JOURNAL OF ARTIFICIAL INTELLIGENCE AND ROBOTICS
Jan - Mar 2026, VOL. 3, ISSUE 1, pp. 27-29
https://doi.org/10.61577/jaiar.2026.100004

RESEAPRO JOURNALS

TECHNICAL NOTE

OPEN ACCESS

Cubic spline interpolation for smooth trajectory generation in autonomous motion systems

Aarush Mandayam and Elliott Tammaro

10X Accel Mentor, Chestnut Hill College, Philadelphia, Pennsylvania, United States

ABSTRACT

Generating a continuous reference trajectory from prescribed waypoints is a basic task in autonomous motion systems. Piecewise-linear position interpolation generally introduces velocity discontinuities at interior waypoints, whereas cubic spline interpolation can provide continuous position, velocity, and acceleration. This technical note describes the automated assembly of a direct linear system for computing piecewise cubic polynomial coefficients from position–time pairs. The formulation combines endpoint interpolation, continuity of the first two derivatives, and two endpoint boundary conditions. A stepwise computational procedure makes the individual constraints explicit. A general dense solution has cubic computational cost in the number of segments, while the classical tridiagonal formulation offers a more efficient alternative. The scope is limited to smooth interpolation: velocity, acceleration, jerk, obstacle avoidance, and system dynamics are not constrained. Consequently, the resulting curves require additional checks before use as executable robot or drone commands. The note presents the mathematical construction and its implementation requirements; numerical benchmarking and physical trajectory tracking are outside its scope.

KEYWORDS

Cubic splines; Trajectory generation; Numerical interpolation; Waypoint interpolation; Autonomous motion systems

ARTICLE HISTORY

Received 19 January 2026;
Accepted 08 February 2026;
Published 15 February 2026

Introduction

A sequence of waypoints specifies where a trajectory should pass, but does not by itself define how motion should evolve between those points. When consecutive position–time samples are connected by straight segments, segment velocities are constant and generally differ at their junctions. Exact tracking would therefore require instantaneous velocity changes. Cubic spline interpolation addresses this continuity problem by joining cubic polynomial segments with matching first and second derivatives [1].

Smoothness is only one requirement in motion planning. Minimum-snap methods use higher-order polynomial trajectories and optimize a cost involving the fourth derivative of position for quadrotor applications [2]. Jerk-limited online trajectory generation explicitly incorporates velocity, acceleration, and jerk bounds [3]. Time-optimal path parameterization addresses the separate question of how a specified path should be traversed under motion constraints [4]. These approaches establish why a smooth interpolant alone should not be described as a complete dynamically feasible motion plan.

This note describes the programmatic assembly of the classical cubic spline equations in the polynomial-coefficient basis. Its purpose is to make the constraints and their implementation explicit, rather than to introduce a new interpolation theory. Established scientific software provides interpolation and linear algebra routines, including a twice continuously differentiable cubic spline interpolator with several boundary-condition options [5,6]. The formulation presented here provides an explicit constraint-based account of this established construction.

Free-form curves and sampled pen traces are potential applications of waypoint interpolation. Signature trace reconstruction should be distinguished from signature verification, which concerns identity classification and may employ learned representations [7]. The present formulation addresses geometric reconstruction only; it does not establish authentication performance. No signature dataset or reconstructed signature is evaluated in this note.

Mathematical Formulation and Implementation

Inputs and polynomial representation

Let $n + 1$ scalar waypoints be specified by position values x_i at strictly increasing parameter values t_i, for $i = 0, \ldots, n$. When the parameter represents physical time, the first and second derivatives represent velocity and acceleration. With an arbitrary curve parameter, the derivatives describe parameterized curve behaviour and cannot be interpreted directly as physical motion quantities.

For each interval, define $h_i = t_{i+1} - t_i > 0$ and the local coordinate $u = t - t_i$. The segment polynomial is defined by four coefficients per segment and $4n$ unknowns overall. Local coordinates avoid unnecessary powers of absolute time values, although numerical conditioning still depends on the input scales and interval lengths.

Interpolation and continuity constraints

The two interpolation conditions for each segment are

At each interior junction, the first and second derivatives must agree:

Together these provide $4n - 2$ equations. Two additional boundary conditions complete the system. Natural boundary conditions set the endpoint second derivatives to zero:

Alternatively, clamped conditions prescribe endpoint first derivatives v_start and v_end:

Natural conditions impose zero endpoint acceleration when t is time; they do not imply zero endpoint velocity. Starting and ending at rest requires the corresponding velocity conditions. Boundary choices must reflect the intended application and match those used by any reference implementation [1,6].

Linear system and implementation procedure

***Correspondence:** Dr. Elliott Tammaro, 10X Accel Mentor, Chestnut Hill College, Philadelphia, Pennsylvania, United States. e-mail: info@10Xlearningacademy.com

© 2026 The Author(s). Published by Reseapro Journals. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Stack the coefficients in the vector $q = (a_0, b_0, c_0, d_0, \ldots, a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1})^T$. Assemble the interpolation, continuity, and selected boundary equations into

Here f contains waypoint values and boundary constants. Although coefficient recovery can be written formally using the inverse of A, numerical implementation should solve the linear system directly rather than explicitly forming the inverse. The procedure below specifies the assembly and evaluation steps independently of a particular programming language or software library.

- Validate finite input values and strictly increasing parameter values.
- Allocate A and f and insert the two endpoint interpolation equations for each segment.
- Insert first- and second-derivative continuity equations at interior junctions.
- Insert the two selected boundary equations and solve Aq = f.
- Store the four coefficients for each segment.
- For each query parameter, locate its interval and evaluate the polynomial and requested derivatives.

A general dense factorization has $O(n^3)$ arithmetic cost and $O(n^2)$ storage. Although the assembled constraints are sparse, actual performance depends on the storage format and solver. The classical moment formulation instead solves a tridiagonal system in $O(n)$ time [1]. These are algorithmic complexity statements, not measured runtime results for this implementation.

Discussion and Limitations

The direct coefficient formulation explicitly links each interpolation and continuity requirement to a row of the linear system. This can aid implementation inspection and teaching. It solves the classical spline problem and does not establish mathematical novelty or computational superiority over existing spline routines. The contribution is the explicit organization of the constraints and their computational assembly.

For multiple coordinates, separate splines may be fitted using the same knots to produce a vector curve with continuous first and second derivatives. That extension does not by itself impose collision avoidance, kinematic coupling, actuator limits, or dynamic balance. A scalar formulation therefore provides a building block for trajectory generation rather than a validated autonomous motion system.

Within a cubic segment, the third derivative is constant, but it can jump at a junction. Thus, C^2 continuity does not provide continuous jerk or enforce a jerk limit. Applications requiring bounds on higher derivatives or motion rates need additional planning methods, such as those considered in [2–4]. Time allocation also changes derivative magnitudes and must be justified before interpreting a curve as an executable motion command.

Exact interpolation reproduces measurement noise at the fitted points. It should not be described as noise filtering unless an additional smoothing procedure is specified. Cubic splines may also overshoot between samples. Shape-preserving cubic interpolation offers an alternative when monotonicity is important, although it does not generally provide the same C^2 continuity as the present construction [8].

The analytical continuity conditions establish smoothness of the mathematical interpolant, but do not establish reconstruction accuracy, runtime performance, or physical tracking capability. Numerical benchmarking, comparisons with established implementations under identical boundary conditions, and simulation or hardware testing are directions for further evaluation. For geometric applications, interpolation residuals at fitted waypoints alone would not establish accuracy between those points.

Data Availability Statement

No original datasets, numerical benchmark results, or executable software implementations are reported in this technical note. The mathematical formulation, constraint equations, and stepwise computational procedure are fully described in the text.

Funding Statement

The authors received no financial support for the research or preparation of this article.

Ethics Declaration

Not Applicable.

Author Contributions

Conceptualization: A.M. and E.T.; Methodology: A.M. and E.T.; Formal analysis: A.M. and E.T.; Writing – original draft: A.M. and E.T.; Writing – review & editing: A.M. and E.T. All authors have read and agreed to the published version of the manuscript.
Conflict of Interest.

Conflict of Interest

The authors declare no conflict of interest.

Preprint Disclosure

An earlier version of this manuscript was posted on arXiv as arXiv:2610.12826v1. The present manuscript has been revised as a technical note.

Conclusion

This technical note sets out the direct polynomial-coefficient formulation of classical cubic spline interpolation and a procedure for assembling its constraints automatically. The resulting mathematical construction passes through prescribed scalar waypoints while matching first and second derivatives at interior junctions. Its value lies in making the formulation explicit and inspectable. A general dense solution is less scalable than the established tridiagonal approach, and smoothness alone does not guarantee feasible robot or drone motion. Numerical benchmarking and physical validation remain future work.

References

1. C DB. A Practical Guide to Splines, Revised ed. New York. 2001.
2. Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors. 2011 IEEE International Conference on Robotics and Automation. 2011:2520–2525. https://doi.org/10.1109/icra.2011.5980409
3. Berscheid L, Kröger T. Jerk-limited real-time trajectory

© 2026 Reseapro Journals
https://doi.org/10.61577/jaiar.2026.100004

generation with arbitrary target states. 2021. https://doi.org/10.15607/rss.2021.xvii.015.
4. Pham H, Pham Q. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. IEEE Transactions on Robotics. 2018;34(3):645–659. https://doi.org/10.1109/tro.2018.2819195
5. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods. 2020;17(3):261–272. https://doi.org/10.1038/s41592-019-0686-2
6. SciPy Developers. CubicSpline. 2026.
7. Saffar MH, Fayyaz M, Sabokrou M, Fathy M. Online Signature Verification using Deep Representation: A new Descriptor. 2018. https://doi.org/10.48550/arxiv.1806.09986.
8. Fritsch FN, Carlson RE. Monotone Piecewise Cubic Interpolation. SIAM J. Numer. Anal. 1980;17(2):238–246. https://doi.org/10.1137/0717021

© 2026 Reseapro Journals

https://doi.org/10.61577/jaiar.2026.100004