Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-98194

USING A POINT-LINE-PLANE REPRESENTATION FOR UNIFIED SIMULATION OF PLANAR AND SPHERICAL MECHANISMS

Shashank Sharma, Anurag Purwar*
Computer-Aided Design and Innovation Lab
Department of Mechanical Engineering
Stony Brook University
Stony Brook, New York, 11794-2300

ABSTRACT

This paper presents a geometric constraints driven approach to unified kinematic simulation of n-bar planar and spherical linkage mechanisms consisting of both revolute and prismatic joints. Generalized constraint equations using point, line and plane coordinates have been proposed which unify simulation of planar and spherical linkages and are demonstrably scalable to spatial mechanisms. As opposed to some of the existing approaches, which seek to derive loop-closure equations for each type of mechanism separately, we have shown that the simulation can be made simpler and more efficient by using unified version of the geometric constraints on joints and links. This is facilitated using homogeneous coordinates and constraints on geometric primitives, such as point, line, and plane. Furthermore, the approach enables simpler programming, real-time computation, and ability to handle any type of planar and spherical mechanism. This work facilitates creation of practical and intuitive design tools for mechanism designers.

Keywords: Kinematic analysis, *n*-bar simulation, prismatic and revolute joints, indeterminate mechanism analysis, optimization, planar mechanisms, spherical mechanisms, finite displacement problem.

1 Introduction

Kinematic simulation of a mechanism is the 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 [1]. The graphical analysis method is based on dyadic decomposition, i.e., identification of four-bar loops in mechanisms [2]. Although this approach is prominently used in simulation packages like Linkages [3] and PMKS [4], its limitations are well known [5]; 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 nonlinear equations [6]. Most analytical methods use the Polynomial continuation method [7, 8], elimination method or Grobner bases [9] to solve the simulation problem. Although, these methods are able to find all the possible assembly configurations of a given mechanism, they are not general in nature. The motion equations of each mechanism need to be derived manually on a case by case basis.

On the other hand, numerical simulation methods can handle extremely complex mechanism [10, 11]. They use iterative numerical methods like the Newton-Rhapson method to solve the system of non-linear equations for one solution only instead of all possible ones [12]. These methods accept the mechanism joints and link information as inputs. Subsequently, the algorithm repeatedly solves the finite displacement problem, i.e., the input

*Address all correspondence to this author at anurag.purwar@stonybrook.edu

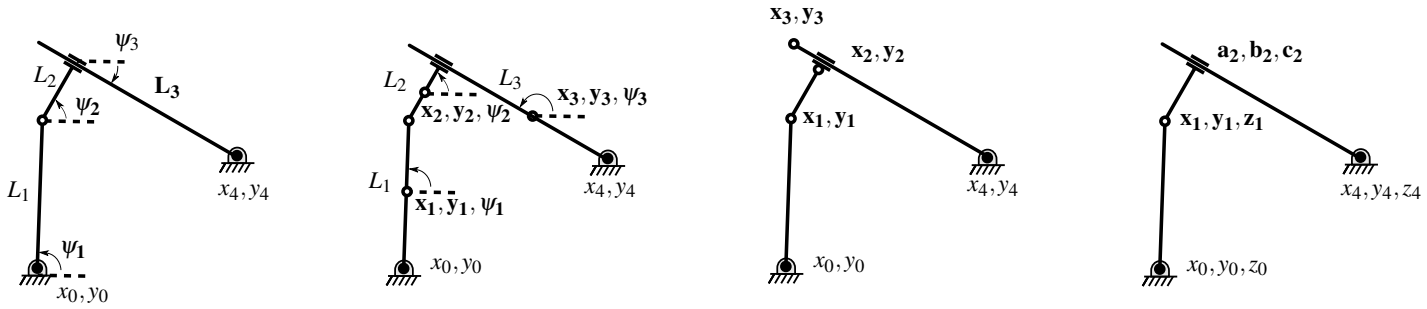


FIGURE 1: Different types of mechanism representations

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 [13]. However, the approach is limited to n-bar planar mechanism with revolute joints only. Radhakrishnan and Campbell [14] 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 [15, 16, 17, 18]. 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 packages which supports n-bar simulation for planar linkages with both revolute and prismatic joints [26, 27]. Furlong et al. [19] 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 [28, 29, 30, 31].However, currently there are no approaches which unify n-bar planar and spherical mechanism analysis and can be demonstrated to simulate more complex linkage systems. The proposed algorithm hopes to bridge this gap and augment the capability of pure kinematic design systems like MotionGen [32].

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 [10]. 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 reference point coordinates enables a scalable representation while relative 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 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 paper, 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 unknown vari-

ables smaller while also enabling construction of simple geometric constraint equations.

In this paper, 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 [20, 21, 22, 23, 24]. In [25], 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 paper, 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 paper 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 paper is organized as follows. Section 2 discusses the representation and constraints required to describe the motion of a general planar and spherical mechanism. Section 3 demonstrates the algorithm required to simulate a mechanism using the iterative numerical approach. Finally, in Section 4, we present a few examples to demonstrate the use of proposed algorithm.

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. 2. Its joints are represented as points and lines while its links are defined as a group of joints as shown in Table 1.

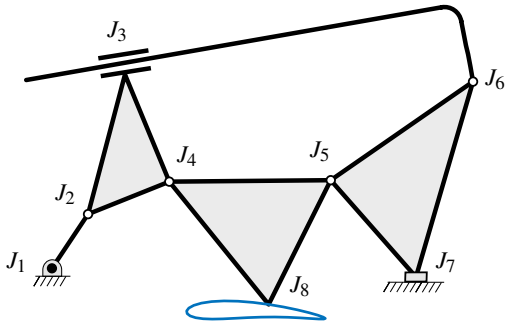


FIGURE 2: Planar Stephenson II six-bar linkage

TABLE 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

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 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. 3. Its joints are represented as points and planes while its links are defined as a group of joints as shown in Table 2.

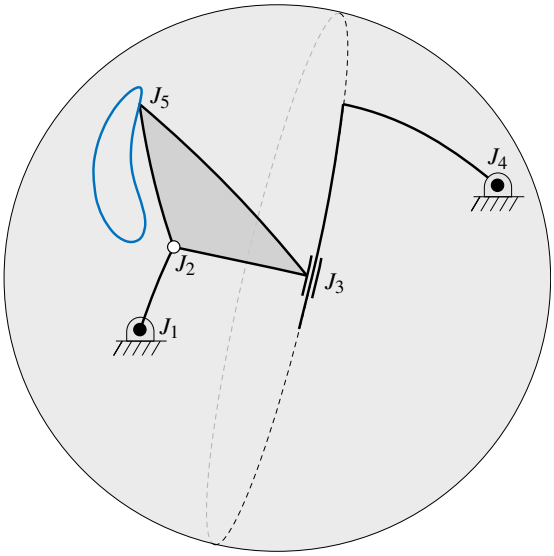


FIGURE 3: Spherical RRPR four-bar linkage

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 used to enforce the rigidity of any link in a general mechanism. Figure 4 and Fig. 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;

TABLE 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

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.

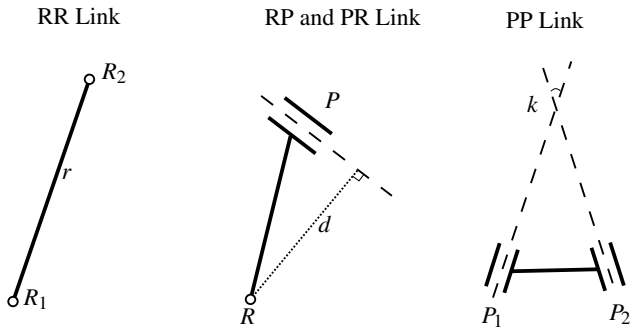


FIGURE 4: Types of binary planar links

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

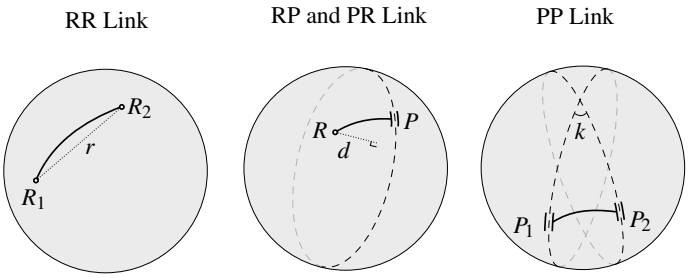


FIGURE 5: Types of binary spherical links

factors. The constraint equation is given as

$$C_{S,RR} : 2a_1b_1 + 2a_2b_2 + 2a_3b_3 + a_0b_4 = a_4\left(\frac{b_1^2+b_2^2+b_3^2}{b_4}\right), \tag{1}$$

where a_0 is given as

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

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

$$C_{P,RR} : 2a_1b_1 + 2a_2b_2 + a_0b_4 = a_4\left(\frac{b_1^2+b_2^2}{b_4}\right), \tag{3}$$

where a_0 is given as

$$a_0 = a_4r^2 - \frac{a_1^2+a_2^2}{a_4}. \tag{4}$$

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

$$C_{RP} : a_1L_1 + a_2L_2 + a_3L_3 + a_4L_4 = da_4\sqrt{L_1^2+L_2^2+L_3^2}, \tag{5}$$

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

$$C_{S,RP} : a_1L_1 + a_2L_2 + a_3L_3 = da_4\sqrt{L_1^2+L_2^2+L_3^2}. \tag{6}$$

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

$$C_{P,RP} : a_1L_1 + a_2L_2 + a_4L_4 = da_4\sqrt{L_1^2+L_2^2}. \tag{7}$$

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

$$C_{S,PP} : L_1M_1 + L_2M_2 + L_3M_3 = k\sqrt{L_1^2+L_2^2+L_3^2}\sqrt{M_1^2+M_2^2+M_3^2}, \tag{8}$$

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

$$C_{P,PP} : L_1M_1 + L_2M_2 = k\sqrt{L_1^2+L_2^2}\sqrt{M_1^2+M_2^2}. \tag{9}$$

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 paper.

It can be seen that the Eqs. 3, 6, 7, 8, 9 are all degenerate case of the Eq. 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. 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

$$C_P: \quad L_1^2 + L_2^2 + L_3^2 - 1 = 0. \tag{10}$$

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

$$C_{S,R}: \quad a_1^2 + a_2^2 + a_3^2 - 1 = 0, \tag{11}$$

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. (1), (3), (6), (7), (8), (9), (10) and (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.

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.

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 paper, 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

$$\begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix} = [\mathbf{T}]^{-1}[\mathbf{R}][\mathbf{T}] \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}, \tag{12}$$

where

$$[\mathbf{R_x}] = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{and} \quad [\mathbf{T}] = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

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

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = ([\mathbf{T}]^{-1}[\mathbf{R}][\mathbf{T}])^{-\mathbf{T}} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{14}$$

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. (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

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{b}{\sqrt{a^2+b^2}} d \\ \frac{-a}{\sqrt{a^2+b^2}} d \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{15}$$

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. (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

$$\begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{a_1 b_2 - a_2 b_1}{\sqrt{a_1^2+b_1^2}} d \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{16}$$

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

$$[\mathbf{R}]_{(l,m,n)} = [\mathbf{R_x}]^{-1}[\mathbf{R_y}]^{-1}[\mathbf{R_z}][\mathbf{R_y}][\mathbf{R_x}] \tag{17}$$

$$[\mathbf{R_x}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{n}{\sqrt{m^2+n^2}} & \frac{-m}{\sqrt{m^2+n^2}} \\ 0 & \frac{m}{\sqrt{m^2+n^2}} & \frac{n}{\sqrt{m^2+n^2}} \end{bmatrix} \tag{18}$$

$$[\mathbf{R_y}] = \begin{bmatrix} \sqrt{m^2+n^2} & 0 & -l \\ 0 & 1 & 0 \\ l & 0 & \sqrt{m^2+n^2} \end{bmatrix} \tag{19}$$

$$[\mathbf{R_z}] = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

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

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = [\mathbf{R}]_{(x_1,y_1,z_1)} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{21}$$

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

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = [\mathbf{R}]_{(x,y,z)} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{22}$$

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.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [\mathbf{R}]_{(a,b,c)} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{23}$$

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

$$\begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = [\mathbf{R}]_{(a_1,b_1,c_1)} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{24}$$

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.

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

$$\mathbf{\Phi}(\mathbf{q}) = 0 \tag{25}$$

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 nonlinear 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

$$\mathbf{q}_{i+1} = \mathbf{q}_i - [\boldsymbol{J}^{-1}(\mathbf{q}_i)]\mathbf{\Phi}(\mathbf{q}_i) \tag{26}$$

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

$$[\boldsymbol{J}(\mathbf{q})] = \begin{bmatrix} \frac{\partial \phi_1}{\partial q_1} & \frac{\partial \phi_1}{\partial q_2} & \cdots & \frac{\partial \phi_1}{\partial q_n} \\ \frac{\partial \phi_2}{\partial q_1} & \frac{\partial \phi_2}{\partial q_2} & \cdots & \frac{\partial \phi_2}{\partial q_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \phi_m}{\partial q_1} & \frac{\partial \phi_m}{\partial q_2} & \cdots & \frac{\partial \phi_m}{\partial q_n} \end{bmatrix} \tag{27}$$

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. (1) can be given as follows

$$\frac{\partial C_{RR}}{\partial a_1} = 2(a_1 - b_1) \tag{28}$$

$$\frac{\partial C_{RR}}{\partial a_2} = 2(a_2 - b_2) \tag{29}$$

$$\frac{\partial C_{RR}}{\partial a_3} = 2(a_3 - b_3) \tag{30}$$

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. (6) can be given as follows

$$\frac{\partial C_{S,RP}}{\partial a_1} = L_1, \frac{\partial C_{S,RP}}{\partial a_2} = L_2, \frac{\partial C_{S,RP}}{\partial a_3} = L_3 \tag{31}$$

$$\frac{\partial C_{S,RP}}{\partial L_1} = a_1 - \frac{dL_1}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{32}$$

$$\frac{\partial C_{S,RP}}{\partial L_2} = a_2 - \frac{dL_2}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{33}$$

$$\frac{\partial C_{S,RP}}{\partial L_3} = a_3 - \frac{dL_3}{\sqrt{L_1^2 + L_2^2 + L_3^2}} \tag{34}$$

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. (8) can be given as follows

$$\frac{\partial C_{S,PP}}{\partial L_1} = M_1 - \frac{kL_1\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{35}$$

$$\frac{\partial C_{S,PP}}{\partial L_2} = M_2 - \frac{kL_2\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{36}$$

$$\frac{\partial C_{S,PP}}{\partial L_3} = M_3 - \frac{kL_3\sqrt{M_1^2 + M_2^2 + M_3^2}}{L_1^2 + L_2^2 + L_3^2} \tag{37}$$

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. (10) can be given as follows

$$\frac{\partial C_P}{\partial a} = 2a, \quad \frac{\partial C_P}{\partial b} = 2b, \quad \frac{\partial C_P}{\partial c} = 2c \tag{38}$$

The first order partial derivatives for unit circle Revolute joint constraint given in Eq. (11) can be given as follows

$$\frac{\partial C_{S,R}}{\partial x} = 2x, \quad \frac{\partial C_{S,R}}{\partial y} = 2y, \quad \frac{\partial C_{S,R}}{\partial z} = 2z \tag{39}$$

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 1.

Algorithm 1: Algorithm for planar and spherical mechanism simulation

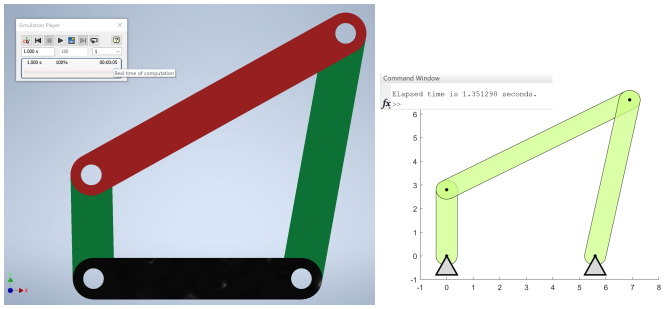
Input: Initial mechanism configuration
Calculate initial rigidity metric for each link
for *range of input link motion* **do**
  Perturb input link
  for *max iterations* **do**
    Calculate the constraint residual
    if *residual* $\leq \varepsilon$ **then**
      Solution found
      break
    end
    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

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.

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 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 6: Speed comparison for a planar crank rocker mechanism

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. 2 and its joint and link data is given in Table 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. 2. The run-time of this simulation was 3.79s.

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. 7 and its joint and link data is given in Table 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. 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.

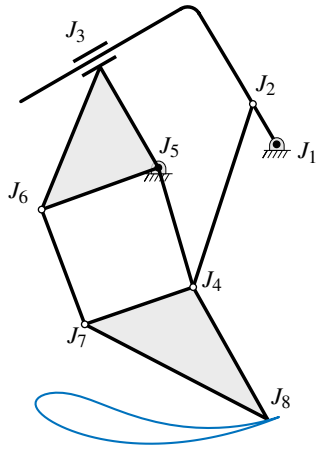


FIGURE 7: Planar modified Theo Jansen with floating prismatic joint

TABLE 3: Joint and Link data for Modified Theo Jansen linkage

Joint	Coordinates	Link	Constituent joints
$J_{1,input}$	2.77, 2.31	L_1	J_1, J_2
J_2	2.17, 3.33	L_2	J_2, J_3
J_3	-0.50, 0.87, -4.80	L_3	J_2, J_4
J_4	.66, -1.3	L_4	J_3, J_5, J_6
J_5	-.22, 1.72	L_5	J_5, J_4
J_6	-3.17, .66	L_6	J_6, J_7
J_7	-2.08, -2.24	L_7	J_4, J_7, J_8
J_8	2.54, -4.64	$L_{8,ground}$	J_1, J_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. 3 and it's joint and link data is given in Table 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 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. 3. The run-time of this simulation was 1.49s.

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. 8 and its link and joint data is given in Table 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. 8. The run-time of this simulation was 3.33s.

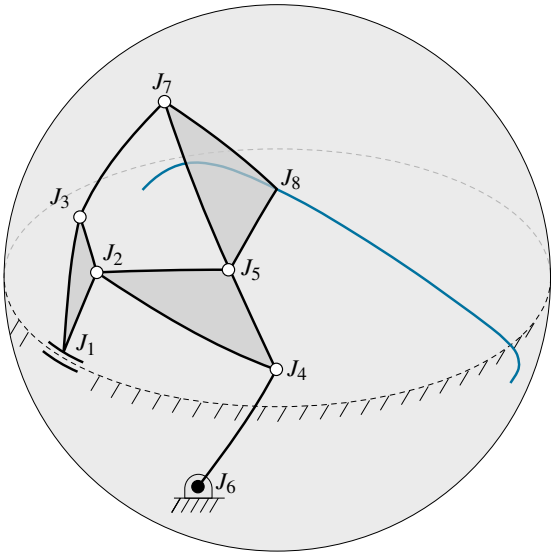


FIGURE 8: Spherical Watt I six-bar linkage

TABLE 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

5 Conclusion

In this paper, we have presented unified equations for motion simulation of planar and spherical $n-$bar mechanisms and an ef-

ficient 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.

REFERENCES

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

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

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

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

[5] 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.

[6] 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.

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

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

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

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

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

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

[32] 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.