ASME Journal of Mechanical Design
Online journal at:
https://asmedigitalcollection.asme.org/mechanicaldesign

A Unified Real-Time Motion Generation Algorithm for Approximate Position Analysis of Planar N-Bar Mechanisms

Zhijie Lyu
Computer-Aided Design and Innovation Lab,
Department of Mechanical Engineering,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: zhijie.lyu@stonybrook.edu

Anurag Purwar[1]
Computer-Aided Design and Innovation Lab,
Department of Mechanical Engineering,
Stony Brook University,
Stony Brook, NY 11794-2300
e-mail: anurag.purwar@stonybrook.edu

Wei Liao
Mechanismic Inc.,
St James, NY 11780
e-mail: wei@mechanismic.com

This paper presents a novel real-time kinematic simulation algorithm for planar N-bar linkage mechanisms, both single- and multi-degrees-of-freedom, comprising revolute and/or prismatic joints and actuators. A key feature of this algorithm is a reinterpretation technique that transforms prismatic elements into a combination of revolute joint and links. This gives rise to a unified system of geometric constraints and a general-purpose solver which adapts to the complexity of the mechanism. The solver requires only two types of methods—fast dyadic decomposition and relatively slower optimization-based—to simulate all types of planar mechanisms. From an implementation point of view, this algorithm simplifies programming without requiring handling of different types of mechanisms. This versatile algorithm can handle serial, parallel, and hybrid planar mechanisms with varying degrees-of-freedom and joint types. Additionally, this paper presents an estimation of simulation time and structural complexity, shedding light on computational demands. Demonstrative examples showcase the practicality of this method. [DOI: 10.1115/1.4064132]

Keywords: kinematic simulation, geometric constraints, graph-based constraint solvers, mobility analysis, computational kinematics, linkages

1 Introduction

The kinematic simulation of multi-body systems is crucial for the rapid design and evaluation of mechanisms in computer-aided design (CAD) systems. Faster simulation algorithms can significantly reduce the time required for mechanism design and enable the creation of high-quality datasets for machine learning research, which has garnered interest in recent years [1–9]. A recent position paper by Purwar and Chakraborty [10] emphasizes the importance of generating high-quality datasets for designing robot mechanisms that rely on robust and fast kinematic simulation. Planar linkage mechanisms are widely utilized in products due to their practicality, simplicity, and performance characteristics. While several commercially available CAD systems have general-purpose multi-body dynamics capabilities, the development of stand-alone kinematic simulation programs with comprehensive capabilities has been limited, with most software applications originating from academic research groups. Examples of such programs include LINCAGES [11,12], KINSYN III [13], Kihonge et al. [14], Spades [15], Sphinx [16], Sphinxpc [17], Osiris [18], and Synthetica [19]. Although these projects are no longer active, a few notable commercially available systems include SAM [20], MechGen [21], Linkages [22], Ch Mechanism Toolkit [23], MechDesigner [24], and Universal Mechanism [25]. Geometer's Sketchpad [26] and Geogebra [27], which are primarily geometry tools, have also been used for simulating mechanisms. Several other kinematic simulation tools, such as Linkage Mechanism Simulator [28], PMKS+ [29,30], GIM [31], Simionescu [32,33], and Schmidt and Lax [34], provide varying levels of capabilities and have filled a critical need for the mechanism simulation. This paper presents a novel simulation algorithm implemented in a web-based kinematic design and simulation application called MOTIONGEN,[2] which provides real-time simulation of N-bar planar linkage mechanisms with an arbitrary number of revolute and prismatic joints, rotary and linear actuators, and topological structures. Real-time simulation requires prompt feedback and data updates with low latency. While there are no rigid rules for the time to update, it is generally expected that this updating will happen quickly and typically at a rate that mimics real-world time, such as seconds or milliseconds per simulation time-step.

Kinematic simulation of planar linkages requires computing the unknown positions of moving joints, which leads to solving a set of geometric constraints—a system of equations constructed from the mechanism. Generally, there are two main issues in reducing the simulation time. The first is to create a fast computation method, or *solver*, to solve a specific set of geometric constraints. One of the simplest analytical methods for planar linkages is the dyadic decomposition [35,36], which solves one planar joint position, or two variables (x and y), at a time. While this works for simpler mechanisms, researchers have developed other methods for handling more complex structures. Dhingra et al. [37] employed

[1]Corresponding author.

Contributed by Mechanisms and Robotics Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received July 6, 2023; final manuscript received November 15, 2023; published online December 15, 2023. Assoc. Editor: Bin Zi.

[2]http://www.motiongen.io

symbolic computation and followed the computation of a Gröbner basis with a solution based on a Sylvester-type determinant. On the other hand, Wampler [38,39] proposed a combination of the Dixon determinant procedure from Nielsen and Roth [40] with a complex plane formulation. He presented a method for formulating kinematic equations in complex planes using isotropic coordinates, which results in loop equations in a simplified form that can be solved using the Sylvester-type formula. Hernández and Petuya [41] have introduced a novel geometric iterative (GI) technique for solving the position problem with only revolute joints, which does not require an initial guess, as in the case of local optimization methods. However, the Newton–Raphson (NR) method is still a popular choice for solving systems of nonlinear equations [42] due to its numerical search for a particular solution, with quadratic convergence in the neighborhood of the solution.

The second issue is the identification and joint position analysis of subassemblies within a mechanism using a divide-and-conquer approach. This technique involves identifying subsets within the geometric constraints of the system, which can be used to solve some of the unknown variables, or unknown joints. The subset can then be substituted back into the system to solve the remaining unknown variables. This approach enables the solution of a set of kinematic constraints to be broken down into a sequence of solutions of the kinematic constraint subsets, a process that typically takes less time than solving the entire set of constraints together. This process of identifying subsets is also referred to as the *decomposition of the system*. Researchers have proposed various methods for implementing the divide-and-conquer approach in system analysis. Ait-Aoudia et al. [43,44] demonstrated a decomposition method that utilizes bipartite graphs under-laid by systems of geometric constraint equations. Bouma et al. [45] and Fudos and Hoffmann [46] proposed a graph-theory-based method to reduce the number of variables that need to be solved in each step. Additionally, Fudos and Hoffmann developed a phase called the *reduction of the system* that simplifies the system of equations by merging fixed links together into a single link.

Most simulation software apply one or more of these aforementioned techniques. For example, LINKAGES [22] uses the dyadic decomposition extensively, while PMKS/PMKS+ [29,30] and GIM [31] use the GI technique. AUTOCAD based simulation [47] and MEKIN2D [32] use the modular approach to directly identify the subassemblies with a saved Assur group library.

Compared to these simulation packages, our method's originality lies in reducing all geometric constraints to a unified form and converting all elements of a mechanism into revolute only joints and links, thereby allowing us to require only two types of solvers for all degrees-of-freedom (DOF) planar N-bar mechanisms. We also developed a method to analyze the complexity of a certain mechanism. Specifically,

(1) The prismatic joints and actuators are converted to a combinations of revolute joints and links. This novel reinterpretation means that an additional solver for the prismatic joint is not necessary. Furthermore, this also means a simpler optimization function to program.
(2) Our algorithm requires only two solvers. One provides the fastest computation speed possible using dyadic decomposition and the other is the optimization-based method for solving complex subassemblies.
(3) This algorithm also allows arbitrary placement of actuators and arbitrary combinations of revolute joints and prismatic joints.
(4) A mobility analysis method is presented to analyze the system of equations, which includes the reduction phase (static link merge) and the decomposition phase (dynamic and static link merge).

A flowchart of the complete simulation process can be seen in Fig. 1. The details of various steps in this figure are described in the following sections. Section 2 presents a matrix representation of the kinematic structures. Following that, Sec. 3 presents the reinterpretation trick and a derivation of the accuracy error incurred during the reinterpretation. Next, the general formulation of the system of equations is presented in Sec. 4. Then, Sec. 5 provides an overview of the two types of solvers. After that, the analysis of the kinematic system, an estimate of the time cost, and a measure of the structural complexity of mechanisms are presented in Sec. 6. Finally, we present a few typical example mechanisms in Sec. 7, which seek to illustrate the overall method.

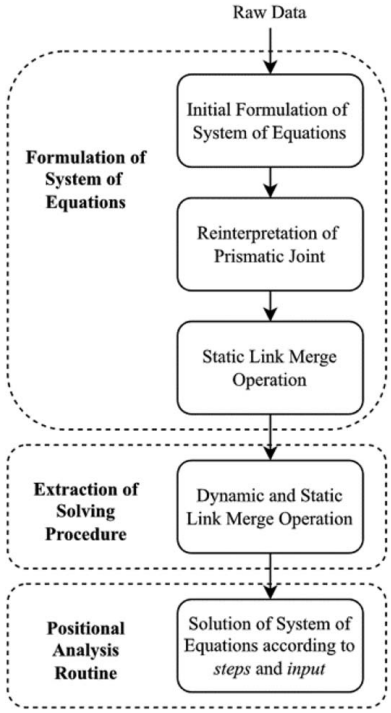

Fig. 1 The complete simulation process

In summary, the main contributions of this work are in (1) proposing a reinterpretation trick to reduce all geometric constraints to a single type, (2) topological modification of N-bar mechanisms to only include revolute joints, (3) formulation of a general-purpose solver which can automatically select the fastest method for computing joint positions, and (4) implementation in a cross-platform browser-based software to satisfy the needs of users with varied expertise.

2 General Matrix Representation of Linkages

The mechanisms in this paper are described by their topological configurations. To fully represent a planar linkage mechanism, four basic components are considered:

(1) Joints: Joints in this work are considered as points of interest characterized by their positional variables, denoted as x_p and y_p whether or not there are two links connected at that joint. Each joint is represented by j_p, where p is the joint identifier. Coupler or tracer points, which are points only on one link, are also considered joints since they can potentially function as joints. They are not distinguished from joints shared by multiple links.
(2) Links: A link is a rigid body consisting of one or more joints. The relative positions of the joints in a link remain constant throughout the mechanism's operation. A link can be binary, containing two joints, or n-ary, containing n specified number of joints. Additionally, a binary link with joints i and j can be denoted as $l_{i,j}$.
(3) Slots: Slots are assumed to be straight and finite constraints placed on binary links to represent linear constraints. A

slot is defined by its two endpoint joints, which determine the slot axis.

(4) Actuators: Actuators introduce changes to the mechanism by varying either the angle or the length. There are two types of actuators: rotary and linear. An actuator is defined by its type and three joints; see Fig. 2 for an illustration.

Throughout this paper, subscripts are used to represent matrix elements, with indexing starting from 0. For example, $K_{i,j}$ refers to the element in the ith row and jth column of the matrix K. If a single number represents a link, it indicates the index of that link. For instance, l_n represents the $(n+1)$th link.

The relationship between links and joints can be represented using the incidence matrix B as mentioned in Tsai's book [48]. This matrix is also known as the LJ-matrix [49]. Its definition is shown in Eq. (1). The two subscripts mean this notation corresponds to the ith row and the jth column in the matrix.

$$B_{i,j} = \begin{cases} 1, & \text{if link } i \text{ contains joint } j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Then, two vectors are used to save the property of the joints and the links, respectively. The link property vector is a column vector, while the joint vector is a row vector. These two vectors are defined by Eqs. (2) and (3), respectively.

$$L_{i,0} = l_i = \begin{cases} 1, & \text{if link } i \text{ is a ground link} \\ 0, & \text{if link } i \text{ is a moving link} \end{cases} \tag{2}$$

$$J_{0,i} = j_i = \begin{cases} 1, & \text{if joint } i \text{ is a ground joint} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The arrangement of the link property vector and joint property vector as described allows them to stack with the incidence matrix B in a specific format. The stacked format is as follows:

$$\begin{matrix} \left(J-\text{vector}\right)_{1\times n} \\ \left(L-\text{vector}\right)_{l\times 1}\left(B-\text{matrix}\right)_{l\times n} \end{matrix}$$

Next, the distance matrix D to save the distance between any two joints is defined in Eq. (4) as

$$D_{i,j} = \begin{cases} l_{i,j}, & i \neq j \\ 0, & i = j \end{cases} \tag{4}$$

The reduced and unweighted form of the matrix D is denoted as matrix T, as defined in Eq. (5). This format is used during the initialization of distances. In the initial state, the matrix D can be calculated according to Eq. (6). Moreover, the joint property vector J can be saved in the diagonal of T for space efficiency concerns.

$$T_{i,j} = \begin{cases} 1, & \text{if } B_{k,i} = 1 \text{ and } B_{k,j} = 1, \text{ excluding } i = j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \cdot T_{i,j} \tag{6}$$

Next, the slots in the mechanism are stored in the slot matrix S. The inclusion of a slot constraint relationship depends on the presence of a joint within the slot, referred to as a "slot joint." If there are a total of m slot joints, the slot matrix S is defined according to Eq. (7). Before the analysis of the kinematic system phase, each slot is converted into a combination of links and joints.

$$S = \begin{bmatrix} p_0 & q_0 & r_0 \\ \vdots & \vdots & \vdots \\ p_{m-1} & q_{m-1} & r_{m-1} \end{bmatrix}_{m\times 3} \tag{7}$$

Next, we discuss a representation of rotary and linear actuators. Figure 2 illustrates these two types of actuators. A rotary actuator constrains the angle between two links. More specifically, it gives a constraint relationship between the common joint and two joints in the two links. Next, a linear actuator constrains the slope (slot axis) and the distance between the output joint (slot joint) and one of the end joints. All a number of actuators are saved in a $a \times 4$ matrix according to Eq. (8). In the I matrix, the first three columns specify the joint indices and the last column specifies their corresponding actuator type. Here, R denotes a rotary actuator, while P stands for a linear actuator, with P being an abbreviation for prismatic.

$$I = \begin{bmatrix} i & j & k & R \\ \vdots & \vdots & \vdots & \vdots \\ p & q & r & P \end{bmatrix}_{a\times 4} \tag{8}$$

It should be noted that the joints are order sensitive. For example, for the rotary actuator $\alpha_{i,j,k}$ shown in the first row of I in Eq. (8), the angle constraint is between vector $\vec{v}_{j,i}$ and $\vec{v}_{j,k}$. Furthermore, we use the cross-product value to represent this constraint so that the angle is not flipped. For the linear actuator $\alpha_{p,q,r}$ in the last row of I, the direction is defined by vector $\vec{v}_{p,r}$, and the output distance is defined by the length of $\vec{v}_{p,q}$. Figure 2 follows this description.

Last, it should be pointed out that each actuator brings one *dynamic link* (a similar concept is the active pair in Ref. [49]). For the rotary actuator $\alpha_{i,j,k}$, when the link lengths of $l_{i,j}$ and $l_{j,k}$ and the input angle are given, $l_{i,k}$ is the dynamic link and can be computed according to the law of cosine. For the linear actuator $\alpha_{p,q,r}$, $l_{p,q}$ is the dynamic link, and the length is the input itself.

Storage of Simulation Input and Results: A state corresponds to a specific set of inputs and the number of inputs is unchanged throughout the simulation. All the state input can be stacked in a state matrix V according to Eq. (9). Here α_i is the $(i-1)$th actuator in matrix I. In the jth state, its input is $\alpha_{i-1}(t_{j-1})$. When there are s number of states successfully simulated, the size of V is $a \times s$.

$$V = \begin{matrix} \begin{matrix} s_0 & s_1 & \cdots & s_{s-1} \end{matrix} \\ \begin{pmatrix} \alpha_0(t_0) & \alpha_0(t_1) & \cdots & \alpha_0(t_{s-1}) \\ \alpha_1(t_0) & \alpha_1(t_1) & \cdots & \alpha_1(t_{s-1}) \\ \alpha_2(t_0) & \alpha_2(t_1) & \cdots & \alpha_2(t_{s-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{a-1}(t_0) & \alpha_{a-1}(t_1) & \cdots & \alpha_{a-1}(t_{s-1}) \end{pmatrix}_{a\times s} \end{matrix} \tag{9}$$

Correspondingly, the simulation result is the joint positions, which can be saved in a position tensor P defined in Eq. (10).

$$P = [P_0, P_1, P_2, \ldots, P_{s-1}]_{s\times n\times 2} \tag{10}$$

For the kth state, the joint positions can be saved in a slice of the P tensor according to Eq. (11).

$$P_{k-1} = \begin{matrix} & x & y \\ \begin{matrix} j_0 \\ j_1 \\ j_2 \\ \vdots \\ j_{n-1} \end{matrix} & \begin{pmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}_{n\times 2} \end{matrix} \tag{11}$$

In conclusion, generally, for a mechanism M, if there are l number of links, a number of actuators, n number of joints, m

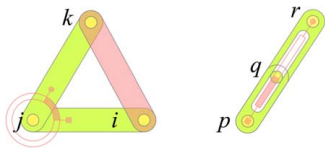

Fig. 2 The letters denote the joints. For the rotary actuator (left), $l_{i,k}$ is the corresponding dynamic link, and for the linear actuator (right), $l_{p,q}$ is the corresponding dynamic link.

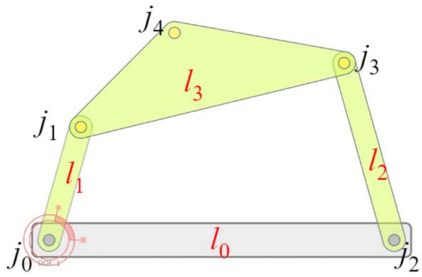

Fig. 3 An example RRRR mechanism

number of slot relationships, and s number of states that are successfully simulated, then all these can be stored in a compact format denoted in Eq. (12). This representation is a concise saving structure: in brief, matrices J, L, B, D, S, and I save all types of constraints. Matrices V and P store the simulation results of all states for input and the joint positions, respectively. Furthermore, the B matrix is a standard representation of kinematic structures [48]: this means that graph isomorphism detection and mobility analysis can be readily applied to the system.

$$M = \begin{cases} J & : J_{1\times n} \\ L & : L_{l\times 1} \\ B & : B_{l\times n} \\ D & : D_{n\times n} \\ S & : S_{m\times 3} \\ I & : I_{a\times 4} \\ V & : V_{a\times s} \\ P & : P_{s\times n\times 2} \end{cases} \tag{12}$$

As an example, the matrices to save the geometric constraints for a typical RRRR mechanism shown in Fig. 3 are as follows. First, according to this figure, the joint property vector J, and the link property vector L are, respectively,

$$J = \begin{matrix} j_0 & j_1 & j_2 & j_3 & j_4 \end{matrix} \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \text{and } L^T = \begin{matrix} l_0 & l_1 & l_2 & l_3 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

There are no slots in an RRRR mechanism, meaning the S matrix is empty. Additionally, there is only one actuator in this mechanism. Therefore, the input matrix I is

$$I = \begin{bmatrix} 2 & 0 & 1 & R \end{bmatrix}$$

The B matrix is

$$B = \begin{matrix} l_0 \\ l_1 \\ l_2 \\ l_3 \end{matrix} \begin{matrix} j_0 & j_1 & j_2 & j_3 & j_4 \end{matrix} \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

And finally, the T matrix is shown below. Additionally, the D matrix can be computed according to Eq. (6) and the initial joint positions.

$$T = \begin{matrix} j_0 \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{matrix} \begin{matrix} j_0 & j_1 & j_2 & j_3 & j_4 \end{matrix} \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}_{5\times 5}$$

3 Reinterpretation of Slots

The idea of reinterpreting prismatic joints as revolute joints arises from the fact that PR and RP links can be considered as sufficiently large RR links with one of their joints located either proximate or distal from the workspace of the mechanism. In the case of a PR link with a fixed line, the RR link can be chosen to be large enough within a desired accuracy with a distal fixed pivot while for the RP link with a fixed pivot, the link length of the approximated RR link should be large enough, but this time with the fixed pivot proximate. By using homogeneous coordinates, a planar joint position is represented in a projective plane as (x, y, w) with its position in Euclidean space given by (x/w, y/w). By setting the homogenizing factor w to a small value, the joint is moved far away from the origin (0, 0) and effectively converted into a prismatic joint. In the kinematic simulation, we perform the inverse operation by converting the prismatic joint back into revolute joints and links. This allows solvers that can only handle revolute joints to solve systems that include prismatic joints.

For ease of implementation and intuitive sketching, we use joints in a slot to represent prismatic joints. Figure 4(*a*) shows an example mechanism with a slot, which restricts the slot joint to a fixed-line segment. Slots can be fixed or can be moving so as to represent RP or PR links.

Here, the slot joint j_2 performs a linear motion along the slot. A straight line can be reinterpreted as the arc of a circle whose radius is large enough such that the curvature is near zero. Consequently, the straight line is represented as the arc of a circle, and the intersections between the straight line and the arc occur at the slot's edges (end joints). Furthermore, the center of the circle lies on the perpendicular bisector of the straight line.

The resultant reinterpretation can be seen in Fig. 4(*b*). First, the slot is removed, and a new far joint (j_5, or j_f) is added to the slotted link. Then, two constraining links ($l_{3,6}$ and $l_{2,6}$) and a constraining joint (j_6, or j_c) are added to constrain the motion along the straight line. These two binary links together can also be seen as an RRR robot arm so that when the arm is stretched, the slot joint j_2 goes near j_4, and when two binary links overlap, j_2 goes near j_3. During simulation, the slot is removed, and the joint previously in the slot now moves along an arc. The farther the joint j_5, the flatter the arc is. Therefore, the difference between the arc and the straight line can be controlled by changing the distance of j_5 from the two end joints j_3 and j_4.

The far joint position and the reinterpretation error are derived as follows. Apropos Fig. 4(*b*), since the bisector is perpendicular to the slot axis, we can compute the perpendicular unit vector as

$$\begin{cases} v_x = -(y_4 - y_3)/\sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2} \\ v_y = (x_4 - x_3)/\sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2} \end{cases} \tag{13}$$

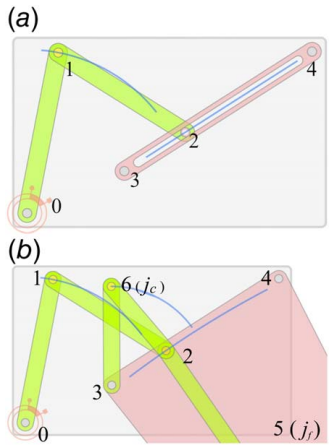

Fig. 4 The reinterpreted RRRP mechanism in (*a*) is shown in (*b*). The far joint j_5 (or j_f) is in the far bottom-right: (*a*) joint notations before the P-joint conversion and (*b*) joint notations after the P-joint conversion.

The intersection point p_{mid} of the bisector and the slot axis and correspondingly the far joint j_5 are given by, respectively

$$\begin{cases} x_{\text{mid}} = (x_4 + x_3)/2 \\ y_{\text{mid}} = (y_4 + y_3)/2 \end{cases} \rightarrow \begin{cases} x_5 = x_{\text{mid}} + v_x \cdot h \\ y_5 = y_{\text{mid}} + v_y \cdot h \end{cases} \tag{14}$$

The value of h depends on the required accuracy. In MOTIONGEN, the value of h is set dynamically such that the error of approximation is within a threshold. The expressions below give the deviation from the curve to the straight line. First, the distance between two edge joints is

$$d = \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2} \tag{15}$$

The reinterpretation error e is measured with the distance between the point on the arc and its corresponding point in the straight line. At its maximum, where the slot joint is close to one of the end joints, we have

$$e = \sqrt{\left(\frac{d}{2}\right)^2 + (h)^2} - h \tag{16}$$

which can be simplified to

$$(e + h)^2 = e^2 + h^2 + 2e \cdot h = \frac{d^2}{4} + h^2 \tag{17}$$

Since the e^2 term is extremely small compared to other terms, by removing e^2 and canceling h^2 on both sides, we get

$$h = \frac{d^2}{8e} \tag{18}$$

The next to compute is the location of the constraining joint, which connects the two constraining links and is denoted as j_c (or j_6 in the figure). Computing its position requires the offsets from the two end joints. For example, j_2 should be distance o_4 away from j_4 when stretched and distance o_3 away from j_3 when one constraining link is on top of the other. We can derive a set of equations with these two additional conditions. Then, the two link lengths can be obtained.

$$\begin{cases} l_{3,6} + l_{2,6} = d - o_4 \\ l_{3,6} - l_{2,6} = o_3 \end{cases} \rightarrow \begin{cases} l_{3,6} = \dfrac{d + o_3 - o_4}{2} \\ l_{2,6} = \dfrac{d - o_3 - o_4}{2} \end{cases} \tag{19}$$

With these two link lengths and the positions of j_2 and j_3 in the initial state, the initial position of j_6 can be computed through arc intersection. Of course, there are usually two solutions, but both work for the simulation, and choosing either is sufficient.

The reinterpretation changes the topological structure of the mechanism. Therefore, the J, L, B, and D matrices will need to be changed accordingly. Here, we show the change of B matrix in Eq. (20) and the change of T matrix in Eq. (21). J is in the diagonal of T, and L adds three more zeros at its end because there are three moving links. The D matrix can be computed according to Eq. (6).

$$\begin{array}{c} \\ l_0 \\ l_1 \\ l_2 \\ l_3 \end{array} \begin{array}{c} \begin{array}{ccccc} j_0 & j_1 & j_2 & j_3 & j_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 \end{array} \\ \left(\begin{array}{ccccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right) \end{array} \tag{20}$$

$$\begin{array}{c} \\ j_0 \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{ccccc} j_0 & j_1 & j_2 & j_3 & j_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ j_0 \\ j_1 \\ j_2 \\ j_3 \\ j_4 \\ j_5 \\ j_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 \end{array} \\ \left(\begin{array}{ccccc|cc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{21}$$

4 System of Equations

When slots are converted into a combination of links and joints, the planar mechanism has only two types of constraints. One is the distance constraint from the links' lengths, and the other is the angle constraint from the rotary actuators. An n-ary link has $\frac{n(n-1)}{2}$ number of binary links (its two-joint sub-collections), and each of these binary links corresponds to one specific distance constraint. Furthermore, the actuators are reinterpreted as dynamic links, which are also distance constraints in the system. For a binary link $l_{p,q}$, this distance constraint is written as shown in Eq. (22). The i in the equation means it is the ith distance constraint. When this constraint is satisfied, $f_i = 0$.

$$f_i = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} - l_{p,q} \tag{22}$$

As mentioned, we use the cross-product value to represent the angle constraint to avoid the flipping issue. Therefore, the jth angle constraint function between vector $\vec{v}_{q,p}$ and $\vec{v}_{q,r}$ can be written as shown in Eq. (23). Here, $\alpha_{p,q,r}$ is to specify the three joints that define the actuator together. Similarly, when this angle constraint is satisfied, $g_j = 0$.

$$g_j = (x_p - x_q)(y_r - y_q) - (x_r - x_q)(y_p - y_q) - l_{p,q} \cdot l_{q,r} \cdot \sin(\alpha_{p,q,r}) \tag{23}$$

The system of equations to solve can be now derived accordingly. The positional analysis problem is formulated as given a specific state, find the stack of variables $\mathbf{x}$ that satisfies $\mathbf{f} = \mathbf{0}$. All the states are solved one by one. Vectors $\mathbf{x}$ and $\mathbf{f}$ are shown in Eq. (24). Here, k number of binary links and a number of rotary actuators are taken into account. It is worth noting that not all binary links need to be taken into account to solve the system because some of them are redundant constraints and can be automatically satisfied.

$$\mathbf{x} = \begin{bmatrix} x_p \\ y_p \\ x_q \\ y_q \\ \vdots \\ x_r \\ y_r \end{bmatrix}_{2n \times 1} \qquad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_k \\ g_1 \\ \vdots \\ g_{a-1} \\ g_a \end{bmatrix}_{(k+a) \times 1} \tag{24}$$

5 Solvers

We utilize two types of solvers for kinematic analysis: (1) the analytical (arc intersection) and (2) the numerical-based (Newton's method). These methods are employed to handle mechanisms with varying complexity and optimize computational efficiency.

The analytical method, specifically the arc intersection method, is highly time-efficient. However, it is not capable of solving mechanisms with complex loop structures. On the other hand, the numerical-based method, i.e., Newton's method, can handle mechanisms with arbitrary complexity, but requires more computational time compared to the arc intersection method, especially when the number of joints increases.

We also experimented with two variations of Newton's method for kinematic structures when multiple joints need to be solved simultaneously. The first variant is the NR method, which aims to find the extremum of an overall cost function. This cost function is iteratively computed using Eq. (25).

$$\mathbf{x_{t+1}} \leftarrow \mathbf{x_t} - \gamma[\mathbf{H}_{F(\mathbf{x_t})}]^{-1}\nabla F(\mathbf{x_t}),\ \ t \geq 0,\ \ 0 \leq \gamma \leq 1 \tag{25}$$

Here, $F(\mathbf{x_t}) = \mathbf{f}(\mathbf{x_t})^T\mathbf{f}(\mathbf{x_t})$, which is the sum of squared error (SSE). This value is close to zero when vector $\mathbf{x}$ is a good numerical approximation, which also serves as the termination of the iteration. Term $\nabla F(\mathbf{x_t})$ is the gradient of $F(\mathbf{x_t})$ and $[\mathbf{H}_{F(\mathbf{x_t})}]$ is the Hessian matrix of $F(\mathbf{x_t})$.

Another variant of Newton's method is the Levenberg–Marquardt (LMA or LM) algorithm [50]. This algorithm introduces the damping factor into the Gauss–Newton algorithm to make the update robust. Specifically, it ensures the existence of the inverse matrix. The updated policy for this numerical method is shown in Eq. (26).

$$\mathbf{x_{t+1}} \leftarrow \mathbf{x_t} - \gamma(\mathbf{J}^T\mathbf{J} + \lambda \cdot \mathrm{diag}(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{f}(\mathbf{x_t}) \tag{26}$$

Here, the Jacobian matrix $\mathbf{J}$ is given by Eq. (27) and $\lambda = 1$. The updated policy for γ is the same as the NR method.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_p} & \frac{\partial f_1}{\partial y_p} & \cdots & \frac{\partial f_1}{\partial x_r} & \frac{\partial f_1}{\partial y_r} \\ \frac{\partial f_2}{\partial x_p} & \frac{\partial f_2}{\partial y_p} & \cdots & \frac{\partial f_2}{\partial x_r} & \frac{\partial f_2}{\partial y_r} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_{a-1}}{\partial x_p} & \frac{\partial g_{a-1}}{\partial y_p} & \cdots & \frac{\partial g_{a-1}}{\partial x_r} & \frac{\partial g_{a-1}}{\partial y_r} \\ \frac{\partial g_a}{\partial x_p} & \frac{\partial g_a}{\partial y_p} & \cdots & \frac{\partial g_a}{\partial x_r} & \frac{\partial g_a}{\partial y_r} \end{bmatrix}_{(k+a)\times(2n)} \tag{27}$$

These two methods are both computationally expensive because they require matrix multiplications. Furthermore, the optimization methods take several iterations to converge. So, for the same number of joints to solve, they are much slower compared to the arc intersection solver. Through our experiments, we found that LM tended to converge faster in most scenarios.

6 Analysis of System and Cost

The previous section presented a system of multi-variable equations, which represent geometric constraints of planar mechanisms. However, solving all the variables simultaneously is time-consuming and often unnecessary. There have been several efforts to develop different analysis algorithms to look for optimum solving paths [43–47]. Each solution step deals with part of the unknown variables in a given particular subassembly. Then, the already solved variables can be substituted back into the system to help solve the remaining variables. Generally, an analysis algorithm involves two processes. The first is the reduction of the system. This process tries to find the link set where all members do not change their relative positions with respect to other members; in other words, we try to find Assur groups [51,52] within the mechanism. We can merge these links into one link (rigid body), reducing the number of links and simplifying the system. The second process is the decomposition. This process starts from the ground link/known link, and tries to find the least number of links that formulate a zero DOF system; in other words, we find the smallest possible Assur group that contains the ground link. The joints on these links are marked to be solved in one step. As a result, solving the joint positions on each step leads to solving all the joint positions. Links being amenable to be merged into a rigid body and a subassembly being a zero degrees-of-freedom system mean the solution for the involved variables bounded. We use the Chebychev–Grubler–Kutzbach criterion to determine the DOF of a system [53], i.e.,

$$\mathrm{dof} = 3(n-1) - 2 \cdot l - h \tag{28}$$

where

$$\mathrm{DOF} = \begin{cases} < 0 & \text{At least one sub-kinematic chain is over-constrained.} \\ = 0 & \text{Undecided if its sub-kinematic chain is undecided; is properly constrained if each sub-kinematic chain is properly constrained.} \\ > 0 & \text{At least one sub-kinematic chain is under-constrained.} \end{cases}$$

According to the description above, to ensure a specific set of links having 0DOF, we must start the examination from the smallest subset of this link set. In the worst-case scenario, all subsets except the empty set and one-member sets need to be examined. This leads to a possibly unacceptable time cost, which is one of the reasons why some modern simulation software uses the Assur Group library. However, the Assur Group is infinite, and the library saves the most common kinematic assembly. As for our algorithm, we employed the aforementioned criterion to maintain generality.

Then, the overall time cost can be divided into three parts. The first part is the position analysis part, which depends on multiple things, e.g., the number of states to compute, the number of steps, and the variables to compute in these steps. The second part is the system analysis. Finally, the third part includes all the remaining processes because they usually take insignificant time compared to the first two. Therefore, the overall time cost can be estimated with Eq. (29).

$$\mathrm{Time} = \left(\sum_{i=0}^{len(steps)} (2n_i)^{t_i} \cdot (a_i + l_i) \cdot C_i\right) \cdot s + \mathrm{AoS\text{-}time} + \mathrm{Prep\text{-}time} \tag{29}$$

Here n_i denotes the number of joints to solve in step i and t_i is determined by the type of solver for the ith step, shown in Eq. (30). Moreover, a_i and l_i are the number of actuators and the number of binary links to solve in the system of equations in step i, respectively. Term s represents the number of states, and C_i is the estimated time constant for the solver used in the step i. This constant is related to the solver's convergence speed and the time to stack all equations in this step.

$$t_i = \begin{cases} 1 & \text{If the arc intersection is used for the } i\text{-th step} \\ 2 & \text{If a Newton solver is used for the } i\text{-th step} \end{cases} \tag{30}$$

With the time analysis stated above, we can even use the time cost for one state of position analysis to determine the complexity of a certain mechanism, i.e.,

$$\mathrm{Complexity} = \sum_{i=0}^{len(steps)} (2n_i)^{t_i} \cdot (a_i + l_i) \cdot C_i \tag{31}$$

7 Case Analysis

In this section, we present a few examples to illustrate the working of our algorithm. The first example includes all steps of the solving procedure for a mechanism with a combination of

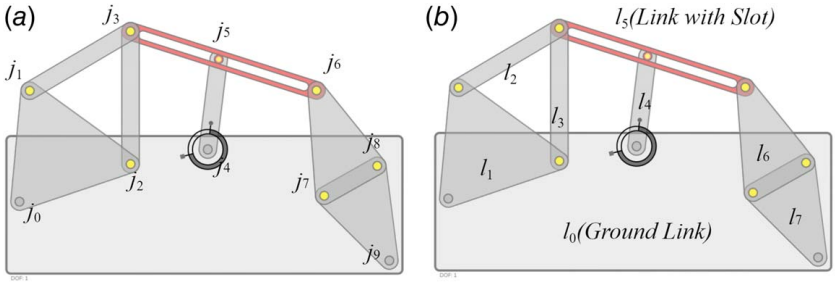

Fig. 5 This figure shows (*a*) links and (*b*) joint labels

binary, ternary, quaternary links, and a moving slot connected by revolute and prismatic joints. Then, this section will go through several cases as subsections to discuss each aspect of this simulation algorithm. Furthermore, in Sec. 7.6, we present the time complexity of each of the example mechanisms.

7.1 Example 1: A Seven-Bar Mechanism With a Moving Slot and Revolute and Prismatic Joints. Figure 5 shows the example mechanism to be analyzed in this section. We note that this is purely an academic example chosen to illustrate several steps of position analysis presented earlier.

Initial Topological Structure: The topological structure for this mechanism is given by several matrices presented earlier

$$B = \begin{array}{c} \\ l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \end{array} \begin{array}{c} \begin{array}{cccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

$$\begin{cases} S = \begin{pmatrix} 3 & 5 & 6 \end{pmatrix} \\ I = \begin{pmatrix} 0 & 4 & 5 & R \end{pmatrix} \\ J = \begin{array}{c} \begin{array}{cccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \\ L^T = \begin{array}{c} \begin{array}{cccccccc} j_0 & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \end{cases}$$

$$D = \begin{array}{c} \\ j_0 \\ j_1 \\ j_2 \\ j_3 \\ j_4 \\ j_5 \\ j_6 \\ j_7 \\ j_8 \\ j_9 \end{array} \begin{array}{c} \begin{array}{cccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 \end{array} \\ \begin{pmatrix} 0 & l_{0,1} & l_{0,2} & 0 & l_{0,4} & 0 & 0 & 0 & 0 & l_{0,9} \\ l_{0,1} & 0 & l_{1,2} & l_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ l_{0,2} & l_{1,2} & 0 & l_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & l_{1,3} & l_{2,3} & 0 & 0 & 0 & l_{3,6} & 0 & 0 & 0 \\ l_{0,4} & 0 & 0 & 0 & 0 & l_{4,5} & 0 & 0 & 0 & l_{4,9} \\ 0 & 0 & 0 & 0 & l_{4,5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_{3,6} & 0 & 0 & 0 & l_{6,7} & l_{6,8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{6,7} & 0 & l_{7,8} & l_{7,9} \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{6,8} & l_{7,8} & 0 & l_{8,9} \\ l_{0,9} & 0 & 0 & 0 & l_{4,9} & 0 & 0 & l_{7,9} & l_{8,9} & 0 \end{pmatrix} \end{array}$$

Reinterpretation of Slot and Prismatic Joint: This step focuses on representing a slot with a combination of revolute joints and links. Figure 6 shows the converted mechanism and the notations for all its elements. The new system has two more joints and three more links. Furthermore, one of the new joints, i.e., the far joint, is added to l_5, which previously was a binary link. The topological notations for this new mechanism are as follows:

$$B = \begin{array}{c} \\ l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 & j_{10} & j_{11} \end{array} \\ \left(\begin{array}{cccccccccc|cc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$

$$\begin{cases} I = \begin{pmatrix} 0 & 4 & 5 & R \end{pmatrix} \\ J = \begin{array}{c} \begin{array}{cccccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 & j_{10} & j_{11} \end{array} \\ \left(\begin{array}{cccccccccc|cc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right) \end{array} \\ L^T = \begin{array}{c} \begin{array}{cccccccccc} l_0 & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & l_8 & l_9 & l_{10} \end{array} \\ \left(\begin{array}{cccccccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \end{cases}$$

$$D = \begin{array}{c} \\ j_0 \\ j_1 \\ j_2 \\ j_3 \\ j_4 \\ j_5 \\ j_6 \\ j_7 \\ j_8 \\ j_9 \\ j_{10} \\ j_{11} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} j_0 & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 & j_8 & j_9 & j_{10} & j_{11} \end{array} \\ \left(\begin{array}{cccccccccc|cc} 0 & l_{0,1} & l_{0,2} & 0 & l_{0,4} & 0 & 0 & 0 & 0 & l_{0,9} & 0 & 0 \\ l_{0,1} & 0 & l_{1,2} & l_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l_{0,2} & l_{1,2} & 0 & l_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & l_{1,3} & l_{2,3} & 0 & 0 & 0 & l_{3,6} & 0 & 0 & 0 & l_{3,10} & l_{3,11} \\ l_{0,4} & 0 & 0 & 0 & 0 & l_{4,5} & 0 & 0 & 0 & l_{4,9} & 0 & 0 \\ 0 & 0 & 0 & 0 & l_{4,5} & 0 & 0 & 0 & 0 & 0 & l_{5,10} & l_{5,11} \\ 0 & 0 & 0 & l_{3,6} & 0 & 0 & 0 & l_{6,7} & l_{6,8} & 0 & l_{6,10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{6,7} & 0 & l_{7,8} & l_{7,9} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & l_{6,8} & l_{7,8} & 0 & l_{8,9} & 0 & 0 \\ l_{0,9} & 0 & 0 & 0 & l_{4,9} & 0 & 0 & l_{7,9} & l_{8,9} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & l_{3,10} & 0 & l_{5,10} & l_{6,10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & l_{3,11} & 0 & l_{5,11} & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

The length of the triad, l_8, l_9 and l_{10} are computed according to Sec. 3 in the paper.

Static Link Merge: In the reduction phase, some of the links in the mechanism can be merged, resulting in less number of links. Furthermore, through this merge operation, two new binary links are found, which are $l_{0,3}$ and $l_{6,9}$. The subscripts of the new binary links represent the endpoint joint location of the links. Figure 7 shows the complete process.

Decomposition of System: At the beginning of this step, the actuator is converted into a dynamic link, which is $l_{0,5}$, as shown

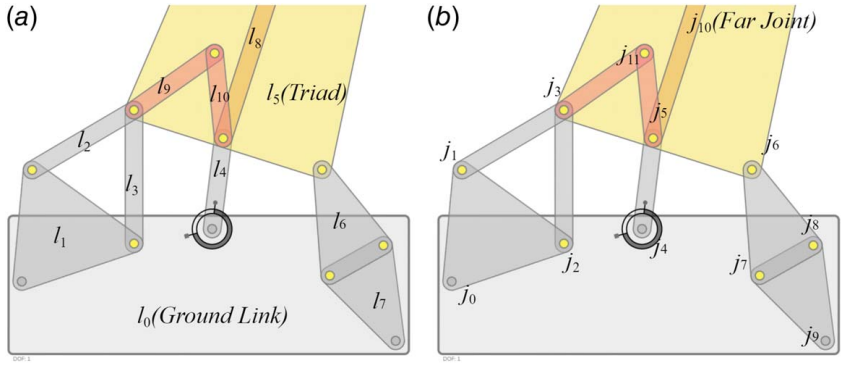

Fig. 6 This figure shows the notation of (*a*) joints and (*b*) links, respectively

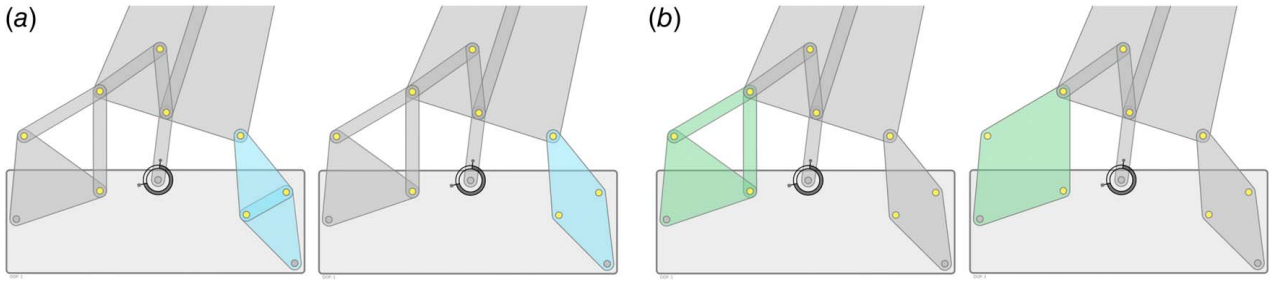

Fig. 7 This figure shows the merge process of links with two different rules: (*a*) l_6 and l_7 are merged because they share more than one joint and (*b*) the mobility formula shows the DOF of l_1, l_2, and l_3 is zero, meaning they can be merged into one rigid body

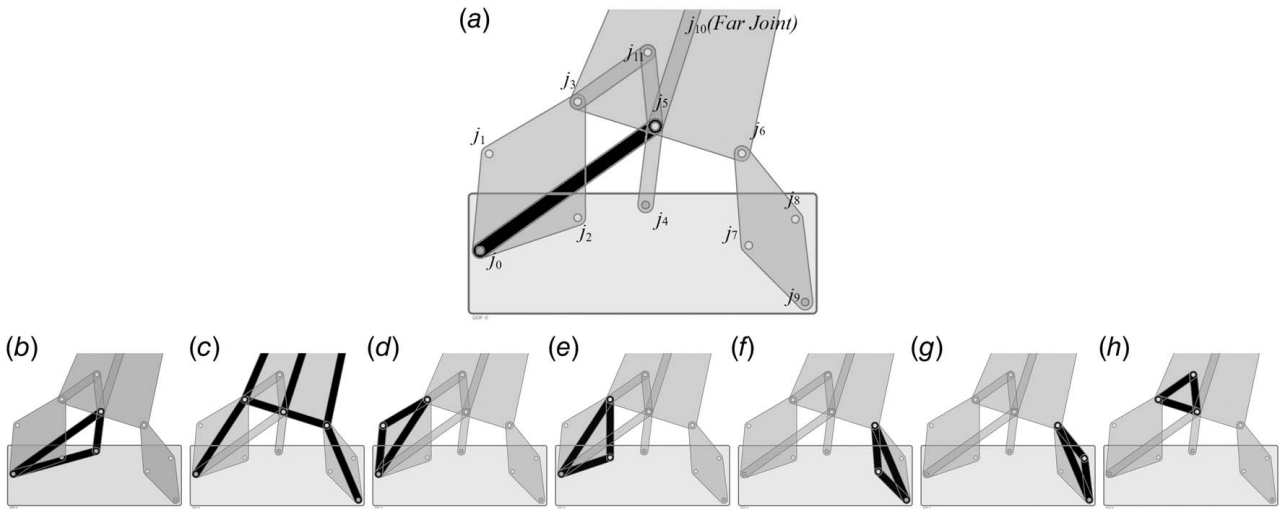

Fig. 8 (*a*) Initial state for the mechanism's decomposition process and (*b*)–(*h*) shows the decomposition process, where the used binary links are colored in dark/black for each step

in Fig. 8(*a*). The input angle is $\theta_{(0,4,5)}$, meaning it is the angle between vector $\vec{v}_{4,0}$ and $\vec{v}_{4,5}$, then the link's length is

$$l_{0,5} = \sqrt{l_{4,5}^2 + l_{4,0}^2 - 2 \times l_{4,0} \times l_{4,5} \times \cos\theta_{(0,4,5)}} \tag{32}$$

Following the conversion, the next step is to find the least number of joints to solve per step, as shown in Fig. 8. The decomposition starts from the known joints j_0, j_4, and j_9, as shown in Fig. 8(*a*).

First, in Fig. 8(*b*), j_5 can be solved with two ground joints (j_0, j_4) and two binary links $l_{0,5}$ and $l_{4,5}$. An actuator is within this triangle formed by j_0, j_4, and j_5. Therefore, a cross-product calculation is needed. The system of equations for this step is

$$\begin{cases} f_1 = \sqrt{(x_0 - x_5)^2 + (y_0 - y_5)^2} - l_{0,5} \\ f_2 = \sqrt{(x_4 - x_5)^2 + (y_4 - y_5)^2} - l_{4,5} \\ g_1 = (x_0 - x_4)(y_5 - y_4) - (x_5 - x_4)(y_0 - y_4) \\ \quad -l_{0,2} \cdot l_{0,4} \cdot \sin(\theta_{(0,4,5)}) \end{cases} \tag{33}$$

It is worth noting that a dynamic link does not replace its corresponding actuator. Instead, it is a necessary condition to satisfy the angle constraint. Although not shown in this example, a dynamic link can be useful sometimes to help decompose the system, i.e., giving more possible solving paths.

Next, in Fig. 8(*c*), the next solvable subassembly is a triad with three binary links. $l_{0,3}$ is a subcollection of the four-ary link

Table 1 This table specifies which joints are used to solve for the unknown joint with the arc intersection method for the last five steps

Step	Known joints	To solve	Need cross-product
d	j_0, j_3	j_1	Yes
e	j_0, j_3	j_2	Yes
f	j_6, j_9	j_7	Yes
g	j_6, j_9	j_8	Yes
h	j_3, j_5	j_{11}	No

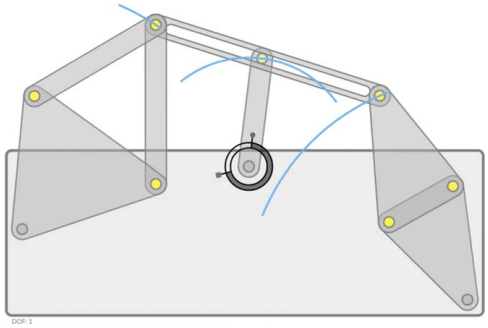

Fig. 9 Result of the positional analysis

formed by j_0, j_1, j_2, and j_3, and $l_{6,9}$ is a subcollection of the four-ary link formed by j_6, j_7, j_8, and j_9. The known joints that are used in this step are j_0, j_5, and j_9 and the joints to solve are j_3, j_6, and j_{10} (the far joint). The system of equations for this step is shown in Eq (34).

$$\begin{cases} f_1 = \sqrt{(x_0 - x_3)^2 + (y_0 - y_3)^2} - l_{0,3} \\ f_2 = \sqrt{(x_5 - x_{10})^2 + (y_5 - y_{10})^2} - l_{5,10} \\ f_3 = \sqrt{(x_6 - x_9)^2 + (y_6 - y_9)^2} - l_{6,9} \\ f_4 = \sqrt{(x_3 - x_6)^2 + (y_3 - y_6)^2} - l_{3,6} \\ f_5 = \sqrt{(x_3 - x_{10})^2 + (y_3 - y_{10})^2} - l_{3,10} \\ f_6 = \sqrt{(x_6 - x_{10})^2 + (y_6 - y_{10})^2} - l_{6,10} \end{cases} \tag{34}$$

The next steps from Figs. 8(*d*)–8(*h*) use the same logic: if two joints are solved and one unknown joint connects to these two joints, this unknown joint can be solved through arc intersection. Table 1 shows the details.

Solving the positions of the joints is in the positional analysis stage, where the actual input $\theta_{(0,4,5)}$ is given. Iteratively solving all these steps with different input values gives the path of all joints as shown in Fig. 9.

Complexity Analysis: In the decomposition phase, there are five steps that use dyadic decomposition with an angle constraint respectively (j_5, j_1, j_2, j_7, and j_8), one step that uses dyadic decomposition without the angle constraint (j_{11}), and one step that needs to solving for three joints. For simplicity, we use C_a and C_n to represent the

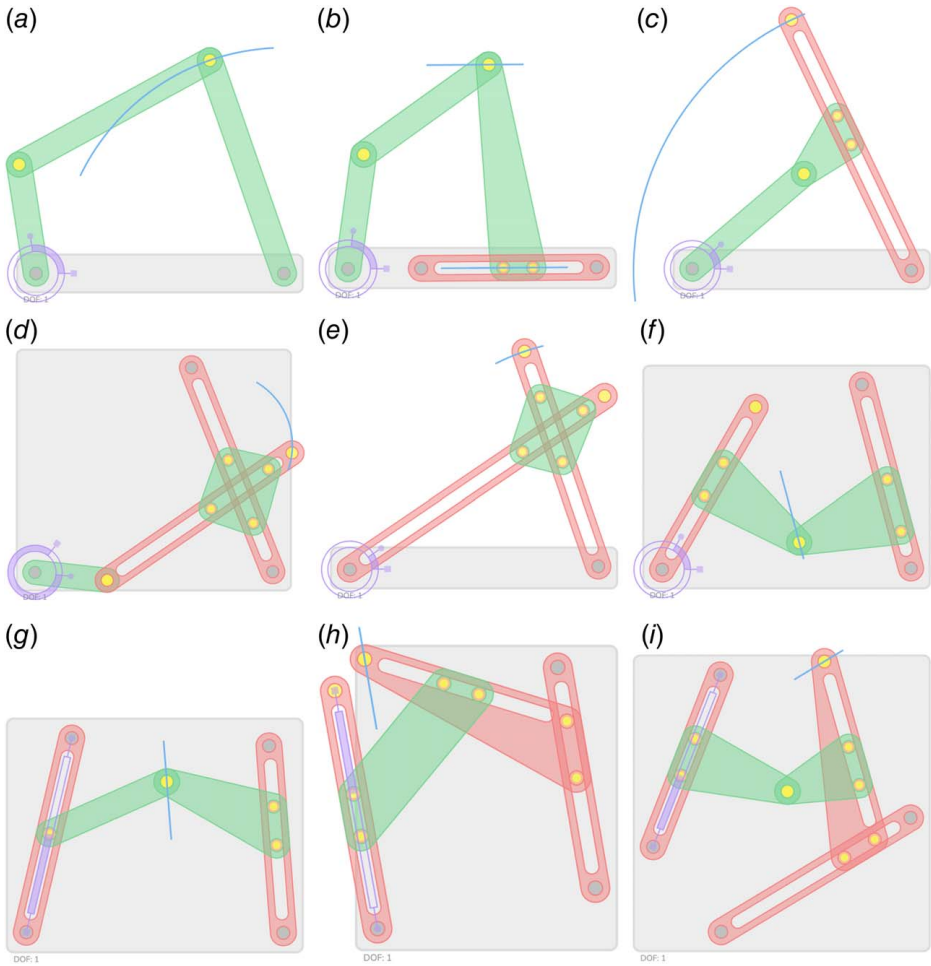

Fig. 10 All nine combinations of RR, PR, RP, and PP dyads are displayed. All nine mechanisms can be simulated successfully: (*a*) RRRR, (*b*) RRRP, (*c*) RRPR, (*d*) RRPP, (*e*) RPPR, (*f*) RPRP, (*g*) PRRP, (*h*) RPPP, and (*i*) PRPP.

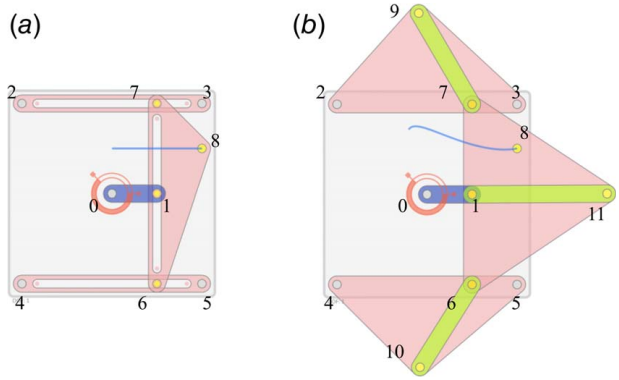

Fig. 11 (*a*) The multi-slot mechanism and (*b*) the interpretation of such a mechanism. The constraining joints and constraining links are not displayed for visual clarity.

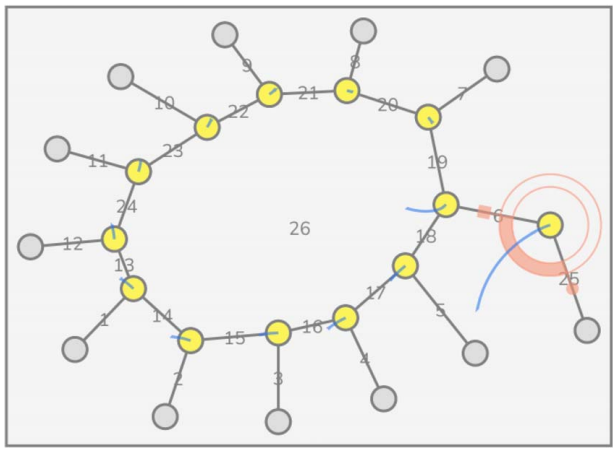

Fig. 12 This millipede-like mechanism can only be solved by an optimization method, and all movable joints have to be solved all at once. $l_1 \sim l_5$, l_6 together with the actuator and link l_{25}, and $l_7 \sim l_{12}$ are the "leg" links. Link $l_{13} \sim l_{24}$ are the "body" links. Link l_{26} is the ground link.

time constants for the arc intersection solver and Newton's solver (regardless of which one), respectively, i.e.,

$$C_i = \begin{cases} C_a & \text{If the arc intersection is used for the } i\text{-th step} \\ C_n & \text{If a Newton solver is used for the } i\text{-th step} \end{cases} \quad (35)$$

According to Eq. (30), the structural complexity is

$$(2^1 \times 3 \times 5 + 2^1 \times 2) \times C_a + (2 \times 3)^2 \times 6 \times C_n = 34C_a + 216C_n$$

7.2 Example 2: Mechanisms With RR, RP, PR, and PP Links. With the reinterpretation of prismatic joints, our algorithm can simulate all combinations of RR, RP, PR, and PP links, whether fixed or moving; see Fig. 10. Notably, a true slider in a slot is represented by two joints. This is how a prismatic joint is drawn in MOTIONGEN. Comparatively, the three-joint relationship for a slot joint and two end joints is equivalent to an RP link.

7.3 Example 3: A Multi-Slot Mechanism. Our algorithm supports moving slots and their combinations as well. Furthermore, a slot can be embedded in other slots. An example is shown in Fig. 11. For this mechanism, j_8 in the right figure shows that when the reinterpreted far joints (j_9, j_{10}, and j_{11}) are avoidably placed too close to the two end joints, the resulting path is curved and incurs large approximation error.

7.4 Example 4: Extremely Complex Structure. In this example, we present a complex kinematic structure, which requires

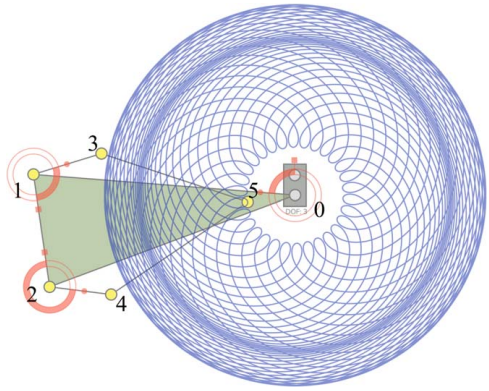

Fig. 13 A pantograph machine

Table 2 This table shows the statics of mechanisms

Mechanism	Estimated structural complexity	Time (# of simulated states)
RRRR	10 C_a	~4 ms (360)
RRRP	26 C_a	~4 ms (90)
RRPR	30 C_a	~4 ms (149)
RRPP	46 C_a	~4 ms (81)
RPPR	50 C_a	~3 ms (14)
RPRP	46 C_a	~4 ms (33)
PRRP	44 C_a	~5 ms (50)
RPPP	52 C_a + 216 C_n	~10 ms (90) for NR
PRPP in Fig. 10	60 C_a	~5 ms (93)
Multi-slot in Fig. 11	22 C_a + 216 C_n	~30 ms (360) for both NR and LM
Millipede in Fig. 12	17,576 C_n	400 ms (122) for NR 200 ms (122) for LM
Pantograph in Fig. 13	26 C_a	60 ms (14,400)

Note: The second column is computed according to Eq. (31). Here NR and LM represent Newton–Raphson and Levenberg–Marquardt algorithm, respectively. The simulation is done in MOTIONGEN and is coded using JAVASCRIPT.

that all the moving joints are solved simultaneously. Figure 12 shows such a mechanism, where there are 12 "legs" that connect with the ground and 12 serial links that each connect two legs one after another. In this case, all the distance constraints (links) and all the angle constraints (the actuator between l_6 and link l_{25}) need to be accounted for in the system of equations in order to solve for positions of movable joints. Additionally, removing any moving link will result in an under-constrained system. Therefore, this system cannot be further decomposed. Furthermore, this example also shows that an actuator can be placed on a moving joint and that an Assur group can be an infinite group because there can be n number of legs in a certain subassembly.

7.5 Example 5: Extremely Large Number of States. Pantographs are harmonic graphs that use electric motors and links to move a pen to create artistic drawings. Figure 13 is an example of such a machine. This mechanism is a rotating two-DOF five-bar mechanism with revolute joints only, where the path of j_5 is a long curve. This case is a simple system (only the arc intersection method is needed). However, it has many states (i.e., a big s in Eq. (30)). Therefore, the simulation time is comparatively long.

7.6 Time Complexity Analysis. Table 2 presents the time complexity statistics for the mechanisms demonstrated in Secs. 7.2– 7.5. Again, for simplicity, we use C_a and C_n to represent the time constants for the arc intersection solver and Newton's solver (regardless of which one), respectively.

From this table, it is clear that when a mechanism can be solved with the arc intersection solver only, the time to compute is short, usually less than 10 ms. The outlier is the Pantograph machine, which has a comparatively much larger number of states. If an optimization-based solver is used, the computation time is always more than 10 ms. Furthermore, its path is not long because the millipede structure is mechanically inefficient. However, it takes more than 200 ms for this small number of states.

We also tested the speed difference between the arc intersection solver and Newton's solvers for the same question (system of two/three equations). For the pantograph machine in Fig. 13, the LM takes around 1750 ms to finish, and the Newton–Raphson solver takes around 1450 ms to complete, compared to 60 ms for the arc intersection solver.

8 Conclusions

In this paper, we have introduced a simulation algorithm that reinterprets prismatic joints in planar linkage mechanisms as a combination of links and revolute joints. We provided representations of the modified mechanisms and showed that there are only two methods needed to perform positional analysis of any planar mechanism. One of the methods, dyadic decomposition is computationally efficient, but can perform simulation only on relatively simpler mechanisms, while the optimization-based method works in all of the other cases. Both of the methods leverage a unified form of geometric and actuation constraints to find the unknown positions of moving joints, while the error incurred in approximating prismatic elements with revolute joints and links is user-controllable. Additionally, this paper presented expressions for estimating simulation time and structural complexity based on mobility analysis. Several examples demonstrated the algorithm's efficiency and effectiveness in simulating mechanisms with various joint types and actuators. Moreover, this work highlights the conversion potential of Assur groups from mixed revolute joints and prismatic joints to exclusively revolute joints. Future extensions of this work include simulation of one and multi-degrees-of-freedom spatial and spherical mechanisms.

Acknowledgment

This work has been financially supported by the National Science Foundation under research grant STTR phase II #2126882 to co-author and Co-PI Purwar who also holds stocks in Mechanismic Inc. The research findings included in this publication may or may not necessarily relate to the interests of Mechanismic Inc. The terms of this arrangement have been reviewed and approved by Stony Brook University in accordance with its policy on objectivity in research.

All findings and results presented in this paper are those of the authors and do not represent those of the funding agencies.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Deshpande, S., and Purwar, A., 2019, "A Machine Learning Approach to Kinematic Synthesis of Defect-Free Planar Four-Bar Linkages," ASME J. Comput. Inf. Sci. Eng., **19**(2), p. 021004.

[2] Deshpande, S., and Purwar, A., 2019, "Computational Creativity Via Assisted Variational Synthesis of Mechanisms Using Deep Generative Models," ASME J. Mech. Des., **141**(12), p. 121402.

[3] Deshpande, S., and Purwar, A., 2020, "An Image-Based Approach to Variational Path Synthesis of Linkages," ASME J. Comput. Inf. Sci. Eng., **21**(2), p. 021005.

[4] Nobari, A. H., Srivastava, A., Gutfreund, D., and Ahmed, F., 2022, "LINKS: A Dataset of a Hundred Million Planar Linkage Mechanisms for Data-Driven Kinematic Design," Volume 3A: 48th Design Automation Conference (DAC), St. Louis, MO, Aug. 14–17, American Society of Mechanical Engineers, pp. 1–14.

[5] Regenwetter, L., Nobari, A. H., and Ahmed, F., 2022, "Deep Generative Models in Engineering Design: A Review," ASME J. Mech. Des., **144**(7), p. 071704.

[6] Yu, S.-C., Chang, Y., and Lee, J.-J., 2022, "A Generative Model for Path Synthesis of Four-Bar Linkages Via Uniform Sampling Dataset," Proc. Inst. Mech. Eng., Part C: J. Mech. Eng. Sci., **237**(4), p. 095440622211237.

[7] Vermeer, K., Kuppens, R., and Herder, J., 2018, "Kinematic Synthesis Using Reinforcement Learning," Volume 2A: 44th Design Automation Conference, Quebec, Canada, Aug. 26–29, American Society of Mechanical Engineers, pp. 1–12.

[8] Sharma, S., and Purwar, A., 2022, "A Machine Learning Approach to Solve the Alt-Burmester Problem for Synthesis of Defect-Free Spatial Mechanisms," ASME J. Comput. Inf. Sci. Eng., **22**(2), p. 021003.

[9] Khan, N., Ullah, I., and Al-Grafi, M., 2015, "Dimensional Synthesis of Mechanical Linkages Using Artificial Neural Networks and Fourier Descriptors," Mech. Sci., **6**(1), pp. 29–34.

[10] Purwar, A., and Chakraborty, N., 2023, "Deep Learning-Driven Design of Robot Mechanisms," ASME J. Comput. Inf. Sci. Eng., **23**(6), p. 060811.

[11] Erdman, A., and Gustafson, J., 1977, LINCAGES: Linkage INteractive Computer Analysis and Graphically Enhanced Synthesis Packages, Technical Report.

[12] Erdman, A. G., and Riley, D., 1981, "Computer-Aided Linkage Design Using the Lincages Package," ASME Design Engineering Technical Conferences, Paper No. 81-DET-121.

[13] Rubel, A. J., and Kaufman, R. E., 1977, "Kinsyn III: A New Human-Engineered System for Interactive Computer-Aided Design of Planar Linkages," ASME J. Eng. Ind., **99**(2), pp. 440–448.

[14] Kihonge, J., Vance, J., and Larochelle, P., 2001, "Spatial Mechanism Design in Virtual Reality With Networking," ASME 2001 Design Engineering Technical Conferences, Pittsburgh, PA, Sept. 9–12, pp. 1–8.

[15] Larochelle, P., 1998, "Spades: Software for Synthesizing Spatial 4C Linkages," CD-ROM Proceedings of the ASME DETC'98, Paper No. DETC98/Mech-5889.

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

[17] Ruth, D., and McCarthy, J., 1997, "Sphinxpc: An Implementation of Four Position Synthesis for Planar and Spherical 4R Linkages," ASME Design Engineering Technical Conferences, Sacramento, CA, Sept. 14–17.

[18] Tse, D., and Larochelle, P., 1999, "Osiris: A New Generation Spherical and Spatial Mechanism CAD Program," Florida Conference on Recent Advancements in Robotics, Gainesville, FL, Apr. 29–30.

[19] Su, H.-J., Collins, C., and McCarthy, J., 2002, "An Extensible Java Applet for Spatial Linkage Synthesis," ASME International Design Engineering Technical Conferences, Montreal, Quebec, Canada, Sept. 29–Oct. 2, pp. 1–5.

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

[21] Associates, M. D., "MechGen, http://mechanicaldesign101.com/mechanism-generator-2-0/#MechGen3.

[22] Norton Associates Engineering, "Linkages," http://www.designofmachinery.com/Linkage/index.html.

[23] SoftIntegration, Ch Mechanism Toolkit, http://www.softintegration.com/products/toolkit/mechanism/, http://www.softintegration.com/products/toolkit/mechanism/.

[24] Ltd., P. M., "MechDesigner," http://www.psmotion.com/.

[25] Laboratory of Computational Mechanics, R., Bryansk State Technical University, "Universal Mechanism".

[26] KCP Technologies, "The Geometer's Sketchpad," http://www.dynamicgeometry.com/.

[27] International GeoGebra Institute, "Geogebra," http://www.geogebra.org/cms/.

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

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

[30] Trevor Dowd, H. Z., Robert Dutile, "PMKS+," https://pmksplus.mech.website/, Accessed October 16, 2022.

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

[32] Simionescu, P.A., 2016, "MeKin2D: Suite for Planar Mechanism Kinematics," Volume 5B: 40th Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. v05BT07A083.

[33] Simionescu, P., 2004, "Enhancing Programming Skills to Engineering Students Using MeKin2D Modular Kinematics Subroutines".

[34] Schmidt, P., and Lax, P., 2018, "Use of Computer Coding to Teach Design in a Mechanics Course," Resulting in an Implementation of a Kinematic Mechanism Design Tool Using PYTHON.

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

[36] Hain, K., 1967, *Applied Kinematics*, McGraw Hill, New York.

[37] Dhingra, A. K., Almadi, A. N., and Kohli, D., 1999, "A Gröbner-Sylvester Hybrid Method for Closed-Form Displacement Analysis of Mechanisms," ASME J. Mech. Des., **122**(4), pp. 431–438.

[38] Wampler, C. W., 2000, "Solving the Kinematics of Planar Mechanisms by Dixon Determinant and a Complex-Plane Formulation," ASME J. Mech. Des., **123**(3), pp. 382–387.
[39] Wampler, C. W., 1999, "Solving the Kinematics of Planar Mechanisms," ASME J. Mech. Des., **121**(3), pp. 387–391.
[40] Nielsen, J., and Roth, B., 1999, "On the Kinematic Analysis of Robotic Mechanisms," Int. J. Rob. Res., **18**(12), pp. 1147–1160.
[41] Hernández, A., and Petuya, V., 2004, "Position Analysis of Planar Mechanisms With R-Pairs Using a Geometrical-Iterative Method," Mech. Mach. Theory, **39**(2), pp. 133–152.
[42] Javier García de Jalón, E. B., 2001, *Kinematic and Dynamic Simulation of Multibody Systems-The Real-Time Challenge*, Springer-Verlag, New York.
[43] Ait-Aoudia, S., Jegou, R., and Michelucci, D., 1993, *Reduction of Constraint Systems*, Compugraphics, Alvor, Algarve, Portugal, pp. 331–340.
[44] Ait-Aoudia, S., and Foufou, S., 2010, "A 2D Geometric Constraint Solver Using a Graph Reduction Method," Adv. Eng. Soft., **41**(10–11), pp. 1187–1194.
[45] Bouma, W., Fudos, I., Hoffmann, C., Cai, J., and Paige, R., 1995, "Geometric Constraint Solver," Comput. Aided Des., **27**(6), pp. 487–501.
[46] Fudos, I., and Hoffmann, C., 2004, "A Graph-Constructive Approach to Solving Systems of Geometric Constraints," ACM Trans. Graph., **16**(2), pp. 179–216.
[47] Simionescu, P., 2014, Computer-Aided Graphing and Simulation Tools for AutoCAD Users.
[48] Tsai, L., 2001, *Mechanism Design: Enumeration of Kinematic Structures According to Function*, CRC Press LLC, Boca Raton, FL.
[49] Yamamoto, T., Iwatsuki, N., and Ikeda, I., 2020, "Automated Kinematic Analysis of Closed-Loop Planar Link Mechanisms," Machines, **8**(3).
[50] Moré, J. J., 1978, The Levenberg-Marquardt Algorithm: Implementation and Theory, G. A. Watson, ed., *Numerical Analysis*, Springer, Berlin, Heidelberg, pp. 105–116.
[51] Galletti, C., and Giannotti, E., 2009, Assur's-Groups-Based Simulation for Teaching Kinematics of Planar Linkages.
[52] Galletti, C. U., 1979, "On the Position Analysis of Assur's Groups of High Class," Meccanica, **14**(1), pp. 6–10.
[53] Gogu, G., 2005, "Chebychev–Grübler–Kutzbach's Criterion for Mobility Calculation of Multi-Loop Mechanisms Revisited Via Theory of Linear Transformations," Eur. J. Mech. - A/Solids, **24**(3), pp. 427–441.