177/02783649922067771.

[55] Wampler, C. W., 1999, "Solving the kinematics of planar mechanisms", Journal of Mechanical Design, **121(3)**, pp. 387–391.

[56] Nielsen, J. and Roth, B., 1999, "Solving the input/output problem for planar mechanisms", Journal of Mechanical Design, **121(2)**, pp. 206–211.

[57] Raghavan, M. and Roth, B., 1995, "Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators", Journal of Mechanical Design, **117(B)**, pp. 71–79.

[58] De Jalon, J. G. and Bayo, E., 2012, Kinematic and dynamic simulation of multibody systems: the real-time challenge, Springer Science & Business Media.

[59] Nikravesh, P. E., 1988, Computer-aided analysis of mechanical systems, volume 186, Prentice-hall Englewood Cliffs, NJ.

[60] Kreyszig, E., 2007, Advanced engineering mathematics, John Wiley & Sons.

[61] Hernández, A. and Petuya, V., 2004, "Position analysis of planar mechanisms with R-pairs using a geometrical–iterative method", Mechanism and machine theory, **39(2)**, pp. 133–152.

[62] Radhakrishnan, P. and Campbell, M. I., 2012, "An Automated Kinematic Analysis Tool for Computationally Synthesizing Planar Mechanisms", ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1553–1562.

[63] Inc., A., 2016, "Autodesk Inventor", URL http://www.autodesk.com/products/inventor/overview.

[64] Dassault Systems, "SolidWorks Software", http://www.solidworks.com, accessed Jan 27, 2014.

[65] MSCsoftware, 2015, "Adams/Solver Help - DOC10902, https://simcompanion.mscsoftware.com/infocenter/index?page=content&id=DOC10902&actp=RSS", .

[66] Software, M., "Adams, http://www.mscsoftware.com/product/adams", URL http://www.mscsoftware.com/product/adams.

[67] Artas Engineering, "SAM (Synthesis and Analysis of Mechanisms)", URL http://www.artas.nl/en.

[68] Petuya, V., Macho, E., Altuzarra, O., and Pinto, C., 2011, "Educational Software Tools for the Kinematic Analysis of Mechanisms", Comp. Appl. Eng. Education, **6(4)**, pp. 261–266.

[69] Furlong, T. J., Vance, J. M., and Larochelle, P. M., 1999, "Spherical mechanism synthesis in virtual reality", ASME Journal of Mechanical Design, **121(4)**, pp. 515–520.

[70] Larochelle, P., Dooley, J., Murray, A., and McCarthy, J. M., 1993, "SPHINX: Software for Synthesizing Spherical 4R Mechanisms", Proc. of the 1993 NSF Design and Manufacturing Systems Conference, volume 1, pp. 607–611.

[71] Ruth, D. and McCarthy, J., 1997, "Sphinxpc: An implementation of four position synthesis for planar and spherical 4r linkages", ASME Design Engineering Technical Conferences.

[72] Furlong, T., Vance, J., and Larochelle, P., 1999, "Spherical Mechanism Synthesis in Virtual Reality", ASME Journal of Mechanical Design, **121(4)**, pp. 515–520.

[73] Tse, D. and Larochelle, P., 1999, "Osiris: a new generation spherical and spatial mechanism cad program", Florida Conference on Recent Advancements in Robotics.

[74] Sharma, S., Purwar, A., and Ge, Q. J., 2019, "An Optimal Parametrization Scheme for Path Generation Using Fourier Descriptors for Four-Bar Mechanism Synthesis", ASME Journal of Computing and Information Science in Engineering, **19(1)**, p. 014501.

[75] Sharma, S., Purwar, A., and Ge, Q. J., 2019, "A Motion Synthesis Approach to Solving Alt-Burmester Problem by Exploiting Fourier Descriptor Relationship Between Path and Orientation Data", ASME Journal of Mechanisms and Robotics, **11(1)**, p. 011016.

[76] Li, X., Ge, X., Purwar, A., and Ge, Q. J., 2015, "A Unified Algorithm for Analysis and Simulation of Planar Four-Bar Motions Defined With R- and P-Joints", ASME Journal of Mechanisms and Robotics, **7(1)**, pp. 011014–011014–7, 10.1115/1.4029295.

[77] Inc, A., 2008, "Force Effect Motion", .

[78] Fielding, R. T. and Taylor, R. N., 2000, Architectural styles and the design of network-based software architectures, University of California, Irvine Doctoral dissertation.

[79] MongoDB, Inc., "Mongodb", URL https://www.mongodb.com/.

[80] The Nodejs Foundation, "Express.js", URL https://expressjs.com/.

[81] Google, "Angular.js", URL https://angularjs.org/.

[82] The Nodejs Foundation, "Node.js", URL https://nodejs.org/en/about/.

[83] Foundation, T. A. S., "Cordova", URL http://cordova.apache.org/docs/en/5.0.0/guide_overview_index.md.html#Overview.

[84] W3C, "HTML 5", URL http://www.w3.org/TR/html51/.

[85] W3C, "CSS 3", URL http://www.w3.org/Style/CSS/.

[86] (W3C), T. W. W. W. C., "JavaScript, http://www.w3.org/standards/webdesign/script", URL http://www.w3.org/standards/webdesign/script.

[87] Adobe Systems Inc, "PhoneGap", URL http://phonegap.com.

[88] Foundation, T. A. S., "ASF", URL http://www.apache.org/foundation/.

[89] Apache, "cordova-plugin-file", URL https://github.com/apache/cordova-plugin-file.

[90] Apache, "cordova-plugin-camera", URL https://github.com/apache/cordova-plugin-camera.

[91] Katzer, "cordova-plugin-email-composer", URL https://github.com/katzer/cordova-plugin-email-composer.

[92] Pwlin, "cordova-plugin-file-opener2", URL https://github.com/pwlin/cordova-plugin-file-opener2.

[93] Chaudhary, G., "Asset2SD", URL https://github.com/gkcgautam/Asset2SD.

[94] MrRio, "jsPDF", URL https://github.com/MrRio/jsPDF.

[95] MrRio, "jsPDF Image Plugin", URL https://github.com/MrRio/jsPDF/blob/master/plugins/addimage.js.

[96] W3C, "File API", URL http://www.w3.org/TR/FileAPI/.

[97] Verbruggen, E., "SocialSharing-PhoneGap-Plugin", URL https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin.

[98] Kwok, K., "jsgif", URL https://github.com/antimatter15/jsgif.

[99] Group, N. N., 1993, "Response Times: The 3 Important Limits", URL https://www.nngroup.com/articles/response-times-3-important-limits/.

[100] Miller, R. B., 1968, "Response time in man-computer conversational transactions", Proceedings of the December 9-11, 1968, fall joint computer conference, part I, ACM, pp. 267–277.

[101] Wickens, C., Lee, J., Liu, Y., and Becker, S., 2004, An Introduction to Human Factors Engineering. Second ed., Pearson Prentice Hall.

[102] Wikipedia, 2017, "Human-computer interaction — Wikipedia, The Free Encyclopedia", URL https://en.wikipedia.org/w/index.php?title=Human%E2%80%93computer_interaction&oldid=791784316, [Online].

[103] Lodha, A., 2017, "Web-Based Design and Simulation of One- and Multi-Degrees of Freedom Planar n-Bar Linkages", .

[104] npm Inc., "npm", URL https://www.npmjs.com/about.

[105] TJ Holowaychuk, C., "Express", URL github.com/expressjs/express.

[106] Charlie Robbins, C., Jarrett Cruger, "Http-proxy", URL github.com/nodejitsu/node-http-proxy.

[107] darrachequesne, C., "Socket.io", URL github.com/socketio/socket.io.

[108] darrachequesne, C., "socket.io-redis", URL github.com/socketio/socket.io-redis.

[109] darrachequesne, C., "socket.io-emitter", URL github.com/socketio/socket.io-emitter.

[110] mbroadst, C., "MongoDB driver", URL github.com/mongodb/node-mongodb-native.

[111] tknew, C., "pm2", URL github.com/Unitech/pm2.

[112] remy, C., "nodemon", URL github.com/remy/nodemon.

[113] benfrederickson, "fmin", URL github.com/benfred/fmin.

[114] Loisel, S., "Numeric Javascript", URL http://www.numericjs.com/.

[115] de Jong, J., "Math.js", URL http://mathjs.org/index.html.

[116] dougwilson, C., "body-parser", URL github.com/expressjs/body-parser.

[117] gevorg, C., "http-auth", URL github.com/http-auth/http-auth.

[118] jindw, C., "xmldom", URL github.com/jindw/xmldom.

[119] stevegill, C., "cordova", URL github.com/apache/cordova-cli.

[120] Tangelder, J., "Hammer JS, http://hammerjs.github.io", .

[121] The jQuery Foundation, "jQuery, http://jquery.com", .

[122] Lathoud, G., "JS Quartic", URL https://github.com/glathoud/js.quartic/blob/master/solve_quartic.js.

[123] Mclaughlin, M., "Decimal JS", URL http://mikemcl.github.io/decimal.js/.

[124] Lathoud, G., "Complex", URL https://github.com/glathoud/js.quartic/blob/master/complex.js.