

Efficient Algorithm for Spatial Trajectories Combined from Double-S Motions of Arbitrary Order based on a Unified Equation

Rico Zöllner[✉],
Heiner Ludwig[✉],
Thorsten Schmidt[✉]

Chair of Materials Handling and Logistics Engineering
TU Dresden

The paper presents an algorithm for generating and optimizing spatial trajectories based on a unified equation applicable to one-dimensional double-S motion profiles of arbitrary order. The proposed algorithm leverages an analytical equation for the time horizon computation, avoiding traditional nested case distinctions, and integrates numerical computation and visualization tools. This framework allows for real-time optimization of time, velocity and higher-order derivatives (with constraints taken into account). High-order trajectories are commonly used in robotics, high-precision systems, or aggressive drone maneuvers. We highlight the generalizability to other kinds of trajectories.

[Keywords: Motion Planning, Trajectory Optimization, On-line Trajectory Generator, Double-S Motions of Higher Order]

1 INTRODUCTION

Motion planning and trajectory optimization are two extensively researched scientific fields due to their numerous applications. Against this backdrop, it seems surprising that gaps still exist even for seemingly simple problems. Considering one-dimensional trajectories with a symmetrical profile, vanishing boundary values for velocity and higher derivatives, and piecewise constant highest order (so-called double-S curves), which play a fundamental role in time-minimal and similar motions, the traditional approach was mainly guided by nested case distinctions and decision trees, which – when higher orders come into play – become opaque and unwieldy (see [1] and [2]). Although by now high-order motions (considering orders beyond the jerk: snap as 4th derivative, or crackle as 5th derivative, for instance) have applications, e.g., in high-precision robotics,

drone maneuver planning, astronautics or even amusement engineering (see [3, 4]), systematic comprehensive studies on general orders are rare (but they do exist, e.g. [5, 6, 7]).

For one-dimensional trajectories of the above type there is a unifying equation for the time horizon of the motion [8]. For example, if we consider 3rd order motions (jerk-controlled), then the duration T of the motion is given by

$$T = \frac{s}{v} + \frac{v}{a} + \frac{a}{j} \quad (1)$$

regardless of how many phases the motion consists of. Hereby, s denotes the distance, and v, a, j stand for the achieved velocity, acceleration and jerk, respectively (sometimes called “peak values” and to be distinguished from the given constraint values). The generalization of (1) to arbitrary orders seems quite obvious.

Although we are dealing with one single equation, it indeed contains enough information allowing optimization, decisions which constraints are active, and insights into the general systematics. A very brief summary is provided in Section 2. Section 3 presents the online generator and optimizer for spatial trajectories and some selected results are shown. We summarize in Section 4.

All in all, our paper addresses a topic that has already been studied (previously handled primarily through case-by-case analysis) but contrasts this by presenting a universal language of movement.

2 REVIEW OF THE MATHEMATICAL MODEL

We briefly collect the mathematics in one dimension, see [8] for details.

Let $x(t)$ denote the position at time t , and let N be the order of the motion (e.g., $N = 2$ means acceleration con-

tol). The trajectories under consideration fulfill the following conditions:

- (i) $x(T) = s$, where T is the time horizon and s is the distance of the motion.
- (ii) $x^{(n)}(0) = 0$ for all $0 \leq n \leq N - 1$.
- (iii) $x^{(n)}(T) = 0$ for all $1 \leq n \leq N - 1$.
- (iv) There are bounds b_n (with $0 \leq n \leq N$) such that $|x^{(n)}(t)| \leq b_n$ for all $t \in [0, T]$.
- (v) There is a constant $0 < c \leq b_N$ such that $x^{(n)}(t) \in \{-c, 0, c\}$, i.e. the highest order is piecewise constant and discontinuous.
- (vi) The motion is symmetric (time reversible, e.g., acceleration = deceleration).

While b_n encode the given bounds for each order, let $x_n = \max_{t \in [0, T]} |x^{(n)}(t)|$ denote the actually achieved values (hence, $x_n \leq b_n$) and furthermore let T_n be the duration until x_n is reached, i.e. $T_1 = \min\{t \in [0, T] | x^{(1)}(t) = x_1\}$. Note that the central general equation is

$$T_n = \sum_{k=n}^{N-1} \frac{x_k}{x_{k+1}} \quad (2)$$

for $0 \leq n \leq N - 1$. Figure 1 provides a schematic example to clarify the introduced notation. This equation is valid for all orders, for all number of phases and optimizations of time, velocity, ...

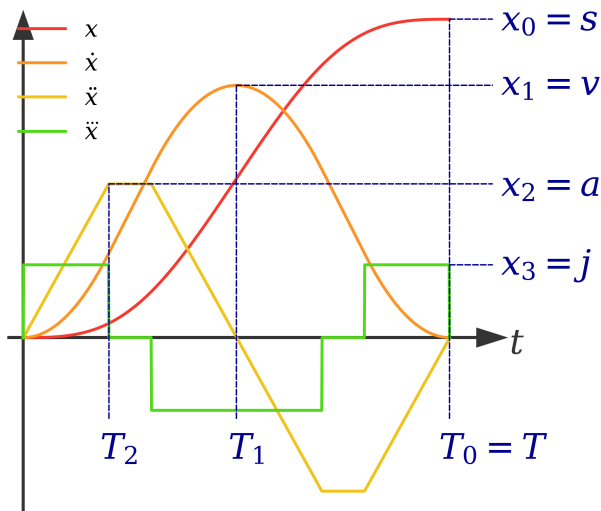


Figure 1: Schematic example of a 3rd-order motion, where the time profiles of all orders are shown. For each order, the achieved (peak) value x_n and the respective time point of achievement T_n is marked.

To become a bit familiar let us discuss jerk-controlled motions ($N = 3$). Introducing $x_0 = s$, $x_1 = v$, $x_2 = a$, $x_3 = j$ as well as $b_0 = s$, $b_1 = v_{\max}$, $b_2 = a_{\max}$, $b_3 = j_{\max}$ we arrive at

$$T = T_0 = \frac{s}{v} + \frac{v}{a} + \frac{a}{j}, \quad (3)$$

$$T_1 = \frac{v}{a} + \frac{a}{j}, \quad (4)$$

$$T_2 = \frac{a}{j} \quad (5)$$

yielding the time $T = T_0$ of the entire motion, the time T_1 for achieving the velocity v , and the time T_2 for achieving the acceleration a (see Fig. 1). To get insights into optimizations and systematics we employ an elegant graphical representation. To this end, we chose units such that $s = j = 1$ (which always can be done by an appropriate transformation) and display $T = \frac{1}{v} + \frac{v}{a} + a$ in a contour plot (see “Diving Fish Diagram” in Fig. 2). This diagram reveals sev-

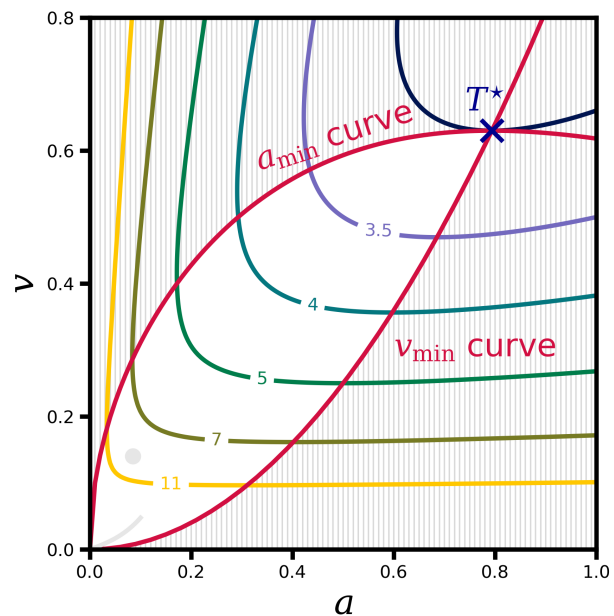


Figure 2: Universal contour plot of the time horizon of jerk-controlled motions as a function of the achieved acceleration a and velocity v (in transformed units). The $v_{\min}$ and $a_{\min}$ curve frame the region of all principally possible motions.

eral points:

- The two red curves frame the region of all principally possible motions, where the condition $T_1 = 2T_2$ determines the $v_{\min}$ curve, and $a_{\min}$ curve is fixed by $T_0 = 2T_1$.
- The constraints $a_{\max}$ and $v_{\max}$ correspond to vertical and horizontal cut-offs, respectively. The loca-

tion of their intersection relative to the fish yields the number of motion phases (see Fig. 3).

- If we seek for the velocity-minimal motion – for instance – we have to move on the given T -contour curve downward until we reach either the $v_{\min}$ curve or the cut-off induced by $a_{\max}$ (analogously for the acceleration-minimal motion).

We refrain here from further graphical interpretation and turn to the algorithmic aspects. The key feature is the observation that conditions as $T_{n-1} = 2T_n$ represent those cases where the peak value x_n is only touched without a cruising phase. The algorithm for determining the time-minimal motion can be therefore very briefly summarized as:

$$\begin{cases} x_n = b_n & \text{if } T_{n-1}(b) > 2T_n(b) \\ \text{else } T_{n-1} = 2T_n & \text{as equation for determining } x_n \end{cases} \quad (6)$$

where $T_n(b)$ means that $b = (b_0, \dots, b_N)$ enters T_n given by (2). In addition $x_0 = b_0 = s$ and $x_N = b_N$ have to be fixed. This allows a direct computation of the motion parameters x_n as well as the characteristic time points T_n . The algorithm for determining the motion with minimal M^{th} order works analogously. The main difference is that x_M exhibits a cruising phase (i.e. $T_{M-1} > 2T_M$) but $x_M < b_M$. Thus, the case $n = M$ is excluded from (6) and replaced $T_0 = T_{\text{given}}$ since a time horizon must be given if a particular derivative should be minimized.

We again refer the interested reader to [8, 9, 10] providing more details. For the direct calculation of all relevant information, we established a real-time capable framework for online computation of spatial trajectories. This is the topic of the following section.

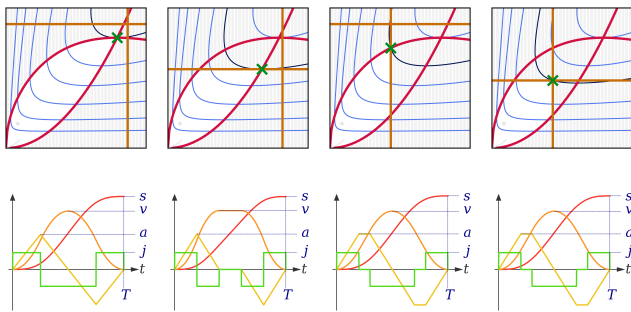


Figure 3: Schematic overview of the four possible cases of a jerk-controlled motion. The time-minimal motion is indicated by the green cross in the Diving Fish Diagrams. First column: neither $a_{\max}$ nor $v_{\max}$ is reached (4 phase motion); second column: only $v_{\max}$ is reached (5 phase motion); third column: only $a_{\max}$ is reached (6 phase motion); fourth column: both $a_{\max}$ and $v_{\max}$ are reached (7 phase motion).

3 ONLINE GENERATOR FOR SPATIAL TRAJECTORIES

3.1 Technical Details

The trajectory calculation framework is a Python-based server architecture which exposes the underlying computational methods via a web API. The API is implemented using FastAPI and can be served using an ASGI server, such as Uvicorn. This separation of the computational backend from its clients enables different applications and user interfaces to access the same trajectory calculation functionality, regardless of their implementation or target platform.

The server provides dedicated HTTP endpoints for the individual computational tasks:

- `/trajectory`: Calculates a trajectory based on a set of time horizons, the highest derivative value, and the desired number of sampling points.
- `/optimized-trajectory`: Performs the corresponding trajectory optimization based on specified motion limits and an optimization mode.
- `/fish-diagram`: Provides the data required for the Diving Fish diagram.
- `/trajectory-3d`: Calculates three-dimensional trajectories by independently processing the motion constraints for the three spatial axes and combining the resulting trajectories into a common three-dimensional path.

The interfaces are intentionally restricted to the parameters required for the respective calculation. Consequently, clients only need to transmit compact request objects containing parameters such as motion limits, time horizons, optimization settings or the desired trajectory resolution. Computationally intensive operations, including trajectory generation and optimization, are performed entirely on the server side. The resulting trajectory data and associated metadata are then returned to the client in a structured form. This design keeps the communication interface lightweight and avoids transferring computational responsibilities to the client.

3.2 Performance

The computational performance was evaluated for trajectory orders ranging from 1 to 7 and resolutions ranging from 100 to 10,000 sampling points. For each configuration, 2 warm-up runs were followed by 10 measured executions, and the median runtime was used as the primary performance metric.

The benchmark separately measures trajectory optimization, time-grid generation, sampling-point generation and trajectory calculation, and also reports the total computation time. Increasing the number of sampling points results in a

direct trade-off between trajectory resolution and computational effort.

The results demonstrate a roughly linear increase in runtime as resolution increases, while maintaining stable performance across higher trajectory orders. The resulting calculation times are short enough to support interactive parameter variation and rapid evaluation of alternative trajectory configurations.

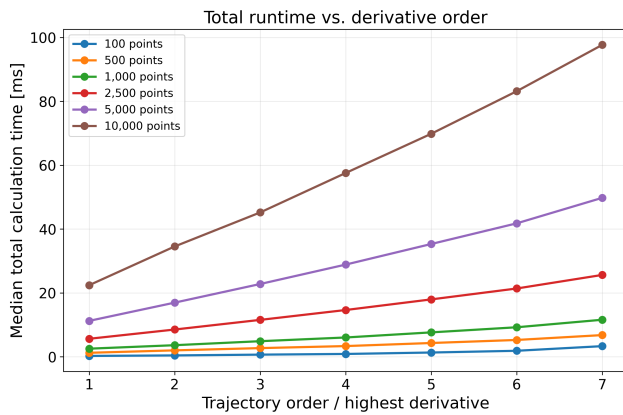


Figure 4: Performance results.

3.3 2D Trajectory Visualizer

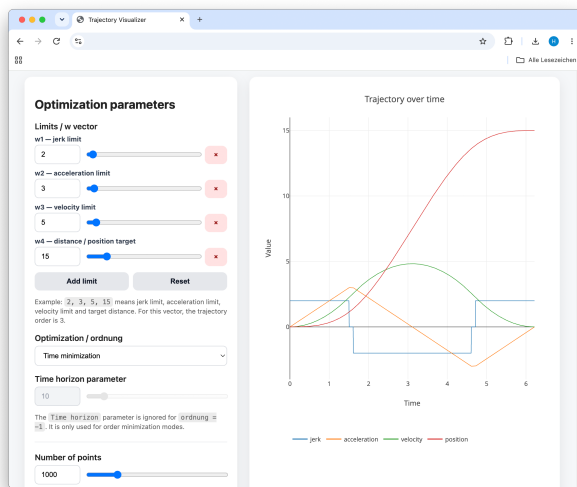


Figure 5: 2D Trajectory Web Visualization.

The two-dimensional Trajectory Visualizer is an interactive, web-based tool for configuring, calculating and inspecting optimized trajectories. Relevant trajectory parameters can be adjusted directly within the dashboard using numerical input fields and synchronized range sliders, enabling the rapid exploration of different parameter combina-

tions. By reducing the effort required to modify and recalculate a trajectory, the front end encourages experimentation with different parameter values and configurations, allowing users to observe their influence on the resulting motion profile. Changes can therefore be evaluated without modifying the underlying implementation or manually issuing API requests.

The calculated trajectory is presented in an interactive Plotly graph, displaying the individual derivatives and the resulting position trajectory over time. Further supporting the detailed analysis of the calculated motion profile, interactive zooming, panning, and value inspection are available.

3.4 3D Trajectory Visualizer

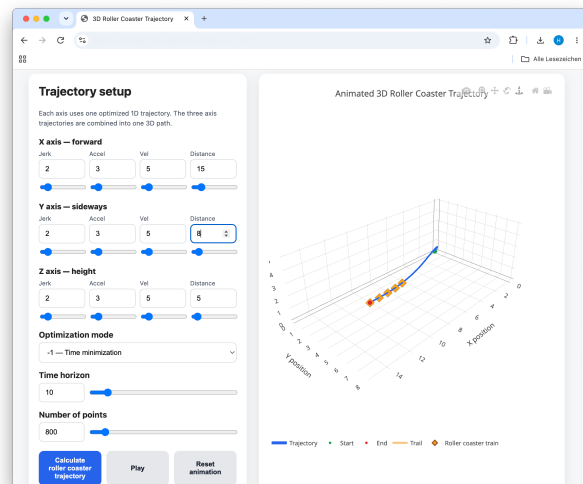


Figure 6: 3D Roller Coaster Web Visualization.

The three-dimensional visualization builds on this concept by combining trajectories for the three spatial axes, which are independently parameterized, into a single 3D path. The corresponding motion is displayed as an animated rollercoaster train travelling along the calculated trajectory, providing an intuitive representation of the motion's temporal progression.

As with the two-dimensional interface, the trajectory parameters can be modified interactively using numerical inputs and range sliders. This direct interaction makes it easier to test alternative spatial configurations and encourages users to explore the effect of different axis-specific parameter settings on the resulting three-dimensional motion. Additionally, characteristic motion vectors such as velocity, acceleration and jerk can be displayed on demand at the current position of the leading vehicle. The interactive 3D view can be rotated and zoomed to allow inspection of the resulting spatial trajectory and its dynamic properties from different perspectives.

3.5 The Diving Fish Diagram

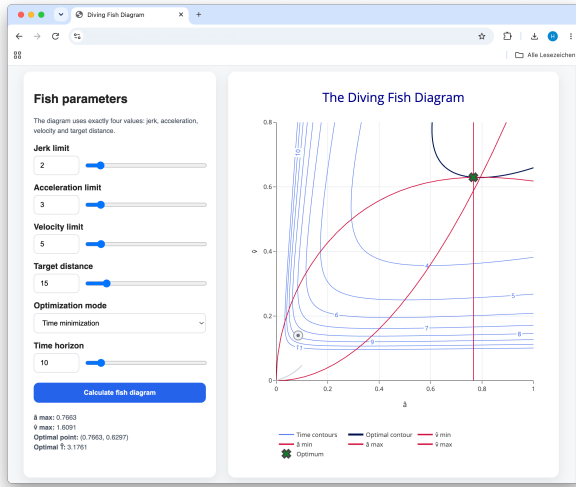


Figure 7: Diving Fish Web Visualization.

We have created a separate interactive web visualization of the Diving Fish Diagram to explore the corresponding normalized parameter space. The interface follows the same interaction concept as the Trajectory Visualizer: relevant input parameters can be modified using numerical fields and synchronized range sliders, updating the visualization accordingly. This immediate visual feedback encourages users to vary the parameters systematically and experiment with different configurations, helping them to develop a more intuitive understanding of how the parameters influence the feasible space and the calculated optimum.

The resulting diagram combines characteristic boundary curves, time contours and the calculated optimum in a single interactive Plotly view, enabling users to examine individual values and regions directly. In addition to the functional visualization, subtle graphical elements, including an animated blinking eye, maintain the distinctive appearance of the diving fish representation, providing a more engaging user interface without impacting the underlying numerical results.

4 SUMMARY AND OUTLOOK

Analytical methods reduce computational efforts, enhance algorithms and reveal deeper insights into the systematics. Starting from the elementary uniform motion ($N = 1$, velocity control) with the one single equation for the time horizon

$$T = \frac{s}{v}, \quad (7)$$

continuing with acceleration-controlled motions ($N = 2$, three phases or trapezoidal motions) with

$$T = \frac{s}{v} + \frac{v}{a} \quad (8)$$

(and furthermore $T_1 = \frac{v}{a}$ giving the time for achieving the velocity) and jerk-controlled motions ($N = 3$, seven-phase motions) with

$$T = \frac{s}{v} + \frac{v}{a} + \frac{a}{j} \quad (9)$$

the underlying principle leads to

$$T_n = \sum_{k=n}^{N-1} \frac{x_k}{x_{k+1}}. \quad (10)$$

Hereby, the condition $T_n = 2T_{n+1}$ is employed to find minimum-time, minimum-velocity, ... motions within the framework of double-S curve trajectories. The use of such trajectories to build spatial trajectories yields a large class of trajectories on the one hand.

However, the symmetry conditions set strong restrictions which should be relaxed in the future. The avenue can be enlightened for instance by generalizing (8) to asymmetric acceleration bound (typically the deceleration is stronger than the acceleration), i.e. $-a_{\min} \leq a(t) \leq a_{\max}$ with $-a_{\min} < 0 < a_{\max}$. Then (8) becomes

$$T = \frac{s}{v} + \frac{v}{2} \left(\frac{1}{a_{\min}} + \frac{1}{a_{\max}} \right). \quad (11)$$

Furthermore, extensions of (10) to rotations are straightforward, and thus, translation-rotation combinations can be used to describe and to analyze trajectories in robotics. However, these visions are left for future work.

REFERENCES

- [1] B. Alpers, "On fast jerk-, acceleration- and velocity-restricted motion functions for online trajectory generation," *Robotics*, vol. 10, no. 1, p. 25, 2021.
- [2] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.
- [3] I. Haouam, "Beyond acceleration: Exploring higher-order time derivatives of position in physical and engineering systems," *WSEAS Transactions on Systems*, vol. 23, p. 354, 2024.
- [4] N. R. D. Eager, A.-M. Pendrill, "Beyond velocity and acceleration: jerk, snap and higher derivatives," *European Journal of Physics*, vol. 37, no. 6, p. 065008, 2016.
- [5] K. D. Nguyen, T.-C. Ng, and I.-M. Chen, "On algorithms for planning s-curve motion profiles," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, p. 11, 2008.

- [6] Y. Fang, C. Gu, Y. Zhao, W. Wang, and X. Guan, "Smooth trajectory generation for industrial machines and robots based on high-order s-curve profiles," *Mechanism and Machine Theory*, vol. 201, p. 105747, 2024.
- [7] M. Boryga and A. Graboś, "Planning of manipulator motion trajectory with higher-degree polynomials use," *Mechanism and Machine Theory*, vol. 44, no. 7, p. 1400, 2009.
- [8] R. Zöllner, "A new perspective on double-s curve motions of higher order and optimal motion planning," *Results in Engineering*, vol. 29, p. 109333, 2026.
- [9] R. Zöllner, F. Schuricht, W. Hofmann, and T. Schmidt, "Semi-analytical approach to trajectory optimization for stacker cranes regarding energy saving," *Cleaner Engineering and Technology*, vol. 30, p. 101141, 2025.
- [10] R. Zöllner, F. Schuricht, T. Schmidt, and W. Hofmann, "Nonsmooth variational approach to trajectory optimization for electrical energy recuperations," *Journal of Optimization Theory and Applications*, vol. 209, no. 2, p. 62, 2026.

Rico Zöllner, is a doctoral researcher at the Chair of Materials Handling and Logistics Engineering at Technical University Dresden. He holds a Dipl.-Math. from the same university. His research interests are applied variational methods for optimization and tribology in materials handling.

Heiner Ludwig, is a research assistant at the Chair of Materials Handling and Logistics Engineering at Technical University Dresden. He completed his degree in Applied Computer Science in 2020. His research interests include the use of language models and semantic data structures in manufacturing and logistics.

Prof. Dr. Thorsten Schmidt, is a professor at Technical University Dresden and owner of the the Chair of Materials Handling and Logistics Engineering since 2008. He received his Dr.-Ing. at Dortmund University. His research interests include factory planning, factory automation and materials handling.

Address: Institute of Material Handling and Industrial Engineering, TU Dresden, 01062 Dresden, GERMANY
 E-Mail: rico.zoellner@tu-dresden.de, heiner.ludwig@tu-dresden.de, thorsten.schmidt@tu-dresden.de,