Proceedings of the ASME 2026
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE 2026
August 23–26, 2026, Houston, TX

DETC2026-193961

WHERE WE ARE IN THE PATH TOWARDS AGI IN KINEMATICS?

Hai-Jun Su[1,*], J. Michael McCarthy[2]

[1]Department of Mechanical & Aerospace Engineering, The Ohio State University, Columbus, Ohio, USA
[2]Department of Mechanical & Aerospace Engineering, University of California, Irvine, CA, USA

ABSTRACT

This paper assesses the current level of artificial intelligence in solving kinematics problems and uses that assessment to discuss progress toward Artificial General Intelligence (AGI) in this domain. We propose a 5-level hierarchy of AI intelligence in kinematics, ranging from routine classroom analysis to open-ended discovery, to categorize problem complexity and autonomous solver capability. The paper surveys representative open problems in kinematic synthesis, parallel mechanisms, compliant mechanisms, and reconfigurable mechanisms as context for this hierarchy; however, the main technical evaluation is intentionally focused on classical benchmark problems, especially the inverse kinematics of general 6R serial robots. For the 6R case, we document the mathematical formulation, the classical Raghavan–Roth dialytic elimination procedure, and the LLM prompting strategy required to generate numerically robust computer algebra and numerical solvers. Evaluation success is defined by executable generated code that reproduces the expected analytical pipeline and passes quantitative FK–IK–FK validation against known benchmark cases. Our findings indicate that current state-of-the-art LLMs operate effectively at Level 3 and can approach Level 4 only when structured prompts encode substantial expert domain knowledge. The results support a cautious but useful view of LLMs as accelerators for implementing difficult analytical kinematics methods, while also showing that human expert intervention remains essential for selecting stable symbolic and numerical strategies.

Keywords: kinematics, large language models, artificial intelligence, open kinematics problems, code generation

1. INTRODUCTION

Kinematics is a foundational discipline in mechanical engineering and robotics that deals with the geometry of motion. The history of kinematics over the past two centuries is characterized by a progressive abstraction of motion and a continuous quest to solve increasingly complex nonlinear polynomial systems. This evolution can be traced through several landmark milestones. During the late 19th century, the "Golden Age of Geometry" witnessed the rigorous graphical and geometric synthesis of linkages, culminating in Burmester Theory for the exact dimensional synthesis of mechanisms traversing discrete poses [1, 2]. As robotics advanced in the late 20th century, the focus shifted to spatial mechanisms during a period of "Computational Conquest." The most celebrated challenge of this era was the closed-form solution to the "*Closed form , algebraic solution of the input/output displacement equation of the general, single-loop, single-degree-of-freedom spatial mechanism consisting of seven links and seven turning pairs of" arbitrary orientation in space*", or 7R kinematics problem, which was considered the "Mount Everest" of kinematics problems by the father of modern kinematics, Ferdinand Freudenstein [3]. This problem is mathematically equivalent to the inverse kinematics of 6R robot manipulators. Researchers definitively proved that the 6R IK problem admits at most 16 distinct assembly configurations [4–6]. This classical milestone problem has historically required sophisticated algebraic elimination techniques and highly robust numerical methods to solve.

The analytical solutions to these problems are well documented in the literature, yet the process of implementing them—deriving equations, coding solvers, and validating results—remains tedious, error-prone, and time-consuming. The application of large language models (LLMs) to engineering and scientific problems has rapidly expanded, opening new paradigms for human-AI collaboration and autonomous discovery. Recent advances in LLMs such as GPT-4 [7] and GPT-5 [8] exhibit emergent capabilities in mathematical reasoning and symbolic manipulation. For instance, Romera-Paredes et al. [9] utilized LLMs to discover novel mathematical constructions, while Trinh et al. [10] developed AlphaGeometry for solving International Mathematical Olympiad geometry problems.

Within engineering and robotics, Guo et al. [11] systematically benchmarked LLM capabilities across diverse engineering

*Corresponding author: su.298@osu.edu

design tasks, including mechanical design, robotics, structural analysis, and optimization. Kim et al. [12] extensively surveyed the integration of LLMs with intelligent robots. Recently, Su presented an LLM-assisted human–AI collaborative framework specifically targeted at developing analytical IK solvers for 6R robots [13]. By having human experts provide strategic algebraic guidance via structured prompts, the LLM agent successfully developed analytical IK solvers for 842 commercial and over 1000 random 6R robots, achieving a 100% success rate and reducing development time from weeks to minutes.

2. DEFINING AGI IN KINEMATICS

Currently, there is no consensus on the specific kinematics problems that would serve as a definitive benchmark for Artificial General Intelligence (AGI). To establish such a standard, we can draw inspiration from the "Einstein test" proposed by Demis Hassabis, CEO of Google DeepMind. Hassabis suggested a rigorous test for AGI: "The kind of test I would be looking for is training an AI system with a knowledge cutoff of, say, 1911, and then seeing if it could come up with general relativity, like Einstein did in 1915. That's the kind of test I think is a true test of whether we have a full AGI system" [14].

Translating this concept to our domain, we propose a similar benchmark for kinematics:

> *If all knowledge after 1980 were removed from its training corpus, could an AI system independently derive the inverse kinematics solution of a general spatial 6R serial manipulator, or equivalently construct a solution framework for a spatial 7R closed-loop linkage?*

Passing this test would imply that the AI surpasses the logical reasoning capabilities of the most brilliant human kinematicians of that era.

Crucially, validating AGI requires **separating computational complexity from symbolic reasoning challenges**. Computational complexity typically implies that the derived kinematic constraint equations require significant computational power—such as high-performance computing facilities—and the application of advanced numerical polynomial solvers, such as polynomial homotopy continuation via Bertini [15] or POLSYS_GLP [16]. Notable examples of such computationally demanding limits include the complete solution of the nine-point path synthesis problem for four-bar linkages by Wampler et al. [17], and the computation of reachable surfaces for spatial open chain synthesis by Su and McCarthy [18]. While solving these problems requires massive parallel computation, it can largely be addressed by scaling computing resources and does not necessarily reflect deep "brain power." In contrast, symbolic reasoning challenges—such as formulating novel algebraic elimination strategies, recognizing profound geometric insights, or mapping a physical constraint to a mathematical manifold without human help—constitute a true test for AI models.

3. OPEN KINEMATICS PROBLEMS

To illustrate the symbolic reasoning challenges previously discussed, we highlight several open or hard problems across various subfields of kinematics and mechanisms. These problems, selected from a series of deep research reports (available at https://github.com/haijunsu-osu/llm_kinematics_deep_research_report) generated by OpenAI's GPT-5 Deep Research model, serve as candidate future benchmarks for evaluating progression toward AGI in kinematics, rather than as reported evaluation cases in this paper. The detailed prompts used to generate these reports are also publicly available in the same GitHub repository.

Please note that these reports reflect personal opinions and exploratory assessments. They are not intended to represent the general consensus of the kinematics and mechanism research community. However, these problems are representative of the challenges that AI models must overcome to achieve AGI in kinematics. We hope these problems can stimulate further discussions and research in the field.

3.1. Kinematic Synthesis of Linkages

Kinematic synthesis is the process of determining the topology and dimensions of a mechanism that will achieve a desired motion or force transmission. This field encompasses both type synthesis and dimensional synthesis, typically utilizing either model-based methods (rooted in kinematic theory and algebraic geometry) or data-driven methods (leveraging learning-based approaches and synthetic datasets). Both paradigms often involve solving complex nonlinear algebraic systems or navigating massive search spaces.

- *What methodologies can guarantee the synthesis of planar linkages that naturally satisfy complex inequality constraints such as branch defect avoidance and physical limits?*
- *By what means can we achieve scalable, complete kinematic synthesis for complex planar mechanisms despite massive polynomial complexity and structural degeneracies?*
- *Is it possible to synthesize mechanisms that robustly approximate continuous trajectories under manufacturing tolerances and noise?*

3.2. Parallel Mechanisms

Parallel mechanisms consist of multiple independent kinematic chains working in parallel to support and manipulate a common platform. They are valued for their high stiffness and precision but suffer from complex workspaces and singularities.

- *What strategies allow for complete and scalable type synthesis and enumeration of non-isomorphic parallel architectures without succumbing to combinatorial explosion?*
- *How might we compute and certify exact, singularity-free workspaces for general parallel mechanisms under realistic parameter uncertainties?*

- *Could low-cost, in-situ calibration methods be developed to robustly identify and compensate for dominant elasto-geometric errors over the entire workspace?*
- *What approaches ensure real-time forward kinematics and assembly-mode tracking that avoids singularities and correctly selects physical branches?*

3.3. Compliant Mechanisms

Compliant mechanisms achieve motion through the deformation of flexible members, necessitating an integration of kinematics and structural mechanics. Research in this field leverages a diverse array of modeling and design paradigms, including Pseudo-Rigid-Body (PRB) models, Beam Constraint Models (BCM), topological optimization, constraint-based approaches such as screw theory and Freedom and Constraint Topology (FACT), and nonlinear finite element analysis (FEA). These methods provide the foundation for synthesizing monolithic structures with high precision across multiple length scales.

- *Can a validated, multi-fidelity modeling stack be constructed to hierarchicalize fast approximations with accurate nonlinear finite element analysis for large deflections?*
- *What advancements are required to make non-linear topology optimization for compliant mechanisms tractable at an engineering scale?*
- *In what ways can crucial time-dependent material behaviors—such as fatigue and stress relaxation—be systematically incorporated into generative design loops?*

3.4. Reconfigurable Mechanisms and Origami/Kirigami Structure Design

Reconfigurable mechanisms can change their physical configuration to adapt to different tasks or environments. Origami- and kirigami-inspired designs leverage folding and cutting patterns to create complex 3D structures from flat sheets.

- *What planning frameworks can guarantee physically feasible folding paths for complex patterns while navigating self-contact and numerous bifurcation modes?*
- *Could geometric and kinematic constraints be integrated into the dimensional synthesis of mechanisms that achieve multiple functional configurations?*
- *Through what computational models can we accurately predict the rugged transition pathways and energy landscapes of thick compliant patterns?*

4. HIERARCHICAL PROBLEM DECOMPOSITION

To frame the evaluation of LLM capabilities, we propose a 5-level hierarchy of AI intelligence in kinematics, progressing from routine analytical execution to profound autonomous discovery:

1. **Level 1: Classroom Intelligence (Routine Analysis):** Capable of solving standard textbook classroom problems and executing well-established, straightforward analytical procedures without requiring advanced algebraic manipulation. *Example: Kinematic analysis of planar 4-bar linkages.*
2. **Level 2: Internalized Research Intelligence (Autonomous Synthesis):** Can solve established research questions autonomously without the need for external references, relying on internalized, pre-trained knowledge to derive constraints and generate solvers. While many synthesis problems are very computationally demanding, the derivation of equations is often less intellectually challenging. *Example: Exact synthesis of spatial dyads (e.g., RR or PR).*
3. **Level 3: Guided Research Intelligence (Complex Reproduction):** Capable of solving highly complex research questions that are already solved in literature, but requires a reference paper or document to follow in order to reproduce dense algebraic pipelines. *Example: Forward kinematics of the general Stewart–Gough platform or inverse kinematics of general 6R robots by following the provided reference paper.*
4. **Level 4: Autonomous Discovery Intelligence (Solving the Solvable Unsolved):** Can solve problems that currently have no existing analytical answer or documented solution in literature. The AI must formulate its own unique elimination strategy from scratch. *Example: Complete analytical IK for novel 6R robot morphologies without a spherical wrist that have not been solved before.*
5. **Level 5: AGI in Kinematics (Solving Open Questions):** Artificial General Intelligence capable of solving foundational, open questions in theoretical kinematics that have eluded human experts. *Example: Derive the analytical IK for the general 6R robot without referring to any existing literature.*

Level 5 is retained here as a conceptual benchmark for AGI-level discovery. It is not treated as an evaluated result in this paper, because the study does not test whether an LLM can independently rediscover the general 6R IK solution without reference material or expert guidance.

Figure 1 illustrates this hierarchy alongside representative kinematics problem examples for each intelligence plateau. In this paper, we systematically evaluate LLM capabilities across a spectrum of classical problems that focus progressively across Levels 1 to 3, touching the boundary of Level 4. We selected five representative problem categories:

1. **Kinematic analysis and synthesis** of planar and spherical linkages [19]. The solution is provided.
2. **Synthesis of spatial SS dyads** [19]. The solution is provided.
3. **Forward kinematics of general Stewart–Gough platforms** [20]: Study quadric formulation yielding a 40th-degree polynomial with up to 40 assembly configurations. The solution is provided.

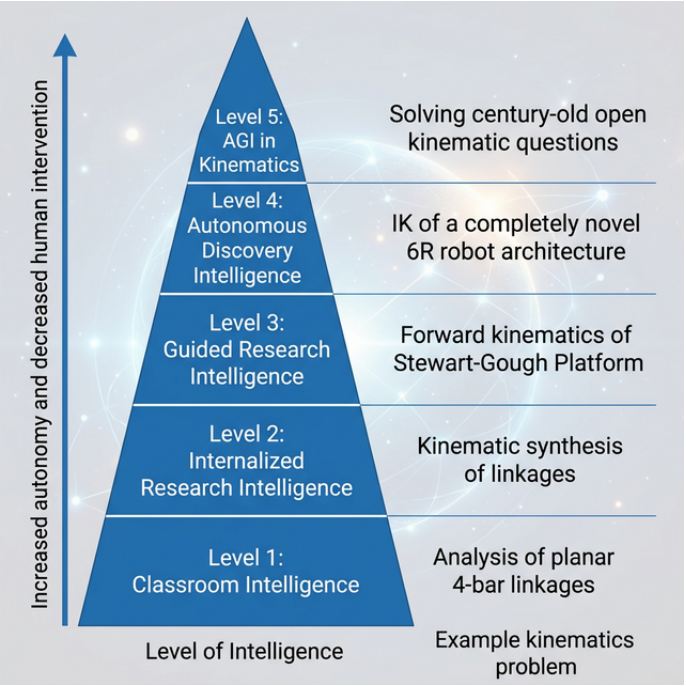

FIGURE 1: A proposed 5-level hierarchy of AI intelligence in kinematics, ranging from routine classroom analysis to Artificial General Intelligence (AGI) capable of solving open kinematics questions without human intervention.

4. **Inverse kinematics of general 6R robots** [6]: Dialytic elimination yielding a 16th-degree characteristic polynomial with up to 16 solutions. The documented solution is provided.

5. **Inverse kinematics of 6R robots with special architectures** (e.g., spherical wrists or parallel axes) [13]. The solution is not provided.

These problems are ordered by increasing difficulty in code generation and the use of computer algebra systems (CAS) for symbolic reasoning.

For consistency, we use the following reporting definitions. A benchmark is counted as successful only when the generated code is executable and passes an independent validation test appropriate to the problem, such as agreement with a published result, recovery of all expected assembly modes, or FK–IK–FK residuals below the stated tolerance. The reported time is the total wall-clock human–AI session time from the initial prompt to a validated solver, including prompt revision, code inspection, debugging, and validation, rather than the runtime of the final generated program. Prompt iterations count substantive follow-up prompts that changed the mathematical strategy, implementation, or validation criteria; minor formatting requests are not counted. Critical expert guidance refers to domain-specific mathematical intervention by the human expert, such as specifying an elimination order, identifying a stable eigenvalue formulation, or correcting an invalid symbolic step.

5. PROMPTING LLMS FOR CODE GENERATION

Recent LLMs such as OpenAI's Codex-5.3 [8] have shown remarkable capabilities in code generation and symbolic manipulation, motivating their application to classical kinematics problems. This section describes the collaborative workflow, agent environment, and prompting strategy used throughout this benchmark.

5.1. Collaborative Workflow

The human-AI collaborative workflow for solving kinematics consists of three phases:

1. **Problem Specification**: The human expert provides the mathematical formulation, identifies the solution strategy, and specifies validation criteria.

2. **Prompt Design**: A structured prompt is crafted with four blocks—context, task, requirements, and deliverables—encoding the expert's domain knowledge.

3. **Iterative Refinement**: The LLM generates code, the expert reviews the output, and follow-up prompts address errors or missing functionality.

5.2. Prompt Structure

Each prompt includes: (1) a **context** block establishing the LLM's role as a domain expert, (2) a **task** block defining the specific problem and solution method, (3) a **requirements** block specifying edge cases, numerical tolerances, and validation tests, and (4) a **deliverables** block describing the expected output format.

To illustrate this structure, consider the following abbreviated example prompt used for the general 6R inverse kinematics solver:

```
[Context] You are an expert in robot kinematics, symbolic
elimination, and numerical computing. Read the three
provided reference PDFs regarding the Raghavan-Roth 1993
elimination method and the companion-matrix generalized
eigenvalue strategy for the general 6R IK problem.
[Task] Your task is to reconstruct the full pipeline in four
phases: 1) extract text/equations via OCR, 2) perform
symbolic derivations in Mathematica, 3) implement a
numeric-only Python solver, and 4) translate the solver into
C++.
[Requirements] For each phase, provide a test verifying the
FK-IK-FK cycle with explicit error thresholds. Run stress
tests across multiple random 6R instances and validate
against the published 16-real-solution reference case.
[Deliverables] Deliver the Python/C++ IK solvers, the Mathe-
matica derivation script, and a final markdown report summa-
rizing validation outcomes and exact numerical residuals.
```

Finally, it is important to note that the quality and result of the generated code depend heavily on four key factors: 1) the specific foundation model used; 2) whether computational tools like a CAS are leveraged; 3) the inherent complexity of the kinematics problem; and 4) the quality and specificity of the provided prompts. Furthermore, users will encounter a consistency problem: due to the non-deterministic nature of generative models, executing the exact same prompt across distinct runs may yield different algorithms, approaches, or success rates.

TABLE 1: Reported LLM code generation performance across evaluated kinematics benchmarks of increasing difficulty. Conceptual Level 5 benchmarks are excluded because they were not evaluated as reported results.

Problem Category	AGI Level	Time (total session)	Prompt Iter.	Critical Expert Guidance
1. Planar & Spherical Linkages [2]	Level 1	< 5 min	Single-shot	None
2. Spatial SS Dyad Synthesis (Solution [21] provided)	Level 2	~ 15 min	Few	Specifying eigenvalue method
3. Stewart–Gough FK (Solution [20] provided)	Level 3	~ 30 min	Single-shot	Directing zero-dimensional ideal elimination
4. General 6R Robot IK (Solution [6] provided)	Level 3	Multi-hour to multi-day total	Many	Derivation of 14 equations, generalized eigenvalue step for numeric robustness
5. 6R IK (Special Architecture) (No solution provided) [13]	Level 4	Multi-hour to multi-day total	Many	Development of helper functions, architectural decoupling and elimination order

Note: Time denotes total wall-clock human–AI session effort, including human prompt engineering, model execution, code review, debugging, and validation, not CAS or solver runtime alone. Prompt iterations count substantive human follow-up prompts after the initial task prompt. "Success" means that generated executable code passed the stated validation criteria for that benchmark, such as reproducing published solution behavior and satisfying FK–IK–FK residual checks.

5.3. CAS-Based Solutions and Expert Supervision

Many kinematics problems require computer algebra systems (CAS) such as Mathematica or SymPy for symbolic derivation before numerical solution. However, AI agents still struggle with the combinatorial explosion of symbolic terms when fully autonomous. Human expert supervision must be injected through the prompts to guide the CAS derivations [13]. Strategic tips encoded in prompts typically include:

- **Variable elimination order**: Suggesting which joint angles to eliminate first based on architectural coupling structures.
- **Algebraic simplification**: Mandating specific trigonometric identities (e.g., $\sin^2\theta + \cos^2\theta = 1$) to prevent term explosion.
- **Matrix operations**: Replacing generic symbolic 4×4 matrix inverses with analytic SE(3) block inverses for numerical stability.
- **Geometric insights**: Leveraging known hardware constraints (e.g., equating twist angles) rather than algebraic brute force.
- **Recognizing solvable patterns**: Guiding the AI to map derived expressions into known solvable systems (e.g., bilinear two-angle systems).

6. INVERSE KINEMATICS OF A GENERAL 6R ROBOT

As summarized in Table 1, the evaluated benchmark kinematics problems are classified into five reported categories of increasing difficulty. Categories 1 through 3, as well as the special architecture 6R IK benchmark (Category 5), have already been successfully solved using this LLM-assisted framework and are detailed in recent papers [13, 19]. In this section, we present the solution process for Category 4: the analytical inverse kinematics of the general 6R serial robot, where a reference solution methodology [6] is provided to guide the LLM. If no solution reference or expert guidance were provided, independently deriving this general 6R IK method would be a conceptual AGI Level 5 challenge; that unevaluated case is therefore not reported as a performance result in Table 1.

6.1. Problem Statement

We reproduce the inverse kinematics solution for a general 6R serial robot following the Raghavan–Roth resultant elimination method [6]. Given a desired end-effector pose ${}^0T_6^d$, the IK problem seeks all sets of joint angles $(\theta_1, \ldots, \theta_6)$ satisfying the forward kinematics equation. The general 6R robot admits up to 16 solutions.

6.2. Kinematics Equations

For a 6R serial chain with Denavit–Hartenberg parameters (a_i, α_i, d_i), each joint transform is

$$A_i(\theta_i) = R_z(\theta_i)\, T_z(d_i)\, T_x(a_i)\, R_x(\alpha_i). \tag{1}$$

The forward kinematics equation $A_1A_2A_3A_4A_5A_6 = A_{\text{hand}}$ is rearranged by splitting the chain at joint 3. Following Raghavan and Roth [6], the matrix A_3 is factored as $A_3 = A_{3v}A_{3s}$, where $A_{3v} = R_z(\theta_3)$ contains the variable θ_3 and $A_{3s} = T_z(d_3)\, T_x(a_3)\, R_x(\alpha_3)$ contains only the structural constants. Pre-multiplying by $A_2^{-1}A_1^{-1}$ and post-multiplying by A_6^{-1} gives $A_{3v}A_{3s}A_4A_5 = A_2^{-1}A_1^{-1}A_{\text{hand}}A_6^{-1}$. Since A_{3s} comprises only structural DH constants, it is absorbed into a redefined joint-4 matrix $A_4' \equiv A_{3s}A_4$, yielding:

$$A_{3v}A_4'A_5 = A_2^{-1}A_1^{-1}\, A_{\text{hand}}\, A_6^{-1}. \tag{2}$$

For clarity, A_4' is referred to as A_4 throughout the remainder of this section. The left-hand side (LHS) is a function of $(\theta_3, \theta_4, \theta_5)$ and the right-hand side (RHS) is a function of $(\theta_1, \theta_2, \theta_6)$.

From Eq. 2, Raghavan and Roth extract the 4th (position) column as $\mathbf{p}$ and the 3rd (approach/z-axis) column as $\boldsymbol{\ell}$ from the matrix on each side. Since θ_6 appears only in two columns of the RHS rotation matrix, $\boldsymbol{\ell}$ and $\mathbf{p}$ are independent of θ_6. This separates θ_6 from the system, leaving only (θ_1, θ_2) on the RHS. In addition, they derive 8 more equations by computing polynomial combinations of $\mathbf{p}$ and $\boldsymbol{\ell}$ that preserve the same bilinear monomial structure. Table 2 summarizes the 14 equations.

TABLE 2: The 14 Raghavan–Roth scalar equations from geometric vectors p and $\boldsymbol{\ell}$

Equation Type	Vector Expression	Count
Position	$\mathbf{p}$	3
Approach	$\boldsymbol{\ell}$	3
Squared distance	$\mathbf{p} \cdot \mathbf{p}$	1
Dot product	$\mathbf{p} \cdot \boldsymbol{\ell}$	1
Cross product	$\mathbf{p} \times \boldsymbol{\ell}$	3
Vector combination	$(\mathbf{p} \cdot \mathbf{p})\boldsymbol{\ell} - 2(\mathbf{p} \cdot \boldsymbol{\ell})\mathbf{p}$	3
Total		**14**

Because each component in Table 2 is already bilinear in the same monomial sets, these polynomial products yield expressions with the *same power products* in (θ_4, θ_5) and (θ_1, θ_2). This algebraic property enables the assembly of the 14 equations into a shared linear structure.

Collecting all constraints, 14 independent scalar equations are obtained. For a fixed θ_3, each equation is bilinear in trigonometric monomials of (θ_4, θ_5) from the LHS and (θ_1, θ_2) from the RHS:

$$P_r\, \mathbf{m}_{45} = Q_r\, \mathbf{n}_{12}, \quad r = 1, \ldots, 14, \tag{3}$$

where the monomial vectors are

$$\mathbf{m}_{45} = [s_4 s_5,\ s_4 c_5,\ c_4 s_5,\ c_4 c_5,\ s_4,\ c_4,\ s_5,\ c_5,\ 1]^T, \tag{4}$$

$$\mathbf{n}_{12} = [s_1 s_2,\ s_1 c_2,\ c_1 s_2,\ c_1 c_2,\ s_1,\ c_1,\ s_2,\ c_2]^T. \tag{5}$$

Note that $\mathbf{m}_{45}$ has 9 entries (including the constant 1 from terms involving only θ_3), while $\mathbf{n}_{12}$ has 8 entries. Stacking all 14 rows yields the matrix system:

$$P(\theta_3)\, \mathbf{m}_{45} = Q(\theta_3)\, \mathbf{n}_{12}, \quad P \in \mathbb{R}^{14\times 9},\ Q \in \mathbb{R}^{14\times 8}. \tag{6}$$

The coefficient matrices P and Q depend on θ_3 (via $\sin\theta_3$ and $\cos\theta_3$) and are linear in the DH constants and target pose entries.

6.3. Solution Process

The solution process eliminates the intermediate joint angles sequentially. First, (θ_1, θ_2) are eliminated using the left null space of Q, yielding a smaller system $E_{45}\, \mathbf{m}_{45} = 0$. Using the tangent half-angle substitution $x_i = \tan(\theta_i/2)$, the trigonometric equations are converted into a polynomial system in variables x_3, x_4, x_5. Dialytic lifting is then applied by multiplying the equations by monomials, resulting in a 12×12 matrix system $M(x_3)\, \mathbf{z}_{12}(x_4, x_5) = 0$.

The requirement that this larger system must have a non-trivial solution dictates that $\det M(x_3) = 0$, forming the characteristic equation of degree 16. To robustly find these roots as recommended by Manocha and Canny [22], M is expressed as a quadratic matrix polynomial $\Sigma(x_3) = A\, x_3^2 + B\, x_3 + C$, which is solved via a 24×24 generalized eigenvalue problem constructed from the companion matrix. Upon identifying the real eigenvalues (the roots x_3), back-substitution yields the remaining joint angles (x_4, x_5), (θ_1, θ_2), and finally θ_6.

Algorithm 1 Inverse Kinematics of General 6R Robot [6]

Require: 6 DH parameters and target transformation A_{hand}
Ensure: Up to 16 joint angle solutions $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$
1: Rearrange kinematics as $A_{3v} A_4' A_5 = A_2^{-1} A_1^{-1} A_{\text{hand}} A_6^{-1}$, where $A_4' \equiv A_{3s} A_4$ absorbs the structural factor of joint 3
2: Extract 3 position $\mathbf{p}$ equations and 3 approach vector $\boldsymbol{\ell}$ equations
3: Generate 8 additional equations via vector products/combinations: $\mathbf{p} \cdot \mathbf{p}, \mathbf{p} \cdot \boldsymbol{\ell}, \mathbf{p} \times \boldsymbol{\ell}, (\mathbf{p} \cdot \mathbf{p})\boldsymbol{\ell} - 2(\mathbf{p} \cdot \boldsymbol{\ell})\mathbf{p}$
4: Form the 14 equations: $P(\theta_3)\, \mathbf{m}_{45}(\theta_4, \theta_5) = Q\, \mathbf{n}_{12}(\theta_1, \theta_2)$
5: Eliminate θ_1, θ_2 via the left null space N^T of Q (i.e., $N^T Q = 0$), forming $E_{45} = N^T P$ such that $E_{45}\, \mathbf{m}_{45} = 0$
6: Substitute $x_i = \tan(\theta_i/2)$ to get polynomials in x_3, x_4, x_5
7: Form 12×12 dialytic lifting matrix $\Sigma(x_3) = A x_3^2 + B x_3 + C$
8: Formulate 24×24 generalized eigenvalue pencil $C_1 \mathbf{w} = x_3 C_2 \mathbf{w}$
9: Compute real eigenvalues to obtain roots for x_3
10: Back-substitute to find (x_4, x_5), then (θ_1, θ_2), and finally θ_6
11: **return** Valid real assembly configurations

6.4. Code Generation by LLM

To initiate the code generation process for the general 6R IK solver, the initial prompting process was quite casual by generally following the standard prompt structure given in Section 5. However, the model failed to reproduce the results due to three primary errors: (1) reading the reference paper without using optical character recognition (OCR), which led to a misunderstanding of the complex mathematical formulas; (2) pitfalls in the symbolic derivation process where the Mathematica script frequently became stuck; and (3) severe numerical instability when attempting to solve the final polynomial directly. Resolving these issues proved to be a highly iterative process, requiring hours or days of debugging by examining intermediate results and providing targeted hints. Specifically, to overcome the numerical instability, the human expert had to explicitly show the AI model the method presented by Manocha and Canny [22], guiding it to use the companion matrix and generalized eigenvalue formulation. This highlights precisely where human expertise is indispensable to guide the AI through complex analytical bottlenecks. After many iterations, a final working Mathematica script is generated as shown in Appendix A.

After the iterative development process produced a validated pipeline, we asked the AI to generate a final refined prompt, similar to the one demonstrated in Section 5, partitioning the necessary domain knowledge into context, task, requirements, and deliverables blocks. This prompt assumes three reference PDF files are available in the LLM's workspace context: the analytical method by Raghavan and Roth [6], the generalized eigenvalue formulation by Manocha and Canny [22], and the 16-real-solution reference case by Manseur and Doty [23]. A detailed version of this prompt is provided in the project's companion GitHub repository [24].

Using this refined prompt, the entire IK code pipeline—ranging from the Mathematica symbolic derivation script to the Python numerical solver, the C++ implementation, and the final reports—can be reproduced via a **single-shot prompt** in the Codex 5.3 (Extra High) model without additional mid-run human intervention. This single-shot reproduction should be interpreted as the result of prior expert prompt engineering, not as evidence that the model autonomously discovered the Raghavan–Roth method or selected the stable numerical formulation from scratch.

6.5. Evaluation Results

The solver was validated using the classical Manseur–Doty example [23], a robot known to possess 16 real IK solutions. Table 3 lists the DH parameters.

TABLE 3: DH parameters for the Manseur–Doty 16-real-root example

i	a_i	d_i	α_i (rad)	α_i (deg)
1	0.3	0	1.57	90
2	1	0	1	57.3
3	0	0.2	1.57	90
4	1.5	0	1	57.3
5	0	0	1.57	90
6	0	0	1	57.3

The reference joint tuple is $\mathbf{q}^* = [0°, 107.46°, 112.46°, -7.66°, 0°, 0°]$. All 16 IK solutions were recovered, as shown in Table 4.

Solution 4 corresponds to the planted reference tuple $\mathbf{q}^*$. All 16 solutions have FK position residuals below 10^{-9}, confirming the solver's numerical accuracy. Note that solutions come in 8 pairs sharing (θ_1, θ_2) values but differing in $(\theta_3, \theta_4, \theta_5, \theta_6)$. This pairing arises because, for each pair of shoulder joint angles (θ_1, θ_2) placing the wrist center, the elbow joint θ_3 and wrist joints $(\theta_4, \theta_5, \theta_6)$ can adopt two symmetric postures, yielding $2 \times 8 = 16$ total configurations.

A C++ implementation of the same algorithm was also generated by the LLM, achieving a 106× speedup over the Python version (mean IK time 0.0015 s vs. 0.161 s) on a random benchmark of 1000 IK queries, while preserving near-complete solve coverage (999/1000). The single failure occurred on a near-singular kinematic configuration where the generalized eigenvalue pencil produced a spurious complex root that did not pass the real-root filter, a known sensitivity of the companion-matrix approach near geometric singularities. The full evaluation result reports are available in the GitHub repository [24].

7. DISCUSSION

7.1. Summary of Evaluation Results

The evaluated benchmark categories reveal a clear and consistent progression in LLM code generation difficulty that tracks with the mathematical depth and sequentiality of the underlying derivation pipeline rather than with algebraic degree alone, as summarized in Table 1. The reported wall-clock session times, ranging from under 5 minutes to multi-hour or multi-day total human–AI sessions, directly reflect this progression and identify where human expert effort is most concentrated.

For foundational problems such as the kinematic analysis and synthesis of planar and spherical linkages [19], the LLM operated highly autonomously, generating working code in a single shot with virtually no expert intervention. As the complexity increased to the spatial SS dyad synthesis [19], the algebraic formulation led to numerical instability, requiring the human expert to explicitly intervene and direct the LLM to adopt a companion-matrix generalized eigenvalue formulation.

In the realm of highly complex spatial mechanisms, the forward kinematics of the Stewart–Gough platform proved surprisingly tractable, reaching a functionally complete solution in *approximately 30 minutes* in a single-shot session. Despite involving a multi-stage resultant cascade to derive a 40th-degree characteristic polynomial, the LLM produced a complete Mathematica script with no iterative correction required. The key enabler was the algorithmic clarity of Husty's elimination procedure [20]: each step (determinant computation, resultant cascade, GCD extraction) is a well-defined symbolic operation with a straightforward Mathematica mapping, allowing the LLM to translate the algorithm into code with high fidelity. This suggests that LLMs are well-suited to symbolic pipelines whose steps are individually simple and independently verifiable, even when the overall algebraic degree is high.

Moving up the difficulty scale, the inverse kinematics of the general 6R robot proved to be a highly demanding benchmark. Even though the human expert provided the explicit solution papers (i.e., Raghavan and Roth [6]) directly to the AI, it still required multi-hour to multi-day total sessions of expert–LLM dialogue to successfully generate validated code. Three compounding sources of difficulty were identified. First, deriving the 14 bilinear Raghavan–Roth constraint equations demands careful bookkeeping of the kinematic split and the chain of matrix manipulations, offering many opportunities for subtle sign and indexing errors. Second, the dialytic lifting step and the subsequent elimination of (θ_1, θ_2) via left null-space projection require precise management of the polynomial degree structure across the trigonometric monomial bases. Third, and most critically, the original Raghavan–Roth paper [6] does not specify a numerically robust method for solving the final characteristic polynomial; the companion-matrix generalized eigenvalue formulation that ensures numerical stability had to be introduced by the human expert via Manocha and Canny's method [22] and then guided through the LLM iteratively. Only once this final step was correctly formulated did the solver reliably recover all 16 solutions.

The inverse kinematics of 6R robots with special architectures like spherical wrists or parallel axes without a solution document [13] proved to be the most demanding reported benchmark, requiring multiple prompting iterations. It is important to emphasize that for this problem, no explicit solution is provided to the LLM. While there are solutions available for specific robot architectures (e.g., PUMA560 or UR5), there is no ready-to-use analytical solution for general cases where the DH parameters can be arbitrary while still satisfying the specific architecture constraint. Because the LLM could not simply transcribe a known

TABLE 4: All 16 recovered IK solutions for the Manseur–Doty example (angles in degrees)

Sol	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Solver res.	Pos. res.	Ori. res. (rad)
1	−148.74	−64.31	−167.76	−54.97	−88.84	44.07	4.24E−15	2.69E−15	2.30E−15
2	63.90	−111.16	−126.95	47.57	−68.51	−55.40	1.08E−14	4.33E−15	7.06E−15
3	−116.98	−75.52	152.12	−44.86	−100.75	105.10	1.28E−14	8.22E−15	6.96E−15
4	0.00	107.46	112.46	−7.66	0.00	0.00	2.22E−14	1.64E−14	1.06E−14
5	−24.19	90.31	123.04	8.55	4.18	−9.97	2.85E−14	1.52E−14	1.71E−14
6	−116.98	−75.52	−27.88	−135.14	−79.25	−74.90	5.99E−14	2.15E−14	3.95E−14
7	63.90	−111.16	53.05	132.43	−111.49	124.60	8.01E−14	4.14E−14	4.85E−14
8	−11.16	−118.68	−38.88	138.63	−77.00	−9.11	9.01E−14	7.88E−14	3.09E−14
9	109.73	121.72	51.12	−102.54	−3.54	1.25	1.07E−13	5.19E−14	6.58E−14
10	−11.16	−118.68	141.12	41.37	−103.00	170.89	1.42E−13	1.41E−13	1.36E−14
11	177.33	74.17	54.09	−150.77	−8.74	35.11	3.60E−13	2.89E−13	1.51E−13
12	−148.74	−64.31	12.24	−125.03	−91.16	−135.93	1.48E−12	1.12E−12	6.85E−13
13	−24.19	90.31	−56.96	171.45	175.82	170.03	2.50E−12	2.57E−13	1.76E−12
14	0.00	107.46	−67.54	−172.34	180.00	−180.00	4.10E−12	1.36E−12	2.73E−12
15	177.33	74.17	−125.91	−29.23	−171.26	−144.89	1.67E−11	1.22E−11	8.01E−12
16	109.73	121.72	−128.88	−77.46	−176.46	−178.75	1.00E−9	7.84E−10	4.41E−10

solution, it necessitated strategic domain guidance regarding spatial decoupling and variable elimination order to correctly isolate the mechanism's simplifications. As a result, finding the solution proved to be a highly iterative process, requiring multi-hour to multi-day total debugging and prompting sessions. However, the final refined prompt based on the prompting process and the generated code can solve the problem in minutes.

7.2. Current AI Intelligence Level

Referring back to the hierarchy proposed in Fig. 1, the current state-of-the-art LLMs evaluated across these benchmark problems operate effectively at Level 3 and approach Level 4 in human-guided settings. As detailed in the companion paper [19], the LLM demonstrated its capability to solve the forward kinematics of the Stewart–Gough platform autonomously without the need for human guidance, a solid Level 3 achievement. In this paper, achieving a complete solver for the general 6R IK problem demanded extensive, structured expert intervention, including guidance on dialytic lifting and numerically stable generalized eigenvalue formulations. As demonstrated by Su [13], solving the inverse kinematics of 6R robots with special architectures, where no explicit analytical solution exists in literature, proved even more demanding and required detailed prompts embedded with domain-specific algebraic knowledge to formulate a novel elimination strategy. These latter interactions indicate Level 4 potential under expert supervision, but they do not demonstrate fully autonomous Level 4 performance. If an LLM could accomplish these tasks without human intervention, it would more clearly reach Level 4 autonomy.

Progressing towards Level 5 AGI, defined here as surpassing the best human experts on foundational open or historically difficult kinematics questions, will require LLMs to master and internalize undocumented heuristic analytical skills that human experts currently rely on through trial and error, thereby reducing the need for iterative human-led refinement. Specifically, if an AI could solve the inverse kinematics of the general 6R problem from scratch, without being provided the solution manual in a reference paper or having explicitly trained on those documents, it would provide strong evidence for AGI Level 5 in this domain. Any forecast about when such capability may appear remains speculative.

7.3. Limitations

This study has several limitations that should be acknowledged. The benchmark is restricted to classical problems with well-known analytical solutions whose correctness can be checked against published results; generalization of these findings to open or unsolved kinematics problems has not been validated. The evaluation reflects a single-expert workflow and therefore does not quantify variability due to prompt phrasing, choice of expert, or non-determinism across independent LLM runs. The prompting strategies described are calibrated to the specific model versions tested and may require adjustment as newer models are released or as different CAS environments are used.

The most important observed failure modes were not superficial coding mistakes, but mathematically plausible errors in the symbolic pipeline. In early 6R IK attempts, the model sometimes misread dense equations from the reference paper, introduced sign or indexing errors in the 14 scalar constraints, selected monomial bases inconsistent with the intended dialytic lifting, or attempted to solve the final characteristic polynomial using numerically unstable direct root-finding. These failures often produced code that looked structurally reasonable but failed benchmark residual tests or recovered an incomplete solution set. Human expert intervention was therefore required not only to debug syntax, but also to identify the correct algebraic split, enforce the elimination structure, and introduce the companion-matrix generalized eigenvalue method. The FK–IK–FK validation checks are effective for rejecting incorrect final solutions, but they do not by themselves teach the model which symbolic route is correct; that methodological guidance remains a central limitation of the current workflow.

A natural concern is whether these results are reproducible, given the non-deterministic nature of LLMs and the significant expert effort invested in the iterative development process. We argue that reproducibility is demonstrated at two complementary levels. First, the final refined prompts—which encode all domain-specific algebraic guidance, validation criteria, and reference materials—are made publicly available in the companion GitHub repository [24]. As demonstrated in Section 6, any practitioner with access to these prompts and the three reference PDFs can regenerate the full IK code pipeline (Mathematica derivation, Python solver, and C++ implementation) in a **single-shot** session using the same model tier, without repeating the hours of iterative development. Second, the correctness of the generated

code is independently verifiable: the solver's output against the Manseur–Doty 16-real-root benchmark and the FK–IK–FK residuals serve as objective, quantitative acceptance criteria that any implementation must satisfy. Taken together, the public prompts and the closed-form validation tests provide a reproducible workflow, even though the exploratory prompt-engineering phase that *produced* those prompts was necessarily non-deterministic and expert-dependent.

LLM hallucination—the generation of plausible-looking but mathematically incorrect code—is a known risk in scientific code generation. This concern is partially mitigated in the kinematics domain by the availability of a cheap, exact verifier: any candidate IK solution $\hat{\mathbf{q}}$ can be checked by computing the forward kinematics $\mathrm{FK}(\hat{\mathbf{q}})$ and comparing it against the target pose T. A position residual above a tight threshold (e.g., 10^{-4} m) immediately flags a spurious solution, regardless of how convincing the surrounding code appears. This FK–IK–FK cycle provides a rigorous numerical pass/fail test for candidate roots. However, it should not be interpreted as a complete safeguard: a generated solver may still omit valid branches, rely on unstable numerical choices, or require expert diagnosis before the verifier is ever satisfied.

7.4. Future Work

These results establish a foundation for two natural extensions. First, expanding the benchmark to cover a broader diversity of kinematics problems—including parallel mechanisms (analysis, synthesis, workspace, and mobility evaluation), compliant mechanisms, origami structures, kirigami structures, reconfigurable mechanisms, overconstrained mechanisms, cable-driven systems, and synthesis under uncertainty—will test whether the patterns observed here generalize beyond the reported benchmark categories. Second, and more ambitiously, applying the LLM-assisted framework to open research problems in kinematics, where no ground-truth analytical solution exists, will probe the limits of the human–AI collaborative paradigm. The development of reusable, domain-specific skill documents (analogous to the `SKILL.md` introduced for the 6R IK solver) that encode expert algebraic guidance for broad classes of elimination problems represents a promising direction for reducing the prompting burden in future sessions.

8. CONCLUSIONS

This paper presented a focused evaluation of large language models for solving classical kinematics benchmark problems, together with a hierarchy for discussing progress toward Artificial General Intelligence (AGI) in kinematics. The primary technical example was the inverse kinematics of the general 6R serial robot, a highly complex spatial mechanism that exemplifies the "Mount Everest" algebraic challenges of the 1980s and 1990s.

Key findings indicate that LLMs can generate working kinematics solvers when guided by structured prompts encoding expert domain knowledge, solution strategies, and validation criteria. However, problem complexity dictates the required level of human involvement. For the 6R IK problem, the algebraic and numerical nuances necessitated multi-step prompting, CAS integration, and iterative expert correction—highlighting the current gap toward AGI. Specifically, the transition from symbolic representation to numerically robust solvers relies heavily on domain-specific best practices (e.g., formulating generalized eigenvalue problems) that LLMs frequently miss. Ultimately, expert guidance remains essential for specifying elimination strategies, identifying solvable subsystems, and selecting appropriate numerical methods in these complex generative tasks.

These results demonstrate that LLMs represent a powerful new tool for accelerating the implementation of analytical kinematics algorithms, provided that appropriate human expertise guides the process. Looking further ahead, the integration of AI tools into kinematics research holds the potential to fundamentally transform how the field advances. Tasks that once required weeks of manual derivation—constructing resultant matrices, managing polynomial degree structures, implementing numerically stable eigensolvers—can increasingly be delegated to an LLM coding assistant, compressing research and development cycles from months to days.

As models continue to improve in mathematical reasoning, we anticipate that the boundary of autonomous LLM capability will shift progressively toward higher-complexity problems, enabling researchers to devote more effort to problem formulation and insight rather than implementation. Based on the intelligence hierarchy proposed in this work, current state-of-the-art LLMs operate effectively at Level 3 and show Level 4 potential under expert supervision: they can reproduce extremely complex, documented analytical pipelines, but still depend on human heuristic guidance to navigate novel algebraic elimination strategies. Progress toward Level 5 AGI should therefore be judged by future systems' ability to choose correct symbolic strategies, select stable numerical methods, and pass independent validation without relying on a human expert to supply the critical reasoning steps.

ACKNOWLEDGMENT

The authors would like to acknowledge Yan Chen, Jian S. Dai, Clement Gosselin, Guangbo Hao, Larry Howell, and Dan Zhang for their valuable comments, inputs, and discussions on the open kinematics problems.

REFERENCES

[1] Burmester, Ludwig. *Lehrbuch der Kinematik*. A. Felix, Leipzig (1888).

[2] McCarthy, J Michael and Soh, Gim Song. *Geometric design of linkages*. Springer Science & Business Media (2010).

[3] Freudenstein, Ferdinand. "Kinematics: past, present and future." *Mechanism and Machine Theory* Vol. 8 No. 2 (1973): pp. 151–160. DOI 10.1016/0094-114X(73)90049-9. URL https://doi.org/10.1016/0094-114X(73)90049-9.

[4] Liao, Q. Z., Liang, C. G. and Zhang, Q. X. "A novel approach to the displacement analysis of general spatial 7R mechanism." *Chinese Journal of Mechanical Engineering* Vol. 22 No. 3 (1986): pp. 1–9.

[5] Lee, Hong-You and Liang, Chong-Gao. "Displacement analysis of the general spatial 7-link 7R mechanism." *Mechanism and Machine Theory* Vol. 23 No. 3 (1988): pp. 219–226.

[6] Raghavan, K. and Roth, B. “Kinematic analysis of general six-revolute manipulators using dialytic elimination.” *The International Journal of Robotics Research* Vol. 12 No. 5 (1993): pp. 469–485.

[7] OpenAI. “GPT-4 Technical Report.” Technical report no. OpenAI. 2023. URL https://cdn.openai.com/papers/gpt-4.pdf.

[8] OpenAI. “OpenAI GPT-5 System Card.” *arXiv preprint arXiv:2601.03267* (2025)URL https://arxiv.org/abs/2601.03267.

[9] Romera-Paredes, Bernardino, Barekatain, Mohammadamin, Novikov, Alexander, Balog, Matej, Kumar, M. Pawan, Dupont, Emilien, Ruiz, Francisco J. R., Ellenberg, Jordan S., Wang, Pengming, Fawzi, Omar, Kohli, Pushmeet and Fawzi, Alhussein. “Mathematical discoveries from program search with large language models.” *Nature* Vol. 625 (2024): pp. 468–475. DOI 10.1038/s41586-023-06924-6.

[10] Trinh, Trieu H., Wu, Yuhuai, Le, Quoc V., He, He and Luong, Thang. “Solving olympiad geometry without human demonstrations.” *Nature* Vol. 625 (2024): pp. 476–482. DOI 10.1038/s41586-023-06747-5.

[11] Guo, Xingang, Li, Yaxin, Kong, Xiangyi, Jiang, Yilan, Zhao, Xiayu, Gong, Zhihua, Zhang, Yufan, Li, Daixuan, Sang, Tianle, Zhu, Beixiao et al. “Toward Engineering AGI: Benchmarking the Engineering Design Capabilities of LLMs.” (2025). URL 2509.16204, URL https://arxiv.org/abs/2509.16204.

[12] Kim, Yoonsoo, Kim, Dongkeun, Choi, Jongrae, Park, Jaeheung, Oh, Namhoon and Park, Dongwoon. “A Survey on Integration of Large Language Models with Intelligent Robots.” *Intelligent Service Robotics* Vol. 17 (2024): pp. 1103–1125. DOI 10.1007/s11370-024-00550-5.

[13] Su, Hai-Jun. “Large language model assisted human-AI collaborative development of analytical inverse kinematics solvers for robot manipulators.” *Mechanism and Machine Theory* Vol. 222 (2026): p. 106392. DOI https://doi.org/10.1016/j.mechmachtheory.2026.106392. URL https://www.sciencedirect.com/science/article/pii/S0094114X26000431.

[14] Hassabis, Demis. “Demis Hassabis on the ‘Einstein test’ for true AGI.” (2025). Various interviews and publications on defining AGI.

[15] Bates, Daniel J., Hauenstein, Jonathan D., Sommese, Andrew J. and Wampler, Charles W. *Numerically Solving Polynomial Systems with Bertini*. SIAM, Philadelphia, PA (2013).

[16] Su, Hai-Jun, McCarthy, J. Michael, Sosonkina, M. and Watson, Layne T. “Algorithm 857: POLSYS_GLP—a parallel general linear product homotopy code for solving polynomial systems of equations.” *ACM Transactions on Mathematical Software (TOMS)* Vol. 32 No. 4 (2006): pp. 561–579.

[17] Wampler, Charles W., Morgan, Alexander P. and Sommese, Andrew J. “Complete Solution of the Nine-Point Path Synthesis Problem for Four-Bar Linkages.” *Journal of Mechanical Design* Vol. 114 No. 1 (1992): pp. 153–159.

[18] Su, Hai-Jun, McCarthy, J. Michael and Watson, Layne T. “Generalized linear product homotopy algorithms and the computation of reachable surfaces.” *Journal of Computing and Information Science in Engineering* Vol. 4 No. 3 (2004): pp. 226–234.

[19] Su, Hai-Jun and McCarthy, J. Michael. “Solving Kinematics Problems by Leveraging the Power of Large Language Models.” *Proceedings of the ASME 2026 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*. 2026. Houston, TX.

[20] Husty, Manfred L. “An algorithm for solving the direct kinematics of general Stewart-Gough platforms.” *Mechanism and Machine Theory* Vol. 31 No. 4 (1996): pp. 365–380.

[21] Liao, Q. and McCarthy, J. M. “Seven-position synthesis of a spatial SS chain.” *Journal of Mechanical Design* Vol. 123 No. 1 (2001): pp. 74–79.

[22] Manocha, D. and Canny, J. F. “Efficient inverse kinematics for general 6R manipulators.” *IEEE Transactions on Robotics and Automation* Vol. 10 No. 5 (1994): pp. 648–657. DOI 10.1109/70.326569.

[23] Manseur, Rachid and Doty, Donald A. “A Robot Manipulator With 16 Real Inverse Kinematic Solution Sets.” *The International Journal of Robotics Research* Vol. 8 No. 5 (1989): pp. 75–83. DOI 10.1177/027836498900800507.

[24] Su, Haijun et al. “IK_6R_RR_1993: A Benchmark for LLM-Assisted Generative Code for General 6R Inverse Kinematics.” https://github.com/haijunsu-osu/IK_6R_RR_1993 (2026).

APPENDIX A. LLM-GENERATED MATHEMATICA SCRIPT FOR 6R IK SOLVER

The following condensed Mathematica script was generated by the LLM to implement the Raghavan–Roth resultant elimination pipeline for the general 6R IK problem. It constructs the 14 bilinear equations, eliminates (θ_1, θ_2), applies dialytic lifting, and solves the quadratic matrix polynomial via a 24×24 generalized eigenvalue pencil.

```
(* Raghavan-Roth 1993 IK solver for general 6R chain *)
ClearAll["Global`*"];

(* DH transform: A_i = Rz(theta) Tz(d) Tx(a) Rx(alpha) *)
Aiv[th_]:={{Cos[th],-Sin[th],0,0},{Sin[th],Cos[th],0,0},
  {0,0,1,0},{0,0,0,1}};
Ais[a_,al_,d_]:={{1,0,0,a},{0,Cos[al],-Sin[al],0},
  {0,Sin[al],Cos[al],d},{0,0,0,1}};
DHMatrix[a_,al_,d_,th_]:=Aiv[th].Ais[a,al,d];

(* Forward kinematics *)
FK[dh_,q_]:=Fold[Dot,IdentityMatrix[4],
  Table[DHMatrix[dh[[i,1]],dh[[i,2]],dh[[i,3]],q[[i]]],{i,1,6}]];

(* Build p,l vectors from chain split at joint 3 *)
BuildPL[dh_,T_,q_]:=Module[{a,left,right},
  a=Table[DHMatrix[dh[[i,1]],dh[[i,2]],dh[[i,3]],q[[i]]],{i,1,6}];
  left=Ais[dh[[2,1]],dh[[2,2]],dh[[2,3]]].a[[3]].a[[4]].a[[5]];
  right=Inverse[Aiv[q[[2]]]].Inverse[a[[1]]].T.Inverse[a[[6]]];
  {left[[1;;3,4]],right[[1;;3,4]],left[[1;;3,3]],right[[1;;3,3]]}];

(* Build 14 equations: 3(p)+3(l)+1(p.p)+1(p.l)+3(pxl)+2(combo)+1(l.l) *)
Build14[dh_,T_,q_]:=Module[{pL,pR,lL,lR},
  {pL,pR,lL,lR}=BuildPL[dh,T,q];
  Join[pL-pR, lL-lR, {pL.pL-pR.pR, pL.lL-pR.lR},
```

```
    Cross[pL,lL]-Cross[pR,lR],
    (pL.pL)*lL-2*(pL.lL)*pL-((pR.pR)*lR-2*(pR.lR)*pR)]];

(* Monomial vectors *)
Mon45[t4_,t5_]:={Sin[t4]Sin[t5],Sin[t4]Cos[t5],Cos[t4]Sin[t5],
  Cos[t4]Cos[t5],Sin[t4],Cos[t4],Sin[t5],Cos[t5],1.};
Mon12[t1_,t2_]:={Sin[t1]Sin[t2],Sin[t1]Cos[t2],Cos[t1]Sin[t2],
  Cos[t1]Cos[t2],Sin[t1],Cos[t1],Sin[t2],Cos[t2]};

(* Extract P,Q matrices by sampling random (t1,t2,t4,t5) *)
BuildPQ[dh_,T_,x3_,samp_]:=Module[{fMat,vals,pR,qR,sol},
  fMat=Table[Join[Mon45[s[[3]],s[[4]]],-Mon12[s[[1]],s[[2]]]],
    {s,samp}];
  vals=Table[Build14[dh,T,{s[[1]],s[[2]],2ArcTan[x3],s[[3]],s[[4]],0.}],
    {s,samp}];
  pR=qR=Table[0.,{14},{9}];
  Do[sol=LeastSquares[fMat,vals[[All,i]]];
    pR[[i]]=sol[[1;;9]]; qR[[i]]=sol[[10;;17]],{i,1,14}];
  {pR,qR}];

(* Eliminate theta1,theta2 via left null space of Q *)
Elim12[p_,q_]:=NullSpace[Transpose[q],Tolerance->10^-12].p;

(* Half-angle substitution map: trig monomials -> polynomial basis *)
HalfAngleMap[]:=Module[{t=ConstantArray[0.,{9,9}]},
  t[[1,4]]=4.; t[[2,3]]=-2.; t[[2,7]]=2.;
  t[[3,2]]=-2.; t[[3,8]]=2.;
  t[[4,1]]=1.; t[[4,5]]=-1.; t[[4,6]]=-1.; t[[4,9]]=1.;
  t[[5,3]]=2.; t[[5,7]]=2.;
  t[[6,1]]=-1.; t[[6,5]]=-1.; t[[6,6]]=1.; t[[6,9]]=1.;
  t[[7,2]]=2.; t[[7,8]]=2.;
  t[[8,1]]=-1.; t[[8,5]]=1.; t[[8,6]]=-1.; t[[8,9]]=1.;
  t[[9,1]]=1.; t[[9,5]]=1.; t[[9,6]]=1.; t[[9,9]]=1.; t];

(* Build 12x12 dialytic matrix from 6x9 E9 *)
Dialytic12[e9_]:=Module[{m=ConstantArray[0.,{12,12}]},
  Do[c=e9[[i]];
    m[[i,{4,5,7,8,6,10,9,11,12}]]=c;
    m[[i+6,{1,2,4,5,3,7,6,8,9}]]=c,{i,1,6}]; m];

(* Decompose P(theta3) = PS*sin(theta3) + PC*cos(theta3) + P1 *)
(* Q is constant with respect to theta3 *)
FitPSinCos[dh_,T_,samp_]:=Module[
  {g={-2.1,-0.2,1.3}, basis,inv,ps={},qs={},pS,pC,p1,qC},
  basis=Transpose[{Sin[g],Cos[g],ConstantArray[1.,3]}];
  inv=Inverse[basis];
  Do[AppendTo[ps,BuildPQ[dh,T,Tan[t/2],samp][[1]]];
    AppendTo[qs,BuildPQ[dh,T,Tan[t/2],samp][[2]]],{t,g}];
  pS=inv[[1,1]]*ps[[1]]+inv[[1,2]]*ps[[2]]+inv[[1,3]]*ps[[3]];
  pC=inv[[2,1]]*ps[[1]]+inv[[2,2]]*ps[[2]]+inv[[2,3]]*ps[[3]];
  p1=inv[[3,1]]*ps[[1]]+inv[[3,2]]*ps[[2]]+inv[[3,3]]*ps[[3]];
  qC=Mean[qs]; {pS,pC,p1,qC}];

(* Build quadratic matrix polynomial M(x3) = M0 + M1*x3 + M2*x3^2 *)
(* where x3 = tan(theta3/2), using sin=(2x)/(1+x^2), cos=(1-x^2)/(1+x^2) *)
(* so (1+x3^2)*P = (P1+PC) + 2*PS*x3 + (P1-PC)*x3^2 = P0 + P1L*x3 + P2*x3^2 *)
BuildM012[dh_,T_,samp_]:=Module[{pS,pC,p1,qC,nMat,map,p0,p1L,p2},
  {pS,pC,p1,qC}=FitPSinCos[dh,T,samp];
  nMat=Module[{piv,qP,qR,x,n,rem,order,nPerm},
    piv=Module[{idx={},rem=Range[14],best,score,cand,blk,rk,sv},
      While[Length[idx]<8&&Length[rem]>0,
        best=Missing[];score=Infinity;
        Do[cand=Append[idx,r]; blk=qC[[cand]];
          rk=MatrixRank[blk,Tolerance->10^-10];
          If[rk<Length[cand],Continue[]];
          sv=SingularValueList[N[blk,40]];
          If[Length[sv]==0||Abs[Last[sv]]<=10^-14,Continue[]];
          If[First[sv]/Last[sv]<score,score=First[sv]/Last[sv];best=r],
          {r,rem}];
        If[best===Missing[],Break[]];
        idx=Append[idx,best]; rem=DeleteCases[rem,best]]; idx];
    rem=Complement[Range[14],piv]; qP=qC[[piv]]; qR=qC[[rem]];
    x=Transpose[LinearSolve[Transpose[qP],Transpose[qR]]];
    nPerm=ArrayFlatten[{{-x,IdentityMatrix[Length[rem]]}}];
    order=Join[piv,rem]; n=ConstantArray[0.,{Length[rem],14}];
    Do[n[[All,order[[j]]]]=nPerm[[All,j]],{j,1,Length[order]}]; n];
  map=HalfAngleMap[];
  p0=p1+pC; p1L=2.*pS; p2=p1-pC;
  {Dialytic12[nMat.p0.map], Dialytic12[nMat.p1L.map],
   Dialytic12[nMat.p2.map]}];

(* Solve via 24x24 generalized eigenvalue pencil *)
SolveIK[dh_,T_]:=Module[{samp,m0,m1,m2,n,matA,matB,
  eigs,x3roots,sols={},th3,pq,e45,map,e9,dMat,ns,x45,
  th4,th5,m45,rhs,n12,sc1,sc2,th1,th2,th6,q,res,fk},
  SeedRandom[73421]; samp=Table[RandomReal[{-Pi,Pi},4],{60}];
  {m0,m1,m2}=BuildM012[dh,T,samp]; n=Length[m0];
  (* Build 24x24 companion pencil for M0+M1*x3+M2*x3^2 *)
  matA=ConstantArray[0.,{2n,2n}]; matB=ConstantArray[0.,{2n,2n}];
  matA[[1;;n,1;;n]]=-m1; matA[[1;;n,n+1;;2n]]=-m0;
  matB[[1;;n,1;;n]]=m2;
  matA[[n+1;;2n,1;;n]]=IdentityMatrix[n];
  matB[[n+1;;2n,n+1;;2n]]=IdentityMatrix[n];
  eigs=Quiet[N[Eigenvalues[{matA,matB}],40]];
  x3roots=Sort[Re[Select[eigs,
    NumericQ[#]&&Abs[Im[#]]<=10^-7&&Abs[Re[#]]<=30&]]];
  (* Back-substitute each x3 root *)
  map=HalfAngleMap[];
  Do[th3=2ArcTan[x3];
    pq=BuildPQ[dh,T,x3,samp];
    e45=Elim12[pq[[1]],pq[[2]]];
    e9=e45.map; dMat=Dialytic12[e9];
    ns=NullSpace[N[dMat,40],Tolerance->10^-16];
    Do[z=Re[v]; x4=z[[9]]/z[[12]]; x5=z[[11]]/z[[12]];
      th4=2ArcTan[x4]; th5=2ArcTan[x5];
      m45=Mon45[th4,th5]; rhs=pq[[1]].m45;
      n12=LeastSquares[pq[[2]],rhs];
      sc1={n12[[5]],n12[[6]]}/Norm[{n12[[5]],n12[[6]]}];
      sc2={n12[[7]],n12[[8]]}/Norm[{n12[[7]],n12[[8]]}];
      th1=ArcTan[sc1[[2]],sc1[[1]]];
      th2=ArcTan[sc2[[2]],sc2[[1]]];
      fk=Fold[Dot,IdentityMatrix[4],Table[
        DHMatrix[dh[[i,1]],dh[[i,2]],dh[[i,3]],
          {th1,th2,th3,th4,th5,0.}[[i]]],{i,1,5}]];
      th6=ArcTan[Inverse[fk].T//N//First//First,
        (Inverse[fk].T)[[2,1]]];
      q={th1,th2,th3,th4,th5,th6};
      res=Norm[Flatten[FK[dh,q]-T]];
      If[res<10^-4, AppendTo[sols,{q,res}]],
    {v,ns}],
  {x3,x3roots}];
  sols];

(* ==================== Example: Manseur-Doty 16-root case ==================== *)
dh16 = {
  {0.3, Pi/2, 0.0},
  {1.0, 1.0, 0.0},
  {0.0, Pi/2, 0.2},
  {1.5, 1.0, 0.0},
  {0.0, Pi/2, 0.0},
  {0.0, 1.0, 0.0}
};
q16 = {0.0, 107.458 Degree, 112.460 Degree, -7.662 Degree, 0.0, 0.0};
target16 = FK[dh16, q16];
Print["Target pose:"];
Print[MatrixForm[target16]];

sols16 = SolveIK[dh16, target16];
Print[Length[sols16], " IK solutions recovered (expected 16)."];
Do[
  Print["  #", i,
    " q=", N[sols16[[i,1]], 8],
    " residual=", N[sols16[[i,2]], 12]],
  {i, 1, Length[sols16]}];

(* ==================== Example: Random general 6R ==================== *)
SeedRandom[20260225];
dhRand = Transpose[{
  RandomReal[{0.08, 0.45}, 6],
  RandomChoice[{-1., 1.}, 6]*RandomReal[{0.35, 1.35}, 6],
  RandomReal[{0.0, 0.30}, 6]}];
qTrue = RandomReal[{-Pi, Pi}, 6];
targetRand = FK[dhRand, qTrue];

solsRand = SolveIK[dhRand, targetRand];
Print[Length[solsRand], " IK solutions for random example."];
dists = Table[Norm[qTrue - s[[1]]], {s, solsRand}];
bestIdx = First@Ordering[dists, 1];
Print["Best match to ground truth: distance=", N[dists[[bestIdx]], 8],
  " residual=", N[solsRand[[bestIdx, 2]], 12]];
```