Proceedings in Manufacturing Systems, Volume 19, Issue 2, 2024, pp. 69-74 ISSN 2067-9238

METHOD FOR QUALITATIVE EVALUATION OF ROBOTIC TRAJECTORIES BASED ON GEOMETRIC AND KINEMATIC METRICS

Cozmin CRISTOIU[1,*], Mario IVAN[2], Marius Valentin DRAGOI[3], Emilia POPESCU[4], Roxana NECHITA[5]

[1), 2), 3), 4)] Lecturer, PhD, Robots and Manufacturing Systems Dep., National University of Science and Technology POLITEHNICA Bucharest, Romania

[5)] Research Eng., Biomedical Mechatronics and Robotics Dep., National Institute of Research and Development in Mechatronics and Measurement Technique, Bucharest, Romania

Abstract: *This paper presents a quantitative framework for the qualitative evaluation of robotic trajectories based on geometric and kinematic metrics. While trajectory optimization in industrial robotics is often judged visually or through execution time, such criteria may overlook aspects of motion smoothness and stability. Starting from a set of ten randomly distributed target points, several alternative trajectories with similar total lengths were generated using cubic spline interpolation through different intermediate points. For each trajectory, specific evaluation metrics were computed, including total path length, mean and maximum curvature, jerk-based smoothness indices, and a composite quality score. The obtained values were compared and analyzed to identify correlations between the proposed metrics and the robot's simulated motion in RoboDK. Results show that even trajectories of comparable length can exhibit significantly different curvature and jerk characteristics, directly influencing execution quality. The proposed metric-based approach enables an objective assessment of trajectory quality and provides a foundation for advanced optimization strategies grounded in physical or data-driven principles.*

Key words: *robotics, trajectory, kinematics, curvature, metrics*.

1. INTRODUCTION

The generation of robot trajectories in industrial applications is increasingly critical not only for achieving target positions, but also for ensuring motion quality, smoothness and stability throughout the path. As manipulators are tasked with more complex and dynamic operations, metrics such as cycle time or endpoint accuracy alone are no longer sufficient: the kinematic behavior along the trajectory – including velocity, acceleration and jerk – play a major role in reducing wear, vibration, and in maintaining repeatable, safe motion. Although a variety of trajectory-planning and optimization methods have been proposed, the objective evaluation of trajectory quality remains largely reliant on visual inspection or subjective assessments rather than on standardized, quantitative criteria.

Existing research has suggested using metrics such as total path length, curvature, and jerk as quantitative indicators of trajectory smoothness and quality. For example, Sharkawy presents a mathematical analysis of minimum-jerk trajectories applied to straight and curved motions, showing that jerk minimization yields smoother motion profiles [1]. Fan et al. propose a Cartesian-based trajectory optimization with joint-jerk limits and demonstrate improved execution quality in industrial manipulators [2]. However, relatively few works explore how multiple trajectories that are geometrically comparable (e.g., similar total length) can nevertheless differ in subtle kinematic metrics, and how these differences can be quantified and correlated with actual simulation or robot execution behavior. A sample illustration is provided in Fig. 1 to visually demonstrate two such alternative trajectories that connect the same target points yet differ subtly in shape and curvature.

This paper addresses this gap by proposing a set of quantitative indicators for the qualitative evaluation of robotic trajectories. Starting from a fixed sequence of ten target points, we generate several alternative trajectories

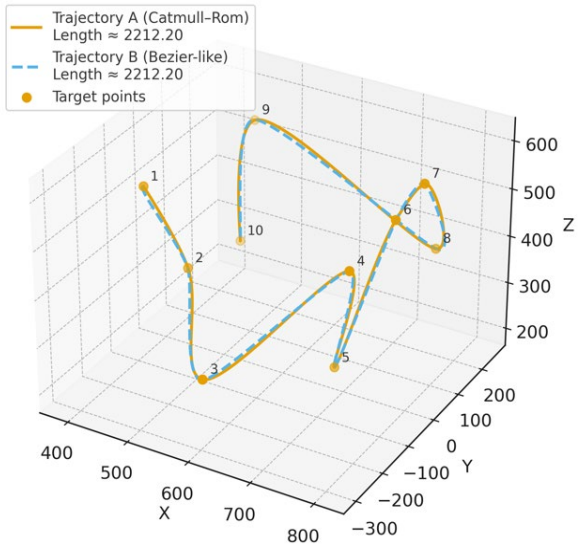

Fig. 1. Two alternative trajectories through the same 10 targets with nearly identical total length.

* Corresponding author: Splaiul Independenței 313, Bucharest 060042, Romania,
Tel.: +40766714482,
E-mail addresses: *cozmin.cristoiu@upb.ro* (C. Cristoiu).

through different sets of intermediate way-points via spline-based interpolation, ensuring that the overall path lengths remain closely matched. Because the trajectories are purposely designed to have very similar lengths, visual discrimination of the "best" trajectory becomes difficult – thereby motivating the need for objective metrics.

The computed metrics for each trajectory – such as mean and maximum curvature, mean and peak jerk, and a composite smoothness score – are then compared and validated through simulation in RoboDK. The results show that trajectories with nearly identical lengths can exhibit significantly different kinematic profiles, and that the proposed metric-based framework provides a reliable basis for distinguishing and selecting higher-quality trajectories.

The remainder of this paper is structured as follows: Section 2 presents the methodology for trajectory generation, metric computation and simulation setup; Section 3 discusses the obtained results and comparative analysis; Section 4 examines the validation of the metrics via simulation; and Section 5 concludes the work with final remarks and future research directions

2. METHODOLOGY

2.1. Trajectory data and target configuration

The study begins from a set of fixed target points, randomly distributed in 3D Cartesian space. These points are stored in a .csv file and represent the same sequence of positions that would be reached by a 6-DOF industrial manipulator (Fig. 2).

The order of points is preserved across all test cases to maintain consistent task logic. To ensure comparability, all trajectories start and end at identical coordinates, with the same boundary orientation of the tool center point (TCP).

In the RoboDK environment, these targets correspond to way-points of a simulated manipulator. The motion between them is typically generated using linear (MoveL) or joint (MoveJ) interpolation. However, for this study, we reproduce these trajectories offline in Python, where several alternative paths are generated between the same targets to enable a purely geometric and kinematic comparison before robot execution.

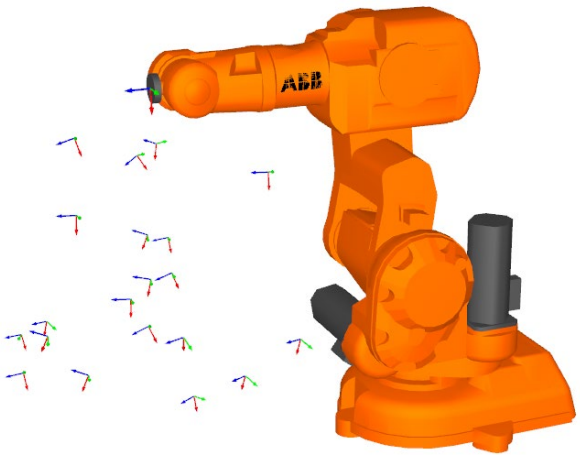

Fig. 2. Random study points in robot workspace.

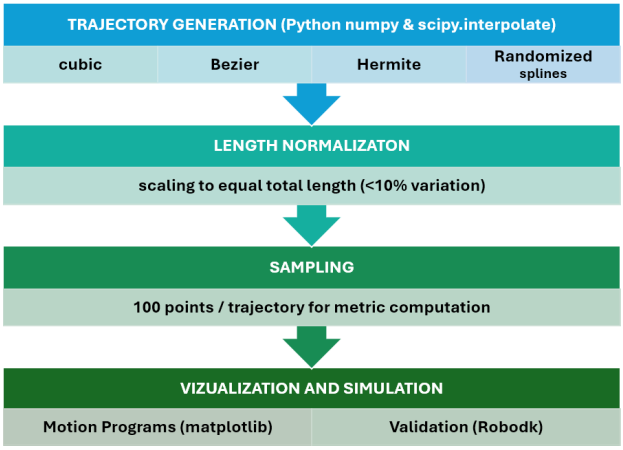

Fig. 3. Schematic pipeline of the process.

2.2. Generation of alternative trajectories

Several classical approaches to robotic path generation exist, ranging from sampling-based planners such as Rapidly-Exploring Random Trees (RRT) [3] to continuous interpolation methods. In this work, we focus on continuous curve representations to ensure differentiable curvature and smoothness metric. For each pair of consecutive target points, we insert intermediate way-points following distinct interpolation logics:

- Uniform cubic spline interpolation, ensuring continuity of first and second derivatives [4].
- Randomized spline interpolation, adding intermediate control points slightly perturbed around the nominal straight line [5].
- Bezier-based interpolation to impose smooth curvature transitions [6].
- Hermite quintic interpolation, providing continuous position, velocity and acceleration profiles [7].

All trajectories were scaled to approximately equal total length, with length variations below 20%. This constraint guarantees that differences between metrics are not dominated by simple path extension but rather by shape and smoothness characteristics. The generation process was implemented in Python 3.11, using the *numpy* and *scipy.interpolate* libraries for curve synthesis, and Matplotlib for visualization. Each trajectory is sampled with 100 points, providing sufficient spatial resolution for curvature and derivative computations. A schematic of the pipeline is illustrated in Fig. 3.

2.3. Computation of evaluation metrics

To quantitatively assess the quality of each generated trajectory, a set of geometric and kinematic metrics was computed. The evaluation process was implemented entirely in Python, using NumPy and SciPy for numerical differentiation and Matplotlib for visualization. Each trajectory was represented as a parametric spline curve:

$$r(s) = [x(s), y(s), z(s)], \quad (1)$$

where $s \in [0,1]$ denotes the normalized arc-length parameter. This representation ensures continuity of first-second-, and third-order derivatives, which are required for curvature and jerk analysis.

The total path length L is obtained as the sum of Euclidean distances between consecutive points along the discretized curve:

$$L = \sum_{i=1}^{N-1} |r_{i+1} - r_i|, \quad (2)$$

where N represents the number of sampled points. This metric provides a geometric baseline for comparing trajectories of similar overall extent.

Curvature is computed from the first and second derivatives of the spline according to classical differential-geometry formulation as defined in classical robot kinematic texts [8]:

$$k = \frac{|r' \times r''|}{|r'|^3}. \quad (3)$$

The curvature quantifies the instantaneous deviation of the trajectory from a straight line. High curvature values indicate sharper turns and potentially higher joint accelerations. From the discretized dataset, two scalar indicators are extracted:

- the mean curvature κ_{mean}, representing the overall smoothness of the path
- the maximum curvature κ_{max}, indicating the most abrupt segment.

To capture higher-order smoothness characteristics, the jerk magnitude is evaluated as the norm of the third derivative of position:

$$J(s) = |r'''(s)|. \quad (4)$$

Since the trajectories were parametrized by normalized arc-length rather than time, jerk values are expressed in dimensionless units, reflecting relative changes in motion continuity rather than absolute accelerations. The mean (J_{mean}) and maximum (J_{max}) values are extracted from the sampled data.

Finally, a composite smoothness index, S, is introduced to provide a unified measure of trajectory quality [9]. The index combines normalized curvature and jerk components:

$$S = \alpha \cdot k_{mean} + \beta \cdot J_{mean}, \quad (5)$$

where weighting coefficients α and β are chosen empirically (in our tests 1 and 0.1) to emphasize geometric rather than purely dynamic smoothness. Lower values of S correspond to smoother, more stable trajectories.

All computed metrics are stored in comparative tables. These metrics form the foundation for the subsequent simulation and validation stage described in Section 3.

3. SIMULATION AND VALIDATION

The resulting trajectories are imported into RoboDK through its Python API, which provides full programmatic control over robot models, targets, and motion commands. Each trajectory is represented as a series of Cartesian targets corresponding to the sampled points of the generated spline curves. For each case, an automated motion program was created using the MoveJ command, ensuring consistent motion type and parameters across all experiments. Figure 4 illustrates the simulation setup in RoboDK, showing the generated targets, robot workspace, and associated control panel used for execution monitoring.

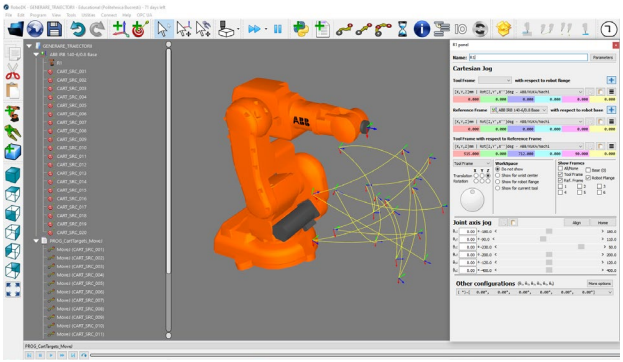

Fig. 4. Simulation layout in RoboDK.

First, the input file containing 10 random points was imported in Robodk and simulation tagets were created. Initial target points and generic cubic spline trajectories between them are presented in Fig. 5. The trajectory was then sampled into 100 pieces as presented in Fig. 6.

In the same manner, the other three trajectories (Bezier, Randomized Spline and Hermetic Quintic) were created in Robodk. These three trajectories are presented in Figs. 7, 8 and 9. From a visual point of view, the differences between the trajectories are extremely difficult to distinguish, but the visual feedback from

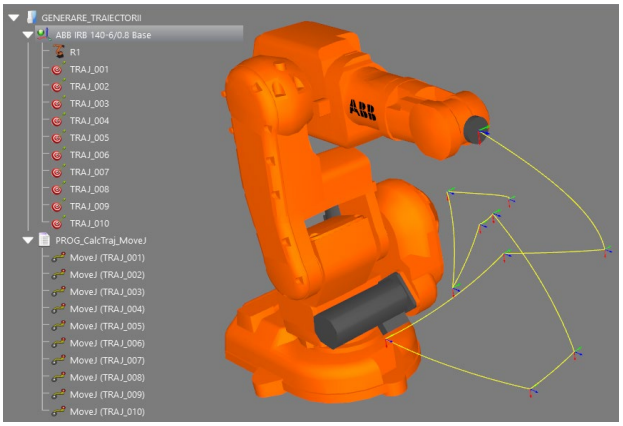

Fig. 5. Initial target points and cubic spline trajectories.

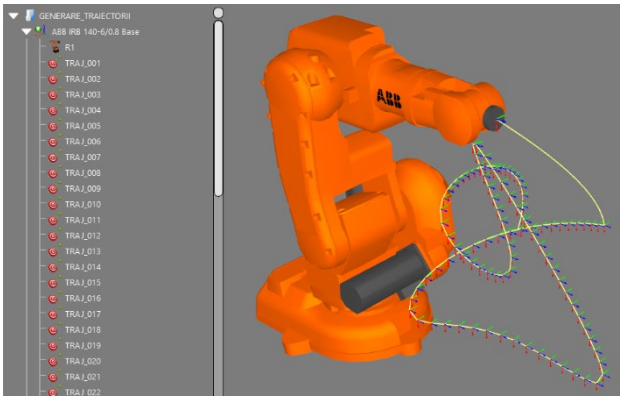

Fig. 6. Sampled cubic spline trajectories.

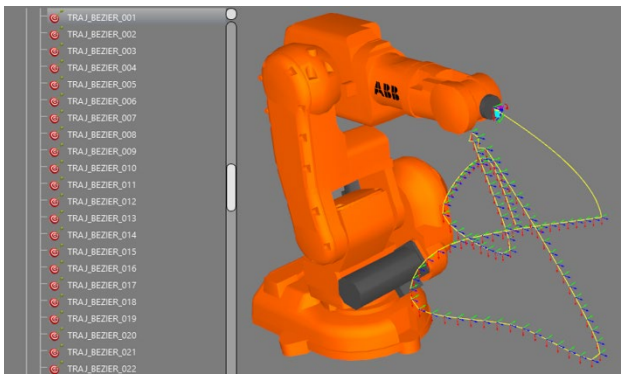

Fig. 7. Bezier trajectories.

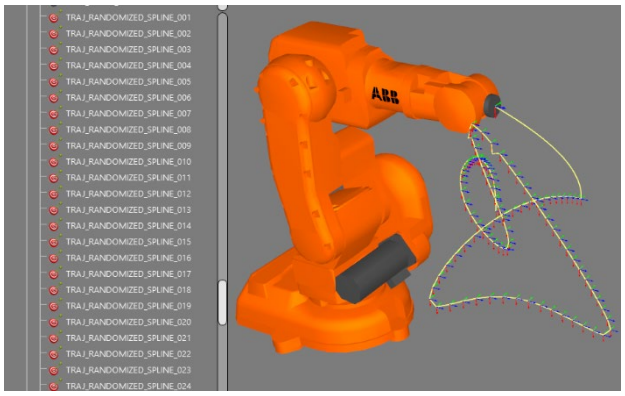

Fig. 8. Randomized spline trajectories.

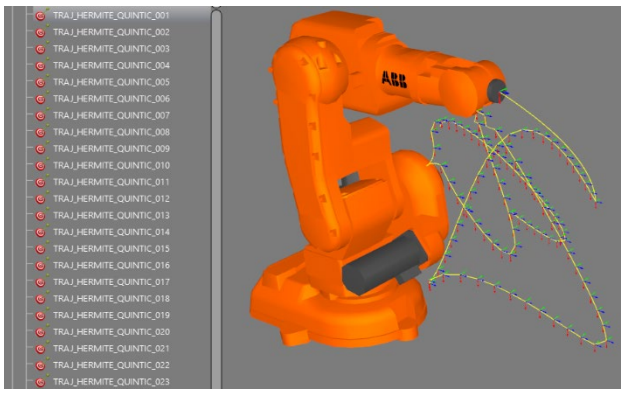

Fig. 9. Hermetic quintic trajectories.

RoboDK is very important because it allows us to assess whether the resulting trajectories are similar (as planned) and whether there are any errors or aberrant trajectories.

In Fig. 10 all four trajectories are presented superimposed. If the size factor is large enough, it can be seen that the trajectories are similar but not exactly identical.

For each set of points and trajectories, motion programs were generated. Absolutely all motion instructions within all programs are of the MoveJ type. To ensure tracking with the same degree of tracking of the trajectories, a smoothing factor of 0.001 mm was set.

Same robot model and reference frame (robot base frame) were maintained in all tests to guarantee comparability. A six-degree-of-freedom industrial robot with a nominal reach of approximately 900 mm was used (ABB IRB140). The base frame and tool center point

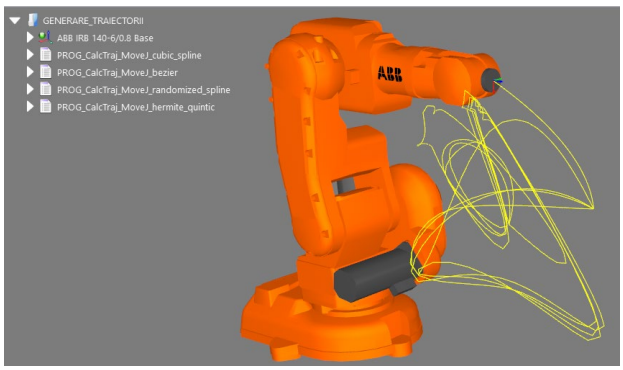

Fig. 10. All trajectories superimposed.

(TCP) were fixed for all trajectories, while the robot's joint configurations were automatically determined by the RoboDK solver. Motion speed and acceleration parameters used are:

- Linear speed = 150 mm/s
- Linear acceleration = 200 mm/s^2
- Joint speed = 200 deg/s
- Joint acceleration = 1000 deg/ s^2

During simulation, RoboDK provided access to time-stamped data on TCP position, velocity, and orientation, as well as joint angles and corresponding velocities. These data were exported automatically in .csv format for each trajectory and subsequently processed in Python. Results are presented in Section 4.

4. RESULTS

For each trajectory type the following metrics were calculated: length, mean curvature κ_{mean}, maximum curvature κ_{max}, mean jerk J_{mean}, maximum jerk J_{max} and smoothness index S. Values of these metrics are presented in Table 1.

Then each robot motion program was executed. The path lengths measured by RoboDK and the execution time of each program were noted. The values are presented in Table 2.

Then we ran the same programs 10 times each to see how the times are scaled when the robot operates in continuous cycle mode. Note that when running in the loop an additional piece of path is added to return from the last point to the first point and resume the cycle.

Table 1

Curvature metrics of trajectories

¶	Cubic	Bezier	R. S.	H. Q.
L [m]	2877.5	2718.3	2895.7	3726.1
κ_{mean}	0.0060	0.0061	0.0061	0.0066
κ_{max}	0.0475	0.070	0.050	0.132
J_{mean}	2.303	2.249	2.369	3.922
J_{max}	10.465	10.625	10.671	17.748
S	0.236	0.231	0.243	0.398

Table 2

Program execution length and duration for one working cycle

	Cubic	Bezier	R. S.	H. Q.
L (mm)	2897	2739	2919	3800
T (s)	4.1	4.0	4.2	6.5

Table 3

Program execution length and duration for 10 working cycles

	Cubic	Bezier	R. S.	H. Q.
L [mm]	31149	29567	31607	40421
T [s]	43.1	41.7	44.1	67.1

Table 4

Program execution length and duration for 100 working cycles

	Cubic	Bezier	R. S.	H. Q.
L [mm]	313904	298081	316068	404213
T [s]	432.9	418.5	441.4	670.8

Results with cumulated lengths and duration for each trajectory type for continuous working cycle (repeating 10 times each motion program) are presented in Table 3.

In order to be able to extrapolate for longer work cycles, we also ran the simulation for 100 cycles for each of the trajectories. The results are presented in Table 4.

For easier interpretation, the results are presented in graphical form in the following figures.

Figure 11 presents the total execution time of each trajectory type for 1, 10, and 100 working cycles. The relationship between the number of cycles and execution time remains nearly linear, confirming the repeatability of the motion programs. However, clear differences are observed between trajectory types: Bézier and Cubic spline exhibit the shortest total durations, while Hermite Quintic trajectories consistently require longer execution times.

Figure 12 shows the average execution time per cycle, which remains practically constant regardless of the number of repetitions. This behavior confirms the numerical stability of the simulation and indicates that the timing differences between trajectories are due to intrinsic kinematic properties rather than accumulated computational effects.

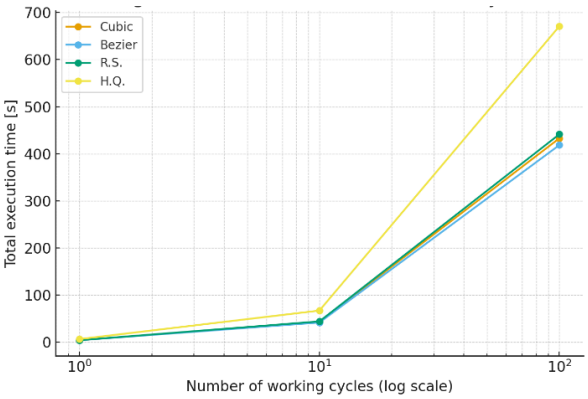

Fig. 11. Program execution times vs. no. cycles.

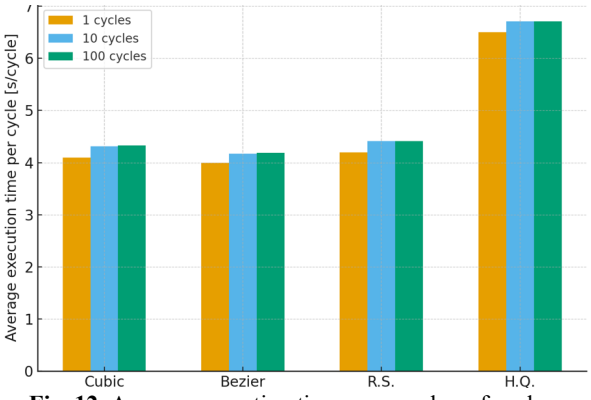

Fig. 12. Average execution times vs. number of cycles.

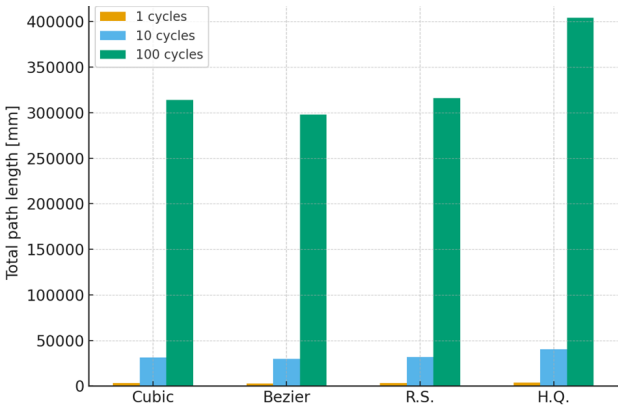

Fig. 13. Path length vs. number of cycles.

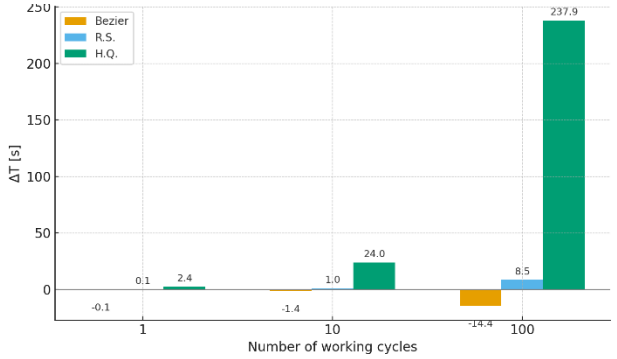

Fig. 14. Execution time difference vs. number of cycles (ΔT= method-cubic).

Figure 13 illustrates the total path length for each trajectory type and cycle count. The Bézier trajectory remains the shortest due to its smooth interpolation, while the Hermite Quintic curve exhibits the highest total length, consistent with its larger curvature amplitude. The Randomized spline and Cubic spline trajectories maintain closely matching values.

Figure 14 shows the execution time difference ΔT between each trajectory and the Cubic spline reference. The Hermite Quintic trajectory exceeds the Cubic case by up to 240 seconds for 100 cycles, while the Bézier curve is faster by approximately 15 seconds for the same test, emphasizing the influence of geometric smoothness on execution efficiency.

Figure 15 presents the path length difference ΔL relative to the Cubic spline trajectory. The Bézier curve results in paths shorter by about 5–7%, while the Hermite Quintic trajectories exceed the reference by more than 25%, confirming that smoother curvature does not necessarily imply shorter paths.

Overall, the obtained results clearly demonstrate that even trajectories with nearly identical start and end configurations can exhibit notable differences in both geometric and kinematic behavior. The cubic and randomized spline trajectories produced comparable lengths and timing, indicating that small spatial perturbations do not significantly affect cycle duration.

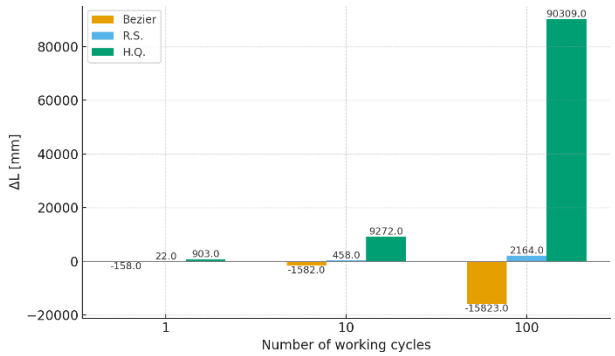

Fig. 15. Path length difference vs. cycles (ΔL = method-cubic).

Conversely, the Hermite Quintic method, while ensuring excellent continuity in position, velocity, and acceleration, led to systematically longer paths and higher execution times, suggesting that an overly smooth profile can compromise efficiency in repetitive industrial cycles.

The Bézier trajectory achieved the most balanced performance, combining reduced curvature and low jerk with the shortest overall path and execution time.

These observations highlight that geometric and kinematic metrics provide an objective and practical means for selecting trajectory types based on desired motion characteristics and operational goals.

5. CONCLUSIONS

This study proposed and validated a quantitative framework for the qualitative evaluation of robotic trajectories based on geometric and kinematic metrics.

The implementation in Python and RoboDK enabled the automatic generation, simulation, and analysis of multiple trajectory types under identical boundary and motion conditions.

The results confirm that even when total path lengths differ by less than 10–20%, the associated curvature, jerk, and execution time can vary substantially. Among the tested methods, the Bézier and Cubic spline trajectories offered the best compromise between smoothness and execution time, while the Hermite Quintic method—although theoretically optimal for motion continuity—proved less efficient for repetitive tasks.

From an industrial perspective, these findings demonstrate that a simple and informed selection of the trajectory type, guided by metric-based evaluation, can yield significant productivity gains. In continuous 24/7 operating conditions, such as those of high-duty industrial robots, the accumulated difference in execution time between less optimal and optimal trajectories could represent several hours or even days of saved cycle time over weeks or months of repetitive operation.

Future work will focus on integrating the metric-based evaluation into real-time optimization frameworks, allowing robots to automatically adapt their motion planning for both energy efficiency and long-term mechanical stability.

REFERENCES

[1] A. N. Sharkawy, *Minimum jerk trajectory generation for straight and curved movements*: *Mathematical Analysis*, Advances in Robotics and Automatic Control: Reviews, Vol. 2, 2021, pp. 187–201.

[2] Z. Fan, K. Jia, L. Zhang, F. Zou, Z. Du, M. Liu, Y. Cao, Q. Zhang, *A cartesian-based trajectory optimization with jerk constraints for a robot*, Entropy, Vol., 25. Issue 4, 2023

[3] J. Kuffner, S. LaValle, *RRT-Connect: An efficient approach to single query path planning*, Proceedings of International Conference on Robotics and Automation, ICRA 2000, 2000, pp. 995-1001.

[4] Carl de Boor, *A practical guide to splines*, Springer New York, ISBN: 978-0-387-95366-3, 2001.

[5] Giannelli, C., Lorenzo S., Alessandra S.,*A local C^2 Hermite interpolation scheme with PH quintic splines for 3D data streams*, University of Florence, Italy, *arXiv preprint arXiv:2108.12948* (2021).

[6] Christian S., Timo T., Tobias M., *Bézier Curve Based Continuous and Smooth Motion Planning for Self-Learning Industrial Robots*, International Conference on Flexible Automation and Intelligent Manufacturing, Procedia Manufacturing 38 (2019) pp. 423–430, 2019

[7] Morten L., *Real-time quintic Hermite interpolation for robot trajectory execution*, PeerJ Computer Science, Vol 6, 2020, doi: 10.7717/peerj-cs.304

[8] M. Dobis, M. Dekan, P. Beno, F. Duchon, A. Babinec, *Evaluation criteria for trajectories of robotic arms*, Robotics, Vol.11, No.1, 2022, pp. 29-39.

[9] L. Sciavicco, B. Sicilliano, *Modeling and control of robotic manipulators*, Springer Science & Businss Media, 2001, ISBN: 1-85233-221-2.