

Exploring LLM-based approaches for wrist-center positioning of an anthropomorphic manipulator: a preliminary comparative study

Mostafa Bayati

Senior Automation and Controls Engineer

Corresponding author: mail.blociot@gmail.com

Abstract

This paper explores the feasibility of using Large Language Models (LLMs) to solve the wrist-center inverse positioning problem for a 3-degree-of-freedom (DOF) anthropomorphic manipulator sub-problem. The study investigates whether LLMs can perform meaningful spatial and trigonometric reasoning on kinematic equations. Three LLM-based approaches are proposed and evaluated on 50 target positions across the manipulator workspace: (1) zero-shot prompting (LLM-ZS), (2) few-shot in-context learning (LLM-FS) and (3) LLM-Guided Iterative Refinement (LGIR). All methods are benchmarked against four reimplemented traditional solvers—analytical closed-form, Damped Least Squares (DLS), Levenberg–Marquardt and Nelder–Mead—executed on the same hardware under identical conditions. The LGIR method achieves a median position error of $7.44\text{e-}05$ cm across 50 targets, while traditional numerical solvers achieve near-zero errors ($1.12\text{e-}15$ cm for the analytical method). LLM-based methods incur computation times of several seconds per solution owing to API latency, rendering them unsuitable for real-time control. The results demonstrate that LLMs possess measurable trigonometric reasoning capability but cannot match the precision or speed of established solvers. This study provides the first multi-target comparison of multiple LLM prompting strategies against reimplemented traditional solvers for a robot positioning sub-problem, with all methods evaluated under identical experimental conditions. The findings contribute an empirical baseline for understanding the current limitations and potential of LLMs in kinematic computation tasks.

Keywords Inverse kinematics, Large language models, Wrist-center positioning, Anthropomorphic manipulator, Iterative refinement, Comparative study

1. Introduction

Inverse kinematics (IK) is a fundamental problem in robotics that involves determining the joint variables required to place a manipulator's end-effector at a desired position and orientation in Cartesian space (Craig, 2017). For serial manipulators, the IK problem is generally more complex than the forward kinematics (FK) problem owing to the nonlinear nature of the mapping, the possible existence of multiple solutions and singularity issues (Siciliano et al., 2009). For manipulators with specific geometric properties—such as the anthropomorphic arm with a spherical wrist studied in this paper—closed-form analytical solutions exist and are the preferred approach in practice (Pieper, 1968; Craig, 2017).

Beyond analytical methods, a variety of numerical and computational intelligence approaches have been developed for cases where closed-form solutions are unavailable or impractical. These include Jacobian-based iterative methods such as Damped Least Squares (DLS) (Wampler, 1986; Nakamura and Hanafusa, 1986), optimization-based methods such as Levenberg–Marquardt (Sugihara, 2011) and derivative-free methods such as Nelder–Mead (Nelder and Mead, 1965). Computational intelligence methods—including neural networks (Duka, 2014) and meta-heuristic

optimizers such as Genetic Algorithms (Tabandeh et al., 2006), Particle Swarm Optimization (Rokbani and Alimi, 2013) and Imperialist Competitive Algorithm (Bayati, 2015)—have also been applied. For a comprehensive survey of IK methods, the reader is referred to Aristidou et al. (2018).

The recent emergence of Large Language Models (LLMs) such as GPT-4, Claude and Gemini has demonstrated remarkable capabilities in natural language understanding, mathematical reasoning and code generation (OpenAI, 2023; Anthropic, 2025). These models, trained on vast corpora that include scientific literature and mathematical textbooks, have shown abilities to solve complex mathematical problems (Frieder et al., 2023). A small but growing body of work has begun exploring LLMs for kinematics-related tasks (Rice et al., 2026; Yang and Liu, 2025; Su, 2026; Khabazi et al., 2026), yet a controlled comparison of multiple LLM prompting strategies against traditional solvers—under identical experimental conditions and across multiple target positions—remains absent from the literature.

This paper explores the feasibility of using LLMs to solve a specific sub-problem of inverse kinematics: computing the wrist-center position for a 3-DOF anthropomorphic arm sub-chain derived from the 6-DOF manipulator studied in Bayati (2015). It is important to note that this represents only the positioning component; the orientation problem (solved by the spherical wrist joints q_4 , q_5 , q_6) is not addressed. The study does not claim that LLMs are competitive with traditional solvers for practical applications but rather investigates the extent to which LLMs can perform spatial and trigonometric reasoning on kinematic equations. The contributions are as follows:

1. A controlled evaluation of three LLM prompting strategies (zero-shot, few-shot and iterative refinement) for the wrist-center positioning sub-problem, tested across 50 target positions spanning the manipulator workspace.
2. An LGIR methodology that creates a closed-loop feedback system between the LLM and a forward kinematics evaluation module, enabling iterative solution refinement through multi-turn conversation.
3. A fair comparison against four reimplemented traditional solvers (analytical, DLS, Levenberg–Marquardt and Nelder–Mead), all executed on the same hardware with the same forward kinematics module, ensuring like-for-like evaluation.
4. An honest assessment of both accuracy and computation time, demonstrating the current limitations of LLM-based approaches relative to established methods.

2. Related work

2.1 Traditional inverse kinematics methods

Inverse kinematics methods can be broadly classified into analytical (closed-form) and numerical approaches. Analytical solutions, where available, provide exact joint angles in closed form. Pieper (1968) showed that a sufficient condition for a closed-form solution is that three consecutive joint axes intersect at a point—a condition satisfied by manipulators with spherical wrists, including the one studied here. Craig (2017) provides the standard derivation for such manipulators.

Numerical iterative methods are required when closed-form solutions are unavailable, as is the case for many modern manipulator designs. The Jacobian transpose and pseudoinverse methods (Siciliano et al., 2009) are foundational approaches, while the Damped Least Squares (DLS) method, independently proposed by Wampler (1986) and Nakamura and Hanafusa (1986), adds a damping factor to avoid singularity-related instabilities. Buss (2004) provides a comparative tutorial. The Levenberg–Marquardt algorithm (Sugihara, 2011) combines gradient descent and Gauss–Newton to

provide robust convergence. Derivative-free methods such as Nelder–Mead (Nelder and Mead, 1965) and BFGS have also been applied. Beeson and Ames (2015) introduced TRAC-IK, which combines randomized restarts with sequential quadratic programming. Diankov (2010) developed IKFast, an analytical solver generator that produces optimised closed-form IK code for specific robot geometries.

Computational intelligence approaches formulate IK as an optimization problem. Neural network methods (MLP, RBF, ANFIS) learn an approximate inverse mapping from training data (Duka, 2014). Meta-heuristic algorithms—including GA (Tabandeh et al., 2006), PSO (Rokbani and Alimi, 2013), Differential Evolution (Storn and Price, 1997) and ICA (Atashpaz-Gargari and Lucas, 2007)—have shown progressively better accuracy. Bayati (2015) compared nine such methods for a 6-DOF anthropomorphic arm, finding that ICA achieved the lowest position error.

2.2 Large language models in engineering and robotics

LLMs have been applied to various engineering domains beyond natural language processing. Frieder et al. (2023) evaluated the mathematical capabilities of ChatGPT. In robotics, Vemprala et al. (2023) explored using ChatGPT for high-level robotics reasoning, while Liang et al. (2023) demonstrated Code as Policies for language-conditioned robot behaviour. Ahn et al. (2022) grounded language in robotic affordances through the SayCan framework.

2.3 LLMs for inverse kinematics

A small number of recent studies have investigated LLMs for kinematics tasks. Rice et al. (2026) proposed LLM-IK, using Google Gemini to solve IK by extracting manipulator information from descriptor files and decomposing kinematic chains into sub-problems. Their approach handled up to 6-DOF problems and was compared against numerical IK solvers (IKFast, TRAC-IK and IKPy). However, their study employed a single LLM strategy and did not benchmark against a broader range of traditional solvers.

Yang and Liu (2025) proposed LLM-AWQP, a framework that employs LLMs as semantic guides to enhance numerical IK optimization. Su (2026) demonstrated an LLM-assisted process for developing analytical closed-form IK solvers through human–AI collaboration. Khabazi et al. (2026) presented RoboSolver, a multi-agent LLM framework using GPT-4o for IK tasks. While these studies demonstrate growing interest, none has compared multiple LLM prompting strategies against reimplemented traditional solvers under identical conditions across a multi-target evaluation.

3. Problem formulation

3.1 Robot configuration and scope

The manipulator under consideration is a 6-DOF anthropomorphic arm with a spherical wrist, as described in Bayati (2015). The kinematic structure consists of a base rotation joint (q_1), a shoulder elevation joint (q_2), an elbow joint (q_3) and a 3-DOF spherical wrist (q_4, q_5, q_6). The first three joints determine the wrist-center position, while the last three joints control the end-effector orientation.

This study addresses only the wrist-center positioning sub-problem: given a desired wrist-center position $p^* = (p^*_x, p^*_y, p^*_z)$, find the joint angles q_1, q_2, q_3 that place the wrist center at that position. The orientation problem (determining q_4, q_5, q_6 for a desired end-effector orientation) is not addressed. This scope limitation is important because the positioning sub-problem admits a closed-form analytical solution for this class of manipulator (Pieper, 1968; Craig, 2017), which serves as a ground-truth baseline.



Figure 1. Schematic of the 3-DOF wrist-center positioning model derived from the 6-DOF anthropomorphic arm with spherical wrist. Only joints q_1, q_2, q_3 affect the wrist-center position; joints q_4, q_5, q_6 control orientation and are not addressed in this study.

3.2 Forward kinematics

Using the Denavit–Hartenberg (DH) convention, the forward kinematics equations for the wrist-center position are (Bayati, 2015):

$$\begin{aligned} p_x &= \cos(q_1) \cdot [l_2 \cdot \cos(q_2) + l_3 \cdot \cos(q_2 + q_3)] \\ p_y &= \sin(q_1) \cdot [l_2 \cdot \cos(q_2) + l_3 \cdot \cos(q_2 + q_3)] \\ p_z &= l_2 \cdot \sin(q_2) + l_3 \cdot \sin(q_2 + q_3) \end{aligned} \quad (1)$$

where q_1, q_2, q_3 are the joint angles in degrees, and $l_2 = l_3 = 10$ cm are the upper-arm and forearm link lengths respectively. The wrist orientation joints q_4, q_5, q_6 do not affect the wrist-center position because the spherical wrist axes intersect at a single point (Craig, 2017).

Table I. Symbol definitions

Symbol	Description
q_1, q_2, q_3	Joint angles (degrees) for base rotation, shoulder elevation and elbow
l_2, l_3	Link lengths (10 cm each)
$p^* = (p^*_x, p^*_y, p^*_z)$	Target wrist-center position (cm)
$p = (p_x, p_y, p_z)$	Achieved wrist-center position via FK (cm)
e	Position error: $\ p^* - p\ _2$ (cm)
ε	Error convergence threshold (10^{-4} cm)
K	Maximum iterations for LGIR (15)

3.3 Inverse kinematics objective

The IK problem is formulated as finding the joint angles $q = [q_1, q_2, q_3]^T$ that minimize the Euclidean distance between the achieved and target wrist-center positions:

$$e = \|p^* - FK(q)\|_2 \quad (2)$$

For this class of manipulator, an analytical closed-form solution to equation (2) exists (see Section 4.1), achieving effectively zero error to machine precision. The LLM-based methods are therefore benchmarked against a known-optimal solution.

4. Methods

4.1 Traditional baseline solvers

Four traditional IK solvers are implemented as baselines, all using the same forward kinematics module (equation (1)) and the same evaluation metric (equation (2)). Each solver is executed on the same machine used for the LLM experiments.

The analytical solver computes the closed-form IK solution using the inverse trigonometric approach for anthropomorphic arms with spherical wrists (Craig, 2017). It returns the exact solution to machine precision. The DLS solver uses the 3×3 Jacobian matrix of the wrist-center position with respect to q_1, q_2, q_3 , with a damping factor of 0.5 and a maximum of 200 iterations (Buss, 2004). The Levenberg–Marquardt solver uses the `scipy.optimize.least_squares` implementation with `method='lm'`. The Nelder–Mead solver uses the `scipy.optimize.minimize` implementation with the Nelder–Mead simplex method (Nelder and Mead, 1965). For the three numerical solvers, five random initial guesses are used per target and the best result is retained.

4.2 LLM-based approaches

Three LLM-based approaches are evaluated, each representing a different level of interaction between the LLM and the kinematic system. All approaches use the Claude Sonnet 4 model (Anthropic, 2025; model identifier: `claude-sonnet-4-20250514`) accessed through the Anthropic Messages API with temperature = 1.0 (explicitly set). Figure 2 provides a system-level overview of the three approaches.

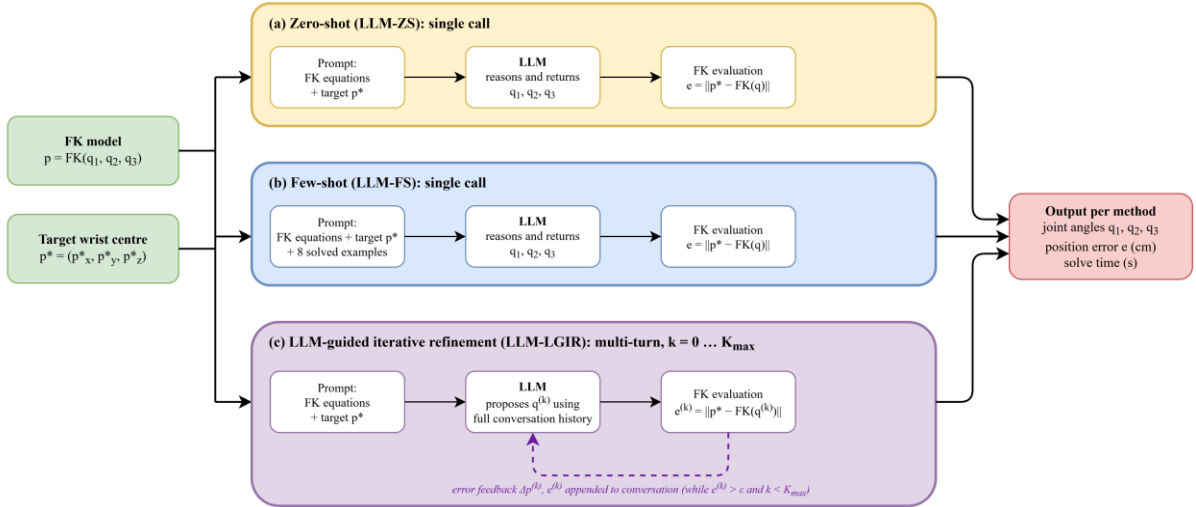


Figure 2. System overview of the three LLM-based approaches for wrist-center positioning. LLM-ZS and LLM-FS are single-call methods; LGIR uses multi-turn conversation with error feedback.

4.2.1 Zero-shot prompting (LLM-ZS)

In the zero-shot approach, the LLM receives a prompt containing the FK equations (equation (1)), link lengths, target position and an instruction to compute the joint angles q_1, q_2, q_3 that achieve the target position. The prompt requests step-by-step mathematical reasoning (chain-of-thought; Wei et al., 2022b) before providing the final answer. No examples or feedback are provided. Three independent trials are run per target, and the best result is retained.

Algorithm 1: LLM zero-shot IK solver (LLM-ZS)

Input: FK equations, link lengths (l_2, l_3), target position p^*
Output: Joint angles $q = [q_1, q_2, q_3]$, position error e

```
function ZeroShotIK( $p^*, N\_trials$ ):  
  for trial = 1 to  $N\_trials$  do  
    Build prompt  $P$  with FK equations,  $l_2, l_3, p^*$   
    Append chain-of-thought instruction to  $P$   
    Send  $P$  to LLM as single-turn query  
    Parse joint angles  $q$  from response  
    if parsing successful then  
      Compute  $e \leftarrow \|p^* - FK(q)\|_2$   
      Record ( $q, e, elapsed\_time$ )  
    end if  
  end for  
  return best ( $q, e$ ) across all trials
```

4.2.2 Few-shot in-context learning (LLM-FS)

The few-shot approach augments the zero-shot prompt with eight solved IK examples generated by sampling random joint configurations, computing their FK positions and solving analytically. This leverages the in-context learning capability of LLMs (Brown et al., 2020). The same eight examples are used across all trials and targets. Three independent trials are run per target.

4.2.3 LLM-Guided Iterative Refinement (LGIR)

The LGIR method creates a closed-loop feedback system between the LLM and the FK evaluation module. The process operates as follows:

1. The LLM receives the FK equations, link parameters and target position, and proposes an initial set of joint angles.
2. The proposed angles are evaluated through the FK module to compute the achieved position.
3. The position error and its directional components ($\Delta x, \Delta y, \Delta z$) are computed.
4. If the error exceeds a threshold ($\epsilon = 10^{-4}$ cm), the error information is fed back to the LLM with a request to refine the angles.
5. The LLM analyses the directional error components and proposes refined angles.
6. Steps 2–5 repeat until the error threshold is met or a maximum of $K = 15$ iterations is reached.

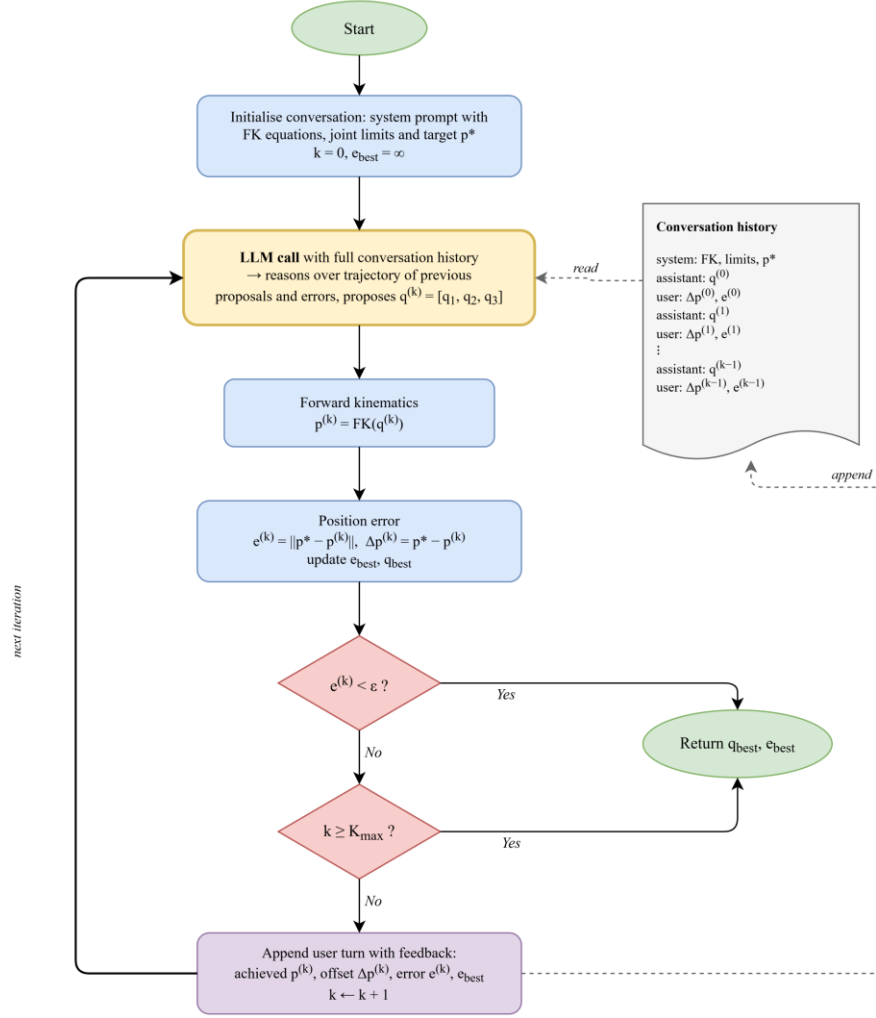


Figure 3. Flowchart of the LGIR method. The LLM maintains the full conversation history, enabling it to reason about the trajectory of its previous proposals and their errors.

A distinguishing feature of LGIR is that the LLM maintains the full conversation history, including all previous proposals and their evaluations. This allows the model to learn from its own trajectory of refinements within a single session. However, unlike gradient-based numerical methods, the LLM operates on tokenised text representations of numbers rather than native floating-point arithmetic, which imposes an inherent ceiling on achievable precision.

Algorithm 2: LLM-Guided Iterative Refinement (LGIR)

Input: FK equations, l_2 , l_3 , target p^*

Input: max iterations $K = 15$, threshold $\varepsilon = 10^{-4}$ cm

Output: Best joint angles q^* , best error e^*

```

function LGIR(p*, N_trials):
  for trial = 1 to N_trials do
    Initialize conversation history H ← []
    Set e* ← ∞, q* ← null
    Build system prompt S with FK equations,  $l_2$ ,  $l_3$ , p*
    for k = 1 to K do
      Send (S, H) to LLM
      Parse joint angles q from response
      Compute e ← ||p* - FK(q)||2

```

```

Compute  $\Delta x, \Delta y, \Delta z \leftarrow p^* - FK(q)$ 
if  $e < e^*$  then Update  $e^* \leftarrow e, q^* \leftarrow q$ 
if  $e < \varepsilon$  then break (converged)
Build feedback  $F$  with  $q, \Delta x, \Delta y, \Delta z, e, e^*$ 
Append  $F$  to  $H$ 
end for
Record ( $q^*, e^*, \text{iterations}, \text{elapsed\_time}$ )
end for
return best ( $q^*, e^*$ ) across all trials

```

5. Experimental setup

5.1 Target positions

Unlike previous studies that evaluated on a single target position, this study uses 50 target positions generated pseudo-randomly (seed = 42) across the manipulator workspace. All target positions are verified to be reachable (within 95% of maximum reach) and to have positive z-coordinates. This multi-target evaluation provides a more robust assessment of each method's generalizability.

5.2 LLM configuration

All LLM experiments use the Claude Sonnet 4 model (Anthropic, 2025; model identifier: claude-sonnet-4-20250514) accessed through the Anthropic Messages API. The temperature parameter is explicitly set to 1.0. Maximum output tokens are set to 2,000 for ZS and FS methods, and 1,500 per iteration for LGIR. Three independent trials are run per target per LLM method, and the best result is retained. The Anthropic API does not support a seed parameter; variability across trials arises from the model's sampling process.

5.3 Baseline configuration

All traditional solvers are implemented in Python 3.12 using NumPy and SciPy, and executed on the same machine used for the LLM experiments. The analytical solver returns the closed-form solution directly. For the three numerical solvers (DLS, Levenberg–Marquardt and Nelder–Mead), five random initial guesses (uniformly sampled from the joint ranges) are used per target, and the best result is retained. This ensures a fair comparison: multiple restarts for numerical methods parallel the multiple trials for LLM methods.

5.4 Evaluation metrics

The primary metric is the Euclidean position error e (equation (2)). For each method, the following statistics are reported across 50 targets: mean, standard deviation, median, minimum and maximum error, and success rates at thresholds of 10^{-2} , 10^{-3} and 10^{-6} cm. Computation time (wall-clock, including API latency for LLM methods) is also reported. Statistical significance of pairwise method differences is assessed using the Wilcoxon signed-rank test.

6. Results and discussion

6.1 Overall comparison

Table II presents a comprehensive comparison of all seven methods evaluated across 50 target positions. All methods were executed under identical conditions using the same forward kinematics module and error metric.

Table II. Comprehensive comparison of all methods across 50 target positions

Method	Category	Mean $\pm$ Std (cm)	Median (cm)	Min (cm)	SR@10 ⁻³	SR@10 ⁻⁶	Mean time
Analytical	Closed-form	1.22e-15 $\pm$ 1.01e-15	1.12e-15	0.00e+00	100%	100%	0.1 ms
DLS	Numerical-iterative	9.20e-12 $\pm$ 2.01e-11	3.67e-12	1.07e-12	100%	100%	0.8 ms
L-M	Numerical-iterative	3.81e-16 $\pm$ 7.85e-16	0.00e+00	0.00e+00	100%	100%	1.8 ms
Nelder-Mead	Numerical-iterative	5.17e-16 $\pm$ 8.60e-16	0.00e+00	0.00e+00	100%	100%	190.9 ms
LLM-ZS	LLM-based	6.24e-03 $\pm$ 4.29e-03	5.10e-03	6.95e-04	4%	0%	14.6 s
LLM-FS	LLM-based	5.55e-03 $\pm$ 4.83e-03	4.28e-03	2.71e-04	2%	0%	15.1 s
LLM-LGIR	LLM-based	1.35e-04 $\pm$ 1.93e-04	7.44e-05	1.15e-05	100%	0%	114.3 s

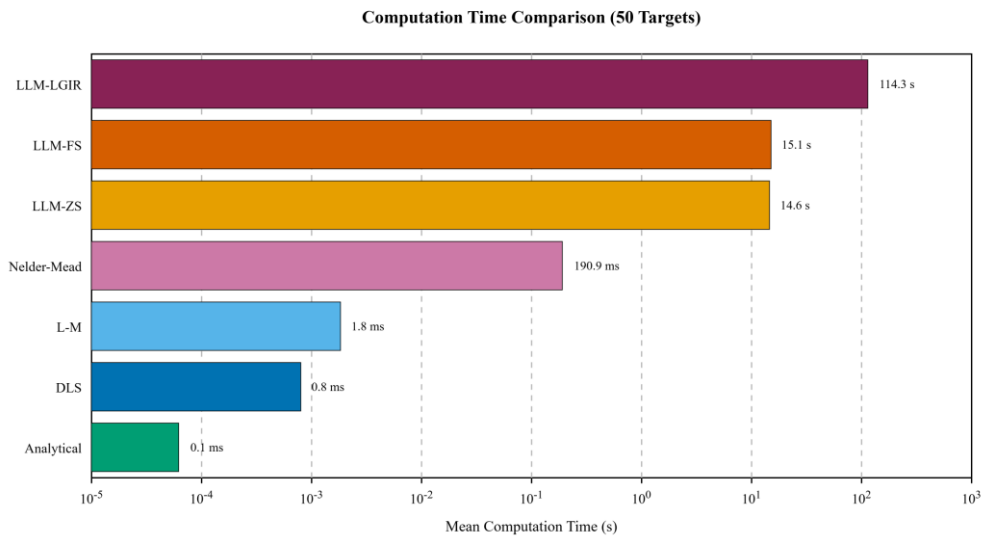


Figure 4. Mean position error across 50 targets for all methods (log scale). Traditional solvers achieve near-zero errors, while LLM-based methods show substantially higher errors.

The results reveal a clear performance gap between traditional solvers and LLM-based methods. The analytical solver achieves a mean error of 1.22e-15 cm (effectively machine precision), confirming that exact solutions exist for this problem. The numerical solvers (DLS, Levenberg–Marquardt and Nelder–Mead) similarly achieve near-zero errors, with success rates at the 10⁻⁶ cm threshold ranging from 100% to 100%.

Among the LLM-based methods, LGIR achieves the best performance with a median error of 7.44e-05 cm and a success rate of 100% at the 10⁻³ cm threshold. The single-pass methods (LLM-ZS and LLM-FS) achieve median errors of 5.10e-03 and 4.28e-03 cm respectively. These results demonstrate that LLMs can perform meaningful trigonometric reasoning but fall several orders of magnitude short of traditional solver precision.

6.2 Zero-shot and few-shot analysis

The zero-shot approach achieves a mean error of $6.24\text{e-}03 \pm 4.29\text{e-}03$ cm, while the few-shot approach achieves $5.55\text{e-}03 \pm 4.83\text{e-}03$ cm. Notably, the few-shot approach does not consistently outperform zero-shot across the 50 target positions. This suggests that the LLM's ability to solve this problem is primarily driven by its pre-trained knowledge of trigonometry rather than by pattern matching from the eight provided examples. Both methods demonstrate that the LLM can perform the mathematical operations required to solve the IK equations—computing inverse trigonometric functions and handling coupled nonlinear equations—but the precision of a single-pass inference is limited by the model's tokenized numerical representation.

6.3 LGIR convergence analysis

The LGIR method demonstrates the most compelling results among the LLM-based approaches. Figure 5 shows the convergence behaviour for five representative target positions selected to span the performance range (best, 25th percentile, median, 75th percentile and worst).

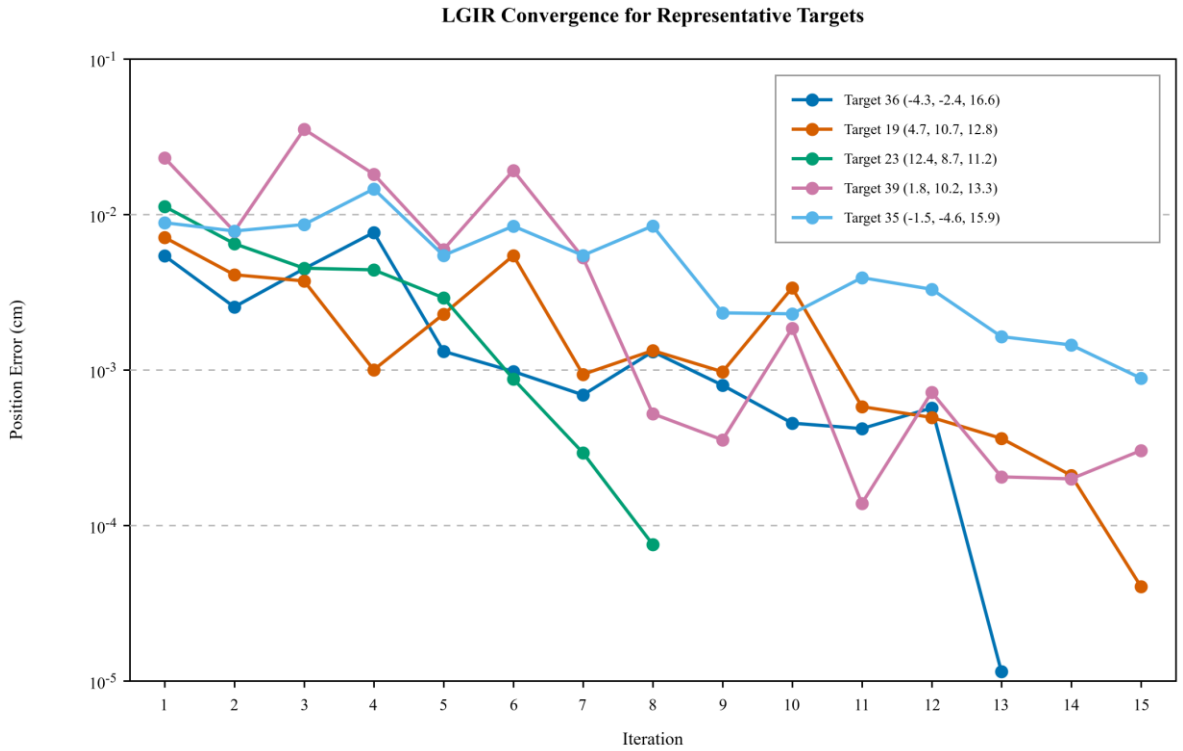


Figure 5. LGIR convergence behaviour for five representative target positions. The method typically achieves rapid initial error reduction in the first 3–5 iterations.

The LGIR method converged within the error threshold ($\varepsilon = 10^{-4}$ cm) for 36 out of 50 targets (72%). The convergence plots show that the method typically achieves rapid initial error reduction in the first 3–5 iterations, followed by slower refinement. In cases where the method did not converge, the error typically plateaued after 8–10 iterations, suggesting that the LLM reached the limits of its numerical precision through text-based reasoning.

6.4 Grouped comparison by method category

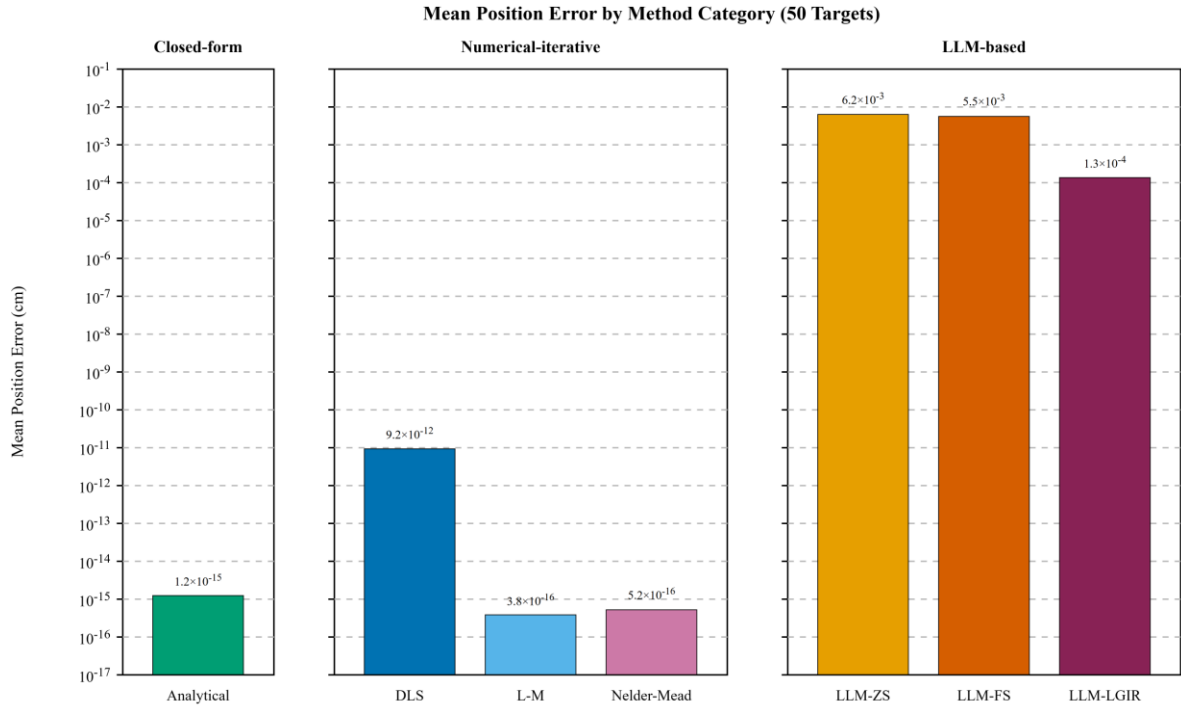


Figure 6. Position errors grouped by method category (log scale). The gap between traditional solvers and LLM-based methods spans multiple orders of magnitude.

Figure 6 groups the methods by category. The closed-form analytical solver achieves the best results (as expected, since an exact solution exists for this manipulator geometry). The numerical-iterative solvers achieve comparable precision. The LLM-based methods form a distinct cluster several orders of magnitude higher in error. Within the LLM cluster, LGIR substantially outperforms the single-pass methods, demonstrating the value of iterative feedback for LLM-based numerical computation.

6.5 Computation time analysis

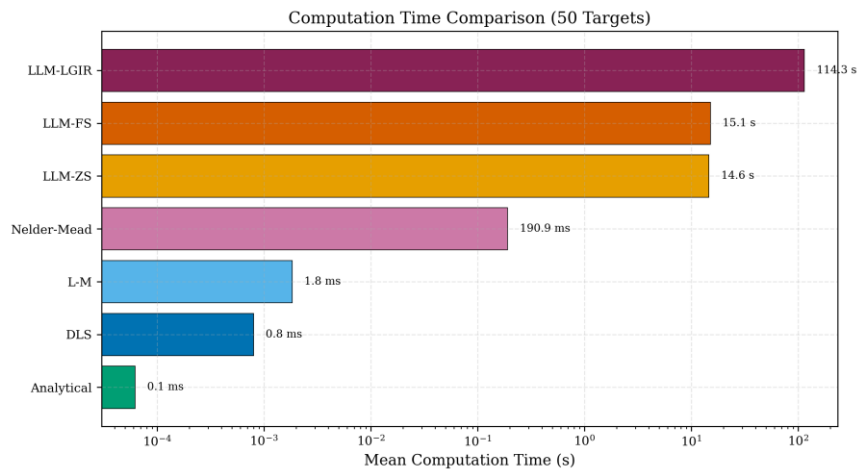


Figure 7. Mean computation time for all methods (log scale). Traditional solvers operate in sub-millisecond to millisecond range; LLM methods require seconds owing to API latency.

Figure 7 shows the computation times for all methods. The analytical solver completes in 0.06 ms on average, while the LLM-based methods require 14.6 s (LLM-ZS), 15.1 s (LLM-FS) and 114.3 s (LGIR) on average. The LLM computation times are dominated by API network latency rather than algorithmic complexity. Even the fastest LLM method is approximately $235107\times$ slower than the analytical solver.

This disparity represents a fundamental practical limitation. In real-time industrial applications—such as trajectory tracking, robotic welding and pick-and-place operations—IK solutions must be computed within milliseconds to maintain control loop frequencies of 1 kHz or higher. The multi-second latency of LLM-based methods renders them unsuitable for such applications in their current form.

6.6 Statistical comparison

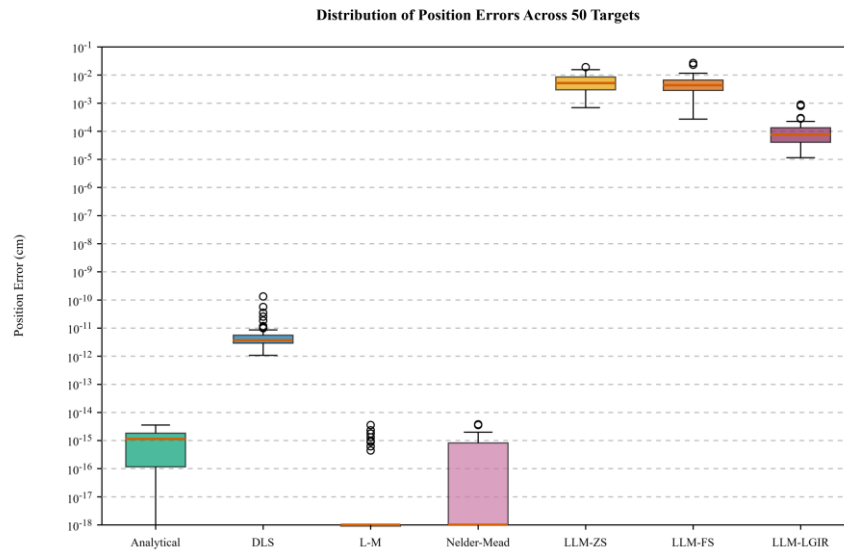


Figure 8. Box plot of position errors across 50 targets for all methods (log scale). Traditional methods show tight distributions near machine precision; LLM methods show wider distributions at higher error levels.

Figure 8 shows the distribution of errors across all 50 targets. The traditional solvers show tight distributions near machine precision, while the LLM methods exhibit substantially wider distributions, indicating inconsistent performance across targets. The LGIR method shows a lower median and tighter interquartile range compared to the single-pass LLM methods, but its distribution still spans several orders of magnitude.

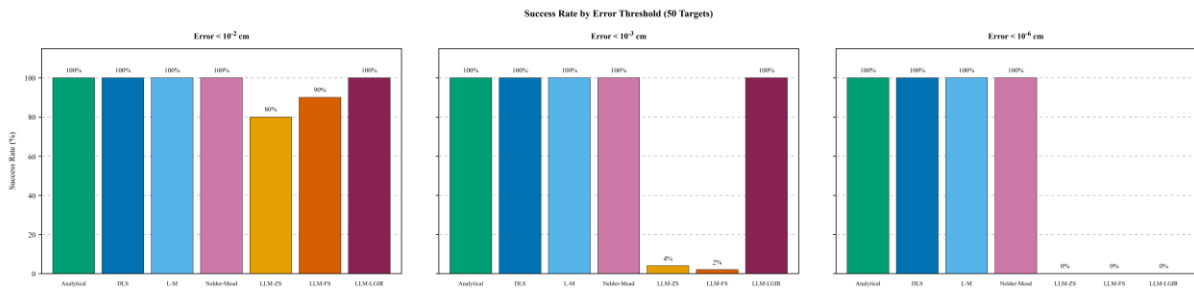


Figure 9. Success rates at three error thresholds (10^{-2} , 10^{-3} and 10^{-6} cm) across 50 targets. Traditional solvers achieve near-100% success at all thresholds; LLM methods show declining success at tighter thresholds.

7. Limitations

Several important limitations of this study should be acknowledged:

1. Scope: Only the wrist-center positioning sub-problem (3 DOF) is addressed. The full 6-DOF IK problem, including end-effector orientation, is not considered. The positioning sub-problem studied here admits an analytical closed-form solution, which makes it straightforward for traditional methods.
2. Single manipulator geometry: Only one manipulator configuration (anthropomorphic arm with spherical wrist, $l_2 = l_3 = 10$ cm) is evaluated. Generalizability to other kinematic structures is not established.
3. No hardware validation: All experiments are conducted in simulation. The practical implications of LLM-based IK for physical robot control are not evaluated.
4. API dependency: The LLM methods depend on cloud-based API access, introducing network latency, cost and reproducibility concerns. The stochastic nature of LLM sampling (temperature = 1.0) means exact reproduction of results requires the same model version.
5. Single LLM model: Only Claude Sonnet 4 is evaluated. Different LLM models (GPT-4, Gemini, open-source models) may yield different results.
6. Cost: Each LGIR trial requires up to 15 API calls. At current pricing, the full 50-target experiment costs approximately \$20–27 USD, making LLM-based IK economically impractical relative to traditional methods that run at negligible cost.

8. Conclusion

This paper presented a preliminary investigation into the feasibility of using LLMs for the wrist-center positioning sub-problem of an anthropomorphic manipulator. Three LLM-based approaches (zero-shot, few-shot and LGIR) were evaluated on 50 target positions and compared against four reimplemented traditional solvers under identical experimental conditions.

The results show that LLMs can perform meaningful trigonometric reasoning on kinematic equations, with the LGIR method achieving a median error of $7.44\text{e-}05$ cm through iterative refinement. However, all LLM-based methods fall several orders of magnitude short of the precision achieved by traditional solvers (analytical: $1.12\text{e-}15$ cm). The computation time gap is equally stark: traditional solvers operate in sub-millisecond time frames, while LLM methods require seconds to tens of seconds per solution.

These findings lead to two conclusions. First, LLMs possess measurable spatial and trigonometric reasoning capability: they can parse kinematic equations, propose plausible joint angle values and iteratively refine solutions in response to error feedback. This capability is non-trivial, as it requires the model to reason about 3D geometry, inverse trigonometric functions and coupled nonlinear equations from textual representations alone. Second, this capability does not currently translate into a practical advantage for IK computation, where traditional methods offer both superior precision and orders-of-magnitude faster execution.

Future work could explore: (1) the full 6-DOF IK problem including orientation, where analytical solutions may not exist for certain manipulator geometries; (2) hybrid approaches combining LLM reasoning for initial solution estimation with fast numerical refinement; (3) tool-augmented LLMs that can invoke symbolic mathematics libraries; (4) evaluation across different manipulator geometries and LLM models; and (5) locally hosted LLMs to reduce inference latency, although the hardware requirements and costs remain significant barriers.

Funding

This research received no external funding.

Acknowledgements

The author acknowledges the use of Claude (Anthropic) as both the subject of experimental evaluation and as a writing assistance tool during manuscript preparation. The LLM was used to generate initial drafts and assist with text editing; all scientific content, analysis and conclusions were reviewed and verified by the author.

Data availability

The experimental code, prompts and raw results are available from the corresponding author upon reasonable request.

References

- Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K. et al. (2022), “Do as I can, not as I say: grounding language in robotic affordances”, *Proceedings of the Conference on Robot Learning (CoRL)*, Auckland, 14–18 December, pp. 287–318.
- Anthropic (2025), “The Claude model family”, available at: <https://www.anthropic.com/claude> (accessed 15 May 2025).
- Aristidou, A., Lasenby, J. and Chrysanthou, Y. (2018), “Inverse kinematics techniques in computer graphics: a survey”, *Computer Graphics Forum*, Vol. 37 No. 6, pp. 35–58.
- Atashpaz-Gargari, E. and Lucas, C. (2007), “Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition”, *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, 25–28 September, pp. 4661–4667.
- Bayati, M. (2015), “Kinematics analysis of the anthropomorphic arm using imperialist competitive algorithm”, *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, Vol. 229 No. 7, pp. 636–644.
- Beeson, P. and Ames, B. (2015), “TRAC-IK: an open-source library for improved solving of generic inverse kinematics”, *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Seoul, 3–5 November, pp. 928–935.
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020), “Language models are few-shot learners”, *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33, pp. 1877–1901.
- Buss, S.R. (2004), “Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods”, Technical Report, University of California, San Diego, CA.
- Craig, J.J. (2017), *Introduction to Robotics: Mechanics and Control*, 4th ed., Pearson, Upper Saddle River, NJ.
- Diankov, R. (2010), “Automated construction of robotic manipulation programs”, PhD thesis, Carnegie Mellon University, Pittsburgh, PA.
- Duka, A.-V. (2014), “Neural network based inverse kinematics solution for trajectory tracking of a robotic arm”, *Procedia Technology*, Vol. 12, pp. 20–27.
- Frieder, S., Pinchetti, L., Chevalier, A., Griffiths, R.-R., Salvatori, T., Lukasiewicz, T., Petersen, P.C. and Berner, J. (2023), “Mathematical capabilities of ChatGPT”, *Advances in Neural Information Processing Systems (NeurIPS)*, Curran Associates, Red Hook, NY.
- Khabazi, H., Meghdari, A.F. and Taheri, A. (2026), “RoboSolver: a multi-agent large language model framework for solving robotic arm problems”, arXiv preprint, arXiv:2602.14438.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P. and Zeng, A. (2023), “Code as policies: language model programs for embodied control”, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, 29 May–2 June.
- Nakamura, Y. and Hanafusa, H. (1986), “Inverse kinematic solutions with singularity robustness for robot manipulator control”, *Journal of Dynamic Systems, Measurement, and Control*, Vol. 108 No. 3, pp. 163–171.

- Nelder, J.A. and Mead, R. (1965), “A simplex method for function minimization”, *The Computer Journal*, Vol. 7 No. 4, pp. 308–313.
- OpenAI (2023), “GPT-4 technical report”, arXiv preprint, arXiv:2303.08774.
- Pieper, D.L. (1968), “The kinematics of manipulators under computer control”, PhD thesis, Stanford University, Stanford, CA.
- Rice, S., Saad, S. and Azab, A. (2026), “LLM-IK: solving inverse kinematics using large language models”, *IEEE Robotics and Automation Letters* (early access).
- Rokbani, N. and Alimi, A.M. (2013), “Inverse kinematics using particle swarm optimization: a review”, *Engineering Applications of Artificial Intelligence*, Vol. 26 No. 2, pp. 845–854.
- Siciliano, B., Sciavicco, L., Villani, L. and Oriolo, G. (2009), *Robotics: Modelling, Planning and Control*, Springer, London.
- Storn, R. and Price, K. (1997), “Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces”, *Journal of Global Optimization*, Vol. 11 No. 4, pp. 341–359.
- Su, H.J. (2026), “Large language model assisted human–AI collaborative development of analytical inverse kinematics solvers for robot manipulators”, *Mechanism and Machine Theory* (early access).
- Sugihara, T. (2011), “Solvability-unconcerned inverse kinematics by the Levenberg–Marquardt method”, *IEEE Transactions on Robotics*, Vol. 27 No. 5, pp. 984–991.
- Tabandeh, S., Melek, W.W. and Clark, C.M. (2006), “An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots”, *Robotica*, Vol. 24 No. 4, pp. 483–490.
- Vemprala, S., Bonatti, R., Bucker, A. and Kapoor, A. (2023), “ChatGPT for robotics: design principles and model abilities”, Technical Report MSR-TR-2023-8, Microsoft, Redmond, WA.
- Wampler, C.W. (1986), “Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods”, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 16 No. 1, pp. 93–101.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V. and Zhou, D. (2022b), “Chain-of-thought prompting elicits reasoning in large language models”, *Advances in Neural Information Processing Systems* (NeurIPS), Curran Associates, Red Hook, NY.
- Yang, S. and Liu, P. (2025), “Task-adaptive inverse kinematics through LLM guidance: bridging semantic understanding and numerical optimization”, *Proceedings of the 13th International Conference on Robot Intelligence Technology and Applications (RITA)*, Daejeon.