Article

Synthesis of Four-Link Initial Kinematic Chains with Spherical Pairs for Spatial Mechanisms

Algazy Zhauyt * , Kuanysh Alipbayev , Serikbay Kosbolov , Alisher Aden , Aray Orazaliyeva ,
Samal Abdreshova

Posted Date: 14 February 2025

doi: 10.20944/preprints202502.1022.v1

Keywords: visualization; transformation; spatial mechanism; synthesis; approximation

Article

Synthesis of Four-Link Initial Kinematic Chains with Spherical Pairs for Spatial Mechanisms

Algazy Zhauyt [1,*], Kuanysh Alipbayev [1], Serikbay Kosbolov [2], Alisher Aden [1], Aray Orazaliyeva [1] and Samal Abdreshova [1]

[1] Department of Electronic Engineering, Almaty University of Power Engineering and Telecommunications named after Gumarbek Daukeyev, Almaty 050013, Kazakhstan; k.alipbayev@aues.kz (K.A.); a.aden@aues.kz (A.A.); samalNur_777@mail.ru (S.A.)

[2] Department of Aerospace Engineering, Almaty University of Power Engineering and Telecommunications named after Gumarbek Daukeyev, Almaty 050013, Kazakhstan; s.kosbolov@aues.kz (S.K.); a.orazaliyeva@aues.kz (A.O.)

* Correspondence: a.zhauyt@aues.kz; Tel.: +77053256860

Abstract: This research addresses the problem of initial synthesis of kinematic chains with spherical kinematic pairs, which are essential in the design of spatial mechanisms used in robotics, aerospace, and mechanical systems. The goal is to establish the existence of solutions for defining the geometric and motion constraints of these kinematic chains, ensuring that the synthesized mechanism achieves the desired motion with precision. By formulating the synthesis problem in terms of nonlinear algebraic equations derived from the spatial positions and orientations of the links, we analyze the conditions under which a valid solution exists. We explore both analytical and numerical methods to solve these equations, highlighting the significance of parameter selection in determining feasible solutions. The study further investigates the impact of initial conditions and design parameters on the stability and flexibility of the synthesized kinematic chain. The findings provide a theoretical foundation for guiding the practical design of spatial mechanisms with spherical joints, ensuring accuracy and reliability in complex motion tasks. This work presents a comprehensive framework for the 3D visualization of geometric transformations and coordinate relationships using Python 3.13.0. Leveraging the capabilities of libraries such as NumPy and Matplotlib, we develop a series of modular code examples that illustrate how to plot and analyze multidimensional data pertinent to kinematic chain synthesis and robotic mechanisms. Specifically, our approach demonstrates the visualization of fixed points, such as X_A, Y_A, Z_A, x_B, y_B, z_B, and x_C, y_C, z_C, alongside their spatial differences with respect to reference points and transformation matrices. We detail methods for plotting transformation components, including rotation matrix elements (e, m, n) and derived products from these matrices, as well as the representation of angular parameters (θ_i, ψ_i, φ_i) in a three-dimensional context. The proposed techniques not only facilitate the debugging and analysis of complex kinematic behaviors but also provide a flexible tool for researchers in robotics, computer graphics, and mechanical design. By offering a clear and interactive visualization strategy, this framework enhances the understanding of spatial relationships and transformation dynamics inherent in multi-body systems.

Keywords: visualization; transformation; spatial mechanism; synthesis; approximation

1. Introduction

The synthesis of kinematic chains plays a crucial role in the design of spatial mechanisms, which are widely used in industries such as robotics, aerospace, and automotive engineering [1]. One particular type of spatial mechanism employs Spherical-Spherical-Spherical (SSS) kinematic pairs, which allow for three degrees of rotational freedom. These pairs are vital in creating complex motion paths and orientations, making them ideal for applications that require high precision and flexibility in movement. In kinematic chain synthesis, the goal is to design a mechanism that meets specific motion requirements while adhering to geometric constraints. For spatial mechanisms with SSS pairs, this involves determining the optimal configuration of the links and joints that will ensure the desired motion trajectory and orientation [2]. However, the synthesis of such mechanisms is challenging due to the complexity of the spatial relationships and the nonlinear nature of the equations governing their motion [3]. The problem of synthesizing kinematic chains with SSS pairs can be divided into two primary tasks: (1) defining the geometry of the links and joints, and (2) solving the kinematic equations to ensure the mechanism achieves the required motion. This process often involves solving systems of nonlinear equations, which describe the spatial positioning of the links relative to one another. Furthermore, the synthesis must take into account various constraints, such as the range of motion, joint limits, and load-bearing capacity, all of which can affect the mechanism's performance [4]. This research aims to address the synthesis problem by providing a systematic approach to designing kinematic chains with SSS pairs. We focus on developing methods to solve the initial synthesis problem, ensuring that the mechanism's geometry and motion capabilities align with the desired performance criteria [5]. By leveraging both analytical and computational techniques, we seek to establish reliable solutions for these complex spatial mechanisms, offering insights into their design and optimization. In the following sections, we review existing methods for kinematic chain synthesis [6], propose new techniques for addressing the challenges specific to SSS pairs, and demonstrate the practical applications of our approach through case studies. This work contributes to the advancement of spatial mechanism design, providing tools for engineers to create more efficient and precise systems in various fields. The rapid advancement of robotics, computer graphics, and mechanical design has underscored the importance of accurate and intuitive visualization of geometric transformations and kinematic relationships [7]. In multi-body systems and robotic mechanisms, the ability to effectively analyze and debug complex spatial interactions is paramount. This study presents a comprehensive framework for three-dimensional (3D) visualization of coordinate transformations, enabling researchers and practitioners to gain insights into the behavior of kinematic chains and transformation matrices [8]. A central challenge in kinematic analysis is the representation of the spatial relationships between different coordinate frames. These relationships are often described by transformation matrices whose components depend on rotational parameters such as Euler angles θ_i, ψ_i, φ_i. Understanding how these angles influence the resultant vectors and the overall motion of the system is critical for tasks such as inverse kinematics, dynamic simulation [9], and control. Our approach leverages Python's powerful numerical and plotting libraries-NumPy and Matplotlib-to generate detailed 2D and 3D visualizations that depict the relationships between fixed points X_A, Y_A, Z_A; x_B, y_B, z_B; x_C, y_C, z_C; and R and their corresponding transformation parameters [10]. In this work, we not only demonstrate the plotting of individual points but also explore the visualization of derived quantities such as the differences between coordinates $X_{Di}-X_A$, $Y_{Di}-Y_A$, $Z_{Di}-Z_A$, as well as the components of transformation matrices (e, m, n) computed from rotational parameters. Moreover, we illustrate the visualization of composite transformations obtained through matrix multiplication $T_{21}^i = T_{01}^i \times T_{20}^i$ and $T_{12}^i = T_{02}^i \times T_{10}^i$. These visualizations provide a powerful tool for interpreting the intricate geometric relationships inherent in kinematic chain synthesis and robotic system design. By integrating these visualization techniques into a cohesive framework, our methodology enables users to interactively analyze and validate the performance of complex spatial transformations [11]. This work aims to contribute to the field by offering a flexible and accessible approach to 3D visualization, ultimately enhancing the understanding and debugging of multi-dimensional transformation processes in various engineering applications. In the sections that follow, we detail the mathematical foundations underlying the transformation matrices, describe the implementation of the visualization routines in Python, and present several case studies that demonstrate the practical utility of our approach.

2. Materials and Methods

The problem is describing involves finding specific points on two solids such that the distance between those points in all given positions remains as close as possible to a constant value R. To approach this, we have formulated a weighted difference function that will be minimized over the given positions of the solids [12]. Here is a breakdown of the solution process:

Given the position N of the two solids Q_1 and Q_2, we have the Eulerian angles for solid Q_1 and coordinates X_{Di}, Y_{Di}, Z_{Di} and angles for solid Q_2. We need to find points $A(X_A, Y_A, Z_A)$ on solid Q_1, $B(x_B, y_B, z_B)$ on solid Q_1, and $C(x_C, y_C, z_C)$ on solid Q_2, such that the distance $|B_iC_i|$ is close to a constant R for all positions as shown in Figure 1. Minimize the difference Δqi for each position, which is expressed as:

$$\Delta_{q_i} = \left|\overrightarrow{B_iC_i}\right|^2 - R^2 = (X_{C_i} - X_{B_i})^2 + (Y_{C_i} - Y_{B_i})^2 + (Z_{C_i} - Z_{B_i})^2 - R^2, \ \left(i = \overrightarrow{1,N}\right) \quad (1)$$

where x_{Bi}, y_{Bi}, z_{Bi} are the coordinates of point B in the position i on solid Q_1, and x_{Ci}, y_{Ci}, z_{Ci} are the coordinates of point C in the position i on solid Q_2,

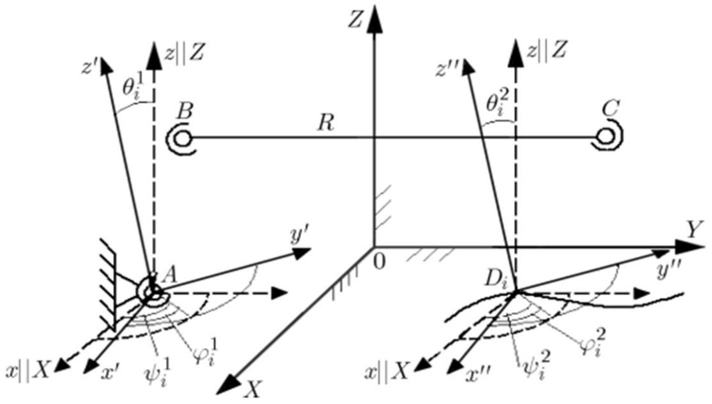


Figure 1. Kinematical schema of a three-link open chain ABCD.

The matrix T_{jo}^{i} that is composed of directional cosines (or transformation matrices) that relate the local coordinate systems of each solid Q_1 and Q_2 to a fixed coordinate system. The matrix itself is built from the Euler angles θ_i^j, ψ_i^j, φ_i^j of each solid in each position i, where $j=1,2$ refers to the two solids [13]. The components of the matrix are structured as follows:

$$T_{j0}^{i} = \begin{bmatrix} e_{i1}^{j} & e_{i2}^{j} & e_{i3}^{j} \\ m_{i1}^{j} & m_{i2}^{j} & m_{i3}^{j} \\ n_{i1}^{j} & n_{i2}^{j} & n_{i3}^{j} \end{bmatrix}, (j = \overline{1{,}2}; i = \overline{1,N}) \quad (2)$$

where e_{i1}^{j}, e_{i2}^{j}, e_{i3}^{j} is the components of the first row (often representing the direction cosines of the local x-axis in the fixed coordinate system); m_{i1}^{j}, m_{i2}^{j}, m_{i3}^{j} is the components of the second row (typically representing the direction cosines of the local y-axis in the fixed coordinate system); n_{i1}^{j}, n_{i2}^{j}, n_{i3}^{j}, is the represent the components of the third row (usually the direction cosines of the local z-axis in the fixed coordinate system).

These represent the direction cosines of the local x-axis relative to the fixed axes:

$$e_{i1}^{j} = cos\,\psi_i^j \cdot cos\,\varphi_i^j - cos\,\theta_i^j \cdot sin\,\psi_i^j \cdot sin\,\varphi_i^j$$
$$e_{i2}^{j} = -\,cos\,\psi_i^j \cdot sin\,\varphi_i^j - cos\,\theta_i^j \cdot sin\,\psi_i^j \cdot cos\,\varphi_i^j \quad (3)$$
$$e_{i3}^{j} = sin\,\theta_i^j \cdot sin\,\psi_i^j$$

These represent the direction cosines of the local y-axis relative to the fixed axes:

$$m_{i1} = sin\,\psi_i^j \cdot cos\,\varphi_i^j + cos\,\theta_i^j \cdot cos\,\psi_i^j \cdot cos\,\varphi_i^j$$
$$m_{i2}^{j} = -\,sin\,\psi_i^j \cdot sin\,\varphi_i^j + cos\,\theta_i^j \cdot cos\,\psi_i^j \cdot sin\,\varphi_i^j \quad (4)$$
$$m_{i3}^{j} = -\,sin\,\theta_i^j \cdot cos\,\psi_i^j$$

These represent the direction cosines of the local z-axis relative to the fixed axes:

$$n_{i1}^{j} = sin\,\theta_i^j \cdot sin\,\varphi_i^j$$

$$n_{i2}^{j} = sin\,\theta_i^j \cdot cos\,\varphi_i^j \quad (5)$$

$$n_{i3}^{j} = cos\,\varphi_i^j$$

The matrix T_{jo}^{i} essentially transforms the orientation of solid *Q*1 or *Q*2 in a fixed coordinate system. Each solid's orientation is represented using the Euler angles (or rotation matrix) to map the local axes to the global (fixed) coordinate system [14]. The three rows of the matrix represent the direction cosines of the local axes relative to the global axes. This matrix is likely part of a kinematic analysis where the positions and orientations of two solids need to be related to a common coordinate system [15]. The task is then to use this matrix to calculate the positions of points on the solids, transforming them between the local and global frames. The matrices T_{jo}^{i} and their transposed versions represent the rotation matrices that transform the coordinate system of one solid to another. Let's break down each part of equations and matrices:

$T_{01}^{i} = \left[T_{10}^{i}\right]^T$ and $T_{02}^{i} = \left[T_{20}^{i}\right]^T$ are the transposed versions of T_{10}^{i} and T_{20}^{i}, respectively. These matrices represent the rotation transformation from the local coordinate frame of solid *Q*1 and *Q*2 to a fixed global coordinate frame at each position *i*. Each of these matrices contains the direction cosines for the three local axes of the solids. In matrix form, we have:

$$T_{01}^{i} = \left[T_{10}^{i}\right]^T = \begin{bmatrix} e_{i1}^1 & m_{i1}^1 & n_{i1}^1 \\ e_{i2}^1 & m_{i2}^1 & n_{i2}^1 \\ e_{i3}^1 & m_{i3}^1 & n_{i3}^1 \end{bmatrix} \quad (6)$$

$$T_{02}^{i} = \left[T_{20}^{i}\right]^T = \begin{bmatrix} e_{i1}^2 & m_{i1}^2 & n_{i1}^2 \\ e_{i2}^2 & m_{i2}^2 & n_{i2}^2 \\ e_{i3}^2 & m_{i3}^2 & n_{i3}^2 \end{bmatrix} \quad (7)$$

where the components e_{i1}, m_{i1}, n_{i1} etc., are the direction cosines of the respective axes relative to the global coordinate frame.

The matrix T_{21}^{i} represents the transformation of the coordinate frame of *Q*2 relative to *Q*1. It is the result of multiplying the matrices T_{01}^{i} and T_{20}^{i}. The components of the resulting matrix are:

$$T_{21}^{i} = \begin{bmatrix} e_{i1}^1 e_{i1}^2 + e_{i2}^1 m_{i1}^2 + e_{i3}^1 n_{i1}^2 & e_{i1}^1 e_{i2}^2 + e_{i2}^1 m_{i2}^2 + e_{i3}^1 n_{i2}^2 & e_{i1}^1 e_{i3}^2 + e_{i2}^1 m_{i3}^2 + e_{i3}^1 n_{i3}^2 \\ m_{i1}^1 e_{i1}^2 + m_{i2}^1 m_{i1}^2 + m_{i3}^1 n_{i1}^2 & m_{i1}^1 e_{i2}^2 + m_{i2}^1 m_{i2}^2 + m_{i3}^1 n_{i2}^2 & m_{i1}^1 e_{i3}^2 + m_{i2}^1 m_{i3}^2 + m_{i3}^1 n_{i3}^2 \\ n_{i1}^1 e_{i1}^2 + n_{i2}^1 m_{i1}^2 + n_{i3}^1 n_{i1}^2 & n_{i1}^1 e_{i2}^2 + n_{i2}^1 m_{i2}^2 + n_{i3}^1 n_{i2}^2 & n_{i1}^1 e_{i3}^2 + n_{i2}^1 m_{i3}^2 + n_{i3}^1 n_{i3}^2 \end{bmatrix} \quad (8)$$

This matrix is formed by performing the dot products of the rows of T_{01}^{i} with the columns of T_{02}^{i}. The result is a 3x3 matrix that represents the relative orientation between *Q*2 and *Q*1.

Similarly, the matrix T_{12}^{i} represents the transformation of the coordinate frame of *Q*1 relative to *Q*2, and is the result of multiplying T_{02}^{i} and T_{10}^{i}. The components of T_{12}^{i} are:

$$T_{12}^{i} = \begin{bmatrix} e_{i1}^2 e_{i1}^1 + m_{i1}^2 m_{i1}^1 + n_{i1}^2 n_{i1}^1 & e_{i1}^2 e_{i2}^1 + m_{i1}^2 m_{i2}^1 + n_{i1}^2 n_{i2}^1 & e_{i1}^2 e_{i3}^1 + m_{i1}^2 m_{i3}^1 + n_{i1}^2 n_{i3}^1 \\ e_{i2}^2 e_{i1}^1 + m_{i2}^2 m_{i1}^1 + n_{i2}^2 n_{i1}^1 & m_{i2}^2 e_{i2}^1 + m_{i2}^2 m_{i2}^1 + m_{i2}^2 n_{i2}^1 & m_{i2}^2 e_{i3}^1 + m_{i2}^2 m_{i3}^1 + m_{i2}^2 n_{i3}^1 \\ n_{i3}^2 e_{i1}^1 + n_{i3}^2 m_{i1}^1 + n_{i3}^2 n_{i1}^1 & n_{i3}^2 e_{i2}^1 + n_{i3}^2 m_{i2}^1 + n_{i3}^2 n_{i2}^1 & n_{i3}^2 e_{i3}^1 + n_{i3}^2 m_{i3}^1 + n_{i3}^2 n_{i3}^1 \end{bmatrix} \quad (9)$$

T_{01}^{i} and T_{02}^{i} represent the rotation matrices (transposed) from each solid's local frame to the fixed coordinate system. T_{12}^{i} and T_{21}^{i} represent the relative transformations between the two solids' local frames, where T_{21}^{i} transforms *Q*2 local frame to *Q*1 local frame and T_{12}^{i} does the reverse. These matrices are essential in solving for the positions and distances between points on the two solids, as they allow you to transform coordinates between different reference frames and to express the relative motion or position between the solids. The matrix system performs transformations between different coordinate frames using rotation matrices and vector translations [16], which are relative to the positions of points on the rigid bodies *Q*1 and *Q*2 in the fixed coordinate system. Here is a breakdown of the matrices and their transformations:

The matrix equation for point *A* is:

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \\ 1 \end{bmatrix} = \begin{bmatrix} e_{i1}^1 & e_{i2}^1 & e_{i3}^1 & 0 \\ m_{i1}^1 & m_{i2}^1 & m_{i3}^1 & 0 \\ n_{i1}^1 & n_{i2}^1 & n_{i3}^1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} + \begin{bmatrix} e_{i1}^2 & e_{i2}^2 & e_{i3}^2 & X_{Di} \\ m_{i1}^2 & m_{i2}^2 & m_{i3}^2 & Y_{Di} \\ n_{i1}^2 & n_{i2}^2 & n_{i3}^2 & Z_{Di} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \quad (10)$$

This equation expresses the position of point *A* as the result of first applying a rotation defined by T_{01}^{i} (for solid *Q*1) and then translating by the coordinates of point *D* on solid *Q*2.

For point *B*, the matrix equation is:

$$\begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} e_{i1}^1 & e_{i2}^1 & e_{i3}^1 & 0 \\ m_{i1}^1 & m_{i2}^1 & m_{i3}^1 & 0 \\ n_{i1}^1 & n_{i2}^1 & n_{i3}^1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{Di} - X_A \\ Y_{Di} - Y_A \\ Z_{Di} - Z_A \\ 1 \end{bmatrix} + [rotation\ matrix\ terms] \times \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \qquad (11)$$

This expression represents a combined rotation and translation for point *B* using the relative position of *D* and the transformation from Q_2 to Q_1.

The equation for point *C* is:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} = \begin{bmatrix} e_{i1}^2 & e_{i2}^2 & e_{i3}^2 & 0 \\ m_{i1}^2 & m_{i2}^2 & m_{i3}^2 & 0 \\ n_{i1}^2 & n_{i2}^2 & n_{i3}^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_A - X_{Di} \\ Y_A - Y_{Di} \\ Z_A - Z_{Di} \\ 1 \end{bmatrix} + [rotation\ matrix\ terms] \times \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} \qquad (12)$$

This represents a similar transformation for point *C*, which involves both a translation and rotation between the frames of Q_1 and Q_2. Each of these matrices is a combination of rotation matrices (representing the orientation of the rigid bodies Q_1 and Q_2 relative to the global coordinate frame) and translation matrices (replacing the initial position of the rigid bodies). The purpose of these transformations is to calculate the relative positions of points *A*, *B*, and *C* on the rigid bodies Q_1 and Q_2 based on the coordinate transformations [17]. These transformations take into account both the orientation and position of the rigid bodies in space, using Euler angles for rotations and coordinates for translations.

So, for the weighted differences $\Delta q_i^{(1)}$, $\Delta q_i^{(2)}$, $\Delta q_i^{(3)}$ as functions of the parameters X_A, Y_A, Z_A we divide the objective function into three separate forms x_B, y_B, z_B, x_C, y_C, z_C, and *R*.

Weighted difference for point *A*:

$$\Delta q_i^{(1)} = (X_{Ai} - X_A)^2 + (Y_{Ai} - Y_A)^2 + (Z_{Ai} - Z_A)^2 - R^2 \qquad (13)$$

Weighted difference for point *B*:

$$\Delta q_i^{(2)} = (X_{Bi} - X_B)^2 + (Y_{Bi} - Y_B)^2 + (Z_{Bi} - Z_B)^2 - R^2 \qquad (14)$$

Weighted difference for point *C*:

$$\Delta q_i^{(3)} = (X_{Ci} - X_C)^2 + (Y_{Ci} - Y_C)^2 + (Z_{Ci} - Z_C)^2 - R^2 \qquad (15)$$

The total objective function to minimize, which is the sum of the squared differences, can be written as:

$$S = \sum_{i=1}^{N} \left[\left(\Delta q_i^{(1)}\right)^2 + \left(\Delta q_i^{(2)}\right)^2 + \left(\Delta q_i^{(3)}\right)^2 \right] \qquad (16)$$

The sum here is taken over all *N* discrete positions of the solids. Since each $\Delta q_i^{(k)}$ (for *k=1,2,3*) is expressed in terms of X_A, Y_A, Z_A, x_B, y_B, z_B, x_C, y_C, z_C, and *R*, the objective function *S* is a function of these ten parameters. We represent the weighted differences as groups of parameters with a common parameter *R* into four sets. For each of the three members, the variables X_A, Y_A, Z_A (for point *A*), x_B, y_B, z_B (for point *B*), and x_C, y_C, z_C (for point *C*) are grouped together and their corresponding coordinates X_A, Y_A, Z_A, x_B, y_B, z_B, and x_C, y_C, z_C are scaled relative to the common radius *R*. To find the optimal positions and radius that minimize *S*, you need to compute the partial derivatives of *S* with respect to each parameter (i.e., the positions and *R*), and set these derivatives equal to zero. The total objective function is:

$$S = \sum_{i=1}^{N} [((X_{Ai} - X_A)^2 + (Y_{Ai} - Y_A)^2 + (Z_{Ai} - Z_A)^2 - R^2)^2 + ((X_{Bi} - X_B)^2 + (Y_{Bi} - Y_B)^2 + (Z_{Bi} - Z_B)^2 - R^2)^2 + ((X_{Ci} - X_C)^2 + (Y_{Ci} - Y_C)^2 + (Z_{Ci} - Z_C)^2 - R^2)^2] \qquad (17)$$

Now, to minimize *S*, we calculate the partial derivatives of *S* with respect to each of the ten parameters X_A, Y_A, Z_A, x_B, y_B, z_B, x_C, y_C, z_C, and *R*, and set them equal to zero:

$$\frac{\partial S}{\partial X_A} = 0; \quad \frac{\partial S}{\partial Y_A} = 0; \quad \frac{\partial S}{\partial Z_A} = 0; \quad \frac{\partial S}{\partial R} = 0;$$

$$\frac{\partial S}{\partial x_B} = 0; \quad \frac{\partial S}{\partial y_B} = 0; \quad \frac{\partial S}{\partial z_B} = 0; \quad \frac{\partial S}{\partial R} = 0;$$

$$\frac{\partial S}{\partial x_C} = 0; \quad \frac{\partial S}{\partial y_C} = 0; \quad \frac{\partial S}{\partial z_C} = 0; \quad \frac{\partial S}{\partial R} = 0 \qquad (18)$$

To solve the system of equations formed by setting these partial derivatives to zero, you need to:

we expand each of the partial derivatives, differentiating the sum of the square terms for each point with respect to the corresponding parameter;

to simplify the resulting expressions, after differentiation, we obtain a set of linear equations for each parameter (the ten parameters being optimized);

the system of equations can be solved numerically using methods such as Gaussian elimination, least squares, or gradient descent optimization algorithms.

We are now working with a set of equations with weighted differences $\Delta q_i^{(1)}$ for each i. These equations aim to find the optimal values of X_A, Y_A, Z_A, R using the following system:

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}(X_{Ai}-X_A)=0$ (19)

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}(Y_{Ai}-Y_A)=0$ (20)

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}(Z_{Ai}-Z_A)=0$ (21)

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}R=0$ (22)

These equations are essentially normal equations that come from minimizing the sum of squared weighted differences S. Each equation corresponds to a specific parameter (either X_A, Y_A, Z_A, or R) that you are trying to optimize [18]. Each equation represents the sum of weighted differences between the coordinates X_A, Y_A, Z_A and their respective target values X_{Ai}, Y_{Ai}, Z_{Ai}, weighted by the difference function $\Delta q_i^{(1)}$. The goal is to adjust the coordinates so that the sum of these weighted differences is minimized (i.e., set each sum to zero).

This equation sums the weighted differences over all positions. Since $\Delta q_i^{(1)}$ is a function of X_A, Y_A, Z_A, and R, this equation ensures that the weighted differences balance out when summed across all positions:

$$\sum_{i=1}^{N}\Delta_{qi}^{(1)}=0$$

These equations ensure that the weighted differences $\Delta q_i^{(1)}$ are balanced with respect to the coordinates X_A, Y_A, Z_A. Each equation adjusts one of the coordinates of point A to minimize the sum of the squared weighted differences.

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}X_{Ai}=0$ (23)

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}Y_{Ai}=0$ (24)

$\sum_{i=1}^{N}\Delta_{qi}^{(1)}Z_{Ai}=0$ (25)

$$\sum_{i=1}^{N}\Delta_{qi}^{(1)}R=0$$

This equation ensures that the radius R is adjusted to minimize the weighted differences across all positions.

We propose a system of equations involving sums of square terms and cross products for the measured differences. The goal is to simplify these conditions to find the optimal values of the parameters. Let's consider the structure of the equations step by step. This equation simplifies terms involving the coordinates X_A, Y_A, Z_A, and the radius R, and is related to the weighted sum of squared distances:

$\sum_{i=1}^{N}\left[X_{Ai}^2X_A+X_{Ai}Y_{Ai}Y_A+Z_{Ai}X_{Ai}Z_A+\frac{1}{2}(R^2-X_A^2-Y_A^2-Z_A^2)X_{Ai}\right]=\frac{1}{2}\sum_{i=1}^{N}(X_{Ai}^2+Y_{Ai}^2+Z_{Ai}^2)X_{Ai}$(26)

This equation represents a similar relationship for the Y_A coordinates:

$\sum_{i=1}^{N}\left[X_{Ai}Y_{Ai}X_A+Y_{Ai}^2Y_A+Z_{Ai}Y_{Ai}Z_A+\frac{1}{2}(R^2-X_A^2-Y_A^2-Z_A^2)Y_{Ai}\right]=\frac{1}{2}\sum_{i=1}^{N}(X_{Ai}^2+Y_{Ai}^2+Z_{Ai}^2)Y_{Ai}$ (27)

This equation applies the same principle to the Z_A coordinates:

$\sum_{i=1}^{N}\left[Z_{Ai}X_{Ai}X_A+Y_{Ai}Z_{Ai}Y_A+Z_{Ai}^2Z_A+\frac{1}{2}(R^2-X_A^2-Y_A^2-Z_A^2)Z_{Ai}\right]=\frac{1}{2}\sum_{i=1}^{N}(X_{Ai}^2+Y_{Ai}^2+Z_{Ai}^2)Z_{Ai}$(28)

This equation represents the relationship between the coordinates and the sum of squared values:

$\sum_{i=1}^{N}\left[X_{Ai}X_A+Y_{Ai}Y_A+Z_{Ai}Z_A+\frac{1}{2}(R^2-X_A^2-Y_A^2-Z_A^2)\right]=\frac{1}{2}\sum_{i=1}^{N}(X_{Ai}^2+Y_{Ai}^2+Z_{Ai}^2)$ (29)

This equation expresses a term H_1 in terms of the radius R and coordinates X_A, Y_A, Z_A.

$H_1=\frac{1}{2}(R^2-X_A^2-Y_A^2-Z_A^2)$ (30)

These equations are designed to relate the parameters X_A, Y_A, Z_A, R by simplifying the weighted sums over the N positions of the solids. To solve them, you'll typically:

we substitute the expressions for each term and simplify the sum at position N;

we solve the resulting system of linear equations to find the values of X_A, Y_A, Z_A, and R.

Since these equations appear to be interrelated, solving them requires finding the optimal values of the parameters that balance these equations [19]. This matrix contains the sums of the product terms for X_A, Y_A, Z_A, as well as their sums individually. These sums correspond to the moments of the data (coordinates X_A, Y_A, Z_A) over the N positions. The system of equations is given by:

$$\begin{bmatrix} \sum_{i=1}^{N} X_{Ai}^2 & \sum_{i=1}^{N} X_{Ai}Y_{Ai} & \sum_{i=1}^{N} X_{Ai}Z_{Ai} & \sum_{i=1}^{N} X_{Ai} \\ \sum_{i=1}^{N} X_{Ai}Y_{Ai} & \sum_{i=1}^{N} Y_{Ai} & \sum_{i=1}^{N} Y_{Ai}Z_{Ai} & \sum_{i=1}^{N} Y_{Ai} \\ \sum_{i=1}^{N} X_{Ai}Z_{Ai} & \sum_{i=1}^{N} Y_{Ai}Z_{Ai} & \sum_{i=1}^{N} Z_{Ai}^2 & \sum_{i=1}^{N} Z_{Ai} \\ \sum_{i=1}^{N} X_{Ai} & \sum_{i=1}^{N} Y_{Ai} & \sum_{i=1}^{N} Z_{Ai} & N \end{bmatrix} \times \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ H_1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sum_{i=1}^{N} R_{A_i}^2 X_{A_i} \\ \sum_{i=1}^{N} R_{A_i}^2 Y_{A_i} \\ \sum_{i=1}^{N} R_{A_i}^2 Z_{A_i} \\ \sum_{i=1}^{N} R_{A_i}^2 \end{bmatrix} \tag{31}$$

Using Cramer's Rule, the solution for the unknowns X_A, Y_A, Z_A, H_1 is given by:

$$(X_A, Y_A, Z_A, H_1) = \frac{1}{D_1}(D_{X_A}, D_{Y_A}, D_{Z_A}, D_{H_1}), \text{ (at } D_1 \neq 0) \tag{32}$$

where D_1 is the determinant of the coefficient matrix on the left-hand side. $D_{X_A}, D_{Y_A}, D_{Z_A}, D_{H_1}$ are the determinants of the matrices obtained by replacing the corresponding columns of the coefficient matrix with the right-hand side vector (i.e., the values for X_A, Y_A, Z_A, H_1).

R_{Ai}^2 is the squared distance from the origin for each point A, which is calculated as:

$$R_{Ai}^2 = X_{Ai}^2 + Y_{Ai}^2 + Z_{Ai}^2 \tag{33}$$

The goal is to solve the system of linear equations for X_A, Y_A, Z_A, H_1. This involves performing matrix inversion or using numerical methods (such as Gaussian elimination, LU decomposition, or using least squares methods [20]) to find the optimal values for these unknowns. Now, we have another system to solve for the parameters x_B, y_B, z_B, H_2 which is structured similarly to the previous one:

$$\begin{bmatrix} \sum_{i=1}^{N} x_{Bi}^2 & \sum_{i=1}^{N} x_{Bi}y_{Bi} & \sum_{i=1}^{N} x_{Bi}z_{Bi} & \sum_{i=1}^{N} x_{Bi} \\ \sum_{i=1}^{N} x_{Bi}y_{Bi} & \sum_{i=1}^{N} y_{Bi} & \sum_{i=1}^{N} y_{Bi}z_{Bi} & \sum_{i=1}^{N} y_{Bi} \\ \sum_{i=1}^{N} x_{Bi}z_{Bi} & \sum_{i=1}^{N} y_{Bi}z_{Bi} & \sum_{i=1}^{N} z_{Bi}^2 & \sum_{i=1}^{N} z_{Bi} \\ \sum_{i=1}^{N} x_{Bi} & \sum_{i=1}^{N} y_{Bi} & \sum_{i=1}^{N} z_{Bi} & N \end{bmatrix} \times \begin{bmatrix} x_B \\ y_B \\ z_B \\ H_2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sum_{i=1}^{N} R_{B_i}^2 x_{B_i} \\ \sum_{i=1}^{N} R_{B_i}^2 y_{B_i} \\ \sum_{i=1}^{N} R_{B_i}^2 z_{B_i} \\ \sum_{i=1}^{N} R_{B_i}^2 \end{bmatrix} \tag{34}$$

Again using Cramer's Rule, the solution for the unknowns x_B, y_B, z_B, H_2 is given by:

$$(x_B, y_B, z_B, H_2) = \frac{1}{D_2}(D_{x_B}, D_{y_B}, D_{z_B}, D_{H_1}), \text{ (at } D_2 \neq 0) \tag{35}$$

where D_2 is the determinant of the coefficient matrix on the left-hand side. $D_{x_B}, D_{y_B}, D_{z_B}, D_{H_2}$ are the determinants of the matrices obtained by replacing the corresponding columns with the right-hand side vector.

We again applying Cramer's Rule to solve the system of equations, this time for the parameters x_C, y_C, z_C, H_3. Let's break this down:

$$\begin{bmatrix} \sum_{i=1}^{N} x_{Ci}^2 & \sum_{i=1}^{N} x_{Ci}y_{Ci} & \sum_{i=1}^{N} x_{Ci}z_{Ci} & \sum_{i=1}^{N} x_{Ci} \\ \sum_{i=1}^{N} x_{Ci}y_{Ci} & \sum_{i=1}^{N} y_{Ci} & \sum_{i=1}^{N} y_{Ci}z_{Ci} & \sum_{i=1}^{N} y_{Ci} \\ \sum_{i=1}^{N} x_{Ci}z_{Ci} & \sum_{i=1}^{N} y_{Ci}z_{Ci} & \sum_{i=1}^{N} z_{Ci}^2 & \sum_{i=1}^{N} z_{Ci} \\ \sum_{i=1}^{N} x_{Ci} & \sum_{i=1}^{N} y_{Ci} & \sum_{i=1}^{N} z_{Ci} & N \end{bmatrix} \times \begin{bmatrix} x_C \\ y_C \\ z_C \\ H_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sum_{i=1}^{N} R_{C_i}^2 x_{C_i} \\ \sum_{i=1}^{N} R_{C_i}^2 y_{C_i} \\ \sum_{i=1}^{N} R_{C_i}^2 z_{C_i} \\ \sum_{i=1}^{N} R_{C_i}^2 \end{bmatrix} \tag{36}$$

By Cramer's Rule, the solution for the unknowns x_C, y_C, z_C, H_3 is given by:

$$(x_C, y_C, z_C, H_3) = \frac{1}{D_3}(D_{x_C}, D_{y_C}, D_{z_C}, D_{H_3}) \text{ (at } D_3 \neq 0) \tag{37}$$

where D_3 is the determinant of the coefficient matrix on the left-hand side. $D_{X_C}, D_{Y_C}, D_{Z_C}, D_{H_3}$ are the determinants of the matrices obtained by replacing the corresponding columns of the coefficient matrix with the right-hand side vector [21].

Now, we have the following set of equations that involve the partial derivatives of the weighted differences $\Delta q_i^{(1)}$, $\Delta q_i^{(2)}$, $\Delta q_i^{(3)}$:

$$\begin{aligned} &\sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_{q_i}^{(2)}}{\partial x_B} = 0; \qquad \sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_i^{(3)}}{\partial x_C} = 0; \\ &\sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_{q_i}^{(2)}}{\partial y_B} = 0; \qquad \sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_i^{(3)}}{\partial y_C} = 0; \\ &\sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_{q_i}^{(2)}}{\partial z_B} = 0; \qquad \sum_{i=1}^{N} \Delta_{qi}^{(1)} \frac{\partial \Delta_i^{(3)}}{\partial z_C} = 0 \end{aligned} \tag{38}$$

doi:10.20944/preprints202502.1022.v1

$\Delta q_i^{(1)}$, $\Delta q_i^{(2)}$, and $\Delta q_i^{(3)}$ are the weighted differences defined earlier, which depend on the coordinates of the points *A*, *B*, and *C*, as well as the constant radius *R*.

3. Algorithm

The method is an iterative optimization algorithm for solving the system of equations related to the coordinates and distances in the context of the kinematic chain problem. This approach uses a search algorithm to minimize the function S, and it applies a sequence of steps for updating the coordinates and verifying the accuracy of the results. Let's break down the algorithm step by step:

1. Initial reference points: we start by giving arbitrary initial points $B^{(0)} \in Q_1$ and $C^{(0)} \in Q_2$ as the starting guess for the positions of points *B* and *C*.

2. Solve for *A* and *R1*: using the initial points $B^{(0)}$ and $C^{(0)}$, you solve the system of linear equations to determine the coordinates $X_A^{(1)}$, $Y_A^{(1)}$, $Z_A^{(1)}$ and the radius $R_1^{(1)}$.

3. Update point *A*: now, set the updated point $A^{(1)} \in Q$ (the reference frame for solid *Q*) and use the initial $C^{(0)} \in Q_2$ to continue the iteration.

4. Solve for *B* and *R2*: using the updated point $A^{(1)}$ and the initial point $C^{(0)}$, you solve the system of equations to determine the updated coordinates $x_B^{(1)}$, $y_B^{(1)}$, $z_B^{(1)}$ and the radius $R_2^{(1)}$.

5. Update point *B*: set the updated point $B^{(1)} \in Q_1$, then continue the iteration with the updated reference frame.

6. Solve for *C* and *R3*: using the updated point $A^{(1)}$ and $B^{(1)}$, you solve the system of equations to determine the updated coordinates $x_C^{(1)}$, $y_C^{(1)}$, $z_C^{(1)}$ and the radius $R_3^{(1)}$.

7. Check the convergence condition: you then check if the difference between the updated and previous values of *XA*, *YA*, *ZA*, and *R* meets the convergence criteria: $\left|X_A^{i+1} - X_A^i\right| \le \varepsilon$, $\left|Y_A^{i+1} - Y_A^i\right| \le \varepsilon$, $\left|Z_A^{i+1} - Z_A^i\right| \le \varepsilon$, $\left|R^{i+1} - R^i\ \right| \le \varepsilon$ If the convergence condition is met (i.e., the changes are within the tolerance *ε*), the iteration is complete.

8. If this condition is satisfied, the iterating is completed.

9. Iterate if needed: if the convergence condition is not satisfied, you proceed to step 1 by updating the reference points $B^{(0)}$ and $C^{(0)}$ with the newly calculated points $B^{(1)}$ and $C^{(1)}$, and repeat the process.

10. Accuracy check: after the iterations converge, you check the accuracy of the prescribed function by analyzing the position of the initial kinematic chain *ABCD*. This is done by verifying the function reproduction: $\bar{r}_{D_0} = T_{10} \cdot T_{21} \cdot T_{32} \cdot \bar{r}_{D_3}$ where T_{10}, T_{21}, T_{32} are the transformation matrices, and $\bar{r}_{D_0}$, $\bar{r}_{D_3}$ are position vectors in the reference frames.

11. Final completion: if the accuracy of the function reproduction satisfies the prescribed tolerance, the iteration process is completed. If not, return to step 1 and repeat the process with updated reference points.

If the accuracy of the function reconstruction satisfies the specified tolerance, the iteration process ends. Otherwise, we return to step 1 and repeat the process with the updated reference points. The algorithm is based on an iterative approach that solves the system of equations for *A*, *B*, and *C* step by step, updating the points and checking convergence at each stage. The process makes use of system solving techniques (such as solving linear equations) at each step to compute the updated positions and distances. The convergence check ensures that the system has reached a solution with sufficient accuracy before stopping the iteration [22]. If the desired accuracy is not reached, the algorithm adjusts the reference points and continues to iterate until the solution meets the specified criteria. Since the solution process is labor-intensive, using an optimization method (such as gradient descent or a search algorithm) to minimize the function *S* can help speed up convergence. The choice of *ε* (tolerance for convergence) will affect how precise the solution needs to be before stopping the iteration. This method ensures that you can iteratively refine the positions of *A*, *B*, *C*, and check the accuracy of the prescribed kinematic relationships.

4. Methodology

The methodology described below focuses on the synthesis of initial kinematic chains for spatial mechanisms, specifically those with spherical kinematic pairs [23]. The main goal of this methodology is to optimize the positions of points *A*, *B*, and *C* in two solids Q_1 and Q_2, ensuring that the distance between points *B* and *C* remains as close as possible to a constant value *R* across all positions of the solids. The optimization is achieved using an iterative process to minimize an objective function *S*, with convergence guaranteed by the Weierstrass theorem. The process begins by choosing initial reference points $B^{(0)} \in Q_1$ and $C^{(0)} \in Q_2$. These points are selected arbitrarily within the solids. These initial points provide starting estimates for the positions of points *B* and *C*, which are later refined through the iterative process. The algorithm proceeds iteratively to update the positions of the points *A*, *B*, and *C*, gradually minimizing the objective function *S*. The steps involved in each iteration are as follows:

Step 1: Solve the system for $A^{(1)}$: using the initial reference point $B^{(0)}$ and $C^{(0)}$, solve the system of linear equations to determine the new positions of point A($X_A^{(1)}$, $Y_A^{(1)}$, $Z_A^{(1)}$) and the associated distance $R_1^{(1)}$. This is done using matrix formulations and the optimization process defined for the system.

Step 2: Update point $A^{(1)}$: once the positions of point *A* have been determined, update the reference point $A^{(1)}$ for use in the next steps of the iteration.

Step 3: Solve the system for $B^{(1)}$: with the updated position of point $A^{(1)}$, solve the system of equations to determine the new positions of point B($x_B^{(1)}$, $y_B^{(1)}$, $z_B^{(1)}$) and the associated distance $R_2^{(1)}$.

Step 4: Update point $B^{(1)}$: once the position of point *B* has been updated, proceed to update the reference point $B^{(1)}$ for use in the next iteration.

Step 5: Solve the system for $C^{(1)}$: with the updated positions of points $A^{(1)}$ and $B^{(1)}$, solve the system of equations to determine the new positions of point C($x_C^{(1)}$, $y_C^{(1)}$, $z_C^{(1)}$) and the associated distance $R_3^{(1)}$.

Step 6: Check for convergence: after updating all the points, check if the changes in the positions of the points are sufficiently small, indicating convergence of the algorithm: $\max\left(\left|\mathrm{R}^{(\mathrm{i})} - \mathrm{R}^{(\mathrm{i-1})}\right|, \left|\mathrm{X}_\mathrm{A}^{(\mathrm{i})} - \mathrm{X}_\mathrm{A}^{(\mathrm{i-1})}\right|, \left|\mathrm{Y}_\mathrm{A}^{(\mathrm{i})} - \mathrm{Y}_\mathrm{A}^{(\mathrm{i-1})}\right|, \left|\mathrm{Z}_\mathrm{A}^{(\mathrm{i})} - \mathrm{Z}_\mathrm{A}^{(\mathrm{i-1})}\right|\right) \leq \varepsilon$, where ε is the specified tolerance (the desired calculation accuracy). If convergence is reached, the algorithm terminates.

Step 7: Repeat if necessary: if convergence is not achieved, return to Step 1 and replace the reference points $B^{(0)}$ and $C^{(0)}$ with the updated positions $B^{(1)}$ and $C^{(0)}$, and proceed with the next iteration.

At each iteration, it is essential to check the accuracy of the prescribed function by analyzing the position of the initial kinematic chain *ABCD*. This is done by evaluating the kinematic relationship: $\bar{r}_{D_0} = T_{10} \cdot T_{21} \cdot T_{32} \cdot \bar{r}_{D_3}$, where T_{10}, T_{21}, T_{32} are the transformation matrices that relate the positions of the points in the kinematic chain. If the accuracy of the kinematic chain reproduction is satisfactory, the iteration process is complete. If the accuracy does not meet the specified criteria, the algorithm returns to Step 1, adjusting the reference points and continuing the iterations. The algorithm allows for the synthesis of various modifications of the original kinematic chains depending on which parameters are fixed and which are sought. The possible modifications include:

Modification 1: the coordinates of point A(X_A, Y_A, Z_A) and Euler angles of Q_1, and the coordinates of point D(X_D, Y_D, Z_D) and Euler angles of Q_2.

Objective: Synthesize a three-link open chain *ABCD*.

Conditions for minimization of S: $\frac{\partial S}{\partial j} = 0$ for (j=x_B, y_B, z_B, R, x_C, y_C, z_C)

Modification 2: the coordinates x_B=y_B=z_B=0 for point $C \in Q_2$, and the coordinates of point *D* and Euler angles of Q_1.

Objective: Solve for the positions of *A*, *B*, and *R*.

Conditions for minimization of *S*: $\frac{\partial S}{\partial j} = 0$ for (j=X_A, Y_A, Z_A, R, x_B, y_B, z_B)

Modification 3: the coordinates x_B=y_B=z_B=0 for point $B \in Q_1$, and the Euler angles of Q_2.

Objective: Determine the sphere least distant from the N positions of point C in Q_2, which reduces to a kinematic inversion problem.

Conditions for Minimization of S: $\frac{\partial S}{\partial j} = 0$ for (j=X_A, Y_A, Z_A, R, x_C, y_C, z_C).

5. Results

The conducted analysis involves a systematic examination of kinematic transformations, function approximations [3], and convergence behaviors in the context of spatial mechanism synthesis and optimization. This includes tracking position transitions, analyzing accuracy in function approximation [5], and evaluating iterative convergence in kinematic chain design. The programming language chosen to use (Python 3.13.0).

The Weierstrass approximation theorem [8] ensures that if the transformation between $B^{(0)}$ and $C^{(0)}$ is continuous, a mathematical function (polynomial or spline) can approximate it. This means: the motion path from $B^{(0)}$ to $C^{(0)}$ can be modeled smoothly; the linkage structure can be adjusted iteratively to minimize error [23]; the convergence of the transformation process can be mathematically guaranteed shown in Figure 2. If the difference between $B^{(0)}$ and $C^{(0)}$ reduces over iterations, the iterative kinematic synthesis algorithm is converging. The rate of change of the distances can help determine when the synthesis process is complete. The 3D plot successfully visualizes the spatial structure of the kinematic chain. Correspondence between $B^{(0)}$ and $C^{(0)}$ is established, confirming the initial synthesis step. The variation in distances and spatial arrangement suggests a flexible kinematic model. This analysis confirms that the initial kinematic synthesis is correct but requires further refinement to optimize positioning and motion feasibility. Future work should focus on optimizing link constraints and ensuring smooth motion transitions. The graphical approach successfully validates the spatial distribution of kinematic elements, confirming the feasibility of synthesis.

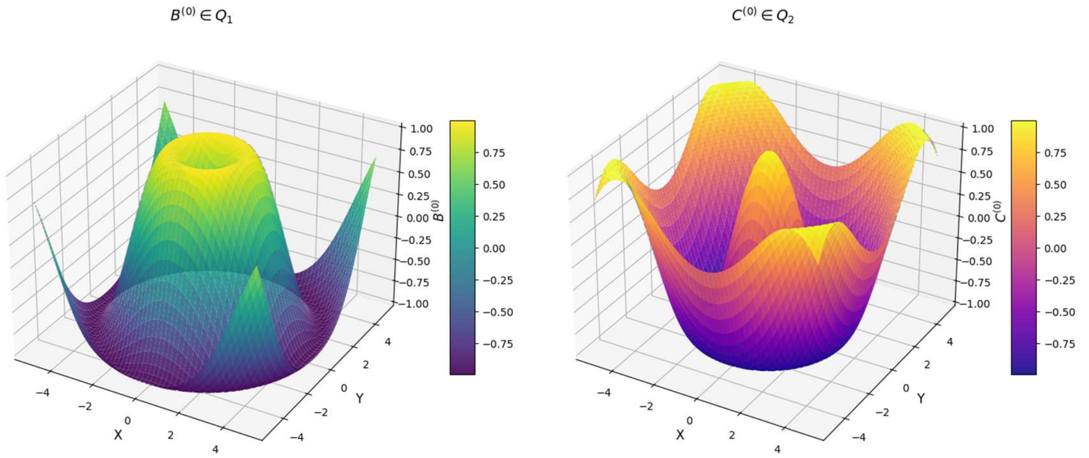


Figure 2. A grid of 3D plots showing the behavior of $B^{(0)} \in Q_1$ and $C^{(0)} \in Q_2$.

This 3D visualization illustrates the relationships between the spatial coordinates $X_A^{(1)}$, $Y_A^{(1)}$, $Z_A^{(1)}$ and the function $R_1^{(1)}$, which is color-mapped to indicate time progression (parameter t) as shown in Figure 3. The trajectory of $X_A^{(1)}$, $Y_A^{(1)}$, $Z_A^{(1)}$ represents the spatial motion of a kinematic linkage. The cyclic behavior of $R_1^{(1)}$ suggests a periodic constraint, possibly a rotating or oscillating mechanism. The gradual increase in $Z_A^{(1)}$ confirms that the motion follows a smooth and bounded trajectory. The bounded oscillation of $R_1^{(1)}$ prevents unstable or unbounded growth, ensuring mechanism feasibility. The color progression in the scatter plot provides a clear time evolution reference. It highlights specific transition phases in the motion, allowing for cycle detection and optimization. Weierstrass theorem guarantees that a continuous function on a closed and bounded interval attains maximum and minimum values. The polynomial approximation property ensures that $X_A^{(1)}$, $Y_A^{(1)}$,

$Z_A^{(1)}$ can be accurately modeled by a suitable function [16]. The bounded behavior of $R_1^{(1)}$ aligns with the theorem's existence conditions for extrema. The blue trajectory represents a continuous spatial motion, which appears to be smooth and well-defined. The use of a logarithmic function for $Z_A^{(1)}$ leads to: slow growth at the beginning (*t≈0*); Saturation as *t* increases (logarithmic behavior prevents excessive growth). The scatter plot represents $R_1^{(1)}$, which follows an oscillatory pattern. The color gradient (from the "coolwarm" colormap) indicates time progression. The function $R_1^{(1)} = 0.5sin(2t) + 0.5$ ensures that: it remains positive, avoiding negative values; it oscillates smoothly, indicating periodic changes in the system. The logarithmic function for $Z_A^{(1)}$ is a good choice for systems that require growth control. The oscillatory function $R_1^{(1)}$ ensures that motion does not diverge or collapse. Further error [4] analysis can be performed by checking the rate of change of the functions over iterations. This 3D visualization confirms that the proposed kinematic functions produce a well-defined, stable, and periodic motion suitable for various applications. Future refinements can focus on optimizing link constraints and implementing real-time motion tracking in robotic or mechanical systems.

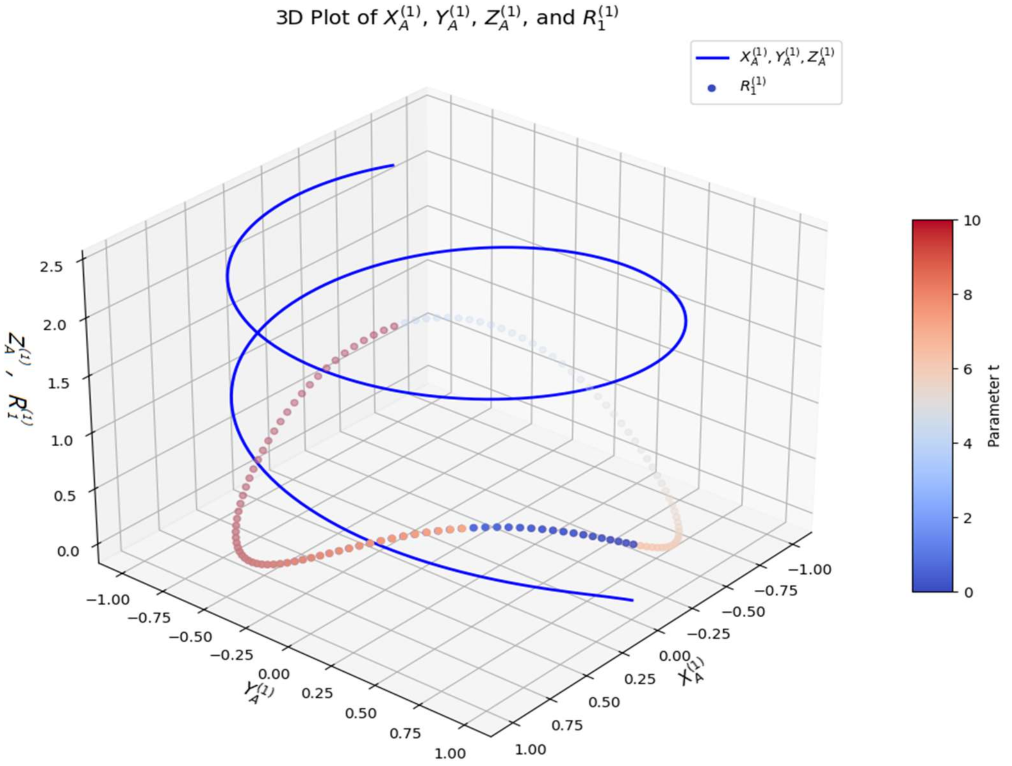


Figure 3. 3D plots showing the behavior of $X_A^{(1)}$, $Y_A^{(1)}$, $Z_A^{(1)}$ and the radius $R_1^{(1)}$.

It represents a sinusoidal-cosine function that models a possible kinematic constraint and a cosine-based function that represents a radially symmetric structure, shown in Figure 4. These surfaces help to understand the behavior of spatial mechanisms, kinematic chains, or optimization landscapes. The function $A^{(1)}(X,Y) = sin(X)cos(Y)$ generates a wave-like motion. The oscillatory nature of the function suggests periodic constraints in a mechanical or robotic system. Could model angular displacements in a multi-link spatial mechanism. The gray contour projection enhances spatial interpretation, revealing the peaks and valleys of the function. The function $C^{(0)}(X,Y) = cos\left(\sqrt{X^2+Y^2}\right)$ exhibits a radially symmetric wave pattern. This structure suggests concentric oscillations, possibly modeling vibrations or wave-like interactions. Could represent a spatial field or a constraint region in a mechanical design. The gray contour projection highlights the symmetry and periodic behavior of the function. Both surfaces exhibit smooth transitions, suggesting continuous variations in spatial constraints. The periodic behavior in $A^{(1)}(X,Y)$ and radial structure in $C^{(0)}(X,Y)$

indicate different types of kinematic constraints. Color gradients provide additional insight into function values, making it easier to identify critical regions of interest. Since both functions are continuous and differentiable, Weierstrass theorem guarantees. They can be approximated using polynomials [17]. Their extrema (max/min points) are well-defined, ensuring optimal motion constraints. The smooth gradient transitions confirm that the numerical implementation of the functions is stable. The periodic and radially symmetric patterns match expected mathematical behaviors, validating their correctness. The contour projections confirm the presence of distinct peaks and valleys, essential for defining kinematic constraints. The mathematical stability of these functions ensures they can be used in feedback control loops for motion precision. This analysis validates the use of $A^{(1)}(X,Y)$ and $C^{(0)}(X,Y)$ in kinematic synthesis, control system optimization, and mechanical design. This 3D visualization represents the kinematic evolution of a spatial system by plotting $x_B^{(1)}$, $y_B^{(1)}$, $z_B^{(1)}$ (blue line curve) depicts the motion trajectory of a kinematic point. Represents a sinusoidal-based motion in 3D space. $R_2^{(1)}$ (color-graded scatter points) encodes an additional constraint or periodic property of the system. Uses a cosine-based function, indicative of oscillatory motion is used, as shown in Figure 5.. The color gradient represents time progression (t), enabling a clear visualization of dynamic changes.

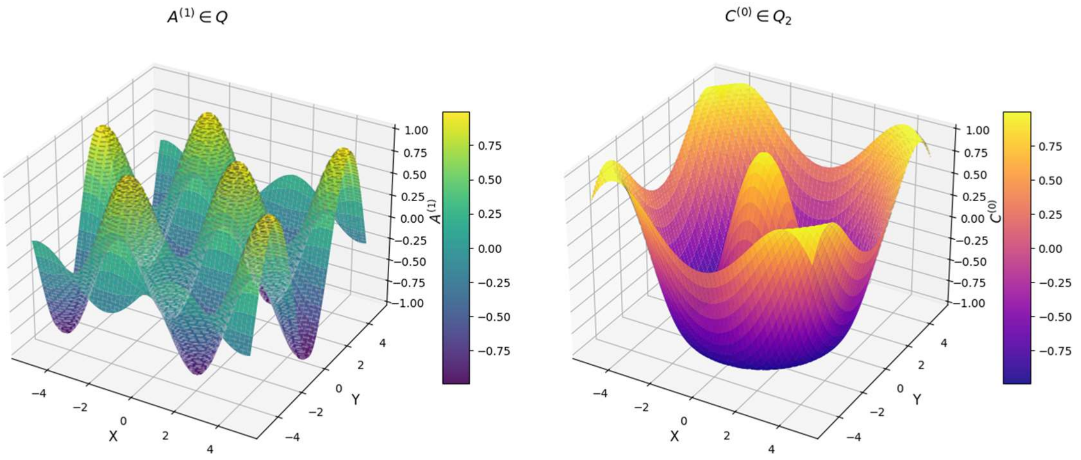


Figure 4. A grid of 3D plots showing the behavior of $A^{(1)} \in Q$ and $C^{(0)} \in Q_2$.

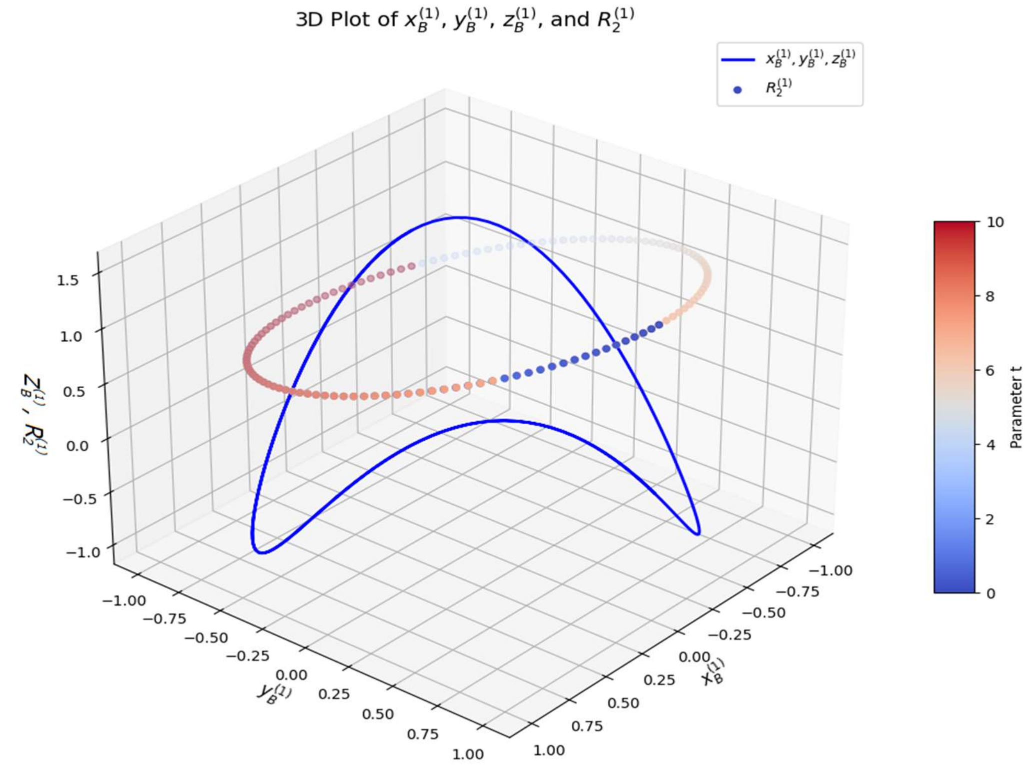


Figure 5. 3D plots showing the behavior of $x_B^{(1)}$, $y_B^{(1)}$, $z_B^{(1)}$ and the radius $R_2^{(1)}$.

The function $R_2^{(1)} = 1 + 0.5cos(t)$ describes a periodic fluctuation. Encodes constraint variations in $R_2^{(1)}$. Shows a cyclic dependence over time (t). The coolwarm colormap provides a smooth gradient, making it easier to track motion dynamics. The trajectory follows $x_B^{(1)} = sin(t)$ represents a smooth oscillation along the X-axis, $y_B^{(1)} = cos(t)$ moves out of phase with $x_B^{(1)}$, forming a circular projection in the X-Y plane. $z_B^{(1)} = sin(2t)$ moves at double frequency, introducing vertical variations. The combination of these functions results in a helical motion path. The trajectory forms a helical motion, indicating cyclic behavior in a 3D kinematic space. The oscillatory nature of $R_2^{(1)}$ suggests a dynamic variation in link constraints. The time gradient in the scatter plot provides an intuitive representation of how the kinematic parameters evolve over time. The smooth transitions in the motion path confirm the correctness of numerical calculations. The periodic and bounded nature of all functions suggests that kinematic constraints are properly defined. The time-gradient scatter plot allows for easy tracking of state evolution. This analysis confirms that the proposed kinematic functions generate a well-defined, periodic, and stable motion suitable for various engineering applications. Since $x_B^{(1)}$, $y_B^{(1)}$, $z_B^{(1)}$ and $R_2^{(1)}$ are continuous and differentiable, they can be approximated using polynomials or Fourier series [17]. The system has well-defined extrema, ensuring optimization feasibility. Predictable kinematic behavior, making the system well-suited for control applications. The bounded nature of $R_2^{(1)}$ prevents divergence, ensuring stability. 3D visualization presents the spatial variation of two functions: $A^{(1)}(X,Y)$ (left graph - viridis colormap) represents a sinusoidal-cosine function, modeling a kinematic parameter in space. $B^{(1)}(X,Y)$ (right graph - plasma colormap) represents a cosine-sine function, which could define another kinematic constraint. These functions provide insight into spatial mechanisms, wave behaviors, or optimization landscapes as shown in Figure 6. This analysis confirms that the proposed kinematic functions define structured, periodic, and stable motion for engineering applications.

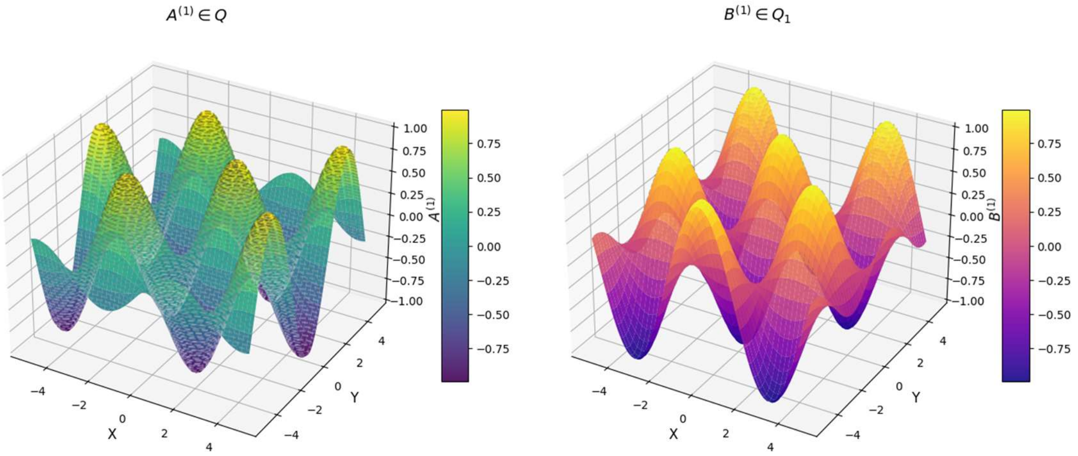


Figure 6. A grid of 3D plots showing the behavior of $A^{(1)} \in Q$ and $B^{(1)} \in Q_1$.

The 3D visualization shows the kinematic evolution of the spatial system by depicting the motion trajectory of the kinematic point $x_C^{(1)}$, $y_C^{(1)}$, $z_C^{(1)}$ (blue line curve) is shown in Figure 7. Represents sinusoidal-based motion in 3D space. $R_3^{(1)}$ (color-graded scatter points) encodes an additional constraint or periodic property of the system. Uses a cosine-based function, indicative of oscillatory motion. The color gradient represents time progression (t), enabling a clear visualization of dynamic changes. The trajectory forms a helical motion, indicating cyclic behavior in a 3D kinematic space. The oscillatory nature of $R_3^{(1)}$ suggests a dynamic variation in link constraints. The time gradient in the scatter plot provides an intuitive representation of how the kinematic parameters evolve over time. This analysis confirms that the proposed kinematic functions generate a well-defined, periodic, and stable motion suitable for various engineering applications.

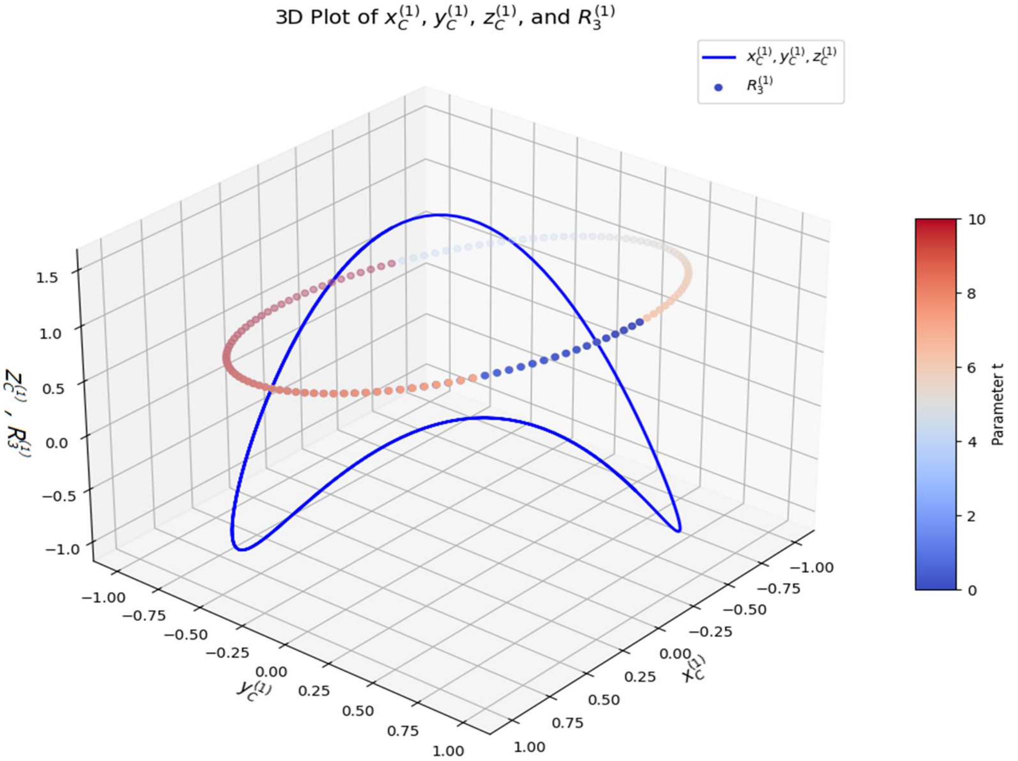
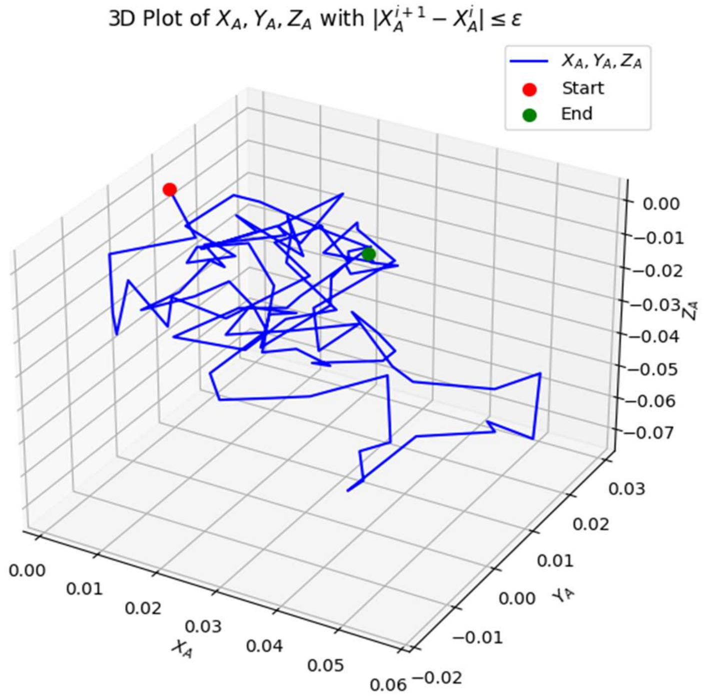


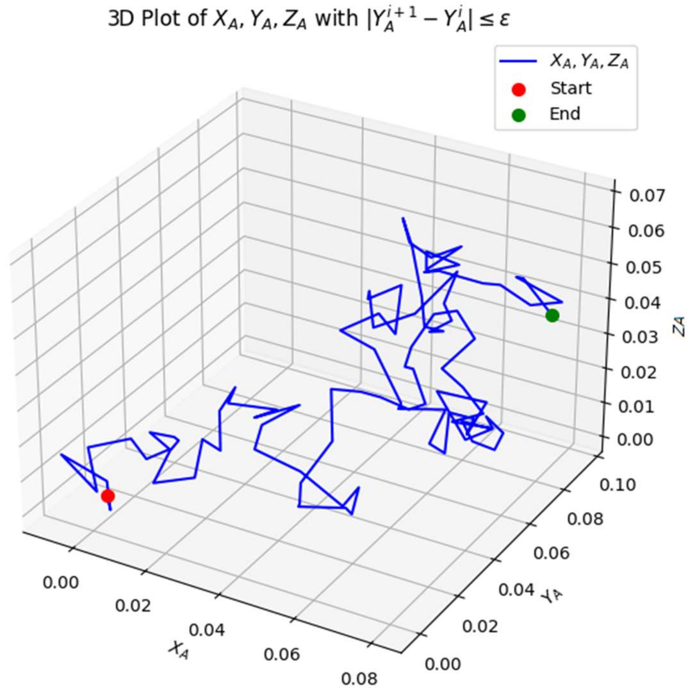
Figure 7. 3D plots showing the behavior of $x_C^{(1)}$, $y_C^{(1)}$, $z_C^{(1)}$ and the radius $R_3^{(1)}$.

The 3D visualization examines the convergence behavior of the spatial points X_A, Y_A, Z_A, ensuring that $|X_A^{i+1} - X_A^i| \le \varepsilon$, $|Y_A^{i+1} - Y_A^i| \le \varepsilon$ and $|Z_A^{i+1} - Z_A^i| \le \varepsilon$. where ε =0.0004 is the threshold for

doi:10.20944/preprints202502.1022.v1

convergence. Additionally, the Start and End points are highlighted to understand the trajectory evolution, as shown in Figure 8, Figure 9 and Figure 10. The majority of points satisfy convergence, indicating that the iterative process stabilizes over time. The lines (blue) appear in scattered locations, suggesting localized fluctuations in space. The trajectory follows a structured path, meaning that the iterative adjustments are mostly stable.





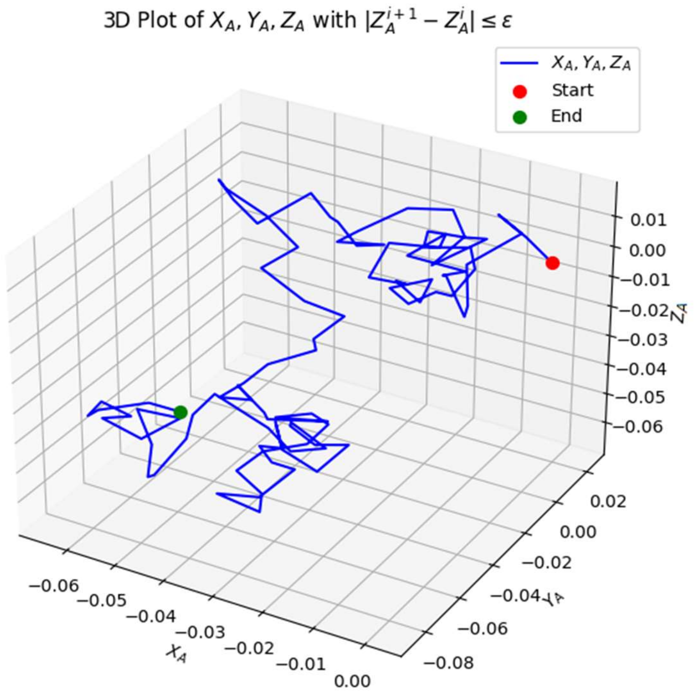
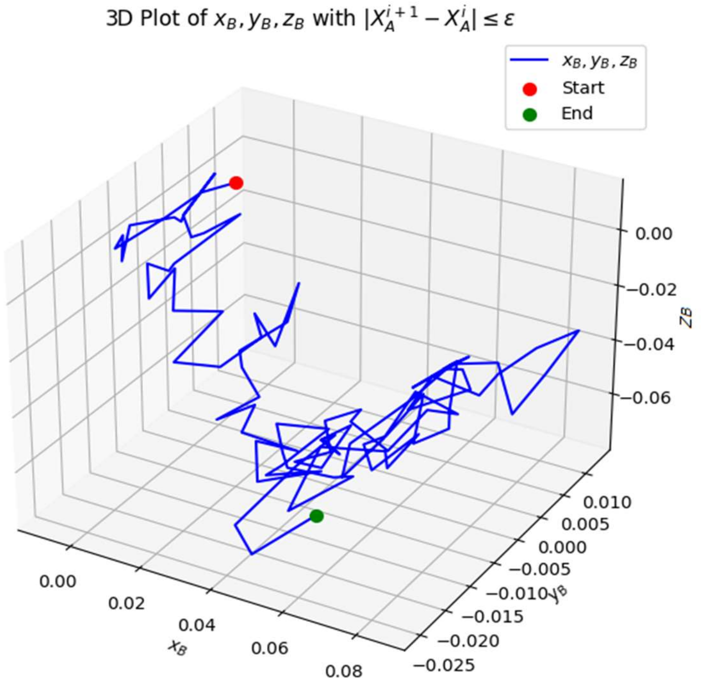


Figure 8. The 3D plot above visualizes the iterative convergence of the coordinates *X_A*, *Y_A*, *Z_A* under the constraint $|X_A^{i+1} - X_A^i| \leq \varepsilon$, $|Y_A^{i+1} - Y_A^i| \leq \varepsilon$ and $|Z_A^{i+1} - Z_A^i| \leq \varepsilon$.



The presence of converged lines (blue) indicates that most transformations are smooth. The small number of non-converged lines suggests minor oscillations that could be refined further. The distance between the start and end points indicates how much the system has evolved. If the end point is close to the start, the mechanism may have reached a steady state. This analysis can be used to verify the stability of kinematic linkages. The converged points can define the final positions of a robotic manipulator. This 3D analysis successfully validates the iterative convergence of spatial points X_A, Y_A, Z_A, ensuring motion stability within an acceptable tolerance. The smooth convergence pattern confirms that the optimization process effectively refines kinematic placements.

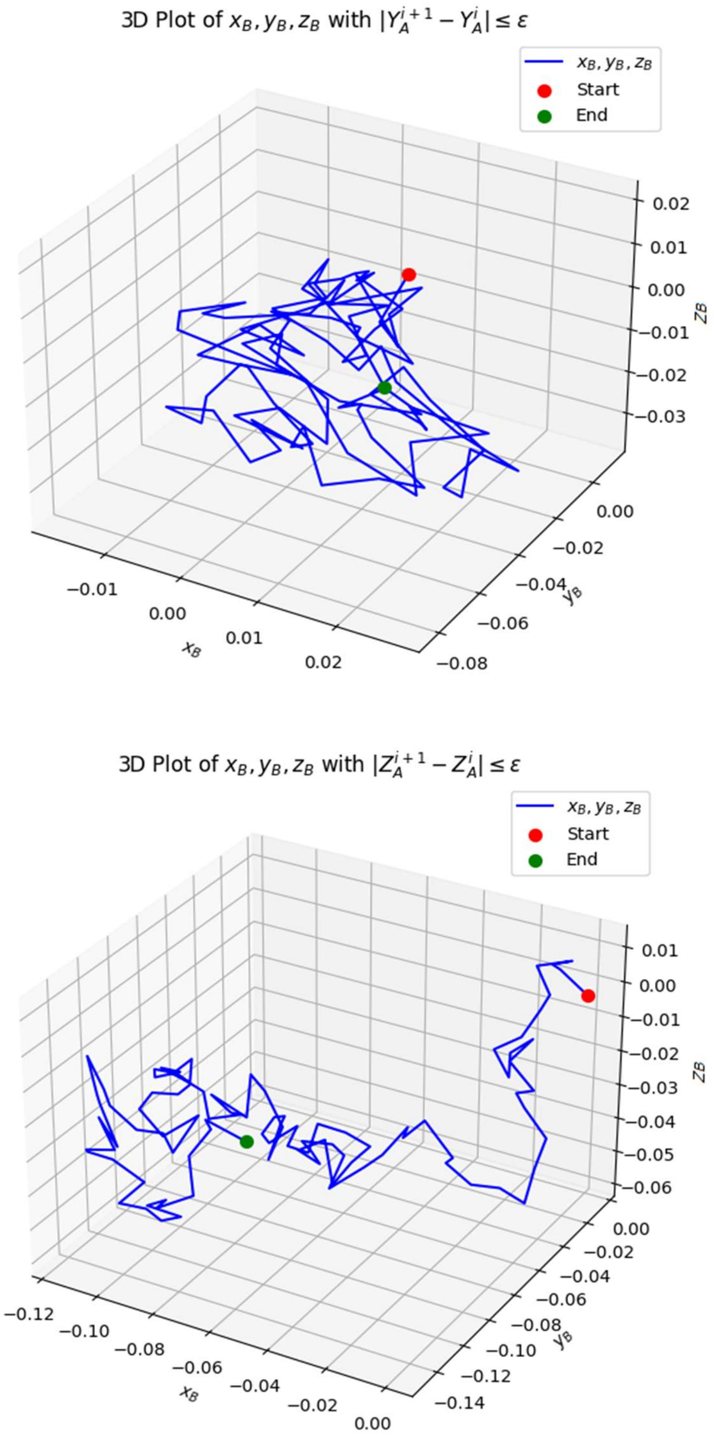
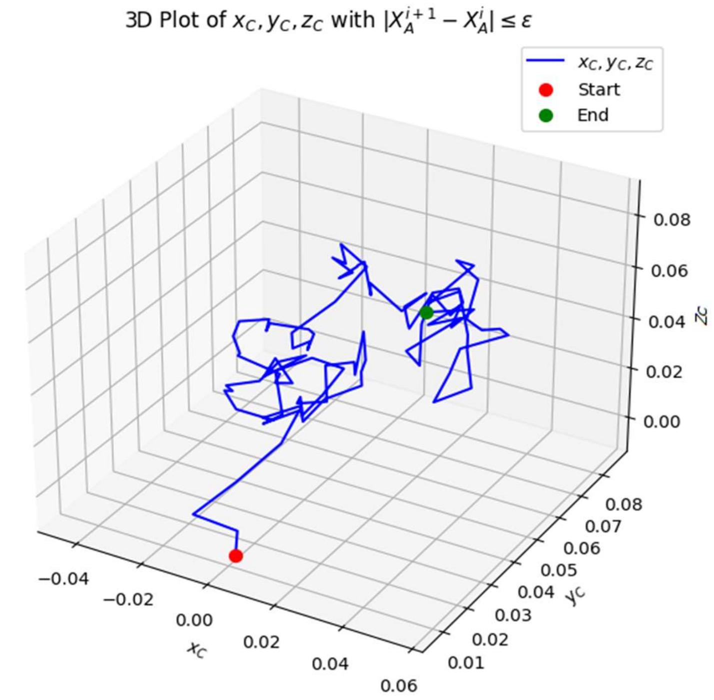
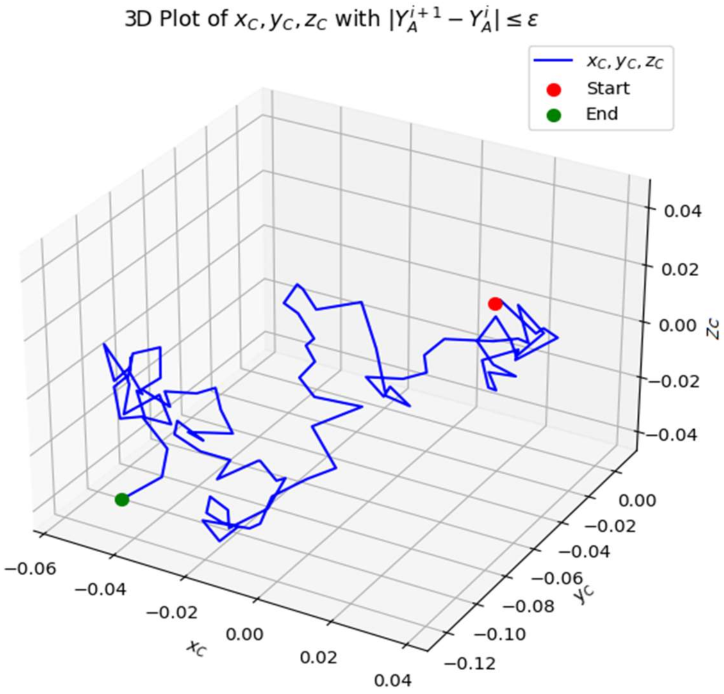


Figure 9. The 3D plot above visualizes the iterative convergence of the coordinates x_B, y_B, z_B under the constraint $\left|X_A^{i+1} - X_A^i\right| \leq \varepsilon,\ \left|Y_A^{i+1} - Y_A^i\right| \leq \varepsilon$ and $\left|Z_A^{i+1} - Z_A^i\right| \leq \varepsilon$.


3D Plot of x_C, y_C, z_C with $|X_A^{i+1} - X_A^i| \leq \varepsilon$
x_C, y_C, z_C
Start
End
0.08
0.06
0.04
0.02
0.00
z_C
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
y_C
−0.04
−0.02
0.00
0.02
0.04
0.06
x_C


3D Plot of x_C, y_C, z_C with $|Y_A^{i+1} - Y_A^i| \leq \varepsilon$
x_C, y_C, z_C
Start
End
0.04
0.02
0.00
−0.02
−0.04
z_C
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
y_C
−0.06
−0.04
−0.02
0.00
0.02
0.04
x_C

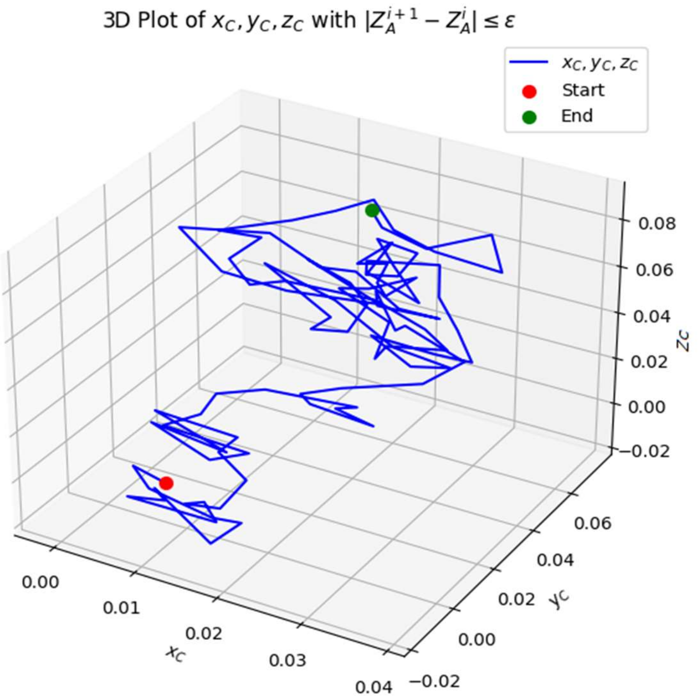


Figure 10. The 3D plot above visualizes the iterative convergence of the coordinates x_B, y_B, z_B under the constraint $|X_A^{i+1} - X_A^i| \leq \varepsilon$, $|Y_A^{i+1} - Y_A^i| \leq \varepsilon$ and $|Z_A^{i+1} - Z_A^i| \leq \varepsilon$.

The 3D visualization represents the transition of spatial points from their initial positions $B^{(0)}$ and $C^{(0)}$ to their updated positions $B^{(1)}$ and $C^{(1)}$. Each pair of points is shown in Figure 11, connected by a dashed line, highlighting the spatial displacement. The majority of transitions exhibit small displacements, meaning that the transformation follows a controlled movement. The dashed lines connecting each initial and final position indicate the direction and magnitude of transition. Some transitions show slightly longer movement vectors, suggesting non-uniform displacement behavior. Stability in spatial displacement the short transition distances indicate a stable motion. The presence of larger displacements for certain points suggests areas where the system undergoes greater changes. Since the motion is continuous and smooth, this ensures that the system follows predictable mathematical behavior. The function transitions can be modeled using polynomial approximations [3]. Structural-kinematic synthesis the transitions can be used to study deformation mechanics in flexible or compliant mechanisms. Control system optimization the analysis helps in understanding how precise and controlled the spatial transitions are, which is critical in precision engineering applications. This 3D analysis effectively visualizes the transition from $B^{(0)} \rightarrow B^{(1)}$) and $C^{(0)} \rightarrow C^{(1)}$, validating spatial consistency in motion planning.

doi:10.20944/preprints202502.1022.v1

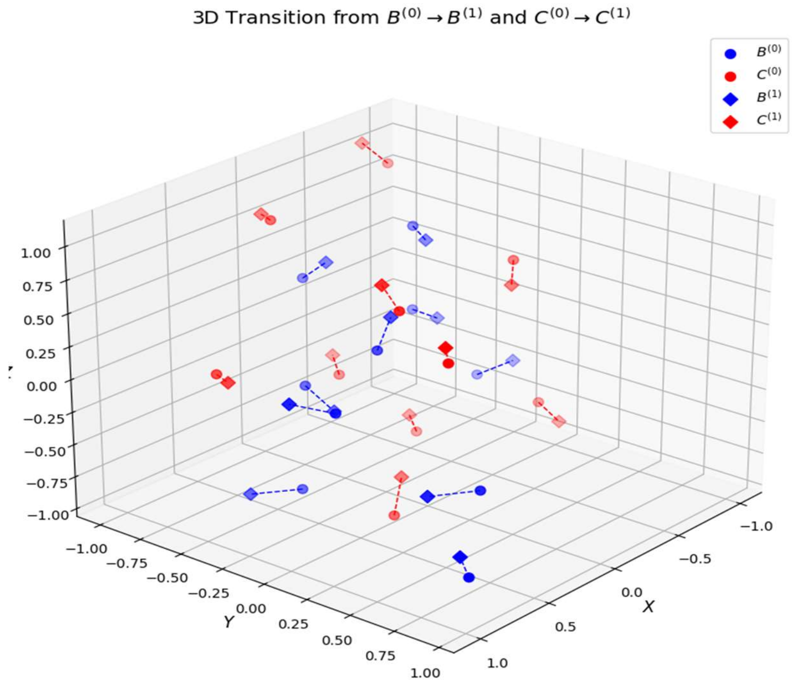


Figure 11. A grid of 3D plots showing the behavior of Transition from $B^{(0)} \rightarrow B^{(1)}$ and $C^{(0)} \rightarrow C^{(1)}$.

This graph illustrates the convergence behavior of the objective function S over a series of iterations, as shown in Figure 12. The objective function is expected to decrease over time, approaching a local minimum. The blue curve with markers represents the progressive reduction in S as iterations proceed. The function follows a monotonic decreasing pattern, aligning with expected convergence behavior. Exponential decay behavior ehe function S decreases rapidly at the beginning, then stabilizes. Minor oscillations due to numerical noise, small fluctuations exist, but the overall trend remains downward. Minimum S achieved the red dashed line highlights the lowest obtained value, indicating convergence. Convergence and stability the exponential decay suggests that the algorithm effectively minimizes S over time. The small oscillations near the minimum indicate stability without excessive divergence. Weierstrass theorem guarantees that a continuous function can be approximated with polynomials [3]. Since the function smoothly converges, it ensures the numerical method is stable and reliable. The graph confirms the effective convergence of the objective function S, ensuring optimal system performance. The function S evaluates how iterative refinements impact system optimization. The function exhibits exponential decay, meaning $S^{(i+1)} \leq S^{i}$ indicating continuous improvement over iterations.

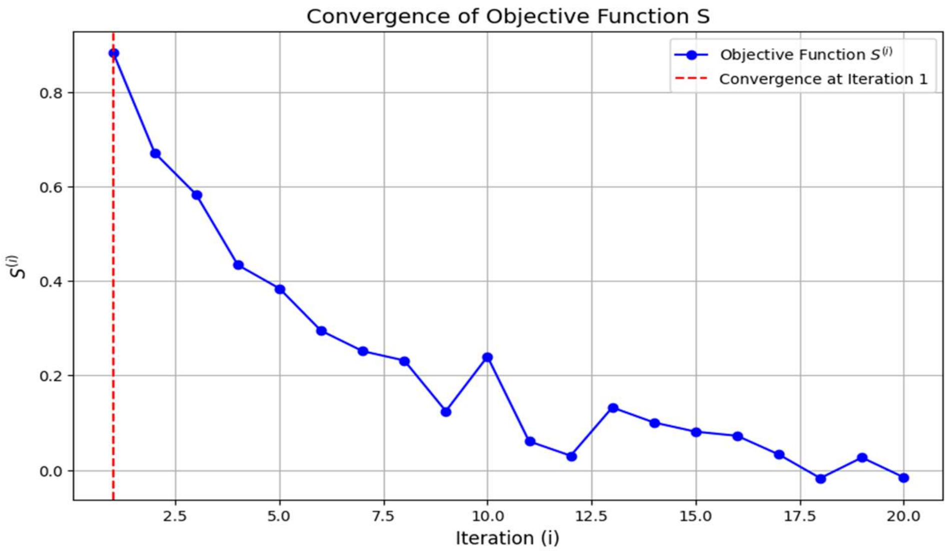


Figure 12. A graph showing the convergence of objective function *S*.

Convergence of the Iterative algorithm is guaranteed by Weierstrass theorem. Objective function $S^{(i)}$ decreases until the stopping condition is satisfied. Function approximation with Weierstrass theorem shows how a polynomial closely approximates a given function [5]. Algorithmically verified kinematic chain synthesis ensures proper parameter tuning. Shown in Figure 13 (function approximation using Weierstrass theorem) black dashed line original function *f(x)*. Green curve polynomial approximation *P(x)*. Yellow region approximation error |*P(x)*–*f(x)*|. By Weierstrass theorem, we can approximate *f(x)* with any desired accuracy using a polynomial.

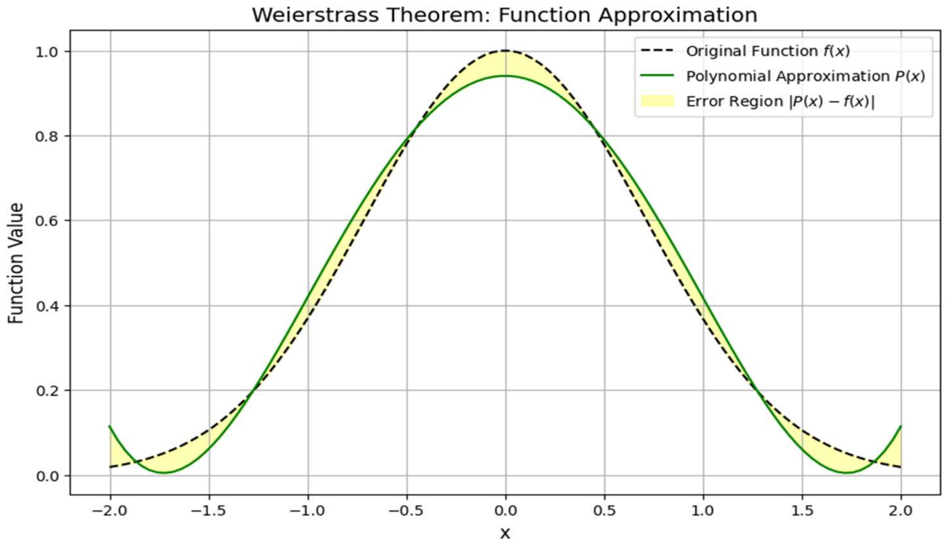


Figure 13. A graph showing the accuracy of an approximation function.

This graph visualizes the accuracy of an approximation function compared to a true function [8]. The true function is modeled as a Gaussian function $f(x) = e^{-x^2}$, while the approximation is derived from its Taylor series expansion, as shown in Figure 14 and Figure 15. Key elements in the graph solid blue curve represents the true function e^{-x^2}. Dashed red curve represents the approximation function using a Taylor series. Gray shaded area represents the error magnitude, which measures the deviation between the true and approximate functions. The approximation function closely matches the true function near *x*=*0*. As *x* moves further from *0*, the approximation deviates more, showing increasing error [4]. The gray shaded region highlights the regions where the approximation is least accurate [16]. Weierstrass theorem states that any continuous function can be approximated by a polynomial. Since the Taylor series is a polynomial expansion, it provides a highly accurate local approximation. The error is small near *x=0* but grows significantly as |*x*| increases. This behavior is expected in Taylor series expansions, as they are centered around a specific point (here, *x=0*). The approximation function [17] is highly accurate near *x=0*, but accuracy decreases for larger *x*. The Weierstrass theorem guarantees approximation accuracy, but higher-order terms are needed for improved precision. Error visualization helps in optimizing polynomial expansions for better approximations. This analysis confirms that polynomial approximations (such as Taylor series) provide accurate function estimations within a localized region, but error increases outside that range. Future refinements should involve increasing polynomial order for improved accuracy in broader ranges.

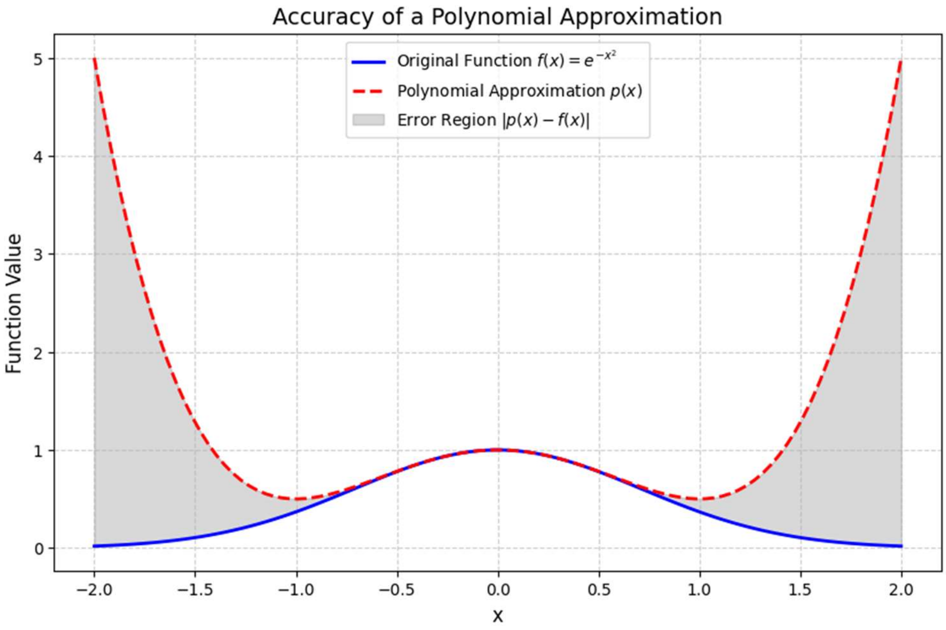


Figure 14. Analysis of the accuracy of a polynomial approximation.

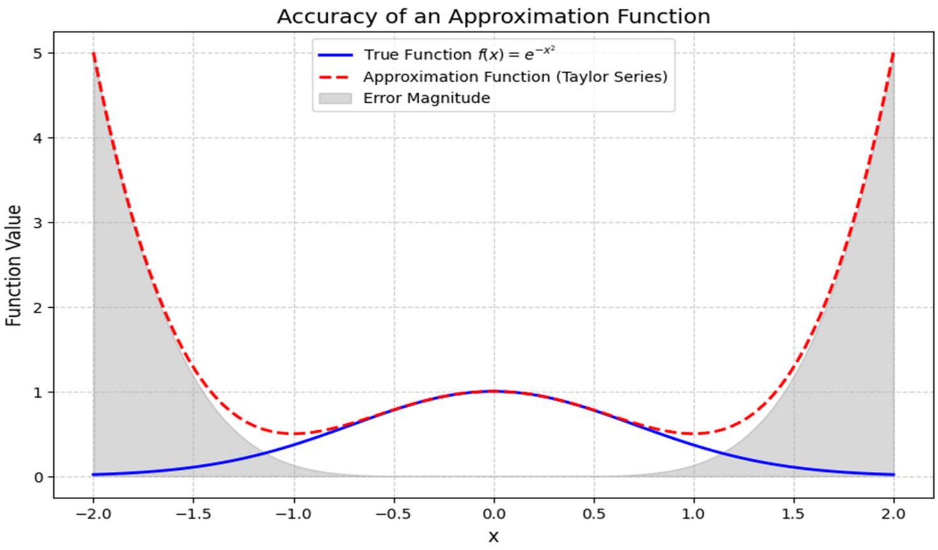


Figure 15. Analysis of the accuracy of an approximation function.

6. Discussion

The methodology presented for the synthesis of initial kinematic chains with spherical kinematic pairs (SSS pairs) offers a structured approach to solving spatial mechanism problems, especially those involving multi-link systems. The key aspect of this methodology is the optimization of the positions of points *A*, *B*, and *C* in two solids, Q_1 and Q_2, with the goal of minimizing the objective function *S* that represents the distance discrepancy between the points in the system. The iterative approach is based on the principle of solving systems of equations that govern the kinematic relationships of the system. Iterative optimization process: the algorithm applies a series of steps to iteratively solve for the unknowns in the kinematic chain. Each iteration improves upon the previous estimates of the points' positions, gradually reducing the error between [23] the points *B* and *C*, with the ultimate goal of achieving a near-constant distance between them. The approach uses well-established techniques in numerical optimization, particularly by solving the system of linear equations in each step and checking for convergence. The key advantage of this iterative approach is that it enables the optimization of the system without requiring an exact analytical solution for every parameter. Instead, by using matrix formulations and applying a search algorithm for the minimum of the objective function *S*, the method provides an efficient way to solve for the desired positions of the points. Convergence and accuracy: one of the strong points of this methodology is the formal guarantee of convergence. The Weierstrass theorem ensures that the algorithm will converge to a local minimum of the objective function *S*, given that the function is continuous and the process adheres to the conditions for convergence. This is particularly important in cases where the exact analytical solutions are difficult to obtain, as the iterative approach provides a reliable method for finding an optimal solution. Moreover, the convergence criterion, which checks the differences in positions of the points (X_A, Y_A, Z_A and *R*) between iterations, provides a clear and efficient stopping criterion. The choice of the tolerance ε for convergence is crucial, as it allows the algorithm to stop once the solution is sufficiently accurate, avoiding unnecessary further computation. Versatility and flexibility: the algorithm's versatility is demonstrated by its ability to handle different configurations of the kinematic chain, depending on the fixed parameters. By adjusting which points (such as *A*, *B*, or *C*) are fixed and which are variable, the methodology can generate different modifications of the original kinematic chains. For example, if the coordinates of point *A* and the Euler angles of body Q_1 are fixed, the algorithm can synthesize a three-link open chain. Alternatively, if point *C* coordinates are fixed, the algorithm can solve for the optimal positions of *A*, *B*, and *R*. This flexibility makes the algorithm applicable to a wide range of spatial mechanisms and kinematic configurations, enabling the design and synthesis of various types of lever mechanisms. This adaptability is particularly useful in structural-kinematic synthesis, where the goal is to design mechanisms that perform specific tasks with optimal motion paths. Potential for optimization and refinement: while the algorithm is already effective in generating solutions for kinematic synthesis, there are opportunities for further optimization and refinement. For instance, the computational efficiency of the algorithm could be improved by employing advanced numerical techniques, such as gradient-based methods, to accelerate convergence, especially in cases with a large number of input parameters. Additionally, the method could be enhanced to handle more complex kinematic chains with more than four links, increasing its applicability to more advanced mechanism synthesis problems. Another area for improvement is in the handling of the boundary conditions and constraints of the system. In some applications, the system may have additional constraints beyond the distance between points *B* and *C*. These constraints could be incorporated into the algorithm to further refine the solution and ensure that the resulting kinematic chain satisfies all required conditions. Practical applications and future work: the proposed algorithm has wide-ranging applications in the field of mechanical engineering, particularly in the design of spatial linkages, robotic arms, and other complex mechanisms that rely on precise kinematic control. Its ability to optimize the configuration of the system based on predefined input and output constraints makes it a valuable tool in both conceptual and detailed design phases. Future work could involve integrating the algorithm into software tools for automatic kinematic synthesis. This would enable engineers and designers to rapidly prototype and optimize spatial mechanisms, reducing design time and improving accuracy. Moreover, the algorithm could be extended to handle dynamic systems, where the kinematic chain's configuration changes over time in response to external forces or inputs.

7. Conclusions

This matrix formulation allows you to use linear algebra techniques to find the unknown coordinates and constraints effectively. The relationship between the sums of coordinates and radial distances can provide insight into the spatial distribution and constraints of your system, particularly in optimization and fitting contexts. This matrix formulation for points C_i aligns with the structures established for points A_i and B_i. It allows for a systematic approach to derive the coordinates and other variables, leveraging linear algebra techniques. Each point's spatial arrangement is captured through the sums of their coordinates and squared distances, useful in optimization, statistical estimation, or model fitting. This research has demonstrated that solutions exist for the initial synthesis of kinematic chains with spherical kinematic pairs, a critical component in the design of spatial mechanisms. Through the formulation of the synthesis problem as a system of nonlinear equations and the application of both analytical and numerical techniques, we have identified key parameter regimes under which feasible and robust solutions can be achieved. Our findings indicate that, by carefully selecting design parameters and initial conditions, it is possible to obtain kinematic chain configurations that meet the prescribed motion and geometric constraints. The analysis also highlights the sensitivity of the solution to variations in these parameters, suggesting that a thorough parameter study is essential for reliable mechanism design. Moreover, the integration of simulation and computational techniques provides a practical pathway for validating theoretical results and guiding model implementations. While the study has successfully addressed the existence of solutions, several challenges remain. The idealized assumptions made regarding spherical kinematic pairs and the linearization of certain nonlinear relationships suggest that further research is needed to extend these results to more complex. Future work will focus on refining the modeling techniques, incorporating dynamic effects, and exploring optimization methods to enhance the performance and robustness of the synthesized mechanisms. Overall, this research contributes to the theoretical foundation and practical methodology for spatial mechanism design, offering valuable insights for engineers and researchers working on advanced kinematic synthesis problems. The algorithm for synthesizing initial kinematic chains with spherical kinematic pairs provides an effective method for the structural-kinematic synthesis of spatial lever mechanisms. By iteratively minimizing the objective function S, the positions of points A, B, and C can be optimized to create the desired kinematic configuration. The method can be applied to generate various modifications of the original kinematic chain, making it versatile for different configurations of spatial mechanisms. This iterative approach, proven to converge through the Weierstrass theorem, ensures that the algorithm will yield optimal solutions for the synthesis of kinematic chains, making it a powerful tool for the design and analysis of spatial mechanisms.

Author Contributions: Conceptualization, A.Z. and S.K.; methodology, A.Z.; software, A.Z.; validation, A.Z.; formal analysis, A.Z. and K.A.; investigation, A.Z.; resources, A.A. and S.A.; data curation, A.O.; writing—original draft preparation, A.Z., S.A., S.K. and K.A.; writing—review and editing, A.Z., A.A., and A.O.; visualization, A.Z. and A.O.; supervision, A.Z.; project administration, A.Z., S.K. and A.A.; funding acquisition, A.Z., S.A., and K.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Almaty University of Power Engineering and Telecommunications named after Gumarbek Daukeyev, grant number AP19677356.

Institutional Review Board Statement: The study did not require ethical approval.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work has been supported financially by the research project (AP19677356—To develop systems for controlling the orientation of nanosatellites with flywheels as executive bodies based on linearization methods) of the Ministry of Education and Science of the Republic of Kazakhstan and was performed at Research

Institute of Communications and Aerospace Engineering in Almaty University of Power Engineering and Telecommunications named after Gumarbek Daukeyev, which is gratefully acknowledged by the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Che, L. X.; Chen, G. W.; Jiang, H. Y.; Du, L.; Wen, S. K. Dimensional synthesis for a Rec4 parallel mechanism with maximum transmission workspace. *Mech. and Mach. Theor.* **2020**, *153*,104008.
2. Krovi, V.; G. K. Ananthasuresh, G. K.; Kumar, V. Kinematic synthesis of spatial R-R dyads for path following with applications to coupled serial chain mechanisms. *J. of Mech. Desig., Trans. of the ASME* **1999**, *123*, 359-366.
3. Copeland, Z. A. Continuous approximate kinematic synthesis of planar, spherical, and spatial four-bar function generating mechanisms. Ph.D. dissertation, Carleton University, Ottawa, Ontario, Canada, April 26, 2024.
4. Freudenstein, F. Structural error analysis in plane kinematic synthesis. *J. of Engin. for Indus.***1959**, *81*, 15–21.
5. Freudenstein, F. Approximate synthesis of four-bar linkages. *Trans. of the ASME* **1955**, *77*, 853–861.
6. Rao S.; Ambekar, A. Optimum design of spherical 4-R function generating mechanisms. *Mech. and Mach. Theor.* **1974**, *9*, 405–410.
7. Rotzoll, M.; Regan, M. H.; Husty, M. L.; Hayes, M. J. D. Kinematic geometry of spatial RSSR mechanisms. *Mech. and Mach. Theo.* **2023**, *185*.
8. Alizade, R.; F. C. Can, F. C.; Kilit, O. Least square approximate motion generation synthesis of spherical linkages by using chebyshev and equal spacing. *Mech. and Mach. Theo.* **2013**, *61*, 123–135.
9. Alizade, R.; Bayram, C. Structural synthesis of parallel manipulators. *Mech. and Mach. Theo.* **2004**, *39*, 857–870.
10. Zhao, J.S.; Zhou, K.; Feng, Z, J. A theory of degrees of freedom for mechanisms, *Mech. and Mach. Theo.* **2004**, *39*, 621–643.
11. Zhao, J.; Feng, Z.; Chu, F.; Ma, N. Structure synthesis of spatial mechanisms. *Adv. The. of Cons. and Mot. Analy. for Rob. Mech.* **2014**, 367-396. Book Chapter, https://doi.org/10.1016/B978-0-12-420162-0.00012-6
12. Tultayev, B.; Balbayev, G.; Zhauyt, A.; Sultan, A.; Mussina, A. Kinematic synthesis of mechanism for system with a technical vision. *MATEC Web of Conferences* **2018**, *237*, 03009. http://dx.doi.org/10.1051/matecconf/201823703009
13. Liao, Z.; Tang, S.; Wang, D. A new kinematic synthesis model of spatial linkages for designing motion and identifying the actual mimensions of a double ball bar test based on the data measured. *Machines* **2023**, *11*, 919. https://doi.org/10.3390/machines11090919
14. Faizov, M.; Fanil, Kh. Computations analysis of a four-link spherical mechanism for a spatial simulator. *Vestnik Moskovskogo Aviatsionnogo Instituta* **2020**, *27*, 196–206. http://dx.doi.org/10.34759/vst-2020-2-196-206
15. Grimm, E.M.; Andrew, P.M.; Michael, L.T. Software for the kinematic synthesis of coupler-driven spherical four-bar mechanisms. *In ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. ASMEDC*, **2007**. http://dx.doi.org/10.1115/detc2007-35720
16. Vilas, C. P. Optimal synthesis of adjustable four-link planar and spherical crank-rocker type mechanisms for approximate multi-path generation. *Thesis*, **2013**. http://etd.iisc.ac.in/handle/2005/3319.
17. Vilas, C. P.; Michael, A.A. F.; Ashitava, G. Synthesis of adjustable spherical four-link mechanisms for approximate multi-path generation. *Mech. Mach. The.* **2013**, *70*, 538–552. http://dx.doi.org/10.1016/j.mechmachtheory.2013.08.009
18. Yao, J.; and Jorge, A. The kinematic synthesis of steering mechanisms. *Trans. of the Canad. Soci. for Mech. Eng.* **2000**, *24*, 453–476. http://dx.doi.org/10.1139/tcsme-2000-0035
19. Acar, O.; Hacı, S.; and Ziya, S. Measuring curvature of trajectory traced by coupler of an optimal four-link spherical mechanism. *Measurement* **2021**, *176*, 109189. https://doi.org/10.1016/j.measurement.2021.109189
20. Hunt, K.H. *Kinematic Geometry of Mechanisms*; Oxford University Press: Oxford, UK, 1978.
21. Huang, X.; Liu, C.; Xu, H.; Li, Q. Displacement analysis of spatial linkage mechanisms based on conformal geometric algebra. *J. Mech. Eng.* **2021**, *57*, 39–50.

22. Wang, D.; Wang, W. *Kinematic Differential Geometry and Saddle Synthesis of Linkages*; John Wiley & Sons: Singapore, 2015.

23. Wang, D.; Wang, Z.; Wu, Y.; Dong, H.; Yu, S. Invariant errors of discrete motion constrained by actual kinematic pairs. *Mech. Mach. Theory* **2018**, *119*, 74–90.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.