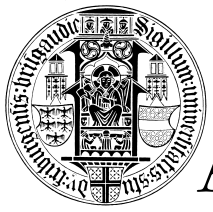


ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Faculty of Applied Sciences
Department of Computer Science
Autonomous Intelligent Systems Lab
Prof. Dr. Wolfram Burgard

Student Project

Planning Motion Trajectories for Mobile Robots Using Splines

Christoph Sprunk
sprunkc@informatik.uni-freiburg.de

October 2008

Supervisor: Dipl. Inf. Boris Lau

Abstract

Motion planning for wheeled mobile robots (WMR) in controlled environments is considered a solved problem. Typical solutions are path planning on a 2D grid and reactive collision avoidance. Active research deals with issues regarding the integration of additional constraints such as dynamics, narrow spaces, or smoothness requirements.

Current approaches to motion planning mainly consider a finite number of actions that can be carried out by the robot at a given time. In this work we present an approach that resorts to a parametric trajectory representation to overcome these limitations. As representation we choose Quintic Bézier Splines. We conduct global, explicit planning for velocities along the robot's trajectory – a prerequisite if smooth kinodynamics along the path are to be included into the planning process, yet mainly unaddressed in current work. Our approach guarantees velocity profiles to respect several kinodynamic constraints and resolves accelerational interdependencies without iteration.

The proposed approach optimizes a curvature continuous trajectory and maintains anytime capability. For actual execution the trajectory is forwarded to a separate feedback controller. The suggested method is able to smoothly update a trajectory to account for unmapped obstacles, moderate errors in localization, or erroneous plan execution. Evaluation of our method on real robots shows its applicability and advantages over a Dynamic Window-based approach.

Contents

1 Introduction

One of the major ambitions in mobile robotics is to enable robots to follow high-level task instructions, i.e., task instructions that do not contain detailed information on the steps to be taken for task fulfillment. To reach this goal, autonomous navigation is a fundamental prerequisite.

The task to navigate autonomously can be divided into the areas of mapping, localization and motion control. The latter is the domain of interest for this work. The localization within a map is herein assumed to be given, but errors have to be accounted for. Furthermore, the problem domain includes the presence of a moderate amount of unmapped obstacles.

The research community has conducted a lot of work in this domain, see for example [15] for an overview. As a result, basic navigation of a mobile robot (i.e., motion from a start to a goal position without any further constraints) is considered a solved problem.

However, practical applications advance to more complex problem domains. Unmanned autonomous vehicles used for heavy cargo load need motion planning to account for their extreme kinodynamics, for example. One observes that with the practical use of mobile robots in increasingly complex scenarios, controllability and predictability of the robot's exact behavior gain importance.

Furthermore, wheeled mobile robots (WMR) are expected to navigate safely and successfully in tight spaces that they share with other moving entities. The ability to plan and predict the state of the robot more precisely with respect to time and position is crucial for navigation in narrow spaces and environments that contain unmapped obstacles for which movements can be anticipated to a certain degree.

In general, increasing weight is put on *how* the robot reaches its goal, planned paths are required to consider costs in domains like smoothness, time of travel, and energy consumption. Car-like robots for instance have strong smoothness requirements if passenger comfort or safety of impulse sensitive payload are accounted for.

Therefore the research area is still active [9, 16, 19, 37, 39] as the classical approaches to mobile robot navigation and motion reach their limits with respect to the requirements stated above.

Limits of existing approaches Reactive approaches to mobile robot navigation like the Dynamic Window [8] and Potential Field methods [12] are unsuitable for application to the more ambitious problem domain. The reactive nature implies a limited path predictability. In addition to that, the Dynamic Window suffers from overshooting [32] and for Potential Field methods local minima are a major issue [14].

The well known 2D A* algorithm and its extension D* provide cost optimal paths on gridmaps. These paths however consist of straight line segments and ignore kinematic constraints. To exactly follow these paths the robot is required to exclusively execute either strictly translational or rotational movement (i.e., to drive straight or turn on the spot). While the action set to move to one of the eight neighbor cells in a 2D

grid is perfectly suited for costs only considering the path's length, it is insufficient for applications aiming to optimize more complex measures like time of travel, energy consumption, or path smoothness. Because of this and due to not accounting for kinematic constraints, the path's optimality is lost when it is taken to the real problem domain.

To cope with the problem of straight line paths and ignored kinematics, applications of A* to higher dimensional state spaces have been suggested. The state space is for instance augmented by the robot's heading and a two-dimensional velocity component. In order to keep the algorithm computationally tractable, heavy discretization has to be applied to the state space and the action set has a limited size to control the branching factor of the search tree. Stachniss and Burgard [32] present an approach where a higher dimensional A* search is guided by a lower dimensional one to further constrain the search space.

The paths computed by these approaches are in general continuous in the robot's heading but still have discontinuities in curvature. This means that for instance car-like robots have to cope with a non-continuous steering angle. While the actual motion of the vehicle is smoothened by inertia and low level controllers, this induces deviations from the assumed path. Only accounting for these effects during planning can improve the predictability of the robot's behavior.

Precise manipulator control There is a domain however, where exact, time dependent motion planning and execution has been reached: the area of multi degree of freedom actuators and manipulators, mostly in the form of robotic arms [30]. While these manipulators in general are more capable of executing commands with high precision than WMR, it is believed that the overall accuracy in trajectory execution is also due to the small feedback loop involved.

Therefore, recent work tries to transfer some of the precision potential to the domain of motion planning for WMR by decoupling trajectory planning and execution. A separate representation of the trajectory is maintained and executed by a feedback controller. This way the feedback loop is small and one can also benefit from the results and progress made in the field of control theory [13, 18]. Furthermore, separation of trajectory and execution of the latter adds a level of abstraction that provides more flexibility, e.g., easy change of the used controller.

Recent developments By the introduction of a separate and smaller feedback loop, (part of) the current work treats a design disadvantage of previously existing motion planning methods: their feedback loop includes the global path planner with the consequence of negative effects on stability. To generate a motion command if the robot has deviated from the previously computed path it is necessary to re-execute the high-level planning method. This takes time and can therefore only be done at a relatively limited frequency. Note that while this limitation amongst others applies to the A* based approaches, methods exist that precompute actions for all positions, e.g., Value Iteration based planners [35].

In the majority of recent work, motion planning is approached by searching in a space that consists of suitable motion primitives that are stitched together. The major challenge for these approaches is to keep the number of primitives within a limit that yields computationally tractable search trees.

An aspect not covered by most of the current approaches is the necessity to globally plan for the velocities along the path: thereby it is possible to slow down early enough when approaching a curve and to account for the effects that changes in the curvature have on kinodynamics.

A detailed discussion of the related work will be given in the next chapter. First, we want to motivate the use of splines for locomotion.

Spline based locomotion planning As means of representation for the trajectories we choose splines (piecewise polynomial parametric curves) because they offer the following favorable properties: splines constitute a compact representation of smooth paths. This allows the generation of paths that can be followed by a robot with high precision. Furthermore, discretization is moved to another dimension. The path between start and goal can be described with arbitrary precision, the actions are not limited by any fixed number. This is gained by only considering curves of a parameterized family for robot movement but does not overconstrain the set of possible paths, which constitutes a major difference to the search based approaches.

Contributions We shortly outline the system we propose for WMR motion planning. Our system takes as inputs a map of the environment and sparse waypoints that describe a collision free line path from the current to the goal position. Based on Quintic Bézier Splines it then creates an initial trajectory along the waypoints and continues to optimize it as long as planning time is available. During the optimization velocities along the curve are explicitly planned for to obtain a 2D path that together with its velocity profile optimizes a cost function, such as the time of travel for example. When planning velocities, kinodynamics of the robot are accounted for.

The output of the proposed system is a time-parameterized trajectory together with its first and second derivatives that can be used by a controller to determine location, speed, and acceleration along the trajectory for every point in time.

As will be shown, the proposed system has anytime capability and we also present a way to account for unmapped obstacles. Finally, it has to be noted that we evaluated our approach with experiments on real robots.

Organization The remainder of this work is organized as follows. Chapter 2 gives an overview about further related work in the domain of spline based motion planning. Chapter 3 provides the theoretical foundations for the suggested approach which is described in detail in Chapter 4. Experiments and corresponding results are presented in Chapter 5. Chapter 6 finally discusses the proposed system and concludes.

2 Related Work

In this section the proposed system will be compared to related work in the area of spline based motion planning for mobile robots.

There exists a variety of motion planning approaches that use splines to represent trajectories. They all concentrate on several aspects of the problem but none accounts for all requirements that we consider important: a path that can be exactly followed by the robot (i.e., curvature continuity), explicit planning for velocities with consideration of kinodynamics, anytime capability, an optimization not prone to local optima, treatment of unmapped obstacles, and an evaluation of the system on a real robot.

Hwang et al. [11] use Bézier Splines as trajectory model. They present an approach that extracts significant points from a touch screen based path input and uses these as pass-through points for a Cubic Bézier Spline. While unmapped obstacles are accounted for, the paths generated by this approach are discontinuous in curvature. The work of Hwang et al. is used by Mandel and Frese [21] for autonomous wheelchair navigation, in their work they provide a non-simulated evaluation of the approach. Our approach relies on Bézier Splines as well but uses a higher degree of these polynomials to overcome the curvature discontinuities.

Another approach to wheelchair navigation is presented by Gulati and Kuipers [10]. They emphasize the need for smooth and comfortable motion and choose B-Splines as trajectory representation. However, the approach only applies to navigation in predefined situations without unmapped obstacles.

B-Splines are also used by Shiller and Gwo [31]. In their work they treat the motion planning task extended to 3D with the help of B-Spline patches and curves. The evaluation of their approach, that accounts for kinodynamic constraints, is based on simulation.

The principle of optimizing a spline based trajectory can be found in the work of Magid et al. [20]. Here, an initially straight-line shaped path is optimized with respect to a cost function by the Nelder Mead Simplex method. The difference to the approach we suggest is that the initial path is not obstacle free and that local minima are a problem for this method, as the simulative evaluation shows.

Sahraei et al. [29] rely on a Voronoi graph to provide information for the generation of a Bézier Curve based path. The path is not continuous in curvature and the approach ignores unmapped obstacles, but it has been evaluated by application on a robot.

Another spline related approach is suggested by Thompson and Kagami [34]. Here, curvature is a polynomial function of the current traveled distance. With these curvature polynomials Thompson and Kagami define obstacle avoiding paths with ad-hoc speed profiles.

An approach that has received extensive evaluation is the work of Arras et al. [1]. It uses Elastic Bands [26] as part of the navigation function for an extension of the Dynamic Window approach and showed reliable performance on an overall traveled distance of more than 3 000 kilometers. With the use of the Dynamic Window, the approach is subject to the aforementioned drawbacks, but the combination with Elastic

Bands as a form of trajectory representation can be interpreted as a first step to separating the trajectory generation from its execution. Here, the Dynamic Window can be seen as controller that is fed the trajectory through the navigation function.

Most of the current work separates trajectory generation and execution even more strongly and employs real controllers, in the sense that these try to exactly follow the trajectory instead of just being guided by it. This considerably improves and simplifies the predictability of the robot's behavior.

A lot of examples for such approaches can be found in the work triggered by the DARPA Grand and Urban Challenge [3, 16, 36, 39]. All these systems have undergone a lot of practical testing and show the small feedback loop that results from decoupling trajectory generation and execution.

Likhachev and Ferguson [16] stitch together actions from a precalculated set, the action space is searched by anytime dynamic A*. The curvature discontinuities in the resulting paths are claimed to be non-critical and handled by reducing speed at high curvatures. A step in the direction of velocity planning is done by a lookahead mechanism that ensures smooth deceleration upon a change of vehicle movement direction (forwards vs. backwards).

Curvature continuous paths are generated by Ziegler et al. [39] through incorporation of the steering angle into the state space. The path is constructed via an A* search, the state space does not contain any velocity information. Path execution is handled by two separate controllers, one for steering angle and velocity each.

An example for searching the space of stitched paths with Rapidly-exploring Random Trees is given by the work of Macek et al. [19] who emphasize safety and collision avoidance. Here, translational velocity is part of the state space, and to treat the costly search an approach called "Partial Motion Planning" is applied.

There are approaches that, like our suggestion, do not have to cope with limiting the action set, as they rely on parametric trajectory representations [3, 36].

Thrun et al. [36] generate a Cubic Bézier Spline from GPS data describing the course of the road. Contrary to our approach the vehicle does not exactly follow the spline but travels at an intended varying lateral offset instead. Offset changes are employed for obstacle avoidance and realized through a discrete set of maneuvers resembling lane changes. While in this approach a spline is used to smoothly interpolate between given waypoints, in our approach optimization of the spline is part of the planning process.

The approach of Connors and Elkaim [3, 4] also uses a parametric trajectory representation. Paths are generated by starting with straight-line segments between waypoints. Control points are then moved out of obstacles repeatedly until a collision free path is generated. A drawback is that a solution cannot be guaranteed by this method.

Summary As shown in this literature review, the research community investigates the benefits of separating the trajectory planning from its feedback execution, yielding a smaller feedback loop. One also finds that smooth, curvature continuous trajectories are possible to employ, yet not a priority for the successfully tested approaches, e.g., the unmanned DARPA challenge vehicles. Advantages of a parameterized trajectory representation over one that is stitched together from a finite set of precomputed parts have emerged. Furthermore, we have seen that little is done in the domain of explicit velocity planning and optimization.

3 Spline Based Trajectories

For our system we will use splines as means to determine the exact 2D path for the robot. The actual trajectory that controls the exact speed and acceleration at every single 2D point will then consist of this 2D path and a separately created velocity profile, thus generating a 3D representation over space and time.

Without reducing the scope of the approach, we only consider forward motion of the robot along the 2D path. Therefore, the non-smooth 2D path shapes that correspond to forward motion interrupted by backward motion, e.g., to a 3-point turn, are neither needed nor desired for our application.

3.1 2D Paths Using Splines

For this work, we define a spline as a *piecewise polynomial parametric curve* in the two-dimensional plane $Q(t) \in \mathbb{R}^2$. All segments of the spline are required to be polynomials of the same degree n. A central aspect of splines is the fulfillment of certain *continuity constraints* (see for instance [6, p. 480]).

Parametric continuity A parametric curve is said to have n-th degree parametric continuity in parameter t, if its n-th derivative $\frac{\mathrm{d}^n}{\mathrm{d}t^n}Q(t)$ is continuous. It is then also called C^n continuous.

Geometric continuity n-th degree geometric continuity does not require the left-hand limit and right-hand limit of the n-th derivative $\frac{\mathrm{d}^n}{\mathrm{d}t^n}Q(t)$ to be equal but demands the direction of these vectors to match, i.e., they have to be scalar multiplies of each other. Note that geometric continuity is trivial for one-dimensional parametric curves (e.g., $Q(t) \in \mathbb{R}$).

For points on a polynomial these continuity constraints are always fulfilled (as derivatives of arbitrary degree are continuous), so the join points of the polynomial segments are the critical points. In general geometric continuity is a consequence of parametric continuity of the same degree, unless the particular derivative yields the zero vector. Then situations can arise in which the left-hand limit points in a different direction than the right-hand limit although both evaluate to the zero vector at the join point.

3.1.1 Desired Properties

To ensure that the robot is able to follow a spline based reference path and that the path can be efficiently planned, the splines are required to have several properties. The properties necessary for successful employment of splines in the domain of trajectories are discussed in the following.

3.1.1.1 Drivability

Several properties are necessary to guarantee that a wheeled robot can exactly follow a trajectory defined by a spline. For the most part, these are continuity properties.

Continuity in heading In addition to the self-evident C^0 continuity it makes sense to require the spline to be G^1 or C^1 continuous, which has the effect that the heading of the robot changes continuously. If this wasn't the case the robot would have to be able to execute infinite rotational accelerations.

Continuity in curvature Trajectories intended to be driven by wheeled robots should also be continuous in their curvature. A discontinuity in curvature would require infinitely high accelerations for robots equipped with synchro drive, differential drive and even for holonomic robots. For car like robots it would correspond to a discontinuity in steering angle. Apart from this, even if a robot can approximately cope with these accelerations, curvature discontinuities imply strong, abrupt forces to the robot itself as well as to potentially fragile payload. With unfavorable consequences such as possible skidding or damaged payload we want to prohibit curvature discontinuities in our trajectories.

The curvature c is the inverse of a curve's radius and defined in [33, Sect. 13.3] as follows:

$$c = \left| \frac{\mathrm{d}T}{\mathrm{d}s} \right|,$$

where T is the tangent vector function to the curve and s the curve's arc length. [33, Sect. 13.3, Theorem 10] allows a formulation in an arbitrary parameterization of the curve Q:

$$c(t) = \frac{Q'(t) \times Q''(t)}{|Q'(t)|^3}. \tag{3.1}$$

Note that, in difference to the original definition, we drop the application of the absolute value in the numerator to allow a distinction of the curvature's direction (in relation to an increasing curve parameter t). When moving forward, positive values for curvature correspond to turning left while negative values indicate a right turn.

As can be seen in Eq. (3.1), control over the first and second derivatives of a parametric curve provides control over its curvature. Choosing first and second derivative suitable for C^2 continuity implies curvature continuity of the spline.

The requirements for a robot driveable spline are summarized and guaranteed with the demand of C^2 continuity. Note however, that curvature continuity can also be reached when falling back to G^1 continuity. For this, the second derivative has to meet certain constraints, for details refer to Section 4.3.4.2.

First and second derivative at start To smoothly update a trajectory that is already being carried out by a robot, one needs to be able to generate trajectories for a moving robot. More specifically, it must be possible to match the first and second derivatives to the required heading and orientation/steering angle of the robot.

Property	Hermite	Cubic Bézier	Catmull-Rom	B-Spline
stitching	manual	manual	inherent	inherent
C^1 continuity	✓	✓	✓	✓
C^2 continuity	-	-	-	✓
strong correlation	-	✓	✓	-
localism	✓	✓	✓	✓
freely set 1^{st} deriv. start	✓	✓	✓	✓
freely set 2^{nd} deriv. start	-	-	-	-

Table 3.1: Key properties of cubic spline families presented in Appendix A. None of them has all of the desired properties, therefore, higher-order splines are considered.

3.1.1.2 Planning

In addition to the properties that enable a wheeled robot to follow a trajectory defined by a spline we also require some properties that will facilitate the process of planning such a spline.

Localism The first property is called *localism*. Localism is a property that becomes important when modifying one segment of a multi-segment spline. We will say that a spline has localism if changes made at a single segment of the spline only affect a bounded neighborhood of this particular segment and do not result in changes for all segments of the spline. This property provides the possibility to locally change the shape of a spline (for instance to circumvent an obstacle) without the need to reconsider its global shape. This property can also translate into a computational gain: a large part of the spline's parameters will remain the same after a local change, so one can expect the computation for the update to be less costly than for a global change of the spline's shape.

Correlation between shape and parameters The second property that will facilitate planning trajectories with splines is the extent to which the parameters of a spline correspond to its shape on an intuitive level. When planning a trajectory through an environment we face the task to define a spline that passes through certain points and stays clear of certain areas. This task is facilitated a lot if there are relatively direct correlations between the spline's shape and its parameters. Otherwise involved inverse math had to be employed to retrieve spline instances with given shape properties.

3.1.2 Choosing a Spline Type

An introduction to splines and an examination of specific families can be found in Appendix A. The discussion is summarized by Tab. 3.1 which lists the key properties of the presented cubic spline families.

Hermite Splines and Cubic Bézier Splines are defined on a per-segment basis, i.e., to generate splines that consist of multiple segments (multiple polynomials) manual stitching is required. In contrast, Catmull-Rom Splines and B-Splines are defined in a way that already incorporates the stitching process. As can be seen in the table,

no spline family possesses all required properties. While B-Splines are the only family to provide C^2 continuity they lack a strong and intuitive correlation between shape and control points, which is on the other hand provided by Bézier and Catmull-Rom Splines.

Note that when giving up on localism during the stitching process in Cubic Bézier Splines and Hermite Splines it is possible to connect them in a C^2 continuous fashion, however, the resulting splines are highly unstable and respond with heavy oscillation to single parameter changes. Furthermore, for all spline families in Tab. 3.1 setting the start point's second derivative influences the remaining derivatives of the segment.

Since all of the splines discussed so far consist of cubic polynomials, they have the same expressive power. To explain why none of the presented families is able to fulfill all our requirements and that this is impossible for cubic polynomials, we closer examine the degrees of freedom available.

3.1.2.1 Degrees of Freedom

A cubic spline consists out of segments S_i that are cubic polynomials, generally defined as

$$S_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i, \quad t \in [0, 1]. \tag{3.2}$$

Each segment therefore introduces four degrees of freedom. Assuming our spline consists of m segments $S_i,\ i \in [0, m-1]$, there are $4m$ degrees of freedom.

First, C^0 continuity is demanded, i.e.,

$$S_i(1) = S_{i+1}(0), \quad i \in [0, m-2]. \tag{3.3}$$

Analogously we demand C^1 and C^2 continuity:

$$S_i'(1) = S_{i+1}'(0), \quad i \in [0, m-2], \tag{3.4}$$

$$S_i''(1) = S_{i+1}''(0), \quad i \in [0, m-2]. \tag{3.5}$$

Each of the continuity levels states $m-1$ conditions, reducing the degrees of freedom still available to $4m - 3(m-1) = m+3$. If additionally the start and end points of the segments are demanded to have specific values (e.g., to match a control point)

$$S_0(0) = p_0, \quad S_i(1) = p_{i+1},\ i \in [0, m-1], \tag{3.6}$$

this results in $m+1$ additional constraints, leaving 2 degrees of freedom.

Therefore a spline should be possible that consists of cubic polynomials, passes through control points and is C^2 continuous. While this indeed is possible, the task becomes unachievable when demanding localism at the same time. Localism requires the individual segments to be independent in a way that changing the constraints for one segment will only affect control points of segments in the direct neighborhood. This is impossible when demanding C^2 continuity for splines consisting of cubic polynomials, as will be shown.

To maintain localism, a change at segment S_i may only propagate to the segments S_{i-1} and S_{i+1}. With cubic polynomials, four degrees of freedom are at disposal for each segment. This is enough to independently set start and end point as well as the tangents at these locations. As thereby all degrees of freedom are used, the second derivative depends on the values for start/end point and tangents. Therefore a change

in any of these four constraints will immediately result in a change of the second derivative at both, the start and end point of the segment. To maintain C^2 continuity changes are necessary to the following second derivatives: $S''_{i+1}(0)$ and $S''_{i-1}(1)$. As these cannot be set independently from the other parameters, this introduces shape changes in S_{i-1}, S_{i+1} that themselves lead to changes of the second derivatives $S''_{i-1}(0)$ and $S''_{i+1}(1)$. These changes now propagate across the whole spline according to the same mechanism.

In conclusion, splines consisting of cubic segments can never provide all of the required properties at the same time.

3.1.2.2 Higher Order Splines

The degree of the polynomials forming the spline has to be raised for the spline to meet all our requirements. While quintic polynomials will turn out to be the solution, quartic polynomials do not suffice: with the additional degree of freedom that a polynomial of degree four introduces it is possible to set the second derivative independently from start and end point and respective tangents. However, the second derivatives at start and end point depend on each other. If it was now necessary to change the second derivative at one single point in the spline this change would propagate over the spline in the same way as before, only that it is not mediated by the parameters for start and end points and their respective tangents this time. Due to these instabilities and because changing the second derivative is for instance necessary when a start curvature has to be matched for a moving robot, quartic polynomials are discarded as segment type.

As mentioned above, quintic polynomials are the segment type of choice to obtain both, localism and C^2 continuity. The additional two degrees of freedom introduced compared to cubic polynomials allow to set the second derivative at start and end point independently from each other. As the Bézier formulation of splines additionally to the demanded properties provides the convex hull property (Appendix A.3), Quintic Bézier Splines will be introduced as the final choice of spline type for trajectory planning.

3.1.2.3 Quintic Bézier Splines

Intuition Quintic Bézier Splines follow the same intuition as Cubic Bézier Splines. The first and last control point constitute the start and end point of a segment. The tangents at start and end are proportional to difference vectors of two particular control points with the first and last control point, respectively. The second derivatives at beginning and end of a segment are defined by the corresponding tangent and a difference vector induced by an additional control point each.

Definition Cubic Bézier Splines can be formulated in the basis spline notation with the help of the Bernstein polynomials of third degree (refer to Eq. (A.19)). Taking the

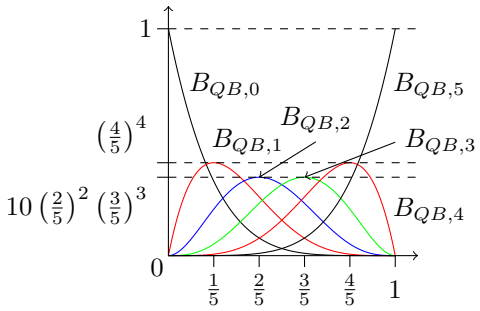


Figure 3.1: The basis splines of a Quintic Bézier Spline's segment, also known as the Bernstein Polynomials of fifth degree. They are defined in Eq. (3.7).

Bernstein polynomials to degree five, one obtains the *Quintic Bézier basis splines*

$$B_{QB} = \begin{pmatrix} B_{QB,0} \\ B_{QB,1} \\ B_{QB,2} \\ B_{QB,3} \\ B_{QB,4} \\ B_{QB,5} \end{pmatrix}^t = \begin{pmatrix} (1-t)^5 \\ 5(1-t)^4 t \\ 10(1-t)^3 t^2 \\ 10(1-t)^2 t^3 \\ 5(1-t)t^4 \\ t^5 \end{pmatrix}^t, \tag{3.7}$$

which Fig. 3.1 provides a visualization for. Using six control points, a Quintic Bézier segment is defined as

$$\begin{aligned} S(t) = (1-t)^5 P_0 + 5(1-t)^4 t P_1 + 10(1-t)^3 t^2 P_2 \\ + 10(1-t)^2 t^3 P_3 + 5(1-t)t^4 P_4 + t^5 P_5. \end{aligned} \tag{3.8}$$

One can verify that the convex hull property as introduced with the Cubic Bézier Splines (see Appendix A.3) still holds with Bernstein polynomials of fifth degree as basis splines. Factoring out the powers of the parameter yields formulation with the *Quintic Bézier coefficient matrix*:

$$\begin{aligned} S(t) &= T \cdot C_{QB} \\ &= \begin{pmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} -P_0 + 5P_1 - 10P_2 + 10P_3 - 5P_4 + P_5 \\ 5P_0 - 20P_1 + 30P_2 - 20P_3 + 5P_4 \\ -10P_0 + 30P_1 - 30P_2 + 10P_3 \\ 10P_0 - 20P_1 + 10P_2 \\ -5P_0 + 5P_1 \\ P_0 \end{pmatrix}. \end{aligned} \tag{3.9}$$

Derivatives Derivation of Eq. (3.8) yields

$$S'(0) = 5(P_1 - P_0), \qquad S'(1) = 5(P_5 - P_4) \tag{3.10}$$

$$S''(0) = 20(P_0 - 2P_1 + P_2), \qquad S''(1) = 20(P_3 - 2P_4 + P_5). \tag{3.11}$$

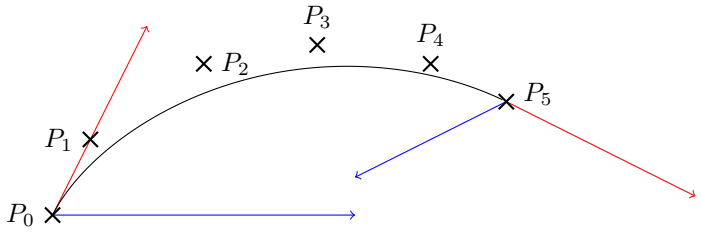


Figure 3.2: Quintic Bézier Spline segment. The tangents are depicted in red, for display the blue second derivatives in the figure have been shortened in their magnitude by factor 5, the tangents by factor 2, respectively. Please refer to Eqs. (3.10) and (3.11) for correlation between the control points and the derivatives.

Fig. 3.2 depicts a Quintic Bézier Spline segment and its tangents at the start and end point.

During trajectory planning the task arises to set specific tangents and second derivatives at start and end point, respectively. A short inverse calculation shows how to set the control points to retrieve a requested first derivative t_s and second derivative a_s at the start point and t_e, a_e at the end point:

$$P_1 = \frac{1}{5}t_s + P_0, \qquad P_4 = P_5 - \frac{1}{5}t_e \tag{3.12}$$

$$P_2 = \frac{1}{20}a_s + 2P_1 - P_0, \qquad P_3 = \frac{1}{20}a_e + 2P_4 - P_5. \tag{3.13}$$

We now have the means to define a two-dimensional trajectory for the robot to drive on. In the remainder of this chapter we address the issue of determining the robot's speed along the trajectory.

3.1.3 Separating Spline and Velocity Profile

As introduced in Section 1, a feedback controller is used to let the robot follow the trajectory defined by the spline. For precise error feedback we provide to the controller the 2D position and the velocities and accelerations of the robot at any point along the trajectory.

An intuitive approach is to use the spline's first and second derivatives to determine translational velocity and acceleration. For this the spline, which is a function of its internal parameter, has to be reparameterized to be a function over time. A naïve approach to this is to linearly map time to the internal spline parameter. This approach has a serious drawback: Fulfillment of kinodynamic constraints hast to be established by tuning *one* scalar factor. If accelerational limits are exceeded at a sharp curve for instance, there is no option other than to change the mapping to yield a slower traversal of the *whole* trajectory.

Because we want closer control over velocities and thereby constraint fulfillment, we split the spline and the velocity profile. A separate structure will be maintained to

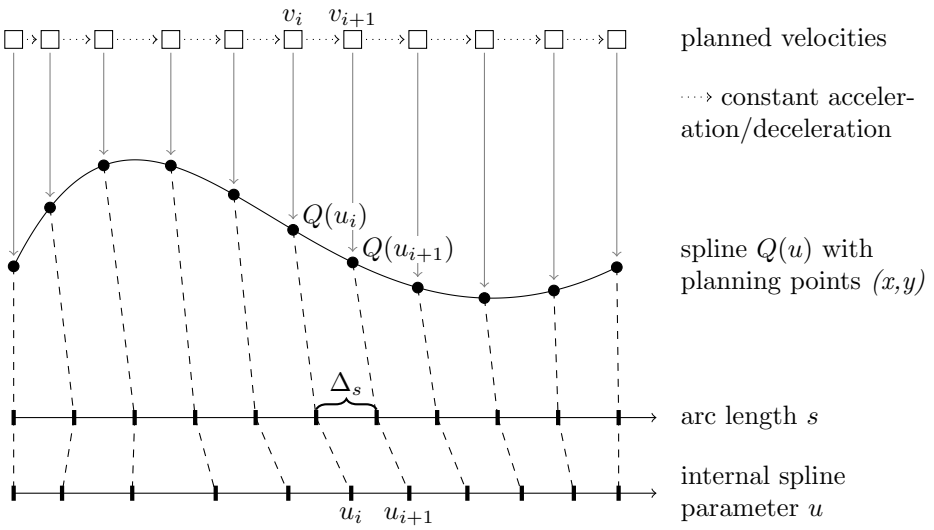


Figure 3.3: Planning points for the velocity profile. Placed at equidistant arc length Δ_s along the curve, velocities are planned for these points and constant acceleration/deceleration is assumed between them. Dashed lines connect planning points to their corresponding values for arc length and internal spline parameter u. In practice, planning points are more densely distributed.

determine which speed and acceleration the robot should have at which point of the trajectory. By doing so we loose the compact and intuitive character of the spline induced velocity profile. The critical advantage however is, that we gain strong control over the velocities and therefore can actively assure compliance with the robot's dynamic constraints.

The next section introduces a method to compute such a separate velocity profile.

3.2 Numerical Velocity Profile

This section describes a way to generate the velocity profile that corresponds to the fastest traversal of the trajectory while obeying constraints. The method applies to *any* parametric curve for which the curve function itself and its first and second derivatives are given.

3.2.1 Overview

3.2.1.1 Planning for Discrete Points

A velocity profile consists of discrete planning points $p_i = Q(u_i)$ distributed along the trajectory $Q(u)$ and planned translational velocities v_i assigned to them. As visualized by Fig. 3.3, the planning points are distributed equidistant with respect to arc length, two adjacent planning points p_i, p_{i+1} inscribe an interval on the trajectory with arc

length Δ_s. For this way of planning point placement, the spline's arc length has to be determined by numerical integration.

Between the planning points, the robot is assumed to move with constant acceleration or deceleration. Once the velocities v_i, v_{i+1} are planned for an interval bound by the planning points p_i, p_{i+1}, the assumption of constant accelerations and the interval's arc length Δ_s can be used to determine the time the robot needs to travel through the interval. Then it is possible to associate with each planning point p_i the time t_i at which the robot will arrive at the planning point. As a consequence, a velocity profile immediately provides the total time of travel for a trajectory.

Association of both, a time t_i and the internal spline parameter u_i with planning points $p_i = Q(u_i)$ constitutes the basis for a reparameterization of the spline to be a function over time: as in practice the planning points are deployed more densely than shown in Fig. 3.3, appropriate interpolation can be used to obtain this reparameterization.

3.2.1.2 Constraint Respecting Profile

To ensure that the velocity profile respects the kinodynamic constraints of the robot, these constraints have to be transformed into constraints on the translational velocities at the planning points. We distinguish two kinds of constraints, *isolated* and *accelerational* constraints.

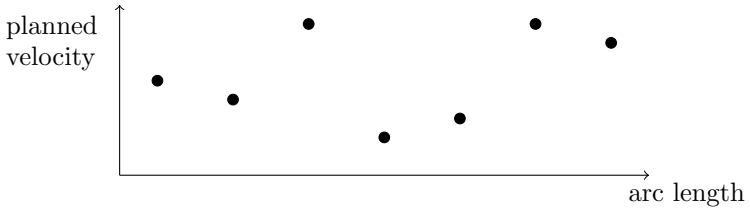
Isolated constraints at a planning point are independent from the translational velocities set at neighboring planning points. An example is the maximum centripetal force allowed for the robot: it depends solely on the curvature and translational velocity at the particular planning point, velocities at neighboring planning points do not affect it.

Accelerational constraints in contrast depend on the velocities set for neighboring planning points. These constraints ensure for instance that the translational velocity at planning point p_{i+1} stays within the bounds determined by maximum translational acceleration or deceleration and the translational velocity at p_i.

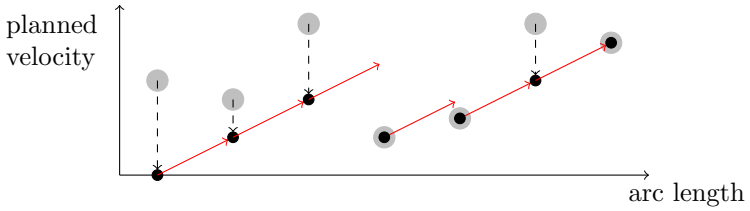
Profile generation To generate a constraint respecting velocity profile, in a first step the maximum translational velocities that respect the isolated constraints are computed for each planning point. These maximum velocities form an initial velocity profile. Fig. 3.4 provides illustrations for this and the upcoming steps, the initial velocity profile is shown by Fig. 3.4a.

The next step assures that the velocity at any planning point p_i can be reached by accelerating from the previous planning point p_{i-1}. We also refer to this step as establishing *forward consistency* for the profile. The process is started by setting the start velocity for the profile. In the example in Fig. 3.4b it is set to zero. Then the maximum velocity at the next planning point is computed (red arrow) and the planned velocity is corrected, if necessary. Now, forward consistency has been established between the first two planning points. The procedure is repeated starting with the second planning point and forward consistency is propagated through the profile this way. Note the velocities that remain unchanged in Fig. 3.4b because the planned velocity is below the one reachable through maximum acceleration.

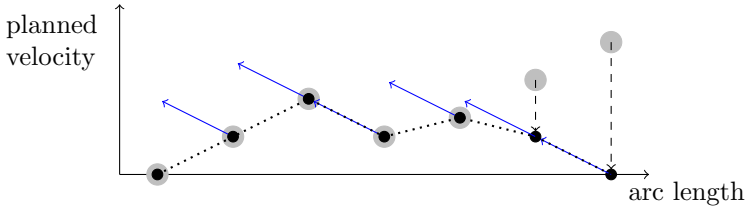
The last phase is visualized by Fig. 3.4c and establishes *backward consistency*. The last planning point's velocity is set to the desired end speed, i.e., zero in this example.



(a) Isolated constraints: in a first step the maximum translational velocity for each planning point has been computed. These velocities are independent from the ones at neighbor points.



(b) Establishing forward consistency by decreasing planned velocities, so that they meet acceleration constraints (red arrows). Maximum velocity at a planning point depends on the one set for the previous point in time. Constraints are being propagated beginning at the start point (left), whose velocity is set to zero in this example.



(c) Establishing backward consistency: The spline is traversed in reverse direction. The velocity at end point (right) is set to zero, now preceding velocities are decreased to meet deceleration constraints (blue arrows). Maximum velocity at a point depends on the one set for the next point in time. The velocity profile is obtained by interpolating between the velocities at the planning points (black dotted lines).

Figure 3.4: A simplified, schematic depiction of velocity profile generation. The discrete planning points for velocities have been set equidistant regarding arc length. Bigger gray dots stand for values inherited from the previous step. This example accounts for one accelerational constraint only.

Now it is ensured that the velocity at each planning point p_i can be reached from the preceding one p_{i-1} with admissible deceleration. Starting at the end of the curve, the consistency is now being propagated in reversed curve direction. The resulting velocity profile is shown with dotted lines in Fig. 3.4c, the velocity at any point on the curve can be retrieved by appropriate interpolation between planning points, as we assume constant accelerations.

Note that for multiple accelerational constraints establishing forward and backward consistency gets more complex, this situation will be accounted for below and is neglected in this overview and the example in Fig. 3.4.

3.2.2 Respected constraints

Our method for velocity profile generation respects the following constraints. The isolated constraints obeyed are maximum translational and rotational velocities, a maximum centripetal force and a maximum translational velocity that depends on the distance to the closest obstacle. As accelerational constraints both, maximum translational and rotational acceleration are respected. The symbols for these quantities can be found in Tab. 3.2.

The remainder of this section will present the generation of a velocity profile in detail.

3.2.3 Definitions

We start the presentation of details on velocity profile generation with the introduction of some notational conventions and definitions. Tab. 3.2 summarizes the ones that will be used throughout this section. The maximum values for rotational velocity and accelerations are given by absolute values, we do not distinguish different maxima for rotation to the left and right, respectively. Furthermore, we assume translational acceleration and deceleration to be symmetric which is expressed by a common upper bound $a_{t_{max}}$ given as an absolute value, too.

We will use the convention that a quantity's symbol indexed by i stands for its value at p_i and that we will denote the maximum translational velocity that a constraint on the quantity $\bullet$ imposes as $v_{max}|\bullet$. The maximum translational velocity due to maximum centripetal force f_{max} is for instance written as $v_{max}|f$.

3.2.4 Movement Equations

As mentioned above, we assume constant accelerations between planning points. Therefore, the following equations should hold within the planning interval bounded by p_{i-1} and p_i:

$$s(t) = \frac{1}{2}a_t t^2 + v_{i-1}t, \quad t \in [0, \Delta_t], \quad \Delta_t := t_i - t_{i-1} \tag{3.14}$$

$$a_t = \frac{v_i - v_{i-1}}{\Delta_t} \tag{3.15}$$

$$\Delta_s = s(\Delta_t) \tag{3.16}$$

$$a_\omega = \frac{\omega_i - \omega_{i-1}}{\Delta_t} \tag{3.17}$$

Symbol	Description
v	translational velocity
v_{max}	maximum translational velocity
a_t	translational acceleration
$a_{t_{max}}$	maximum translational acceleration (absolute value)
ω	rotational velocity
ω_{max}	maximum rotational velocity (absolute value)
a_ω	rotational acceleration
$a_{\omega_{max}}$	maximum rotational acceleration (absolute value)
f	centripetal force
f_{max}	maximum centripetal force
c	curvature as introduced in Eq. (3.1)
d_{obst}	distance to closest obstacle
t	time passed since the beginning of the curve
p_i	i-th planning point on the curve
u_i	spline parameter for $p_i = Q(u_i)$
Δ_s	arc length between planning points, i.e., $\int_{u_{i-1}}^{u_i} \|Q'(u)\| \mathrm{d}u$
$\bullet_i$	value of quantity $\bullet$ at p_i
$v_{max}\|\bullet$	maximum translational velocity due to constraint on quantity $\bullet$

Table 3.2: Abbreviations and variables used throughout the discussion of velocity profile generation.

Here $s(t)$ stands for the arc length traveled since the beginning of the interval and a_t is the translational acceleration that is assumed to be constant within the interval. The total arc length of the interval is given by Δ_s (see also Fig. 3.3).

We will now derive a closed form expression for the time $\Delta_t = t_i - t_{i-1}$ traveled between planning points p_{i-1} and p_i and one for the velocity v_i at planning point p_i. Both will be used in the remainder of the section.

Closed forms for $\boldsymbol{\Delta_t}, \mathbf{v_i}$ With Eq. (3.16) and by solving Eq. (3.14) after t one obtains

$$\Delta_t = -\frac{v_{i-1}}{a_t} \pm \sqrt{\frac{{v_{i-1}}^2}{a_t^2} + \frac{2\Delta_s}{a_t}}. \tag{3.18}$$

When ignoring negative values for time and velocities, the solutions for Eq. (3.18) are given by

$$\Delta_t = \begin{cases} -\frac{v_{i-1}}{a_t} + \sqrt{\frac{{v_{i-1}}^2}{a_t^2} + \frac{2\Delta_s}{a_t}} & a_t > 0 \\ \frac{\Delta_s}{v_{i-1}} & a_t = 0 \\ -\frac{v_{i-1}}{a_t} - \sqrt{\frac{{v_{i-1}}^2}{a_t^2} + \frac{2\Delta_s}{a_t}} & a_t < 0,\ v_{i-1} \geq \sqrt{-2\Delta_s a_t}. \end{cases} \tag{3.19}$$

Note that $v_{i-1} \geq \sqrt{-2\Delta_s a_t}$ is a direct consequence of $v_i \geq 0$. For the cases where $a_t \neq 0$ we substitute it using Eq. (3.15) and thus

$$\begin{aligned}
\Delta_t &= -\frac{v_{i-1}\Delta_t}{v_i - v_{i-1}} + \text{sign}(a_t)\sqrt{\frac{{v_{i-1}}^2\Delta_t^2}{(v_i - v_{i-1})^2} + \frac{2\Delta_s\Delta_t}{v_i - v_{i-1}}} \\
\Leftrightarrow \left(1 + \frac{v_{i-1}}{v_i - v_{i-1}}\right)\Delta_t &= \text{sign}(a_t)\sqrt{\frac{{v_{i-1}}^2\Delta_t^2}{(v_i - v_{i-1})^2} + \frac{2\Delta_s\Delta_t}{v_i - v_{i-1}}}.
\end{aligned}$$

Both sides of the equation have the same sign in any case, therefore

$$\begin{aligned}
&\left(1 + \frac{v_{i-1}}{v_i - v_{i-1}}\right)^2 \Delta_t^2 = \frac{{v_{i-1}}^2\Delta_t^2}{(v_i - v_{i-1})^2} + \frac{2\Delta_s\Delta_t}{v_i - v_{i-1}} \\
\Leftrightarrow &\left(1 + \frac{2v_{i-1}}{v_i - v_{i-1}}\right)\Delta_t^2 - \frac{2\Delta_s}{v_i - v_{i-1}}\Delta_t = 0 \\
\Leftrightarrow &\left(v_i + v_{i-1}\right)\Delta_t^2 - 2\Delta_s\Delta_t = 0 \\
\Leftrightarrow &\Delta_t^2 - \frac{2\Delta_s}{v_i + v_{i-1}}\Delta_t = 0.
\end{aligned}$$

Ignoring the negative solution for Δ_t we obtain

$$\Delta_t = \frac{2\Delta_s}{v_i + v_{i-1}}. \tag{3.20}$$

A closer look at Eq. (3.20) reveals that this also holds for the case $a_t = 0$, as it implies $v_i = v_{i-1}$. The side condition that arised for the case $a_t < 0$ can be safely dropped, as it is always ture if $v_i \geq 0$. To see this, we derive an expression for v_i using Eqs. (3.15) and (3.19). For the cases $a_t \neq 0$ it is

$$\begin{aligned}
v_i &= v_{i-1} + a_t\left(-\frac{v_{i-1}}{a_t} + \text{sign}(a_t)\sqrt{\frac{{v_{i-1}}^2}{a_t^2} + \frac{2\Delta_s}{a_t}}\right) \\
\Leftrightarrow v_i &= \text{sign}(a_t)a_t\sqrt{\frac{{v_{i-1}}^2}{a_t^2} + \frac{2\Delta_s}{a_t}}
\end{aligned}$$

which can be simplified to

$$v_i = \sqrt{{v_{i-1}}^2 + 2\Delta_s a_t}. \tag{3.21}$$

Again, this equation also holds for $a_t = 0$. The equation also provides the fulfillment of the side condition in Eq. (3.19) by the existence of a positive v_i. For this

$$\begin{aligned}
&{v_{i-1}}^2 + 2\Delta_s a_t \geq 0 \\
\Leftrightarrow &{v_{i-1}}^2 \geq -2\Delta_s a_t
\end{aligned}$$

has to hold, which is trivial for $a_t \geq 0$ and translates to $v_{i-1} \geq \sqrt{-2\Delta_s a_t}$ for $a_t < 0$.

Note that given a velocity profile, Eq. (3.20) provides the time of travel for each planning point interval and therefore for the complete trajectory.

3.2.5 Isolated constraints

As introduced above, isolated constraints are independent from the velocities set for neighboring planning points. Of course all velocities must obey the robot's maximum translational velocity v_{max}. As we prohibit backward motion this means

$$v_i \in [0, v_{max}]. \tag{3.22}$$

For our method we assume independence of maximum translational velocity v_{max} and maximum rotational velocity ω_{max}. This is the case for synchro drive robots but not true for other drive types, e.g., the differential drive. Without this assumption, an additional constraint has to be included.

With ω being the product of curvature c and translational velocity v, an upper bound on rotational velocity ω_{max} imposes a limit on translational velocities $v_{max|\omega}$:

$$v_i \in [0, v_{max|\omega}], \quad v_{max|\omega} = \frac{\omega_{max}}{|c_i|}. \tag{3.23}$$

For some robots it might be necessary to limit the occurring centripetal forces, which are given by $f = m \cdot |c| \cdot v^2$. Translational velocities that respect a maximum centripetal force f_{max}, fulfill

$$v_i \in [0, v_{max|f}], \quad v_{max|f} = \sqrt{\frac{f_{max}}{m|c_i|}}, \tag{3.24}$$

where m refers to the mass of the robot.

Slow down near obstacles For safety reasons it might be appropriate to require the robot to drive slower when in proximity of an obstacle. We derive a maximum velocity $v_{max|d_{obst}}$ that depends on the distance to the nearest obstacle d_{obst}.

The robot should be able to completely stop before hitting any obstacle. The stopping distance s_{brake} depends on the stopping time t_{brake}:

$$s_{brake} = v t_{brake} - \frac{1}{2} a_{t_{max}} t_{brake}^2. \tag{3.25}$$

Via $v - a_{t_{max}} t_{brake} \stackrel{!}{=} 0$ we get $t_{brake} = v/a_{t_{max}}$ and thereby obtain $s_{brake} = v^2/2a_{t_{max}}$. Under consideration of the distance $s_{react} = v t_{react}$ that the robot travels during its reaction time t_{react}, we now demand:

$$\begin{aligned} & s_{react} + s_{brake} \leq d_{obst} \\ \Leftrightarrow\ & v t_{react} + \frac{v^2}{2a_{t_{max}}} \leq d_{obst} \\ \Leftrightarrow\ & v^2 + 2a_{t_{max}} t_{react} v \leq 2a_{t_{max}} d_{obst} \\ \Rightarrow\ & v \leq -a_{t_{max}} t_{react} + \sqrt{a_{t_{max}}^2 t_{react}^2 + 2a_{t_{max}} d_{obst}}, \end{aligned}$$

or differently noted

$$v_i \in [0, v_{max|d_{obst}}], \quad v_{max|d_{obst}} = -a_{t_{max}} t_{react} + \sqrt{a_{t_{max}}^2 t_{react}^2 + 2a_{t_{max}} d_{obst}}. \tag{3.26}$$

3.2.6 Accelerational constraints

The fulfillment of accelerational constraints at p_i additionally depends on the velocities set at p_{i-1}. Here, the point of view for establishing forward consistency is presented, the equations for backward consistency are obtained by assuming a reversed motion: for backward consistency constraints between p_{i-1} and p_i one simply swaps their roles in the corresponding equations for forward consistency.

3.2.6.1 Translational Acceleration

For further simplification we expect accelerational constraints to be symmetric, i.e., the absolute values of maximum acceleration and deceleration to be identical, denoted $a_{t_{max}}$. Note that this does only affect the planned trajectory that we want to be smooth, for constraints like safety margins for example it is still possible to assume different values for deceleration and acceleration.

Through $a_{t_{max}}$ only $v_i \in [v_{min|a_t}, v_{max|a_t}]$ can be reached, with

$$v_{min|a_t} = v_{i-1} - a_{t_{max}}\Delta_t, \tag{3.27}$$

$$v_{max|a_t} = v_{i-1} + a_{t_{max}}\Delta_t. \tag{3.28}$$

Eq. (3.21) allows to obtain the boundaries for translational velocities that comply with $a_{t_{max}}$:

$$v_{min|a_t} = \begin{cases} \sqrt{{v_{i-1}}^2 - 2a_{t_{max}}\Delta_s} & {v_{i-1}}^2 > 2a_{t_{max}}\Delta_s \\ 0 & else, \end{cases} \tag{3.29}$$

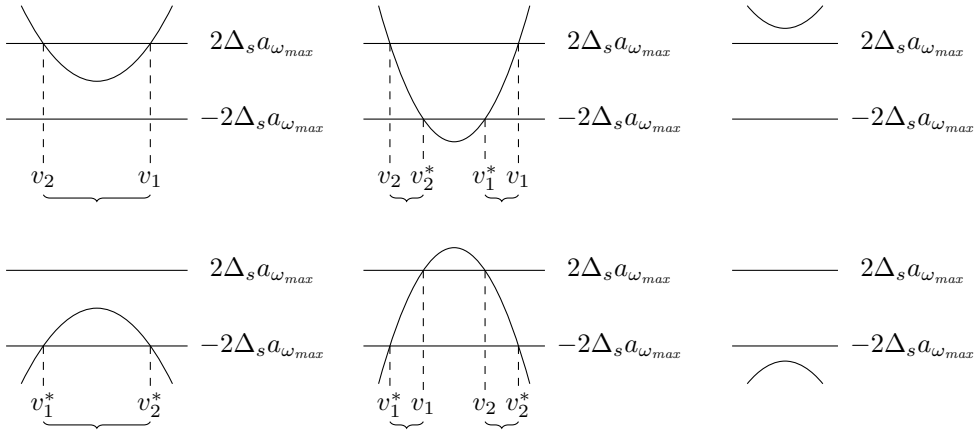
$$v_{max|a_t} = \sqrt{{v_{i-1}}^2 + 2a_{t_{max}}\Delta_s}. \tag{3.30}$$

3.2.6.2 Rotational Acceleration

Our method assumes constraints on translational and rotational acceleration to be independent from each other. While this is true for synchro drives, it is certainly not for the differential drive although suggested so by many controllers. Here, one theoretically has to consider the translational accelerations of the left and right wheel of a differential drive. However, there is a good reason to make the assumption of independent accelerations: this way one can account for the differences in moments of inertia between rotational and translational movements of the robot by imposing appropriate limits for the respective accelerations. These differences in moments of inertia depend on the shape of the robot and the resulting mass distribution and therefore vary a lot among different types of robots.

Despite the fact that it introduces a lot of complexity into velocity profile generation we choose to make the assumption because the errors caused by not accounting for different moments of inertia usually are much bigger than the ones introduced by assumed independence of rotational and translational accelerations.

From the parametric curve it is expected that its curvature does change approximately linearly between the planning points. A prerequisite that for any meaningful curve in the domain of trajectory planning is easily achieved by a sufficient density of planning points.



(a) one interval (b) two intervals (c) empty solution set

Figure 3.5: Schematic figures for the shape of the parabola in v_i defined by Eq. (3.31). The upper row depicts the case $c_i > 0$ (left curve), the lower row the case $c_i < 0$ (right curve). For $c_i > 0$ the parabola's opening is directed upwards, while it is directed downwards for $c_i < 0$. For the solution set (depicted by braces) one can distinguish three basic cases, depicted by (a)-(c). The explicit interval borders are given by Eq. (3.32) and Eq. (3.33).

With the assumptions introduced above, translational acceleration has an effect on the rotational acceleration between p_{i-1} and p_i. As the rotational acceleration is bounded by $a_{\omega_{max}}$, this also implies boundaries on v_i that are mediated through $\omega = c \cdot v$ to Eq. (3.17).

The following has to hold

$$-a_{\omega_{max}} \leq \frac{v_i c_i - v_{i-1} c_{i-1}}{\Delta_t} \leq a_{\omega_{max}},$$

which with the help of Eq. (3.20) can be transformed to

$$-2\Delta_s a_{\omega_{max}} \leq c_i {v_i}^2 + (c_i - c_{i-1}) v_{i-1} v_i - c_{i-1} {v_{i-1}}^2 \leq 2\Delta_s a_{\omega_{max}}. \tag{3.31}$$

This quadratic inequality in v_i describes a parabola whose basic shape is determined by the sign of c_i. Depending on the actual parameters there will be one or two intervals that contain values for v_i that fulfill Eq. (3.31). An illustration is given by Fig. 3.5: the upper row shows the case $c_i > 0$ (left curve) while the lower row depicts the case $c_i < 0$ (right curve). It will will now be shown that the case depicted in Fig. 3.5 (c) can never occur and an explicit formulation for the solution set is derived with the help of fall differentiation.

First, general conditions for intersection of the parabola with the upper and lower bound will be derived. The case $c_i = 0$ (driving straight) will at first be ignored and is accounted for later.

Intersection with upper bound ($2\Delta_s a_{\omega_{max}}$)

$$
\begin{aligned}
& c_i {v_i}^2 + (c_i - c_{i-1}) v_{i-1} v_i - c_{i-1} {v_{i-1}}^2 = 2\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & {v_i}^2 + \frac{(c_i - c_{i-1}) v_{i-1}}{c_i} v_i - \frac{c_{i-1}}{c_i} {v_{i-1}}^2 - \frac{2\Delta_s a_{\omega_{max}}}{ci} = 0 \\
\Rightarrow\ & v_{i1,2} = -\frac{(c_i - c_{i-1}) v_{i-1}}{2c_i} \pm \sqrt{\frac{(c_i - c_{i-1})^2 {v_{i-1}}^2}{4{c_i}^2} + \frac{c_{i-1}}{c_i} {v_{i-1}}^2 + \frac{2\Delta_s a_{\omega_{max}}}{c_i}} \\
\Leftrightarrow\ & v_{i1,2} = -\frac{(c_i - c_{i-1}) v_{i-1}}{2c_i} \pm \sqrt{\frac{(c_i + c_{i-1})^2 {v_{i-1}}^2}{4{c_i}^2} + \frac{8\Delta_s a_{\omega_{max}}}{4c_i}} \\
\Leftrightarrow\ & v_{i1,2} = -\frac{(c_i - c_{i-1}) v_{i-1}}{2c_i} \pm \frac{1}{2|c_i|} \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}}} \\
\Leftrightarrow\ & v_{i1,2} = \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} \pm \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}}} \right)
\end{aligned}
$$

For later use, we restate the result, naming both values for v_i:

$$
\begin{aligned}
v_1 &= \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}}} \right), \\
v_2 &= \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}}} \right).
\end{aligned}
\tag{3.32}
$$

Intersection with lower bound ($-2\Delta_s a_{\omega_{max}}$) Analogously to the derivation of Eq. (3.32) we obtain:

$$
\begin{aligned}
& c_i {v_i}^2 + (c_i - c_{i-1}) v_{i-1} v_i - c_{i-1} {v_{i-1}}^2 = -2\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & \dots \\
\Leftrightarrow\ & v_{i1,2} = \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} \pm \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i \Delta_s a_{\omega_{max}}} \right).
\end{aligned}
$$

We again name the result:

$$
\begin{aligned}
v_1^* &= \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i \Delta_s a_{\omega_{max}}} \right), \\
v_2^* &= \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i \Delta_s a_{\omega_{max}}} \right).
\end{aligned}
\tag{3.33}
$$

Existence of intersections We briefly comment on when the intersections derived above actually exist. The case $c_i > 0$ is depicted by the upper row of Fig. 3.5. For $c_i > 0$ the discriminants in Eq. (3.32) are always positive, the intersections with the upper bound (Eq. (3.32)) therefore always exist and consequently the case in Fig. 3.5 (c) cannot occur.

Whether or not the solution set is split into two intervals (Fig. 3.5 (b) vs. (a)) depends on the existence of intersections with the lower bound (Eq. (3.33)):

$$
v_1^*, v_2^* \text{ exist} \Leftrightarrow \left((c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i a_{\omega_{max}} \Delta_s \geq 0 \right). \tag{3.34}
$$

The lower row of Fig. 3.5 shows the case $c_i < 0$. Here the discriminants in Eq. (3.33) are always positive, therefore the existence of intersections with the lower bound (Eq. (3.33)) is guaranteed. The case in Fig. 3.5 (c) is hereby proved to be irrelevant.

The existence of intersections with the upper bound (Eq. (3.32)) determines whether or not the solution set is split into two intervals (Fig. 3.5 (b) vs. (a)):

$$v_1, v_2 \text{ exist} \Leftrightarrow \left((c_i + c_{i-1})^2 {v_{i-1}}^2 + 8 c_i a_{\omega_{max}} \Delta_s \geq 0\right). \tag{3.35}$$

The degenerate case $\mathbf{c_i = 0}$ For $c_i = 0$, which corresponds to a straight movement, Eq. (3.31) simplifies to

$$-2\Delta_s a_{\omega_{max}} \leq -c_{i-1} v_{i-1} v_i - c_{i-1} {v_{i-1}}^2 \leq 2\Delta_s a_{\omega_{max}}. \tag{3.36}$$

This is a linear function in v_i which intersects the upper bound in the point $\hat{v}_1$, the lower bound in $\hat{v}_2$:

$$\hat{v}_1 = -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1}, \qquad \hat{v}_2 = \frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1}. \tag{3.37}$$

Consequently, the solution set for this case is

$$v_i \in \begin{cases} [\hat{v}_1, \hat{v}_2] & c_{i-1} > 0 \\ [\hat{v}_2, \hat{v}_1] & c_{i-1} < 0 \\ \mathbb{R} & c_{i-1} = 0 \\ \emptyset & \text{else.} \end{cases}$$

Note that for $c_i = c_{i-1} = 0$ any v_i is admissible, as $\omega_i = \omega_{i-1} = 0$ always holds and implies a zero rotational acceleration.

Solution set Summarizing all previous results, the solution set I'_{a_ω} can be stated as

$$v_i \in I'_{a_\omega} := \begin{cases} \begin{cases} [v_2, v_1] & (c_i + c_{i-1})^2 {v_{i-1}}^2 - 8 c_i a_{\omega_{max}} \Delta_s < 0 \\ [v_2, v_2^*] \cup [v_1^*, v_1] & (c_i + c_{i-1})^2 {v_{i-1}}^2 - 8 c_i a_{\omega_{max}} \Delta_s \geq 0 \end{cases} & c_i > 0 \\ \begin{cases} [v_1^*, v_2^*] & (c_i + c_{i-1})^2 {v_{i-1}}^2 + 8 c_i a_{\omega_{max}} \Delta_s < 0 \\ [v_1^*, v_1] \cup [v_2, v_2^*] & (c_i + c_{i-1})^2 {v_{i-1}}^2 + 8 c_i a_{\omega_{max}} \Delta_s \geq 0 \end{cases} & c_i < 0 \\ \begin{cases} [\hat{v}_1, \hat{v}_2] & c_{i-1} > 0 \\ [\hat{v}_2, \hat{v}_1] & c_{i-1} < 0 \\ \mathbb{R} & c_{i-1} = 0 \\ \emptyset & \text{else} \end{cases} & c_i = 0, \end{cases} \tag{3.38}$$

with $v_1, v_2, v_1^*, v_2^*, \hat{v}_1, \hat{v}_2$ according to Eqs. (3.32), (3.33), and (3.37).

Note that we only want to consider positive velocities and therefore define the final solution set as the intersection with the non-negative real numbers:

$$I_{a_\omega} = I'_{a_\omega} \cap \mathbb{R}_0^+. \tag{3.39}$$

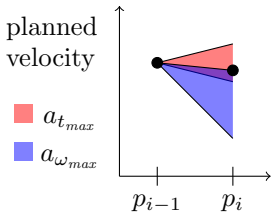


Figure 3.6: Situation in which intervals of admissible translational velocities defined by accelerational constraints $a_{t_{max}}$ and $a_{\omega_{max}}$ overlap. Translational velocity has to be reduced due to increasing curvature. Consider for example a car approaching a curve.

3.2.7 Constraint Satisfaction

All of the isolated constraints are easily satisfied as they yield admissible velocity intervals of the form $[0, b]$, $b \in \mathbb{R}^+$. One takes the minimum over all upper bounds, i.e.,

$$v_i \leftarrow \min\{v_{max}, v_{max|\omega}, v_{max|f}, v_{max|d_{obst}}\}. \tag{3.40}$$

When additionally accounting for accelerational constraints the situation gets more complex: each accelerational constraint defines at least one interval of admissible translational velocities and thereby in general demands a minimum translational velocity for a planning point. In case the intersection of these intervals is non-empty, the maximum velocity contained in the intersection is chosen.

An example is shown in Fig. 3.6: the situation corresponds to a car approaching a curve. The preceeding planning point has a lower curvature and a relatively high translational velocity planned for it. As the next planning point lies within the curve, an increased curvature at this point requires the car to slow down in order to not exceed the maximum rotational acceleration. Here, the intervals defined by maximum translational and rotational accelerations overlap and the translational velocity for the planning point is set to the maximum contained in the intersection.

There is no guarantee, however, that the intervals $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω} overlap in general. This means that when trying to establish forward consistency one can run into a situation where due to the velocity v_{i-1} planned for p_{i-1} and the curvatures c_{i-1} and c_i there is no velocity v_i that respects both, translational *and* rotational acceleration constraints. The analog is true for backward consistency.

Suppose for instance a car approaching a curve at a high speed: Even if skidding is not a problem, it would either have to decelerate quickly or to abruptly change its steering angle in order to stay on the road. If neither is feasible according to the limiting constraints of the car, it will not be able to follow the curve. Fig. 3.7 shows a a schematic, non-exhaustive visualization of the problem. Similar to Fig. 3.6 the intervals of translational velocities that comply maximum translational and rotational acceleration are shown, but they do not overlap.

An ad-hoc solution for this problem would be to iterate the process of readjusting (i.e., decreasing) velocities until an admissible velocity profile has been found. The drawback of this approach would be a potentially long and to some degree unpredictable

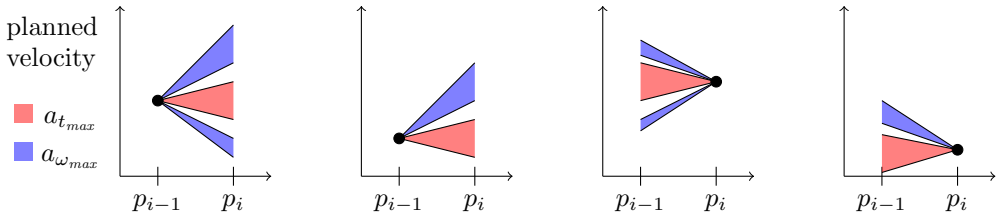


Figure 3.7: Schematic, non-exhaustive depiction of the overlap problem. During velocity profile generation a situation can occur, where the admissible translational velocities respecting constraints for translational acceleration (red) and the ones for rotational acceleration (blue) do not overlap and therefore no admissible v_i or v_{i-1} can be found. This is due to the current combination of the curvatures c_{i-1} and c_i and the planned velocity v_{i-1} or v_i, respectively.

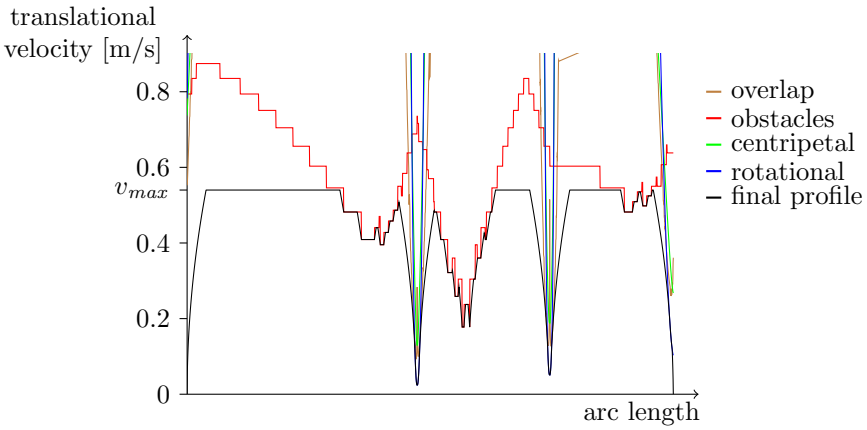
execution time. However, there is another solution that forgoes iteration: given the curvatures c_{i-1} and c_i one can analytically determine the biggest upper bound ${v_{i-1}}^*$ for v_{i-1} such that for all $v_{i-1} \leq {v_{i-1}}^*$ an overlap of $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω} is guaranteed. To assure that v_i then lies within the overlap, the same method can be applied in a backward fashion due to our symmetry assumptions.

The bounds derived thereby can be treated as additional isolated constraints for v_{i-1} and v_i and they are therefore easily integrated into the first phase of velocity profile generation. As the derivation of the bounds is quite lengthy, we skip it here and refer the reader to the appendix. The derivation of the upper bound for v_{i-1} can be found in Appendix B.1, while Appendix B.2 presents pseudo code for its computation.

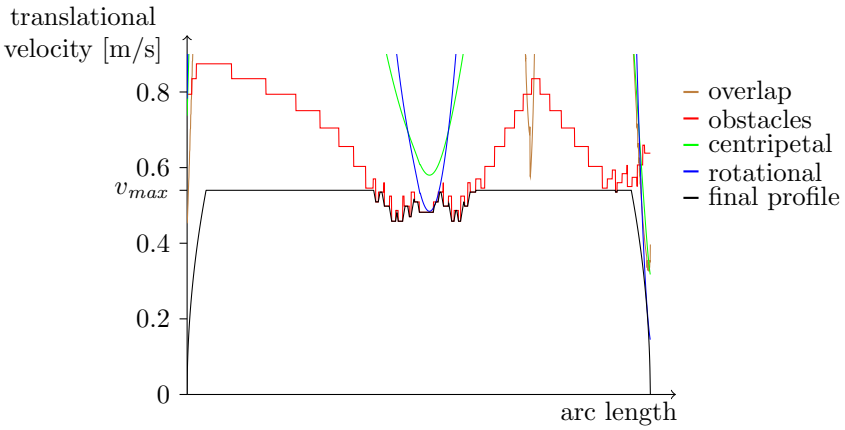
With the overlap of the intervals guaranteed by isolated constraints we can establish forward and backward consistency as presented in Section 3.2.1.2. The difference is that one additionally has to establish consistency with respect to rotational acceleration. Note that attention has to be paid to choosing the appropriate interval from I_{a_ω} when determining the maximum admissible velocity respecting rotational acceleration.

Fig. 3.8 provides a visualization of velocity profiles. It depicts velocity profiles together with the different isolated constraints for two differently shaped trajectories. The plots show how the final velocity profile respects constraints on translational velocities imposed by maximum translational and rotational velocity v_{max} and ω_{max}, maximum centripetal force f_{max}, obstacle distance d_{obst}, and the aforementioned additional constraint that guarantees overlapping of $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω}. Furthermore, it can be observed how accelerational limits affect the final velocity profile: it accelerates and decelerates smoothly between the minima of the isolated constraints and the start and end velocity of zero.

The corresponding splines for the velocity profiles can be found in Fig. 4.5, the smoother spline was generated from the one with sharper curves by the optimization method presented in Chapter 4. Note how the optimization drastically reduces the effects imposed by the constraints.



(a) Values for a trajectory with sharp curves, see Fig. 4.5a.



(b) Values for a smooth trajectory, see Fig. 4.5b.

Figure 3.8: Different sources of isolated constraints imposed on translational velocity. The final velocity profile obeys these constraints as well as accelerational constraints, as can be seen for example by the smooth acceleration from starting velocity zero. The profiles are for the trajectories shown in Fig. 4.5. (a) shows the data for a trajectory with sharp curves whereas the plot in (b) corresponds to a smoothened spline (Fig. 4.5b) generated by the optimization method presented in Chapter 4.

4 Trajectory Generation

This chapter presents how trajectories can be obtained and optimized. Recall that we are given a map and sparse waypoints as an input to our system. The line path defined by the waypoints is required to be traversable by the robot, i.e., it is expected to be free of obstacles.

At first, a trajectory will be generated that closely follows the line path induced by the waypoints. This trajectory will then be iteratively optimized in 2D shape by moving around control points and elongating spline tangents at the latter. Optimization will be stopped if either the optimal trajectory has been found or time constraints prohibit further computation. With this design our system is able to generate an admissible trajectory instantly (anytime capability) while efficiently using available computational resources. Fig. 4.1 provides a schematic overview over this system.

The next section covers the generation of the initial trajectory through the use of heuristics. The remaining sections provide details on the optimization of the initial trajectory and the issue of planning a new trajectory while still following the old one.

4.1 Initial Trajectory

The intuition behind the initial trajectory is to correspond to traversing the line path induced by the waypoints in a drive and turn fashion. As a consequence, we expect the highest curvature near the waypoints, since these are the turning points of the path.

For the generation of the initial trajectory the given waypoints will be used as start and end points of the segments of a Quintic Bézier Spline. To specify tangents and second derivatives at these points, heuristics will be used that extract suitable values from the geometric shape of the line path induced by the waypoints.

4.1.1 First Derivative Heuristic

The first derivatives, or tangents, at inner waypoints (i.e., all but the starting and the end point) are defined by employment of the following heuristic.

In the following we assume computation of the tangent for the waypoint P_i. As it is an inner waypoint, it inscribes an angle with its neighbor points, $\alpha = \angle_{P_{i-1}P_iP_{i+1}}$. The tangent will be set perpendicular to the angular bisector of α, pointing in the direction of travel. To prevent unnecessary oscillations of the spline, the tangent magnitude is set to half of the minimum Euclidean distance to a neighbor control point, i.e., $\frac{1}{2}\min\{|P_{i-1}P_i|, |P_iP_{i+1}|\}$. Fig. 4.2 provides an exemplary visualization of the heuristic. Application of the heuristic means that half of the heading change between the straight line connections $P_{i-1}P_i$ and P_iP_{i+1} has taken place when reaching P_i. This heuristic is well known to yield smooth curves and is for instance referred to in [17, p. 318].

For the first waypoint the tangent's direction is given by the robot's heading, for the last waypoint it is set to match the one of the straight line connection to the previous

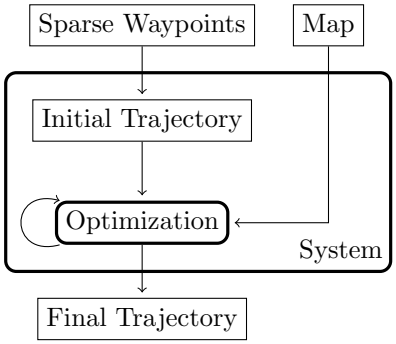


Figure 4.1: Schematic system overview with processes/functional entities (rounded corners) and objects (normal rectangles). With sparse waypoints and a map as input the system first generates an initial trajectory. This trajectory is then optimized in an iterative fashion.

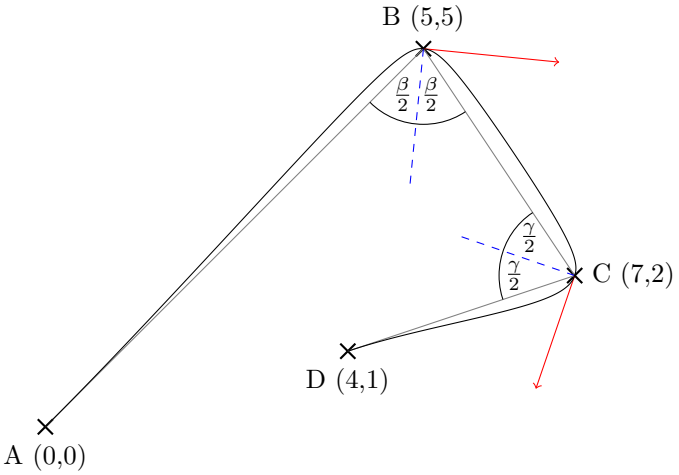
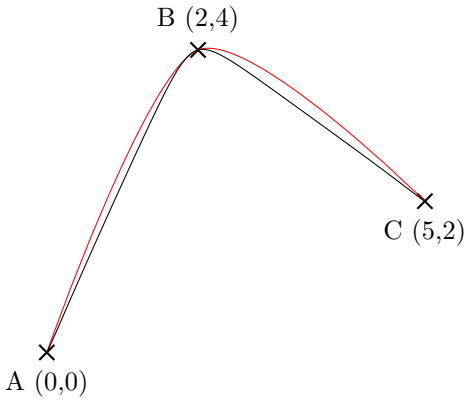
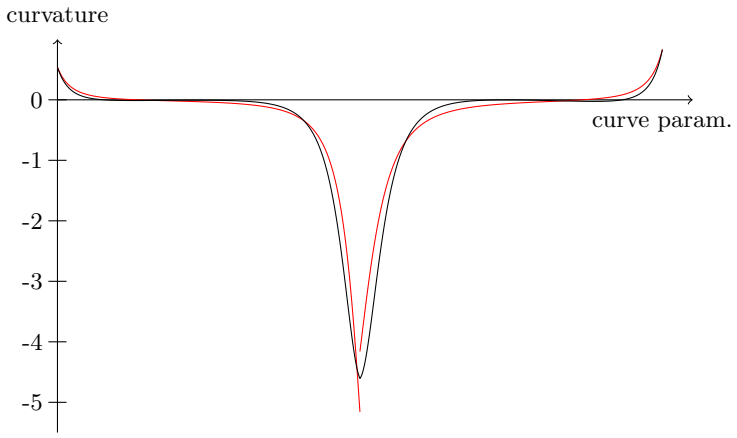


Figure 4.2: Visualization of the tangent (first derivative) heuristic applied to a Quintic Bézier Spline. The tangents at control points B and C (red arrows) have been set according to the heuristic: they are perpendicular to the angular bisector of the angle inscribed by the particular control points and its neighbors (dashed blue). Furthermore their magnitude is proportional to the distance to the nearest neighbor control point.



(a) Splines



(b) Curvature profiles

Figure 4.3: Second derivative heuristic visualization. Black: a Quintic Bézier Spline, red: two Cubic Bézier Splines. (a) shows the splines while (b) displays their curvature profiles. The splines start at point A and B, respectively. The Quintic Bézier Spline's second derivative at point B has been set to a weighted average of the two Cubic Bézier splines. Except for the second derivative at point B the Quintic Bézier Spline's first and second derivatives match the ones of the particular Cubic Bézier Splines. Note the curvature discontinuity at the join point B for the Cubic Bézier Splines.

waypoint. Similar to the inner waypoints the tangents' magnitudes are set to half of the Euclidean distance to the neighbor waypoint.

The suitability of the employed heuristic is shown below during treatment of trajectory optimization (Section 4.2.3).

4.1.2 Second Derivative Heuristic

With the second derivative our goal is to retrieve a spline whose curvature does not change too much. As proved in [24, Sect. 2.3, Corollary 2.8], cubic splines minimize the integral of the second derivative's absolute value. In general, this corresponds to small changes in curvature. Although preconditions are not exactly met (second derivative at start and end point do not necessarily equal zero), we try to mimic the behavior of cubic splines.

If we connected the waypoints P_{i-1}, P_i and P_i, P_{i+1} with Cubic Bézier Splines each, we would have minimal curvature change on each of the segments. As discussed in Section 3.1.2.1, however, this would result in a curvature discontinuity. The intention now is to set the Quintic Bézier Splines' second derivatives at the join point to a weighted average of the values that two Cubic Bézier Splines would have. See Fig. 4.3 for a visualization of this heuristic.

For simplicity we adopt the notation used in Fig. 4.3 for the derivation. Our goal is to determine the second derivative at the point B for the Cubic Bézier Spline connecting A and B (S_{AB}), as well as for the one from B to C (S_{BC}). We consider the tangents t_A, t_B, and t_C at the points A, B, and C to be given.

With this information, the control points of the two Cubic Splines compute to (see Eq. (A.22))

$$\begin{aligned} S_{AB}: &\quad A,\ A+\frac{1}{3}t_A,\ B-\frac{1}{3}t_B,\ B \\ S_{BC}: &\quad B,\ B+\frac{1}{3}t_B,\ C-\frac{1}{3}t_C,\ C. \end{aligned}$$

For the second derivatives of the splines at the point B this yields (refer to Eq. (A.23)):

$$\begin{aligned} S''_{AB}(1) &= 6\left(\left(A+\frac{1}{3}t_A\right)-2\left(B-\frac{1}{3}t_B\right)+B\right) = 6A+2t_A+4t_B-6B \\ S''_{BC}(0) &= 6\left(B-2\left(B+\frac{1}{3}t_B\right)+\left(C-\frac{1}{3}t_C\right)\right) = -6B-4t_B-2t_C+6C. \end{aligned}$$

While one could now compute a simple average of the two second derivatives, it makes even more sense to weight the second derivatives according to the relative size of their respective segments. We will do this in an inversely proportional fashion, which might seem counterintuitive at first glance but yields the effect that a long segment will not dramatically deform a much shorter adjacent segment.

Let the generic weighted average of the second derivatives be

$$a = \alpha S''_{AB}(1) + \beta S''_{BC}(0) = \alpha\left(6A+2t_A+4t_B-6B\right)+\beta\left(-6B-4t_B-2t_C+6C\right).$$

To define weights with the above mentioned properties, let d_{AB} be the distance between A and B, say $d_{AB} = |AB|$, d_{BC} analogously. The weights for the segments

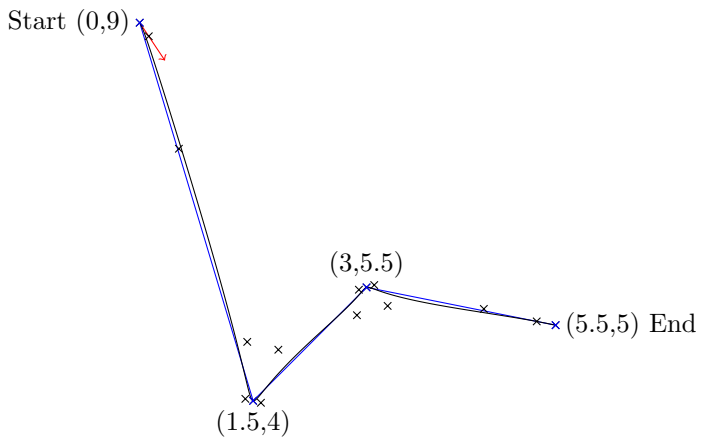


Figure 4.4: Initial trajectory generation: The given waypoints and the induced line path are depicted in blue. The red arrow indicates the robot's heading at the start position. The generated initial trajectory is given by the black solid curve. The black markers indicate the control points of the Quintic Spline that are not given waypoints. Note that the tangent length has been reduced to a quarter of the minimum adjacent segment length to achieve even less deviation from the line path than in Fig. 4.2 for example.

are then given by

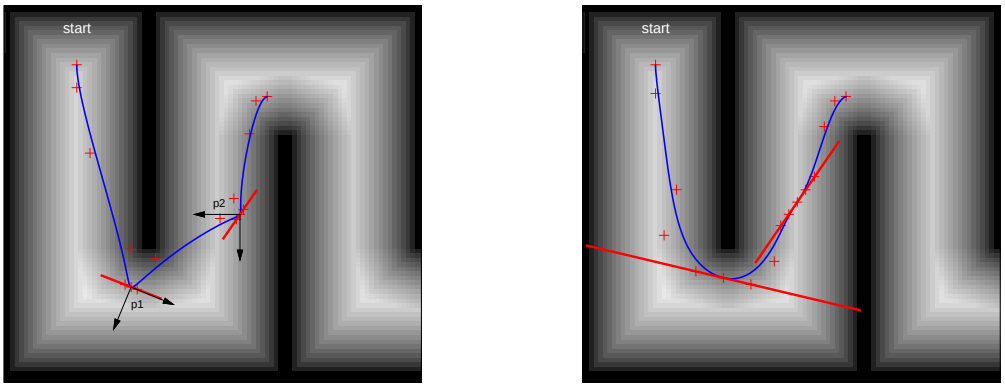
$$\alpha = \frac{d_{BC}}{d_{AB} + d_{BC}}, \quad \beta = \frac{d_{AB}}{d_{AB} + d_{BC}}.$$

As for the first and last waypoint only one segment is available, one simply takes the second derivative corresponding to that segment. Note that the second derivative at the first waypoint might also be predetermined when planning a new trajectory while the robot is already moving.

4.1.3 Generation

To generate an initial trajectory, the given waypoints are chosen as start and end points of Quintic Bézier Spline segments. The remaining control points, that determine first and second derivatives at the waypoints, are obtained by means of the respective heuristics presented above. Compared to the original tangent heuristic, tangents are shortened by factor 2 to achieve less deviation from the waypoint induced line path. Fig. 4.4 gives an example for an initial trajectory. It shows the given waypoints, the induced line path, and the initial trajectory as well as the initial heading of the robot.

We now have a trajectory that closely follows the line path induced by the waypoints in a drive straight and turn fashion and is therefore not colliding with any obstacles. For this trajectory a velocity profile could be instantly generated and together with the trajectory provided to a controller for execution. The next section describes how



(a) Initial trajectory (b) Optimized trajectory

Figure 4.5: Trajectory before and after optimization. (a) shows the initial trajectory with the gradient induced coordinate system for waypoint translation (black arrows). In (b) the result of the optimization process is shown. Red markers stand for spline control points, tangents at inner waypoints are depicted by red lines. Trajectories are shown in blue above the distance map, where darker values are closer to obstacles.

this initial trajectory can be optimized iteratively until an optimal solution has been found or time constraints enforce returning the best trajectory found so far.

4.2 Optimization

The goal of the optimization is to change the initial trajectory's shape in a way that it yields lower costs, for this work this means that it can be traversed faster. This can be reached by smoothing the trajectory to have less sharp curves and by reducing the overall arc length, for example. The optimization has to balance these two potentially conflicting means to reach its goal of a minimum traversal time while respecting the robot's kinematic and obstacle distance constraints.

In this section we will first introduce the parameters of the spline we are going to optimize and present the method that is used to do so. Afterwards, an empirical examination of the search space is conducted to assure the suitability of the employed optimization method.

For a rough idea of what the method does refer to Fig. 4.5. It shows a trajectory before and after optimization, the grayscale background of the figures is the corresponding distance map. The optimized trajectory can be traversed faster and the occurring kinematic constraints are less harsh, as can bee seen in Fig. 3.8. It shows the velocity profiles and isolated constraints for the splines in Fig. 4.5.

4.2.1 Parameters for Spline Optimization

As mentioned above, to influence the shape of the initial trajectory we choose to use the 2D location of the inner waypoints and the tangent magnitude at the waypoints

as optimization parameters. This leaves us with three degrees of freedom per inner waypoint.

Waypoint 2D position For the parameterization of a waypoint's translation we choose the first parameter to be the translation of the waypoint in the direction of the distance map's gradient at the waypoint location. The second parameter will be the translational component orthogonal to this direction. This way we use a local coordinate system that is induced by the distance map gradient. This corresponds to moving a waypoint closer to or further away from an obstacle, and translating the waypoint parallel to the closest obstacle, respectively. With the use of this coordinate system the two parameters for translation now have a meaning, especially with respect to the part of the cost function that addresses obstacle proximity. Note that whether using the meaningful coordinate system or the one provided by the distance map, the reachable 2D translations are the same, of course. In the left part of Fig. 4.5 the induced coordinate systems are shown by black arrows at the inner waypoints p1 and p2. Details on the Sobel Operator based computation of the gradients can be found in Appendix C.

Tangent magnitude The parameterization of the tangent magnitude will be represented by the *tangent elongation factor*, a scalar value indicating by which factor the initial tangent has to be multiplied to retrieve the desired tangent. Elongation of a tangent causes the curve at the particular join point to become less sharp and therefore in general allows faster traversal of the trajectory. See for instance the change in shape of the curve at p1 in Fig. 4.5 and in the magnitude of respective tangents.

One might ask oneself why the tangent's direction is not considered as optimization parameter, the answer is that we found the heuristic introduced in the previous section to yield results that hardly need optimization. Support for this claim will be given below. Note that when translating an inner waypoint, the tangent direction at this point and its neighbors will be corrected according to the heuristic while keeping the elongation factors.

Search space topology The space that we will search for the optimized trajectory is spanned around the initial trajectory by continuous variation of the above mentioned parameters for each of the inner waypoints. The evaluation criterion for each combination of parameters is the total traversal time. For splines that collide with the environment an infinite traversal time is assumed. This causes the optimization procedure to discard these parameter combinations.

Note that we furthermore restrict the search space in the following manner: we do not continue the search in directions that yield collision causing parameter combinations. In practice this scenario occurs if there are two paths around an obstacle: we will only consider the one suggested by the initial trajectory and not incrementally translate waypoints through the obstacle to reach the other path, a part of the search space that is disconnected by colliding splines.

4.2.2 Optimization Method

The optimization method we use to search for the optimal trajectory is inspired by the *RPROP* algorithm. RPROP stands for resilient backpropagation and is an algorithm

```
1  P ← parameters to be optimized;
2  BestTrajectory ← initial trajectory;
3  while time left do
4      forall p ∈ P do
5          CurrentTrajectory ← BestTrajectory;
6          repeat
7              (Δ, CurrentTrajectory) ← RPROP(p, CurrentTrajectory);
8              if evaluate(CurrentTrajectory) > evaluate(BestTrajectory) then
9                  BestTrajectory ← CurrentTrajectory;
10                 break;
11         until Δ < ϵ ;
12     tryWaypointDeletion(BestTrajectory)
```

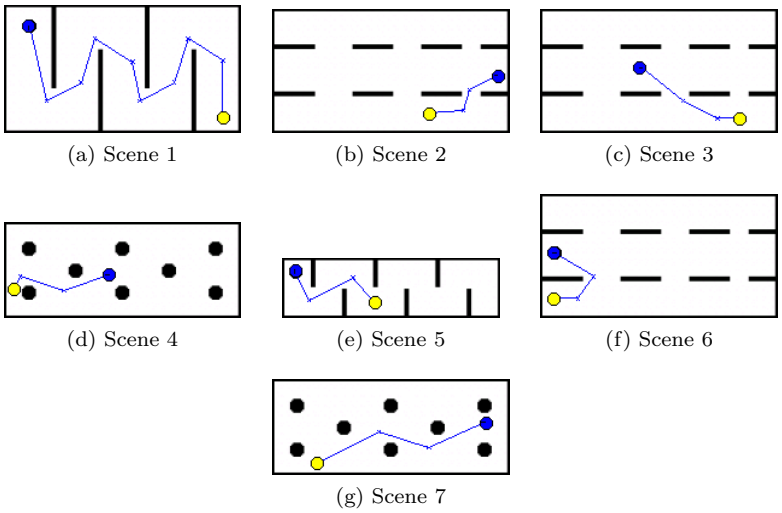
Figure 4.6: Pseudo code for the optimization method based on the RPROP algorithm.

for weight adaption in neural networks. A brief introduction can be found in a technical report [27], while [28] is the original publication of the algorithm. It is a gradient descend method that only uses the sign of partial derivatives to determine the direction of the weight change. The magnitudes of the derivatives are ignored, the actual amount of change in the weights is determined by an additional heuristic. As the sign of partial derivatives can be computed by pointwise evaluation of the cost function, this renders RPROP a derivative free optimization algorithm. Further advantages of the algorithm are its robustness with respect to local minima and its fast convergence compared to standard gradient descent methods.

We base our optimization method on RPROP because of its robust and fast convergence and because it is derivative free. The derivative of our cost function is too involved to be obtained in closed form: deriving the influence of shape parameters on time of travel on its own involves deriving arc length and curvature functions and further complexity is introduced with the influence of the distance map on admissible velocities.

The major difference in our adoption of RPROP is that we do not update all parameters at the same time but choose to iterate through them and treat each parameter as a one-dimensional optimization problem. While doing so, we interweave the optimization of the different parameters, i.e., we do not fully optimize a parameter before proceeding to the next one. The treatment of one dimensional subproblems is chosen to reduce the number of gradient sign computations and thereby the number of velocity profile generations.

Fig. 4.6 gives the pseudo code for the proposed optimization method. Starting with the initial trajectory it always keeps a reference to the best trajectory so far. As long as there is time left for optimization the method cycles through the parameters P of the spline (2D location of the inner waypoints and tangent elongation) and triggers a one-dimensional optimization using RPROP. As soon as a RPROP step successfully improved the trajectory, optimization of the parameter is stopped and the method moves on to the next parameter. The same is true for the case that the step size Δ computed by RPROP falls below a certain threshold (ϵ in the algorithm). In Fig. 4.6 the function `RPROP` stands for one step of RPROP optimization that returns a new step



(a) Scene 1 (b) Scene 2 (c) Scene 3

(d) Scene 4 (e) Scene 5 (f) Scene 6

(g) Scene 7

Figure 4.7: Scenes used for search space inspection. Obstacles are shown in black, the robot's start position is blue and the goal is indicated by a yellow dot. The waypoints and the line path induced by them are shown in blue. For examination of Scene 1 the path has been cut after the first four waypoints (compare Fig. 4.5, it shows initial and optimized trajectory for Scene 1).

size and an updated trajectory. The `evaluate` function rates a trajectory with respect to a cost function. The trajectory with the shorter time of travel is chosen. At this point, other cost measures like anticipated energy consumption or steering effort could be used instead.

Deletion of inner waypoints In addition to parameter optimization using RPROP, we employ the deletion of inner waypoints as mean of optimization. Not seldom, such a deletion results in a more optimal path, as complexity is reduced due to one less waypoint to pass through. Another side effect is that in total less parameters remain to be optimized, which considerably speeds up the process.

We choose to delete a waypoint only if the resulting trajectory is collision free and considered a better one with respect to our cost function. Note that when deleting an inner waypoint, tangent directions at the neighboring waypoints are *not* updated according to the heuristic because this might cause a collision with the environment.

Tests for possible deletion of inner waypoints are conducted after every complete pass through all parameters, the pseudo code in Fig. 4.6 refers to it as the function `tryWaypointDeletion`.

4.2.3 Examination of the Search Space

In this section we will examine the search space defined above with respect to the applicability of the suggested optimization method. Evidence is presented that the optimization method converges to near optimal results and that it is not disturbed

by local minima in the search space. Furthermore, we elaborate on how changes in optimization parameters affect the spline and associated costs.

To examine the search space, we rely on an empirical inspection, as the general space is difficult to handle. Parameter combinations are evaluated in an exhaustive fashion for several scenes. The hereby gained discretized views of the search space are then examined. The input basis for this is constituted by the seven scenes shown in Fig. 4.7. As we expect the search space to be smooth and well behaved for scenes without close obstacles or sharp turns in the waypoint path, we choose scenes that represent critical situations for our examination. Paths have to navigate around obstacles and their shapes are very diverse. For the most part, the obstacles' shape cannot be suitably approximated by a circumcircle. This induces sharp turns into the waypoint path.

Note that for Scene 1 the input path was pruned to contain only the first three segments in order to keep the exhaustive search computationally tractable. This way all input scenes contain paths with two inner waypoints, which leads to a six-dimensional search space (three dimensions for each inner waypoint).

We start the examination with the subspaces the optimization parameters induce when modified exclusively.

Tangent elongation Fig. 4.8 shows the influence of tangent elongation in the initial trajectory for several scenes. For combinations of tangent elongations for both inner waypoints p1 and p2 the durations of the velocity profiles are shown.

Initial trajectories have a tangent elongation factor of 0.5 for both inner waypoints p1 and p2, this combination is found in the the plots' lower left corner. The plots show that elongating tangents in principle allows the spline to be traversed faster. Beyond a certain point, however, further elongation of the tangents again increases the durations or even yields trajectories that collide with the environment (white areas). Increasing durations occur as soon as the trajectory cannot be traversed any faster but the widened curves lead to a bigger overall arc length instead.

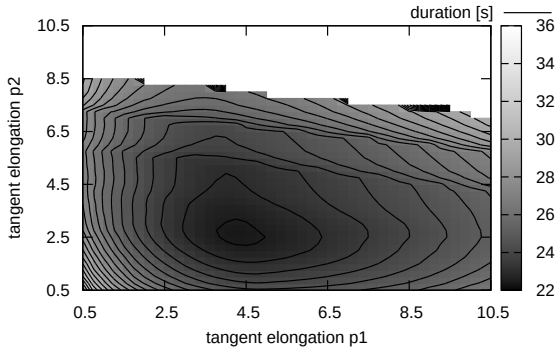
It can be seen that the space is smooth and free of local minima. The plots for the remaining scenes in Fig. 4.8 look similar.

Waypoint translation Translating inner waypoints can affect the time needed to traverse a trajectory in several ways. Firstly, translating waypoints has a direct effect on overall arc length and can furthermore lead to a change in curvature at the join points. Secondly, by translating waypoints, the trajectory can change its distance to nearby obstacles, influencing induced velocity limits (see Section 3.2.5).

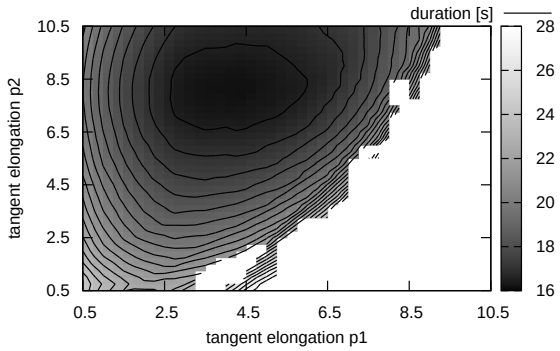
Fig. 4.9 shows how translation of the first inner waypoint (p1) in the initial trajectory changes time of travel for several scenes. The figure shows the corresponding initial trajectories and distance maps.

The plots' axes represent movement along the gradient direction and orthogonal to it, this gradient induced coordinate system is visualized with black arrows in the corresponding distance maps. Looking at these coordinate systems in the right column of Fig. 4.9 one observes that movement along gradient direction generally corresponds to increasing obstacle distance while orthogonal movement corresponds to lateral translation with respect to the closest obstacle.

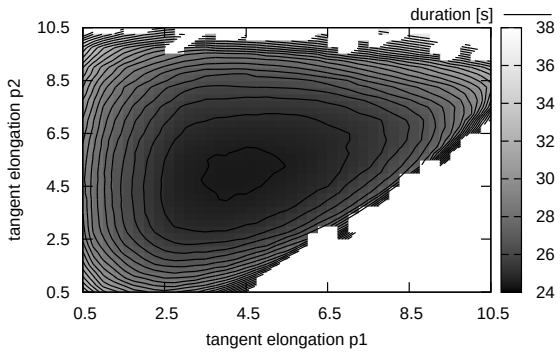
The relation between waypoint translation and time of travel is observable in Scene 4 (Fig. 4.9b). From the plot one can see that the optimal position for the waypoint p1 is



(a) Scene 1

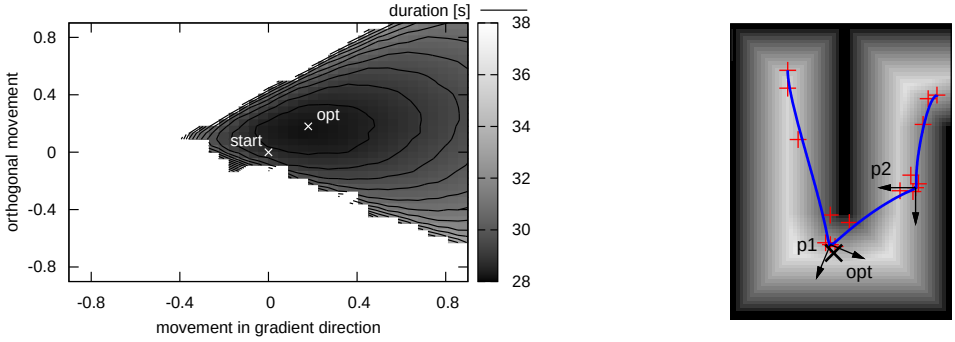


(b) Scene 4

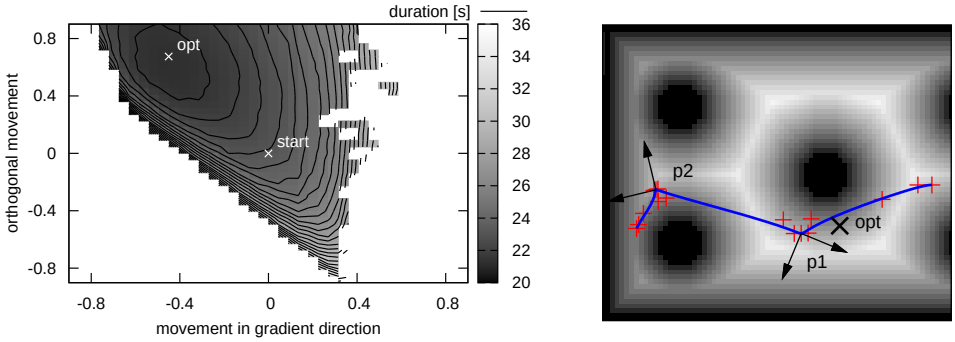


(c) Scene 7

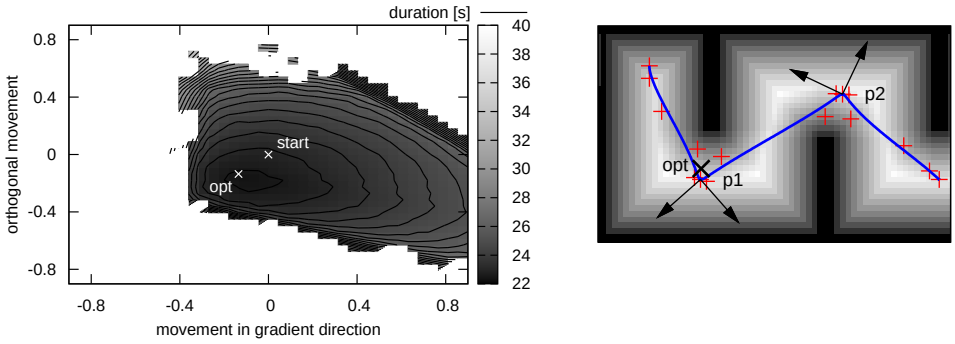
Figure 4.8: Elongation of tangents for initial trajectory and its effects on time of travel. The initial trajectory has elongation factors of 0.5 for both tangents (lower left corner). Darker is better, contours have been added for intervals of 0.5 seconds. Parameters that correspond to white areas yield collisions with the environment. Figures look similar for the remaining scenes.



(a) Scene 1



(b) Scene 4



(c) Scene 5

Figure 4.9: Translation of waypoint p1 for initial trajectory and its effects on duration. Right part shows corresponding initial trajectories. In the left part darker is better, contours have been added for intervals of 0.5 seconds. White areas correspond to parameter combinations that lead to collisions with the environment. Optima and start positions are marked. Figures look similar for the remaining scenes.

in negative gradient direction and positive orthogonal direction from the start position. It is indicated by a white marker in the plot and a black one in the distance map in the right part of the figure.

We also observe that moving p1 closer to the obstacle, i.e., movement in negative gradient direction, is not feasible without prior lateral movement along the obstacle, here in positive orthogonal direction. As can be seen in the distance map, moving p1 towards the obstacle leads to a collision of the trajectory on the right side of p1. In the plot these collisions correspond to the triangular white area in the lower left. Lateral movement of p1 centers the point below the obstacle and thereby allows passing it without collision when moving p1 towards it.

The parameter combinations that yield the rectangular white area in the right part of the plot correspond to moving p1 away from the obstacle beyond a certain threshold. A look at the distance map reveals that a collision of the trajectory with the obstacle at p2 is inevitable when moving p1 in positive gradient direction, regardless of any additional movement in orthogonal direction.

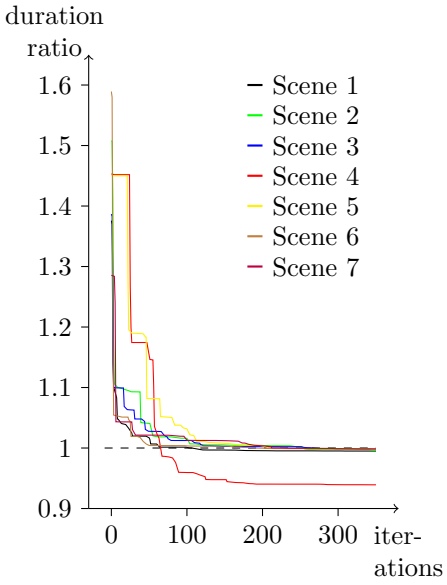
One observes that the search space is smooth and free of local minima for translation of the first inner waypoint. The plots for the remaining scenes in Fig. 4.9 look similar to the presented ones and analog observations have been made for the translation of the second inner waypoint p2.

Combined search space The spaces discussed above only constitute two-dimensional subspaces of the actual search space, which consists of two inner waypoints with three parameters each and therefore has six dimensions. As visualization gets difficult for this dimensionality we have to rely on different means for inspection of the complete space.

The smoothness of the subspaces suggests that this is also the case for the combined search space. However, interaction effects might occur. We therefore conduct an exhaustive search in the discretized six-dimensional space for each scene.

For Scene 5 the discretized space contains only one minimum. For the other scenes several local minima are detected. However, they either yield durations within 0.2 seconds range of the optimum, which is roughly 1% of the total travel time, or they correspond to extreme parameter settings that yield a considerably higher duration.

As another mean to assess the applicability of the optimization method in the combined space we compare its results to the ones obtained by the aforementioned exhaustive search. Fig. 4.10 shows how the proposed optimization method performs in each scene. It is always able to find a trajectory in the vicinity of the optimum in the discretized space. In the left part the duration ratio of the search's trajectory and the one that is the optimum found by the exhaustive search are plotted against the number of optimization steps. As soon as this ratio approaches 1, the search has found a trajectory that is as good as the discretized optimum. For Scene 4 the ratio even drops considerably below 1. Due to not being limited by discretization the search method was able to find a better trajectory than the discretized exhaustive search. The table in the right part of the figure states the absolute numbers. It lists duration of the initial trajectory, the best one found by the discretized exhaustive search, and the one the search method converges to.



	durations [s]		
Scene	initial	exh. search	optimized
1	28.73	20.89	20.78
2	18.10	12.01	11.92
3	26.75	19.29	19.27
4	23.10	15.90	14.92
5	22.85	15.76	15.72
6	21.76	13.69	13.68
7	30.25	23.53	23.48

Figure 4.10: Optimization process for Scenes 1–7. The plot shows the ratio of duration and optimum duration found by the (discretized) exhaustive search against the number of iterations. The table lists the absolute values.

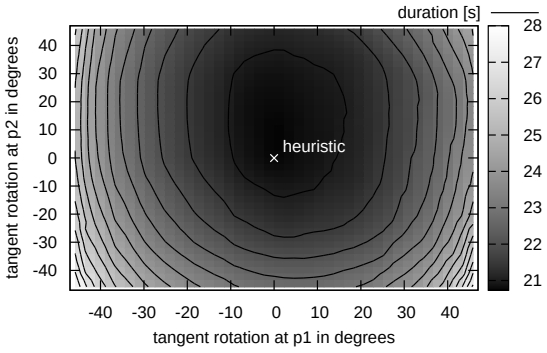
Tangent rotation To conclude the examination of the search space we provide some evidence for the suitability of the employed tangent heuristic. While one could include tangent rotation as an additional parameter in the optimization, we chose to rely on the heuristic for complexity reasons, as the use of the heuristic decreases the number of optimization parameters.

To ensure that by not considering tangent orientation for parametric optimization not too much potential is given away, an exhaustive search of the space including tangent rotation is conducted. The now eight-dimensional space has to be heavily discretized to keep the search tractable.

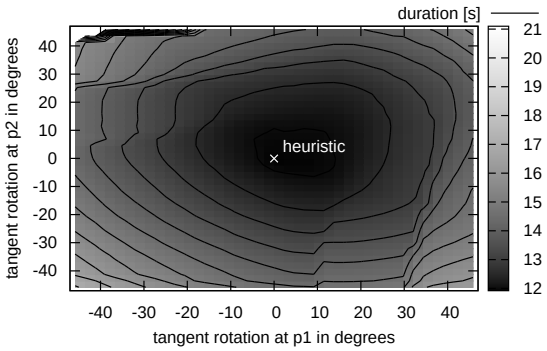
Tab. 4.1 shows the differences in optimal trajectory duration if considering tangent rotation or not. Note that data was generated with a different discretization and slightly different parameters for obstacle imposed velocity limits, therefore the optima listed in Tab. 4.1 do not exactly match the ones in the right part of Fig. 4.10.

In Tab. 4.1 one observes that the durations for the optima of the exhaustive search do not differ dramatically, including tangent rotation into the search space did not yield optima with considerably better durations.

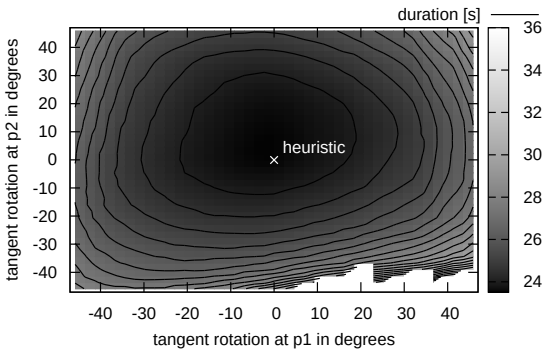
More evidence that tangent rotation is set to a sensible value by the heuristic is given in Fig. 4.11. It shows the two-dimensional space spanned by tangent rotation at both inner waypoints around the trajectory the optimization method converged to. Darker values stand for less time of travel and contours have been added to the figure at levels of 0.5 seconds. White areas correspond to values for rotation that yield trajectories colliding with the environment. One observes that deviation from the rotation set by the heuristic can hardly improve the duration of the trajectory. In fact, examination of the space for Scenes 1–7 results in a maximum improvement of 0.09 seconds, i.e.,



(a) Scene 1



(b) Scene 2



(c) Scene 7

Figure 4.11: Tangent rotation for first and second inner waypoint for final optimized trajectory and its effects on time of travel. Darker colors stand for shorter durations. Contours have been plotted for levels of 0.5 seconds. The tangent heuristic corresponds to values of 0 for both rotations. Figures for scenes remaining look similar.

	exh. search best duration [s]		
Scene	with rotation	without rotation	difference [s]
1	20.71	20.80	0.09
2	11.92	12.15	0.23
3	19.24	19.31	0.07
4	15.72	16.16	0.44
5	15.56	15.58	0.02
6	13.60	13.60	0.00
7	23.50	23.52	0.02

Table 4.1: Differences in optimal duration regarding consideration of tangent rotation for optimization. The data is based on an exhaustive search in the discretized eight-dimensional space.

0.5% of the total time of travel, if one considers tangent rotation only (numbers not shown in the figure). For the evaluation, rotation up to 45 degrees in both directions has been considered, values beyond that tend to yield unfavorably deformed splines as tangents increasingly point against the general direction of travel.

4.2.4 Runtime

We briefly describe the runtime of the proposed method. The predominant part of computational expense is consumed by the generation of the velocity profile. Herein, the numerical integration of the 2D path of the trajectory constitutes constant costs for each segment. Calculations at the planning points yield costs that are linear in a segment's arc length. Costs for calculating the spline from its parameters and for collision checks are linearly depending on the number of spline segments. Consequently, the runtime is linear in the trajectory length and in the number of waypoints provided to the method.

In most cases around 500 RPROP steps can be executed within 0.4 seconds for paths consisting of four waypoints. Significant improvements are in general achieved earlier in the optimization process, as can be seen in Fig. 4.10 as well, the major improvements take place within the first 100 RPROP steps, which corresponds to 0.1 seconds. Note again the anytime capability of the proposed method, i.e., it provides the continuously improving trajectory at any time during the optimization.

The runtimes correspond to execution on an Intel Pentium 2 GHz dual processor. The optimization method used to gather this data can be further improved regarding implementational and conceptual aspects (see also Section 6.2).

4.3 Updating Plans

During the execution of the planned trajectory, it becomes necessary to update the active plan for mainly two reasons. First, sensors might have detected an unmapped obstacle that leads to a collision in the active path or at least to changes in costs. The second reason is that accumulating odometry error leads to a displacement of the

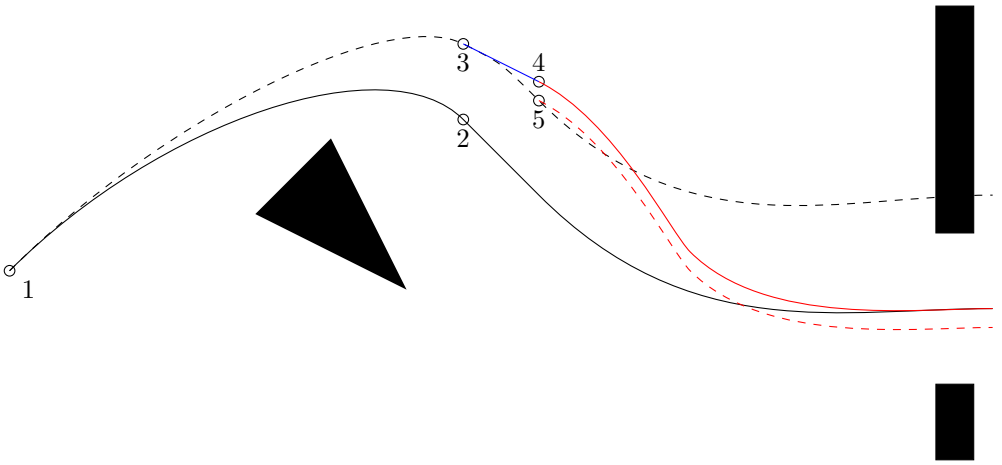


Figure 4.12: Schematic visualization of the effects of accumulated odometry error and a plan update as countermeasure. The reference frame is the ground truth, localization errors are ignored. For details please refer to the text (Section 4.3.1).

robot with respect to the planned trajectory in case that the employed controller uses odometry as feedback exclusively.

This is the case for most controllers and is also due to the fact that in high precision robots, odometry readings are available at a considerably higher frequency than high level position estimates generated by the employed localization method. We now show how updating the active plan can account for the accumulation of odometry errors and for unmapped obstacles.

4.3.1 Odometry Errors

Eventually the robot pose estimated by accumulated odometry readings will deviate from the estimates by localization. As a result, derivation from the planned trajectory increases over time.

Fig. 4.12 provides a schematic visualization for this situation: the reference frame for this figure is the ground truth, localization errors are ignored. The robot starts at position 1. While the black solid line depicts the planned trajectory, the dashed black line stands for the execution of the planned trajectory in case the plan is not updated. Over time the deviation from the planned path increases due to accumulating odometry error and leads to a collision.

At position 3 an update of the plan is initiated. At this time the robot is located at position 3 but the estimate by the odometry based controller is at the originally planned position 2. During the update, the position estimate based on the odometry is reinitialized to the current localization, thus the accumulated error vanishes.

To account for the time consumed by updating the plan, a lookahead is done to determine the anticipated location of the robot at the time the updated plan will be available. In Fig. 4.12 this position is marked with the number 4, the blue solid line represents the anticipated movement of the robot during the planning phase. A new

plan (red solid line) is therefore computed to smoothly fit at this position and prevent the impending collision.

The odometry error that accumulates during the planning phase for the updated trajectory is assumed to be small compared to the one building up to the beginning of the phase. It results in the robot actually being at position 5 at execution time of the updated trajectory. As a consequence the actual execution of the updated trajectory (depicted as dashed red line) slightly differs from the planned trajectory.

Note that Fig. 4.12 only constitutes a schematic visualization of the situation, i.e., it does not claim proportionality. For easier reference we restate the meaning of line colors and numbers in the figure:

black solid original trajectory

black dashed robot's execution of trajectory due to odometry error

blue solid anticipated robot movement during the planning phase

red solid updated plan

red dashed execution of updated plan (due to to a small odometry error during replanning phase)

1 initial robot position

2 ideal robot position at replanning time (controller thinks robot is here)

3 real robot position at replanning time (due to odometry error)

4 robot's assumed position at start time of new plan

5 robot's real position at new plan execute time (due to small odometry error during replanning phase)

4.3.2 Unmapped Obstacles

Through updating the trajectory as described above one can also account for unmapped obstacles. This is done by simply incorporating current sensor readings into the map just before planning the new trajectory. As a result, the updated trajectory accounts for obstacles that are not contained in the original map of the environment but sensed at planning time.

For our work we employ a simple version of sensor data incorporation, only the most current laser scan is added to the original map of the environment.

Adding currently sensed obstacles to the environment map before updating the trajectory not only accounts for unmapped obstacles, moderate localization errors are treated by this technique, too. A displaced position estimate results in accordingly expanded obstacles within the sensor field of view that are then safely circumvented by the updated trajectory. Extending sensor incorporation to not only add obstacles but also mark unexpectedly clear areas of the environment would further improve the system's capabilities to cope with localization errors but is not addressed in this work.

4.3.3 Subdivision of Long Paths

For planning situations where the path to the goal is long, planning the complete path takes a lot of time, as the computational expenses are linear in path length and number of waypoints (see Section 4.2.4). Furthermore, motion planning beyond the sensors' field of view is of limited use only.

Therefore, a trajectory is only planned for the first part of a long path. During plan updates, new trajectory segments are appended to the existing trajectory with smooth join, i.e., with continuity in heading, curvature, and velocity. This way, a smooth path from start to goal is computed iteratively over time in an efficient way.

4.3.4 Generating an Updated Trajectory

The basic idea for the generation of an updated trajectory is to determine the state of the robot, which consists of its current position, velocity, and acceleration, at the point in time that corresponds to the start of the new trajectory. From this state's position a new set of waypoints is requested as a basis for trajectory generation as described earlier in this chapter. Two aspects require attention herein: firstly, the robot state estimation has to be corrected according to the current localization and secondly, a smooth fitting of the updated trajectory to the existing one has to be realized.

We will now develop several aspects of this smooth fitting.

4.3.4.1 Continuity of Derivatives

To ensure continuity in tangent direction and curvature, continuity in first and second derivative (i.e., C^2 continuity) is a sufficient condition. The use of Quintic Bézier Splines allows us to set arbitrary values for the first and second derivative (Eqs. (3.12) and (3.13)), which makes establishing continuity of the derivatives straightforward.

However, it is also possible to achieve tangent direction and curvature continuity with less strict demands.

4.3.4.2 Geometric Continuity of Derivatives

Achieving tangent direction and curvature continuity by geometric continuity instead of parametric continuity of derivatives gives us the possibility to modify the magnitude of the tangent at the new trajectory's starting point. We include this additional dimension into the optimization method. By that, it is possible to straighten out initial trajectories that start with sharp curves, which is of benefit when establishing velocity profile continuity (see below). In the following, constraints are derived for geometric continuity of the derivatives to yield continuity of tangent direction and curvature.

Let P be the active trajectory and Q the updated one, both at the join point of the trajectories. Instead of planning with exactly continuous first and second derivative, say $Q' = P'$ and $Q'' = P''$, one can also consider planning with geometric continuity, i.e., first and second derivatives should point in the same direction, but magnitudes do not have to be identical (G^2 continuity, c.f. Section 3.1). When taking into account a continuity in curvature (Eq. (3.1)), this results in

$$c_Q = c_P \Leftrightarrow \frac{Q' \times Q''}{|Q'|^3} = \frac{P' \times P''}{|P'|^3}.$$

Assuming geometric continuity in the first and second derivative, i.e., $Q' = aP'$, $Q'' = bP'', \quad a, b > 0$:

$$\begin{aligned}
&\Leftrightarrow \frac{aP' \times bP''}{|aP'|^3} = \frac{P' \times P''}{|P'|^3} \\
&\Leftrightarrow \frac{ab(P' \times P'')}{a^3|P'|^3} = \frac{P' \times P''}{|P'|^3} \\
&\Leftrightarrow \frac{b(P' \times P'')}{a^2|P'|^3} = \frac{P' \times P''}{|P'|^3}
\end{aligned}$$

Now the constraints for the first and second derivative are:

$$Q' = aP'' \tag{4.1}$$

$$Q'' = a^2P''. \tag{4.2}$$

If these constraints are met, the two curves P and Q fit smoothly with G^2 and curvature continuity.

4.3.4.3 Velocity Profile Continuity

In addition to continuity regarding the trajectory's shape, continuity in the planned velocities has to be established as well. There are two situations which require more attention than simply setting the start velocity of the updated trajectory to the required value, which are described in the following.

Obstacle proximity Suppose the critical constraint of the active trajectory's velocity at the join point was obstacle proximity. Furthermore, an error in trajectory execution now leads to an anticipated join point that is closer to the obstacle. In this situation, the velocity required at the join point would not be admissible due to obstacle proximity restrictions.

We resolve this situation by allowing the required velocity to be set at the updated trajectory's starting point but force the velocity profile to decelerate at maximum rate until the limit imposed by nearby obstacles is obeyed.

Unfavorable path shape Depending on how sophisticated the method providing the waypoints is, it might be the case that the waypoints for the new trajectory define an initial trajectory that cannot be traversed starting with the required velocity. An example for this situation is a sharp curve that cannot be slowed down for appropriately. Velocity profile generation is therefore unable to maintain the initially set starting velocity.

This issue can be treated by using the difference between the required velocity and the maximum starting velocity as a cost function for the optimization method. The method will then modify the trajectory's shape to allow a higher starting velocity. Once it is possible to set the required velocity, the cost function can be switched back to the normal one.

Note that when treating starting velocity discontinuities with the optimization method, the discontinuity is mostly resolved by elongating the tangent at the start point using geometric continuity as presented above. For this specific case it therefore makes

sense to *not* switch to the next optimization dimension as soon as an improvement was observed but to fully optimize the start tangent's magnitude before proceeding (compare Section 4.2.2).

5 Experiments

The previous chapter has already presented a simulation based validation of the optimization method and its applicability to the search space. Having found that the optimization method is suitable and the search space well behaved, in this chapter we assess the performance of the overall system on real robots.

A number of experiments were conducted to evaluate the performance of our spline based method including a comparison to a Dynamic Window-based approach.

5.1 Experimental Setup

The robots used for our experiments are Albert, a RWI B21r robot with synchro drive, and Friedrich, a differential drive Pioneer 2 system, shown in Fig. 5.1. For environment sensing we rely on laser range finders that both robots are equipped with.

Software The implementation of our approach is integrated into the CARMEN robotics framework [22]. Waypoints for the initial trajectory are supplied by the framework's value iteration based path planner [35].

To execute the planned trajectories we rely on a dynamic feedback linearization controller presented by Oriolo et al. [23]. Note that as this controller relies on odometry readings as error feedback, its performance directly depends on the quality and frequency of odometry readings. It is, like all controllers of its kind, intended for use with robotic hardware that supplies frequent, reliable odometry readings with accurate time stamps.

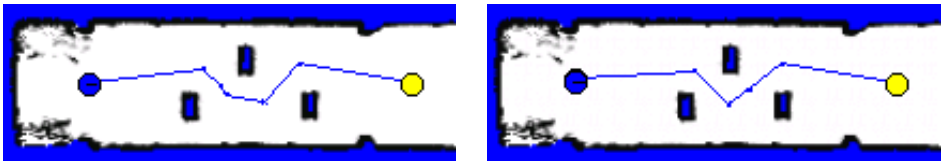
We compare our method to an implementation of the Dynamic Window Approach (DWA) [8]. For the spline based system used for the experiments, only the first four waypoints are used, longer paths are cut after the fourth waypoint. The time assigned to planning and optimizing a trajectory is 0.4 seconds. Furthermore, updating of trajectories is triggered approximately every second. As mentioned in Section 4.3.3 we thereby account for goal locations that are beyond the sensors' field of view by iterative replanning.

Experimental situations Experiments have been conducted in two different situations shown in Fig. 5.2. The maps show a hallway of building 079 of the Department for Computer Science, University of Freiburg. Three boxes are used as obstacles in two different configurations. The one in Fig. 5.2a is referred to as "normal obstacles", the situation depicted by Fig. 5.2b is called "close obstacles".

The situations have been designed to correspond to a hard setting, where balancing out travel velocity, path length, and obstacle distance is crucial and difficult. In the situation with close obstacles, the most narrow part of the map is the passage between the leftmost box and the one in the middle. When subtracting a safety margin of 5 cm, within which the collision avoidance immediately stops the robot, the side distance to

(a) Albert (b) Friedrich

Figure 5.1: Robots used for experiments. Synchro drive robot Albert and differential drive robot Friedrich.



(a) Normal obstacles (b) Close obstacles

Figure 5.2: Experiment situations. Hallway of building 079, University of Freiburg. Three boxes are used as obstacles. (a) is referred to as situation with normal obstacles, obstacles in (b) are closer to each other. The robot starting position is marked in blue, the goal is represented by a yellow dot.

	close		normal	
	DWA	Splines	DWA	Splines
avg. traveled distance [m]	6.83	6.62	7.05	6.43
standard deviation [m]	0.13	0.15	0.20	0.18

Table 5.1: Average traveled distance for Albert in situation with close and normal obstacles. This is data for the runs depicted by Fig. 5.3

the closest obstacle is 5 cm for Albert and 12 cm for Friedrich when driving exactly in the middle of the passage.

As the approach has been successfully applied to a wider range of situations in simulations, we chose one of the hardest settings to be evaluated on real robots.

5.2 Results

In the situations introduced above, several runs were executed for the spline based and the Dynamic Window based approach with both robots, Albert and Friedrich. A general observation for the spline based approach is that the paths driven by the robots were smooth and the updating of trajectories was not perceivable by a human observer, i.e., updated trajectories did fit very smoothly.

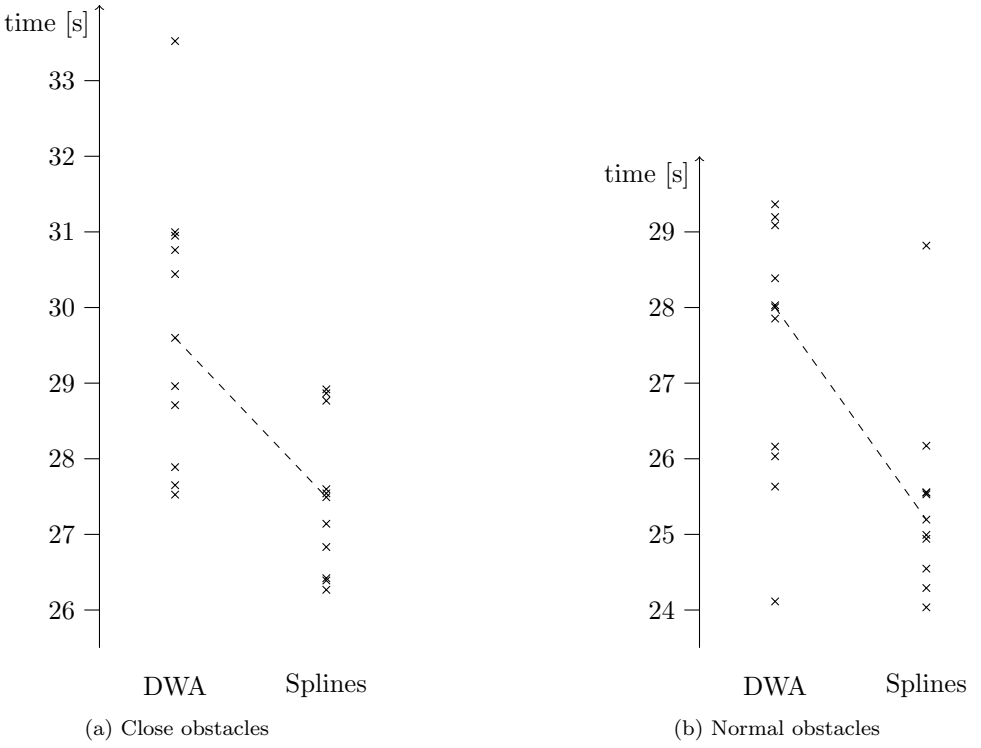
More details on the runs will now be presented along with a comparison to the Dynamic Window-based approach that has been evaluated on exactly the same situations as the one based on splines.

Albert With the synchro drive robot Albert, 11 runs were executed for both experimental situations and approaches, each. Fig. 5.3 shows the recorded times of travel. Each run is represented by a marker in the scatter plots. The left part of the figure depicts the data for the situation with close obstacles, the right part for the one with normal obstacles.

In general, time of travel is higher for the situation with close obstacles than for the other situation, where obstacles are further apart. This is due to the fact that the robot is required to drive slowly when in proximity of obstacles. In both situations the median time of travel is lower for the spline based approach, the medians are connected by a dashed line in both plots. Furthermore, the variance in time of travel is higher for the Dynamic Window, especially for the situation with close obstacles.

The average distance traveled for the runs presented by Fig. 5.3 is given by Tab. 5.1. In both situations the spline based approach travels a shorter path than the Dynamic Window does. Note that this occurs although both approaches are granted the same minimal distance to obstacles.

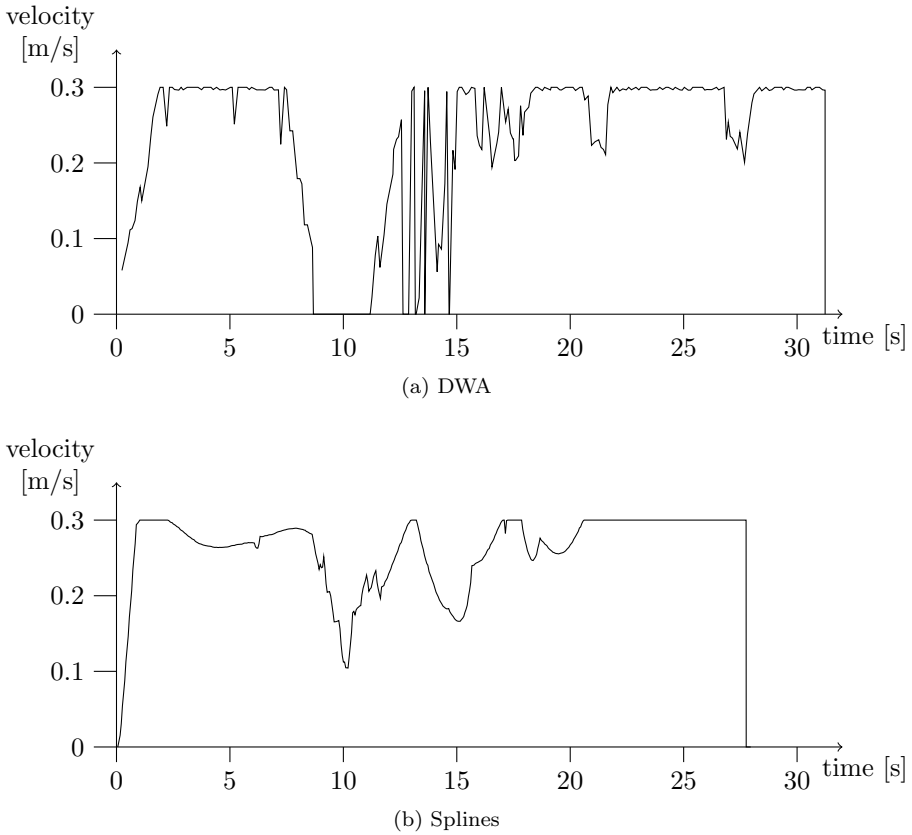
We could observe that the actions executed by the spline based method are smoother than the ones by the Dynamic Window. Support for this observation is given in Fig. 5.4a. It shows the translational velocity commands generated by both approaches on the first particular run in the close obstacles situation with Albert. The commands generated by the spline based method are clearly smoother and lack the heavy oscillation found in the Dynamic Window. For the remaining runs, the plots show similar



(a) Close obstacles

(b) Normal obstacles

Figure 5.3: Results for Albert in two situations with differently spaced obstacles (see Fig. 5.2). Depicted is the total time of travel for 11 runs each. The dashed lines connect the medians of runs with the Dynamic Window approach and the proposed spline based method, respectively. With the proposed method the robot reaches the goal faster and the total time of travel shows lower variance, especially in the situation with close obstacles (a). Note that the time axes have been cropped.



(a) DWA

(b) Splines

Figure 5.4: Translational velocity commands sent to Albert by (a) the Dynamic Window approach and (b) the proposed spline based method. The data is taken from a run for the close obstacles situation (see Fig. 5.2b). Note that the abrupt decelerations at the goal are caused by the global path planner which stops the robot when in vicinity of the goal.

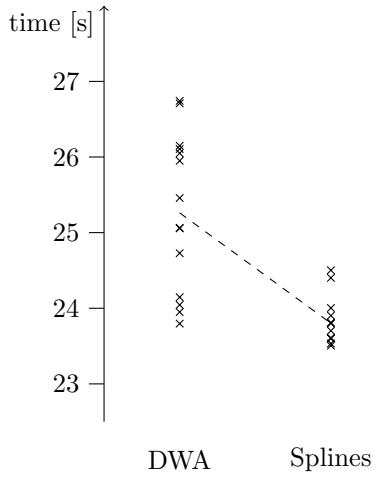


Figure 5.5: Results for Friedrich in the close obstacles situation. The scatter plot shows the total time of travel for every run (14 for each approach). The dashed line connects the median of runs with the Dynamic Window approach and the proposed spline based method, respectively. Note that the time axis has been cropped.

differences. The abrupt deceleration at the end is due to the global path planner which stops the robot when it is near the goal. Note that this suppresses the smooth decelerations that the spline based approach originally plans (see for instance Fig. 3.8).

It has to be noted that the implementation of the Dynamic Window does not employ any smoothing. Of course this can be done to receive smoother velocity commands. However, by smoothing the commands one also "smoothens" the constraints that cause the abrupt commands, i.e., one loosens the constraints as their fulfillment is delayed by the command smoothing.

In the situation with close obstacles, the Dynamic Window-based approach using Albert showed the following behavior: right before passing through the first two boxes (most narrow part of the path) it stops and turns left and right on the spot before entering the narrow passage between the two boxes. This can also be seen in Fig. 5.4a, where translational velocities drop to 0 around the time of 10 seconds. The behavior is caused by the fact that the Dynamic Window is not looking into the future and can therefore not easily find and head towards the narrow corridor of admissible velocities leading through the passage.

Friedrich For the differential drive robot Friedrich, experiments were only conducted for the situation with close obstacles. The observations match those made for Albert, the difficulties of the Dynamic Window however were less evident, yet still observable to a certain degree. This is due to the lower diameter of Friedrich in comparison to Albert which renders the narrow passage less constrained.

Fig. 5.5 shows the times of travel for the runs conducted in the close obstacles situation, the corresponding average traveled distances are presented in Tab. 5.2. As

	DWA	Splines
avg. traveled distance [m]	6.64	6.13
standard deviation [m]	0.07	0.06

Table 5.2: Average traveled distance for Friedrich in situation with close obstacles. This is data for the runs depicted by Fig. 5.5

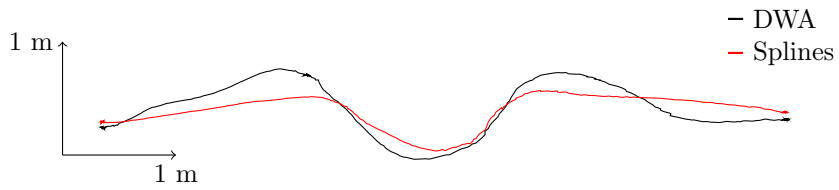


Figure 5.6: CARMEN localization output for the first run of Friedrich in the close obstacle situation (Fig. 5.2b) for the Dynamic Window (black) and the spline based approach (red), respectively. Note that the actually driven path is smoother than the localization output which is subject to errors.

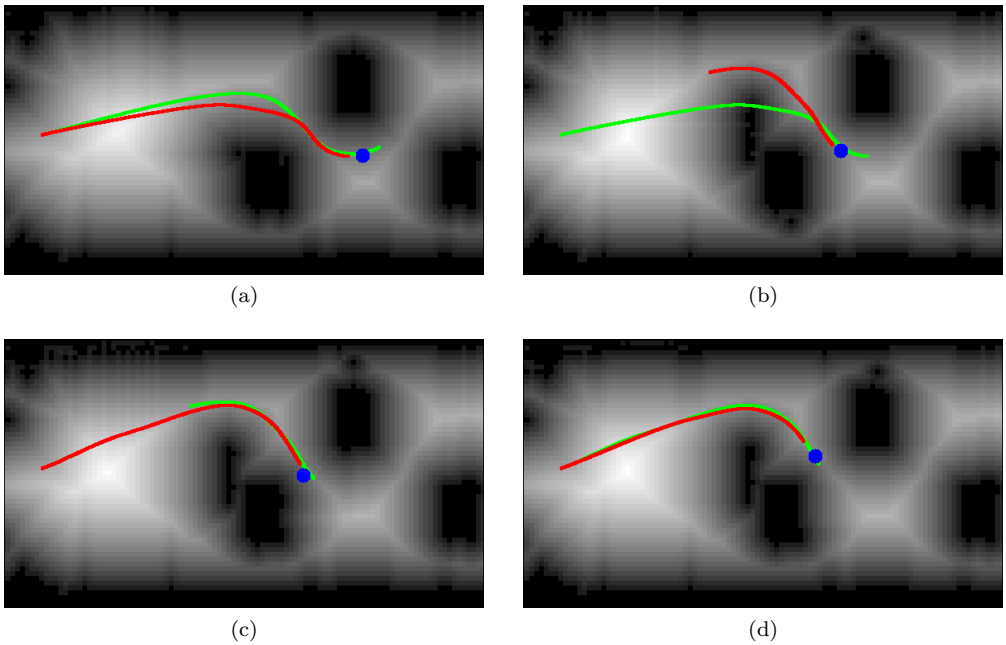
mentioned above, the spline based approach yields lower times of travel together with shorter overall pathlength. For the particular first runs, Fig. 5.6 shows the output of the CARMEN localization for the Dynamic Window approach and the spline based method, respectively. One can observe how the curves taken by the spline based method are less sharp and the curves are in general less distinct, yielding the reported lower traveled distance (compare Tab. 5.2). Note that the noise in the shown trajectories is caused by the localization method, the actually driven paths are smooth.

5.2.1 Unmapped Obstacles

To test how our approach accounts for unmapped obstacles, we conducted several runs with Friedrich in the close obstacles situation (Fig. 5.2b). While the robot was traveling from right to left in the map, we confronted it with an unmapped obstacle near the leftmost of the three boxes.

As Fig. 5.7 shows, the robot successfully evaded the obstacle with a smoothly fitting updated trajectory. The figure visualizes the effect of replanning in four subsequent situations. The blue circles stand for the current belief of the CARMEN localization about the robot's position. The currently active trajectory is depicted in green and the result of the triggered trajectory update is shown in red.

In Fig. 5.7b the robot senses the unmapped obstacle that has not been present at the previous timestep (Fig. 5.7a). As a result the updated trajectory circumvents it. The updated trajectory in Fig. 5.7b is shorter than the other trajectories because here the waypoint planner did provide more densely distributed waypoints. Recall that for the experiments we limited the number of processed waypoints to four.



(a) (b)

(c) (d)

Figure 5.7: Reaction to an unmapped obstacle. The plots show new (red) and active (green) trajectories above the current distance map from (a)–(d) in chronological order. The robot travels from right to left. In (b) it senses an unmapped obstacle near the leftmost box and plans an evasive trajectory. The current belief of the CARMEN localization about the robot's position is depicted with a blue circle. The underlying map is the close obstacles situation (Fig. 5.2b), the data was recorded with Friedrich.

5.3 Summary

The experiments have shown that our spline based approach is applicable to real robots, it yields smooth paths with respect to both, velocity and shape. The approach has been successfully applied to a synchro drive and a differential drive robot. Updating the trajectory while executing it is possible seamlessly, enabling the method to cope with unmapped obstacles and accumulating odometry error.

Comparison to a Dynamic Window-based approach shows that paths generated by the spline based method are shorter and can be traversed faster. At the same time they respect the very same constraints – in kinodynamics as well as regarding obstacle distance – and yet yield a smoother execution behavior of the robot. Furthermore, less variance is observed in the spline based runs.

However, we belive that the full potential of the approach could not be displayed by the experiments. As mentioned above, for any controller that relies on odometry feedback it is crucial that these odometry readings are both frequent and accurate. The robots at our disposal for the experiments lack this high quality odometry. First of all, due to system architecture time stamps for the odometry readings do not correspond to the measurement itself but to points in time that variably deviate up to one reading cycle. In addition to that, readings are only available at fairly low frequencies ($\backsim$ 10 Hz for Friedrich, $\backsim$ 20 Hz for Albert). As a last drawback, the quality of the odometry readings is comparatively low for the used robots.

With improved odometry capabilities we expect our approach to perform even better, especially with increasing velocities, as the odometry error's impact increases there. Note that with higher frequency odometry the original Dynamic Window approach is expected to perform worse as its search window shrinks with increasing execution frequency.

6 Discussion

This chapter comments on how the proposed method can be customized and presents possible improvements before it finally gives a conclusion.

6.1 Customization

The idea to optimize a parametric trajectory representation with respect to a cost function is very generic and therefore highly customizable. All parts can be exchanged and modified to suit a specific need or situation.

When starting from the method proposed here, three main parts emerge for customization: cost function, waypoint input, and controller.

In case that travel time is not the only cost measure of interest for a specific application, the cost function for the optimization method is easily changed to correspond to different measures like anticipated energy consumption or overall steering effort. Costs depending on the curvature integral of the spline are easily computed as curvatures are already calculated for velocity profile generation in the current approach.

Also the method that computes the input waypoints for the approach is of course easily exchanged. Instead of using the employed value iteration planner one can think of using Voronoi graphs as a high level planner, as pursued by [29].

An advantage of the proposed method is that the controller used can be easily exchanged. This way one can benefit from progress in control theory and adapt the controller to specific needs. As long as the assumption of independent translational and rotational velocities and accelerations is a valid approximation (see Section 3.2) one can also easily port the method to different drive types this way without changing velocity profile generation.

6.2 Improvements

The proposed method has a lot of potential for further improvement. Apart from implementational issues there are a number of ideas that are able to extend the method's functionality or considerably improve its performance, which we want to present here.

Spline subdivision Bézier Spline segments have the property that they can be efficiently split into two Bézier segments of the same degree that yield the same curve. A derivation and description of this *subdivision property* can be found in [25, Chapt. 3.3]. So far, our method does not take advantage of this property. However, there are two potential applications for it.

Together with the convex hull property (c.f. Appendix A.3) one can implement collision checking in a hierarchical fashion, applying polygon clipping algorithms to the

respective convex hulls that can be iteratively refined to localize a collision. This hierarchical collision checking entails a lot of speed up potential compared to the currently employed system of pointwise collision checking along the trajectory.

The subdivision property can also be exploited for re-using previously planned trajectories when planning a new trajectory. The current system always computes a completely new trajectory starting at the replanning point. Similar to seeding the queue in A* based algorithms, one can re-use existing segments – or with the help of the subdivision property even parts of them – and append new segments. This speeds up the planning process if already existing trajectory parts are still valid and admissible.

Trajectory updating In the current implementation updating of the trajectory is done in a relatively naïve fashion at a fixed time interval. A more sophisticated way would consider the current state and take different appropriate actions. It would distinguish whether the environment drastically changed or the robot is just a bit off the trajectory due to odometry error. Smart trajectory updating could therefore decide whether to reuse the existing trajectory and expand the planning horizon or to discard the current trajectory and plan an evasive maneuver. This would put computational resources to a more efficient use than repeatedly recalculating most part of the trajectory.

When extending the planning horizon by this method one can also exploit the localism of Bézier Splines to speed up calculations, as changes to a spline segment only affect a limited neighborhood.

Time dependent maps As the proposed method generates time parameterized trajectories it can be extended to use a time dependent map to navigate among obstacles for which movements can be anticipated.

Obstacle distance When computing the velocity limit that nearby obstacles impose on the robot, the current method does not take into account the robot's heading. This way the robot is also forced to drive at a low speed when moving away from obstacles. Accounting for the robot's heading would remove this unnecessary constraint. Furthermore, the system can be extended to account for the robot's actual shape instead of approximating it by a circle.

Spline representation In the current approach, splines are subject to discretization by numerical integration and the planning points. Therefore, also curvature extrema are subject to discretization. These and other extrema can be computed analytically, if splines are represented as done by Connors and Elkaim [3]. They construct a separate coordinate reference frame for each segment and restrict the shape of a segment to a polynomial of the form $\mathbb{R} \to \mathbb{R}$. This can be exploited to minimize discretization effects by placing planning points at curvature extrema.

Furthermore, integration of polynomials that have a one dimensional range and domain can be done analytically to avoid discretization errors that accompany numerical integration.

6.3 Conclusion

We have proposed a method for wheeled mobile robot motion planning that features a parametric representation of the planned actions and thereby overcomes the limitations imposed by finite action sets, which a majority of the current work has to cope with.

The suggested approach plans curvature continuous paths in a time parametric fashion that allows prediction of the robot's position and kinematic state for any given point in time along the trajectory. Therefore, a variety of controllers can be used to execute the trajectory, which allows for precise navigation through a small odometry feedback loop. Furthermore, optimization is employed to receive trajectories that are near optimal with respect to a cost function such as the time of travel.

The input to the proposed method is constituted by a map and a set of sparse waypoints out of which an admissible trajectory is created instantly (anytime capability). The initial trajectory is then subject to optimization in a continuous parameter space for the remainder of the designated planning time. After optimization the time parametric trajectory together with its first two derivatives are forwarded to a controller for execution.

In contrast to Cubic Bézier Splines the Quintic variant is able to provide the desired properties for a trajectory representation, which are localism, curvature continuity, and a strong correlation between control points and shape. The additional degrees of freedom in form of second derivatives at start and end point are handled by a heuristic that mimics the minimizing behavior of Cubic Bézier Splines. Furthermore, Bézier Splines provide the subdivision property which is of use for future extensions of the method.

We have shown how – under certain assumptions – the fastest constraint respecting velocity profile can be computed for a trajectory without resorting to iteration to resolve interdependencies between translational and rotational acceleration constraints.

The RPROP based optimization of trajectories worked well in both simulation and experiments on real robots. It not only shortens the length of the planned trajectory but also permits faster traversal of the latter by deforming it to reduce the constraints' impact as has been seen in Fig. 3.8. This has positive effects on the smoothness of driving behavior.

Our experiments have shown the potential of the proposed method and its real time applicability to robots: the approach works on both synchro and differential drive robots and is able to smoothly update plans to avoid unmapped obstacles. In comparison, the spline based method yields faster and smoother trajectories than a Dynamic Window-based approach. Part of the advantage is due to the lookahead our method performs, but even if a lookahead mechanism was added to the Dynamic Window, it would still suffer from the negative effects of a finite action set.

As a first step to a more precise navigation in constrained spaces the majority of current work has decoupled trajectory generation from execution. While we follow this step, we also believe that a necessary next step is to abandon finite action sets.

Finite action sets and their optimal discretization receive substantial attention in current state of the art work. A lot of effort is put into keeping the size of the action set in bounds, i.e., keeping the resulting search tree tractable, and at the same time spanning the relevant action space.

Our work is able to overcome the limitations of discretization by the choice of a parametric trajectory representation instead of the predominant finite action sets.

Furthermore, we argue that the employment of an optimization method that only provides near optimal results is not to be considered a major drawback regarding optimality compared to existing approaches. Firstly, existing A* based search approaches on 2D gridmaps have to abandon their optimality claims when their solutions are taken to the real, usually higher dimensional and continuous problem domain that involves kinematics and dynamics. Secondly, a strong discretization of the action space and the use of approximate or heuristic costs challenge optimality claims with respect to the original problem, although justified in the discrete subproblem regarding the used costs.

We therefore recommend the presented application of an optimization in a continuous parameter space with respect to the costs of interest. This is a goal oriented approach to trajectory generation when exhaustive search is prohibitive and heuristics on their own do not yield sufficient results. In addition to that, explicit planning for velocities along the trajectory is a prerequisite for consideration of kinodynamics along the path.

The use of a trajectory that defines position, velocity, and acceleration over time not only facilitates the use of state of the art controllers, it also opens the door to the domain of planning with time dependent maps. Solutions are within reach that can efficiently avoid obstacles that follow trajectories which are known to a certain degree.

List of Figures

Bibliography

[1] K. O. Arras, R. Philippsen, N. Tomatis, M. de Battista, M. Schilt, and R. Siegwart. A navigation framework for multiple mobile robots and its application at the Expo.02 exhibition. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003.

[2] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An introduction to splines for use in computer graphics & geometric modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[3] J. Connors and G. Elkaim. Analysis of a spline based, obstacle avoiding path planning algorithm. In *Proc. IEEE 65th Vehicular Technology Conference, VTC2007-Spring*, pages 2565–2569, 22–25 April 2007.

[4] J. Connors and G. Elkaim. Manipulating B-Spline based paths for obstacle avoidance in autonomous ground vehicles. In *Proc. of the 2007 National Technical Meeting of the Institute of Navigation*, pages 1081–1088, San Diego, California, 2007.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[7] J. D. Foley, R. L. Phillips, J. F. Hughes, A. van Dam, and S. K. Feiner. *Introduction to Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[8] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, March 1997.

[9] H. González-Baños, D. Hsu, and J. Latombe. Motion planning: Recent developments. In S. Ge and F. Lewis, editors, *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, chapter 10. CRC Press, 2006.

[10] S. Gulati and B. Kuipers. High performance control for graceful motion of an intelligent wheelchair. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, pages 3932–3938, 19–23 May 2008.

[11] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon. Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1444–1449, 2003.

[12] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[13] I. Kolmanovsky and N. McClamroch. Developments in nonholonomic control problems. *IEEE Control Systems*, 15(6):20–36, Dec. 1995.

[14] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[15] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[16] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proc. of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[17] J. Loustau and M. Dillon. *Linear Geometry with Computer Graphics*. CRC Press, 1992.

[18] A. Luca and G. Oriolo. *Kinematics and Dynamics of Multi-Body Systems*, chapter Modelling and control of nonholonomic mechanical systems. Springer-Verlag, Wien, 1995.

[19] K. Macek, A. Vasquez, T. Fraichard, and R. Siegwart. A hierarchical approach to safe vehicle navigation in dynamic urban scenarios. In *10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2008. To appear.

[20] E. Magid, D. Keren, E. Rivlin, and I. Yavneh. Spline-based robot navigation. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2296–2301, Oct. 2006.

[21] C. Mandel and U. Frese. Comparison of wheelchair user interfaces for the paralysed: Head-joystick vs. verbal path selection from an offered route-set. In *Proc. of the 3rd European Conference on Mobile Robotics (ECMR)*, 2007.

[22] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) Toolkit. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 3, pages 2436–2441, 27–31 Oct. 2003.

[23] G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, Nov. 2002.

[24] R. Plato. *Concise Numerical Mathematics*, volume 57 of *Graduate studies in mathematics*. American Mathematical Society, 2003.

[25] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline techniques*. Springer, 2002.

[26] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proc. of the International Conference on Robotics and Automation*, pages 802–807, 1993.

[27] M. Riedmiller. Rprop – description and implementation details. Technical report, University of Karlsruhe, January 1994.

[28] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE International Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[29] A. Sahraei, M. T. Manzuri, M. R. Razvan, M. Tajfard, and S. Khoshbakht. *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, chapter Real-Time Trajectory Generation for Mobile Robots, pages 459–470. Springer, 2007.

[30] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced textbooks in control and signal processing. Springer, 2nd edition, January 2005.

[31] Z. Shiller and Y. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7:241–249.

[32] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and System*, volume 1, pages 508–513, 30 Sept.–5 Oct. 2002.

[33] J. Stewart. *Calculus: Early Transcendentals*. Brooks Cole, Pacific Grove, 5th edition, 2002.

[34] S. Thompson and S. Kagami. Continuous curvature trajectory generation with obstacle avoidance for car-like robots. In *Proc. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 1, pages 863–870, 28–30 Nov. 2005.

[35] S. Thrun and A. Bucken. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, 1998.

[36] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, June 2006.

[37] K. I. Tsianos, I. A. Sucan, and L. E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1:2–11, August 2007.

[38] A. H. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley Publishing Company, Inc., New York, 1992.

[39] J. Ziegler, M. Werling, and J. Schröder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *IEEE Intelligent Vehicles Symposium (IV 08)*, 2008.

A Splines

This chapter introduces splines as used throughout this work and presents several cubic spline families along with a discussion of their key properties.

A.1 Introduction

We introduce a spline as a *piecewise polynomial parametric curve* $Q(t)$. In general splines can exist in other spaces as well, for this application however, we restrict them to live in a two-dimensional plane: $Q(t) \in \mathbb{R}^2$. Note that nevertheless most of the following is true in arbitrary spaces. Typically one considers splines whose segments are polynomials of identical degree n.

There are three commonly used ways in the literature to define the polynomial segments of a spline. We will briefly introduce them with the example of a cubic segment with domain $\mathbb{R}^2$.

Polynomial in parameter t The most straightforward way is of course the standard notation for a polynomial. The segment $S(t)$ can be expressed by individual cubic polynomials for each dimension, as the following equation shows:

$$\begin{aligned} S(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} &= \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \end{pmatrix} \\ &= \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{pmatrix} = T \cdot C \end{aligned} \tag{A.1}$$

Eq. (A.1) defines the so called *coefficient matrix* C which serves as the characterization of a specific cubic curve in this notation. Furthermore, it introduces the abbreviation T for the vector containing the parameter's powers. Note that in the literature also a reversed variant of the parameter vector is used, e.g., [2]. We adopt the notation used in [6, 7], and [38, Chapt. 3.5.2].

Weighted sum of control vertices $S(t)$ can also be expressed as a weighted sum of control vertices $p_i \in \mathbb{R}^2$, for cubic polynomials there are four of them:

$$S(t) = \sum_{i=0}^{3} p_i B_i(t) \tag{A.2}$$

This notation allows the interpretation of the cubic polynomial as a *linear combination of the control vertices* $p_i \in \mathbb{R}^2$, weighted by the so called *basis splines* $B_i(t)$, which are cubic polynomials. A particular segment is here characterized by its control vertices

whereas the basis splines can be interpreted as defining the family of segments it belongs to.

Product of matrices Combining the ideas of the precedent two notations one can also express a segment through the product of three matrices and vectors, respectively:

$$S(t) = T \cdot M \cdot G \tag{A.3}$$

T is the vector containing the parameter's potencies as introduced in Eq. (A.1). Further comparison with the definition in Eq. (A.1) shows that the coefficient matrix C has been split into a *basis matrix* M and a *geometry matrix* G by $C = M \cdot G$. Defined as

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix}, \; G = \begin{pmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{pmatrix} \tag{A.4}$$

the geometry matrix characterizes the segment by four control vertices $\{G_i |\, i \in 0 \dots 3\}$ whereas the basis matrix can be regarded to define the family the curve belongs to. Having a look at Eq. (A.2) again, one realizes that the product of the parameter vector $T = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix}$ with the basis matrix M is identical with a vector containing the basis splines from Eq. (A.2)

$$T \cdot M = \begin{pmatrix} m_{00}t^3 + m_{10}t^2 + m_{20}t + m_{30} \\ m_{01}t^3 + m_{11}t^2 + m_{21}t + m_{31} \\ m_{02}t^3 + m_{12}t^2 + m_{22}t + m_{32} \\ m_{03}t^3 + m_{13}t^2 + m_{23}t + m_{33} \end{pmatrix}^t,$$

when identifying control vertices p_i with G_i. A^t denotes the transposed of A.

We will now introduce some well known spline types for later discussion about their suitability for trajectory planning. Note that a huge domain of theory that uses splines for interpolation is of no use for us, as it prescribes the location of the sampling points. In this application however, we deal with sparse sampling points at fixed locations.

For the following splines the parameter t lies in $[0, 1]$ for each segment $S(t)$. We do not introduce a unified notation for mapping the global spline parameter t in $Q(t)$ to the segment parameters in order to perform segment selection at this point.

A.2 Cubic Hermite Spline

Intuition A Cubic Hermite Spline consists of segments that are cubic polynomials. Each segment connects two control points. The two remaining degrees of freedom are controlled by specifying the tangent vectors at both, the start and the end point of the segment. At join points, the tangent vectors are manually set to match, resulting in a C^1 continuous spline.

Definition The following definitions are derived in [6, Sect. 11.2.1]. Note that, deviating from conventions in mathematics, we use 0 as the first index to comply with

notation used in Computer Science. The *Hermite geometry matrix* is defined as

$$G_H = \begin{pmatrix} P_0 \\ P_3 \\ R_0 \\ R_3 \end{pmatrix}, \tag{A.5}$$

where P_0 is the start point of a segment and P_3 specifies its end point. R_0 and R_3 are the tangent vectors at the start and the end point, respectively. We follow the somewhat unintuitive notation of [6] for the same reason they introduced it: consistency with notation that will be introduced later on. The *Hermite basis matrix* evaluates to

$$M_H = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{A.6}$$

Now, with Eq. (A.3), the Hermite form segment can be noted as

$$S(t) = T \cdot M_H \cdot G_H. \tag{A.7}$$

This equation can easily be transformed into the corresponding instance of Eq. (A.2):

$$S(t) = (2t^3 - 3t^2 + 1)P_0 + (-2t^3 + 3t^2)P_3 + (t^3 - 2t^2 + t)R_0 + (t^3 - t^2)R_3, \tag{A.8}$$

which yields the vector of the *Hermite basis splines*

$$B_H = \begin{pmatrix} 2t^3 - 3t^2 + 1 \\ -2t^3 + 3t^2 \\ t^3 - 2t^2 + t \\ t^3 - t^2 \end{pmatrix}^t. \tag{A.9}$$

To get the polynomial formulation of our segment, we compute the coefficient matrix $C_H = M_H \cdot G_H$ and proceed according to Eq. (A.1). We get:

$$C_H = \begin{pmatrix} 2P_0 - 2P_3 + R_0 + R_3 \\ -3P_0 + 3P_3 - 2R_0 - R_3 \\ R_0 \\ P_0 \end{pmatrix} \tag{A.10}$$

and therefore

$$S(t) = (2P_0 - 2P_3 + R_0 + R_3)t^3 + (-3P_0 + 3P_3 - 2R_0 - R_3)t^2 + R_0 t + P_0. \tag{A.11}$$

The drawback of the Hermite form for the spline's segments is that the tangent specification by vectors somewhat differs from the specification of the start and end points by position vectors. Furthermore the relation of the tangents' directions and magnitudes to the segment's shape is not very obvious. These considerations lead to the definition of Cubic Bézier Splines.

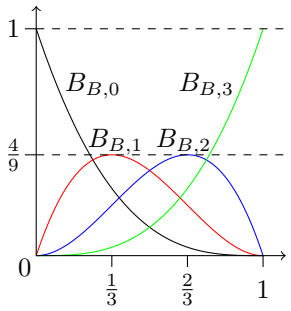


Figure A.1: The basis splines of a Cubic Bézier Spline's segment, also known as the Bernstein Polynomials of third degree. For a definition see Eq. (A.16).

A.3 Cubic Bézier Splines

Intuition Cubic Bézier Splines also consist of cubic polynomials and therefore have the same expressive power as the Cubic Hermite Splines. They differ from Hermite Splines in the formulation of the segments. Instead of explicitly defining the tangent vectors at start and end point, these vectors are extracted from two additional control vertices in the case of Cubic Bézier Splines' segments. C^1 continuity can be reached for Cubic Bézier Splines by arranging the control vertices for the tangents in an appropriate way.

Definition The definitions stated here can be found in [6, Chapt. 11.2.2] together with a derivation. Again we adapt indexing to Computer Science's conventions. The *Bézier geometry matrix* is

$$G_B = \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} \tag{A.12}$$

and the *Bézier basis matrix* is

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{A.13}$$

Following Eq. (A.3) we get the Bézier segment defined by the four control vertices in G_B as

$$S(t) = T \cdot M_B \cdot G_B. \tag{A.14}$$

Further transformation yields a formulation with the help of the Bézier basis splines

$$B_B = \begin{pmatrix} B_{B,0} \\ B_{B,1} \\ B_{B,2} \\ B_{B,3} \end{pmatrix}^t = \begin{pmatrix} -t^3 + 3t^2 - 3t + 1 \\ 3t^3 - 6t^2 + 3t \\ -3t^3 + 3t^2 \\ t^3 \end{pmatrix}^t = \begin{pmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{pmatrix}^t, \tag{A.15}$$

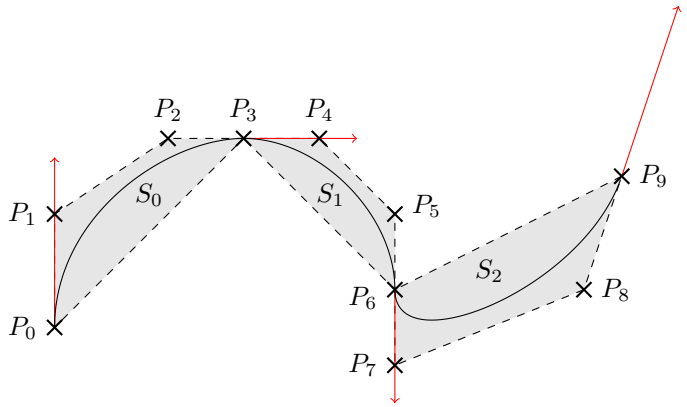


Figure A.2: Cubic Bézier Spline consisting of three segments S_0, S_1, and S_2. The tangents at the join points are depicted in red with half their magnitude and the control points have been arranged for C^1 continuity. The grey areas show the convex hull of the respective segment's control points and thereby the convex hull property of a Cubic Bézier Spline.

which are also known as the Bernstein polynomials of third degree:

$$S(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3. \tag{A.16}$$

They have the property to always sum up to 1 and to never leave the interval $[0, 1]$:

$$\sum_{i=0}^{3} B_{B,i} = 1, \quad B_{B,i} \in [0,1],\ i = 0 \ldots 3. \tag{A.17}$$

As a consequence the linear combination of the control vertices is an interpolation of the control vertices, a property which is referred to as *convex hull property*. Fig. A.1 provides a visualization of the basis splines. Calculating the *Bézier coefficient matrix* from M_B and G_B,

$$C_B = \begin{pmatrix} -P_0 + 3P_1 - 3P_2 + P_3 \\ 3P_0 - 6P_1 + 3P_2 \\ -3P_0 + 3P_1 \\ P_0 \end{pmatrix}, \tag{A.18}$$

enables us to state the instantiation of Eq. (A.1) for a Cubic Bézier segment, the formulation as polynomial in parameter t:

$$S(t) = (-P_0 + 3P_1 - 3P_2 + P_3)t^3 + (3P_0 - 6P_1 + 3P_2)t^2 + (-3P_0 + 3P_1)t + P_0. \tag{A.19}$$

Derivatives Eq. (A.19) makes it easy to state the first and second derivatives of a Cubic Bézier Segment:

$$S'(t) = (-3P_0 + 9P_1 - 9P_2 + 3P_3)t^2 + (6P_0 - 12P_1 + 6P_2)t - 3P_0 + 3P_1 \tag{A.20}$$

$$S''(t) = (-6P_0 + 18P_1 - 18P_2 + 6P_3)t + 6P_0 - 12P_1 + 6P_2 \tag{A.21}$$

For the start and end point of a segment, these evaluate to

$$S'(0) = 3(P_1 - P_0), \qquad S'(1) = 3(P_3 - P_2), \tag{A.22}$$

$$S''(0) = 6(P_0 - 2P_1 + P_2), \qquad S''(1) = 6(P_1 - 2P_2 + P_3). \tag{A.23}$$

These equations clarify the relation between the inner control points of a segment and its derivatives at the start and end point, respectively. Fig. A.2 shows Cubic Bézier Spline with three segments. The light gray polygons visualize the convex hull property of the segments: as the Bézier basis splines compute a weighted average of the control vertices, each point of the segment is contained in the convex hull of its control vertices, i.e., the polygon created by them. For the spline in the figure, the segments' inner control points have been arranged to yield C^1 continuity.

This is achieved by establishing

$$\begin{aligned} & S_i'(1) = S_{i+1}'(0) \\ \Leftrightarrow\ & 3(P_{3i+3} - P_{3i+2}) = 3(P_{3(i+1)+1} - P_{3(i+1)}) \\ \Leftrightarrow\ & P_{3i+3} - P_{3i+2} = P_{3i+4} - P_{3i+3}. \end{aligned}$$

A.4 Catmull-Rom Splines

So far, to reach C^1 continuity it has been necessary to manually adjust the tangents at the join points. Catmull-Rom Splines overcome this by being inherently C^1 continuous.

We will introduce Catmull-Rom Splines according to [6, Chapt. 11.2.6], [38, Chapt. 3.10]. Note that [2] uses the term to refer to a more generalized family of splines that interpolate vector valued functions instead of control vertices. The splines named Catmull-Rom Splines here and in [6] are a member of this family, see [2, Chapt. 21.4] for a definition of this family and [2, p. 427] for a remark on the membership relation.

Intuition A Catmull-Rom Spline is defined by a sequence of control points $P_0, \ldots, P_m$. It passes through the control points $P_1, \ldots, P_{m-1}$ and the tangent vector at point P_i is parallel to the vector $P_{i-1}P_{i+1}$. Catmull-Rom Splines are designed for inherent C^1 continuity and are composed of segments that are cubic polynomials. To reach C^1 continuity, control points are shared between segments to a certain degree.

The curve segment $S_i(t)$ connects the control points P_{i-2} and P_{i-1} via $S_i(0) = P_{i-2}$ and $S_i(1) = P_{i-1}$. However, it is defined by the control points $P_{i-3}, \ldots, P_i$, the additional points are used to determine the tangent at P_{i-2} and P_{i-1}, respectively. As a consequence, the very first and last control point of a sequence are not connected to the curve, it reaches from P_1 to P_{m-1}.

A.4.0.1 Definition

The i-th segment ($i \in [3, m]$) of a Catmull-Rom Spline defined by control points $P_0, \ldots, P_m$ is given by

$$\begin{aligned} S_i(t) &= T \cdot M_{CR} \cdot G_{CR}^i \\ &= \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \frac{1}{2} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix}. \end{aligned} \tag{A.24}$$

The vector containing the basis splines can be derived as

$$B_{CR} = \frac{1}{2} \begin{pmatrix} -t^3 + 2t^2 - t \\ 3t^3 - 5t^2 + 2 \\ -3t^3 + 4t^2 + t \\ t^3 - t^2 \end{pmatrix}^t . \tag{A.25}$$

We realize that although the basis splines sum up to 1, they do exceed the interval $[0, 1]$ and therefore Catmull-Rom Splines do *not* possess the convex hull property.

Calculating the coefficient matrix

$$C^i_{CR} = M_{CR} \cdot G^i_{CR} = \frac{1}{2} \begin{pmatrix} -P_{i-3} + 3P_{i-2} - 3P_{i-1} + P_i \\ 2P_{i-3} - 5P_{i-2} + 4P_{i-1} - P_i \\ -P_{i-3} + P_{i-1} \\ 2P_{i-2} \end{pmatrix} \tag{A.26}$$

allows us to write Equation (A.24) as a polynomial:

$$\begin{aligned} S_i(t) =& \frac{1}{2} \left(-P_{i-3} + 3P_{i-2} - 3P_{i-1} + P_i\right) t^3 \\ &+ \frac{1}{2} \left(2P_{i-3} - 5P_{i-2} + 4P_{i-1} - P_i\right) t^2 \\ &+ \frac{1}{2} \left(-P_{i-3} + P_{i-1}\right) t + P_{i-2}. \end{aligned} \tag{A.27}$$

Derivatives Now, by inserting 0 for the start and 1 for the end point, respectively, we obtain

$$S_i(0) = P_{i-2} \qquad S_i(1) = P_{i-1} \tag{A.28}$$

$$S'_i(0) = \frac{1}{2} \left(P_{i-1} - P_{i-3}\right) \qquad S'_i(1) = \frac{1}{2} \left(P_i - P_{i-2}\right) \tag{A.29}$$

and thereby confirm the C^1 continuity, as $S_{i-1}(1) = S_i(0)$ and $S'_{i-1}(1) = S'_i(0)$. Furthermore, we see that the tangent at point P_i is parallel to the vector $P_{i-1}P_{i+1}$. Calculating the second derivative for a segment of a Catmull-Rom Spline will reveal that they are generally *not* C^2 continuous.

Catmull-Rom Splines guarantee C^1 continuity at the cost of strictly prescribing tangent direction and magnitude at every joint point.

A.5 B-Splines

For the above presented spline classes we were unable to reach our initial goal of C^2 continuity. B-Splines inherently possess this property. However, this comes at the additional cost of not passing through any of their control points.

Intuition Like Catmull-Rom Splines, to ensure continuity constraints at the joint points, a B-Spline's segments have to share some of their control points. A B-Spline is defined by its control points $P_0, \ldots, P_m$. Each segment is a cubic polynomial, the i-th segment ($i \in [0, m-3]$) is defined by the control points $P_i, \ldots, P_{i+3}$. The following definition can be found in [38, Chapt. 3.5]. A remark that addresses the transformability of B-Splines and Bézier curves can be found in [6, p. 510].

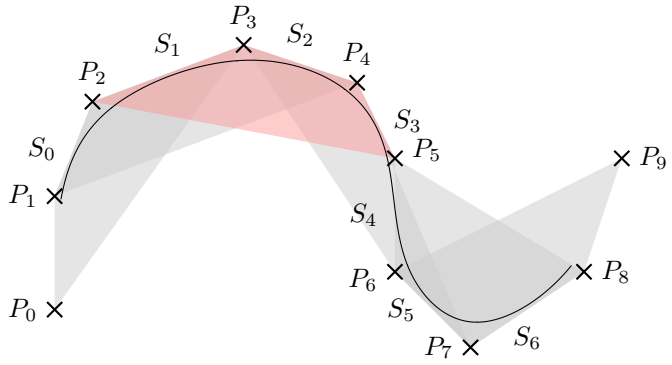


Figure A.3: B-Spline consisting of seven segments $S_0, \ldots, S_6$. The overlapping convex hulls of the segments' control points are depicted in gray, the one of segment S_2 (control points $P_2, \ldots, P_5$) is highlighted in red. It is apparent that a B-Spline in general does not pass through any of its control points.

Definition The i-th segment ($i \in [0, m-3]$) of a B-Spline defined by control points $P_0, \ldots, P_m$ is given by

$$\begin{aligned} S_i(t) &= T \cdot M_{BS} \cdot G^i_{BS} \\ &= \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{pmatrix}. \end{aligned} \tag{A.30}$$

We calculate the vector containing the basis splines

$$B_{BS} = \begin{pmatrix} B_{BS,0} \\ B_{BS,1} \\ B_{BS,2} \\ B_{BS,3} \end{pmatrix}^t = \frac{1}{6} \begin{pmatrix} -t^3 + 3t^2 - 3t + 1 \\ 3t^3 - 6t^2 + 4 \\ -3t^3 + 3t^2 + 3t + 1 \\ t^3 \end{pmatrix}^t, \tag{A.31}$$

where $\sum_{i=0}^{3} B_{BS,i} = 1, \quad B_{BS,i} \in [0,1], \; i = 0, \ldots, 3$, which is the sufficient condition for the convex hull property that was introduced with the Cubic Bézier Splines in Appendix A.3. Fig. A.3 shows an instance of a B-Spline with seven segments. The convex hulls of the segments overlap, the one of segment S_2 (corresponds to control points $P_2, \ldots, P_5$) is highlighted in the figure.

We continue with the calculation of the coefficient matrix

$$C^i_{BS} = M_{BS} \cdot G^i_{BS} = \begin{pmatrix} -P_i + 3P_{i+1} - 3P_{i+2} + P_{i+3} \\ 3P_i - 6P_{i+1} + 3P_{i+2} \\ -3P_i + 3P_{i+2} \\ P_i + 4P_{i+1} + P_{i+2} \end{pmatrix}, \tag{A.32}$$

which leads to the polynomial formulation:

$$\begin{aligned} S_i(t) =& \frac{1}{6}\left(-P_i + 3P_{i+1} - 3P_{i+2} + P_{i+3}\right) t^3 + \frac{1}{6}\left(3P_i - 6P_{i+1} + 3P_{i+2}\right) t^2 \\ &+ \frac{1}{6}\left(-3P_i + 3P_{i+2}\right) t + \frac{1}{6}\left(P_i + 4P_{i+1} + P_{i+2}\right). \end{aligned} \tag{A.33}$$

Derivation is now straightforward and confirms the C^2 continuity. The fact that B-Splines do not pass through any of their control points (compare Fig. A.3) can be revealed by a closer look at the basis splines: none of them ever reaches the value 1, any point on a segment therefore contains information from more than one control point. However, there is a way to enforce passing through control points for a B-Spline: repeated insertion of a control point will lead the spline through this point. The drawback of this method is a resulting discontinuity in the derivatives.

While B-Splines have favorable properties such as the desired C^2 continuity and even the convex hull property, not passing through control points decreases the degree of intuitive correlation between control points and spline shape. We consider this an exclusion criterion.

A.6 Summary

The properties of the cubic spline families presented in this chapter and the ones of Quintic Bézier Splines, which are introduced in Section 3.1.2.3, are summarized by the following table:

	Cubic				Quintic
Property	Hermite	Bézier	Catmull-Rom	B-Spline	Bézier
stitching	manual	manual	inherent	inherent	manual
C^1 continuity	✓	✓	✓	✓	✓
C^2 continuity	-	-	-	✓	✓
visit control points	✓	✓	✓	-	✓
strong correlation	-	✓	✓	-	✓
localism	✓	✓	✓	✓	✓
freely set 1^{st} deriv. start	✓	✓	✓	✓	✓
freely set 2^{nd} deriv. start	-	-	-	-	✓
convex hull	-	✓	-	✓	✓

Only Quintic Bézier Splines can provide all properties as they are constructed from quintic polynomials, whereas the remaining presented spline families resort to cubic polynomials.

B Solving the Overlap Problem

During velocity profile generation, compliance of the planned velocities with translational and rotational acceleration constraints has to be ensured. Due to a combination of curvature change and translational velocity planned for an adjacent planning point, situations can arise where no velocity exists for a planning point that respects both constraints (Section 3.2.7). Suppose for instance a car approaching a curve at a high speed: As descibed in Section 3.2.7, even if skidding is not a problem, it would either have to decelerate quickly or to abruptly change its steering angle in order to stay on the road. If neither is feasible according to the limiting constraints of the car, it will not be able to follow the curve.

More formally, the intervals $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω}, defined by Eqs. (3.29), (3.30), and (3.39), do not overlap in general. This chapter provides a solution to the problem: it derives the biggest upper bound for v_{i-1} that guarantees a non-empty intersection for all velocities smaller than this bound. To ensure that admissible velocity intervals overlap for backward consistency, too, the same bound can be applied in a reverse fashion, assuming motion from p_i to p_{i-1}.

The computation of the bound derived in the following section is summarized by a pseudo code algorithm presented in Appendix B.2. Once the bound has been computed, it can be seamlessly integrated into velocity profile generation as an *additional isolated constraint*.

B.1 Derivation of the Bound

In this section the biggest upper bound for v_{i-1} is derived that guarantees $I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset$.

There are many different possible situations regarding the curvature change between two planning points: curvature can increase or decrease, the curve can be oriented to the left or the right, or it can even change orientation from left to right or vice versa. Due to the occurence of square roots and quadratic expressions in the definitions of I_{a_ω},$v_{min|a_t}$, and $v_{max|a_t}$ (Eqs. (3.29), (3.30), and (3.39) and in consequence Eqs. (3.32), (3.33), and (3.37)), it is important to know the signs of terms during the derivations to conduct them in a mathematically correct way. We therefore have to resort to excessive case differentiation, especially regarding curvature orientation and sign of its change $c_i - c_{i-1}$ between planning points.

In the following case differentiation positive values for curvatures c_{i-1} and c_i correspond to a left curve, negative ones yield a right curve. The cases where $c_i > c_{i-1}$ correspond to an increasing curvature for left curves, for right curves they correspond to a decreasing curvature. For $c_i < c_{i-1}$ these correspondences are swapped. A curvature of zero corresponds to strictly straight movement, cases where the signs of c_{i-1} and c_i differ correspond to transition from a right to a left curve or vice versa.

B.1.1 Case 1, $c_i > 0, c_{i-1} \geq 0$

B.1.1.1 Case 1.1, $c_i > c_{i-1}$

We will first show that $v_2 < 0$, and v_2^* exists $\Rightarrow$ $v_2^* < 0$ which makes both values irrelevant for intersection as $v_{min|a_t} \geq 0$.

From the case definition it follows that $c_{i-1} - c_i < 0$. Looking at the definitions of v_2 and v_2^* in Equations Eq. (3.32), Eq. (3.33) respectively, the claim immediately follows.

The interval that remains to be intersected with $[v_{min|a_t}, v_{max|a_t}]$ is now $[v_1^*, v_1]$ in case v_1^* does exist and $[0, v_1]$ in case it doesn't.

We show that $v_1^* < v_{max|a_t}$: since $v_{max|a_t} > 0$ (Eq. (3.30)), this is clear for $v_1^* < 0$. We can therefore assume $v_1^* \geq 0$ and obtain:

$$\begin{aligned}
& v_1^* < v_{max|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)\,v_{i-1} + \sqrt{(c_i+c_{i-1})^2\,{v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) < \sqrt{{v_{i-1}}^2 + 2a_{t_{max}}\Delta_s}.
\end{aligned}$$

As both sides are greater or equal zero due $v_{max|a_t} > 0$ and the assumption above, they can be squared:

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}} < 4{c_i}^2{v_{i-1}}^2 + 8{c_i}^2 a_{t_{max}}\Delta_s \\
& \Leftrightarrow 2\left({c_{i-1}}^2 - {c_i}^2\right){v_{i-1}}^2 - 8c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right) < -2\left(c_{i-1}-c_i\right)v_{i-1}\sqrt{\dots}.
\end{aligned}$$

The inequality is always true, as the rhs is always positive whereas the lhs always yields a negative value. Therefore $v_1^* < v_{max|a_t}$ holds.

We have to distinguish two cases for $v_{min|a_t}$: For the case that $v_{min|a_t} = 0$ the intersection is always nonempty, as $v_1 > 0$:

$$\begin{aligned}
v_1 &= \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \\
&> \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2}\right) \\
&= \frac{1}{2c_i} 2c_{i-1}v_{i-1} \\
&\geq 0.
\end{aligned}$$

The next case is $v_{min|a_t} > 0$. The intersection will be non-empty as soon as $v_{min|a_t} \leq v_1$. We derive a condition for this:

$$\begin{aligned}
& v_1 \geq v_{min|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \geq \sqrt{{v_{i-1}}^2 - 2a_{t_{max}}\Delta_s}.
\end{aligned}$$

With both sides of the inequality greater than zero, squaring both sides is an equivalence transformation:

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i a_{\omega_{max}}\Delta_s \geq 4{c_i}^2{v_{i-1}}^2 - 8{c_i}^2 a_{t_{max}}\Delta_s \\
& \Leftrightarrow 2\left({c_{i-1}}^2 - {c_i}^2\right){v_{i-1}}^2 + 8c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right) \geq -2\left(c_{i-1}-c_i\right)v_{i-1}\sqrt{\dots} \\
& \Leftrightarrow\ -\left(c_{i-1}+c_i\right)v_{i-1} + \frac{4c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right)}{-\left(c_{i-1}-c_i\right)v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

For this to have any chance to hold, the lhs has to be positive. This translates into a constraint for v_{i-1}:

$$\begin{aligned}
& -\left(c_{i-1}+c_i\right)v_{i-1} + \frac{4c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right)}{-\left(c_{i-1}-c_i\right)v_{i-1}} \geq 0 \\
\Leftrightarrow\ & -\left(c_{i-1}+c_i\right){v_{i-1}}^2 \geq \frac{4c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right)}{c_{i-1}-c_i} \\
\Leftrightarrow\ & {v_{i-1}}^2 \leq -\frac{4c_i\Delta_s\left(a_{\omega_{max}} + c_i a_{t_{max}}\right)}{\left(c_{i-1}-c_i\right)\left(c_{i-1}+c_i\right)}.
\end{aligned}$$

We keep this constraint in mind and assume for now that v_{i-1} fulfills it. Therefore both sides of the original inequality are now positive and we can square them. Note that the term $(c_{i-1}+c_i)^2 {v_{i-1}}^2$ cancels out immediately:

$$
\begin{aligned}
&\Leftrightarrow 2(c_{i-1}+c_i)\frac{4c_i\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})}{c_{i-1}-c_i}+\frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}}+c_i a_{t_{max}})^2)}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\geq +8c_i a_{\omega_{max}}\Delta_s\\
&\Leftrightarrow \frac{8c_i(c_{i-1}+c_i)\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})}{c_{i-1}-c_i}{v_{i-1}}^2+\frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_{i-1}-c_i)^2}\geq 8c_i a_{\omega_{max}}\Delta_s{v_{i-1}}^2\\
&\Leftrightarrow \frac{8c_i\Delta_s\left((c_{i-1}+c_i)(a_{\omega_{max}}+c_i a_{t_{max}})-(c_{i-1}-c_i)a_{\omega_{max}}\right)}{c_{i-1}-c_i}{v_{i-1}}^2\geq -\frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_{i-1}-c_i)^2}\\
&\Leftrightarrow \left((c_{i-1}+c_i)(a_{\omega_{max}}+c_i a_{t_{max}})-(c_{i-1}-c_i)a_{\omega_{max}}\right){v_{i-1}}^2\leq -\frac{2c_i\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{c_{i-1}-c_i}\\
&\Leftrightarrow \left(2c_i a_{\omega_{max}}+(c_{i-1}+c_i)c_i a_{t_{max}}\right){v_{i-1}}^2\leq -\frac{2c_i\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{c_{i-1}-c_i}\\
&\Leftrightarrow {v_{i-1}}^2\leq \frac{2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_i-c_{i-1})\left(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}}\right)}.
\end{aligned}
$$

We remember that there was an additional constraint on v_{i-1} but it is weaker than the final constraint: It is

$$
\begin{aligned}
&-\frac{4c_i\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}>\frac{2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_i-c_{i-1})\left(2a_{\omega_{max}}+(c_i+c_{i-1})a_{t_{max}}\right)}\\
\Leftrightarrow\ &\underbrace{\frac{2c_i}{c_{i-1}+c_i}}_{>1}>\frac{a_{\omega_{max}}+c_i a_{t_{max}}}{2a_{\omega_{max}}+(c_i+c_{i-1})a_{t_{max}}}=\underbrace{\frac{a_{\omega_{max}}+c_i a_{t_{max}}}{a_{\omega_{max}}+c_i a_{t_{max}}+a_{\omega_{max}}+c_{i-1}a_{t_{max}}}}_{<1}.
\end{aligned}
$$

A last consideration for this case reassures that the computed bound fulfills the assumption of $v_{min|a_t}>0$ (see the second case of Eq. (3.29)):

$$
\begin{aligned}
&\sqrt{\frac{2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_i-c_{i-1})\left((2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})\right)}}>\sqrt{2\Delta_s a_{t_{max}}}\\
\Leftrightarrow\ &(a_{\omega_{max}}+c_i a_{t_{max}})^2>a_{t_{max}}((c_i-c_{i-1})\left(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}}\right))\\
\Leftrightarrow\ &a_{\omega_{max}}^2+2c_i a_{\omega_{max}}a_{t_{max}}+{c_i}^2a_{t_{max}}^2>a_{t_{max}}^2({c_i}^2-{c_{i-1}}^2)+a_{t_{max}}(c_i-c_{i-1})2a_{\omega_{max}}\\
\Leftrightarrow\ &a_{\omega_{max}}^2+2a_{\omega_{max}}a_{t_{max}}c_{i-1}+a_{t_{max}}^2{c_{i-1}}^2>0.
\end{aligned}
$$

The last line is a tautology if case assumptions are considered.

We can now conclude for this case:

$$
\begin{aligned}
&\left((c_i>0)\wedge(c_{i-1}\geq 0)\wedge(c_i>c_{i-1})\right)\\
&\qquad\Rightarrow\ \left(I_{a_\omega}\cap[v_{min|a_t},v_{max|a_t}]\neq\emptyset\Leftrightarrow\ v_{i-1}\leq\sqrt{\frac{2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_i-c_{i-1})\left(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}}\right)}}\right).
\end{aligned}
\tag{B.1}
$$

B.1.1.2 Case 1.2, $c_i < c_{i-1}$

Note that the case assumption implies $c_{i-1}>0$. With $c_{i-1}-c_i>0$ as a direct consequence of the case assumptions $v_1>0$ is easily verified by its definition in Eq. (3.32).

For v_2 it holds

$$\begin{aligned} v_2 &= \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i a_{\omega_{max}}\Delta_s}\right) \\ &< \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2}\right) \\ &= \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2}\right) \\ &= -\frac{1}{2c_i} 2c_i v_{i-1} \\ &\leq 0. \end{aligned}$$

Therefore it is $I_{a_\omega} = [0, v_2^*] \cup [v_1^*, v_1]$ or $I_{a_\omega} = [0, v_1]$, depending on whether or not v_1^*, v_2^* exist. For intersection with $[v_{min|a_t}, v_{max|a_t}]$ it will be shown, that $v_1 \geq v_{min|a_t}$:

$$\begin{aligned} & v_1 \geq v_{min|a_t} \\ \Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i a_{\omega_{max}}\Delta_s}\right) \geq \sqrt{{v_{i-1}}^2 - 2a_{t_{max}}\Delta_s}. \end{aligned}$$

As both sides of the inequality are non-negative they can be squared without causing any harm:

$$\begin{aligned} &\Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i a_{\omega_{max}}\Delta_s \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2 a_{t_{max}}\Delta_s \\ &\Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + 8c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}}) \geq 0. \end{aligned}$$

The lhs of the inequality is easily verified to be non-negative through the case assumptions.

As a first result we state that a non-empty intersection exists as soon as v_1^*, v_2^* do not exist and have distinct values:

$$(\neg \mathit{exv1sv2s}) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.2}$$

where

$$\mathit{exv1sv2s} = \left(v_{i-1} > \sqrt{\frac{8c_i a_{\omega_{max}}\Delta_s}{(c_i+c_{i-1})^2}}\right). \tag{B.3}$$

We assume that v_1^*, v_2^* do exist and are not equal (*exv1sv2s*). The first possibility for an intersection is

$$\begin{aligned} & v_1^* \leq v_{max|a_t} \\ \Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i a_{\omega_{max}}\Delta_s}\right) \leq \sqrt{{v_{i-1}}^2 + 2a_{t_{max}}\Delta_s}. \end{aligned}$$

Squaring is an equivalence transformation as both, lhs and rhs are non-negative:

$$\begin{aligned} &\Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_{i-1}+c_i)^2 {v_{i-1}}^2 - 8c_i a_{\omega_{max}}\Delta_s \leq 4{c_i}^2 {v_{i-1}}^2 + 8{c_i}^2 a_{t_{max}}\Delta_s \\ &\Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2) - 8c_i\Delta_s(c_i a_{t_{max}} + a_{\omega_{max}}) \leq -2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\ &\Leftrightarrow\ -(c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} + a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{\dots}. \end{aligned}$$

If the inequality's lhs is negative there won't be an intersection. We assume to lhs to be non-negative. This is the case if

$$\begin{aligned}
&-(c_{i-1}+c_i)v_{i-1}+\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}}\geq 0\\
\Leftrightarrow\ &-(c_{i-1}+c_i){v_{i-1}}^2\geq -\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{c_{i-1}-c_i}\\
\Leftrightarrow\ &{v_{i-1}}^2\leq \frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\\
\Leftrightarrow\ &v_{i-1}\leq \sqrt{\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}.
\end{aligned}$$

With the assumption of the lhs being non-negative we continue transforming the inequality by squaring both sides, note that the term $(c_{i-1}+c_i)^2{v_{i-1}}^2$ immediately cancels out:

$$\begin{aligned}
&-\frac{8(c_{i-1}+c_i)c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{c_{i-1}-c_i}+\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\geq -8c_i a_{\omega_{max}}\Delta_s\\
\Leftrightarrow\ &\frac{-8(c_{i-1}+c_i)c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})+8c_i(c_{i-1}-c_i)a_{\omega_{max}}\Delta_s}{c_{i-1}-c_i}{v_{i-1}}^2\geq -\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2}\\
\Leftrightarrow\ &(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}}){v_{i-1}}^2\leq \frac{2\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})^2}{c_{i-1}-c_i}\\
\Leftrightarrow\ &v_{i-1}\leq \sqrt{\frac{2\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}}.
\end{aligned}$$

We state another intermediary result:

$$\begin{aligned}
\Bigg(exv1sv2s\wedge\Bigg(v_{i-1}\leq\sqrt{\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}\Bigg)\\
\wedge\Bigg(v_{i-1}\leq\sqrt{\frac{2\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}}\Bigg)\Bigg)\\
\Rightarrow\ I_{a_\omega}\cap[v_{min|a_t},v_{max|a_t}]\neq\emptyset.\quad\text{(B.4)}
\end{aligned}$$

Note that also holds:

$$\begin{aligned}
&(exv1sv2s\\
&\wedge\neg\Bigg(\Bigg(v_{i-1}\leq\sqrt{\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}\Bigg)\wedge\Bigg(v_{i-1}\leq\sqrt{\frac{2\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}}\Bigg)\Bigg)\Bigg)\\
&\Rightarrow\ [v_1^*,v_1]\cap[v_{min|a_t},v_{max|a_t}]=\emptyset.\quad\text{(B.5)}
\end{aligned}$$

The second possibility for an intersection is that $[0,v_2^*]\cap[v_{min|a_t},v_{max|a_t}]\neq\emptyset$. This is true for $v_2^*\geq v_{min|a_t}$. Before examining this condition we first demand v_2^* to be positive:

$$\begin{aligned}
&v_2^*>0\\
\Leftrightarrow\ &(c_{i-1}-c_i)v_{i-1}>\sqrt{(c_i+c_{i-1})^2{v_{i-1}}^2-8c_i a_{\omega_{max}}\Delta_s}.
\end{aligned}$$

Taking both sides to the power of two:

$$\begin{aligned}
&\Leftrightarrow -4c_{i-1}c_i{v_{i-1}}^2 > -8c_i a_{\omega_{max}}\Delta_s \\
&\Leftrightarrow {v_{i-1}}^2 < \frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}} \\
&\Leftrightarrow v_{i-1} < \sqrt{\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}}.
\end{aligned}$$

We define

$$v2spos = \left(v_{i-1} < \sqrt{\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}}\right). \tag{B.6}$$

We have to distinguish two cases for $v_{min|a_t}$: $v_{min|a_t} = 0$ and $v_{min|a_t} > 0$. A non-empty intersection exists for the case $v_{min|a_t} = 0$ and $v_2^* > 0$ and of course for the case $v_{min|a_t}, v_2^* > 0$ and $v_2^* \geq v_{min|a_t}$.

Assume $v_{min|a_t} = 0$. This means

$$vmatzero = \left(v_{i-1} \leq \sqrt{2a_{t_{max}}\Delta_s}\right). \tag{B.7}$$

We state an intermediary result:

$$(exv1sv2s \wedge vmatzero \wedge v2spos) \Rightarrow I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \tag{B.8}$$

Now, assuming $v_2^* > 0$ and $v_{min|a_t} > 0$:

$$\begin{aligned}
&v_2^* \geq v_{min|a_t} \\
&\Leftrightarrow \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2{v_{i-1}}^2 - 8c_i a_{\omega_{max}}\Delta_s}\right) \geq \sqrt{{v_{i-1}}^2 - 2a_{t_{max}}\Delta_s}
\end{aligned}$$

to the power of two

$$\begin{aligned}
&\Leftrightarrow (c_{i-1}-c_i)^2{v_{i-1}}^2 - 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2{v_{i-1}}^2 - 8c_i a_{\omega_{max}}\Delta_s \geq 4{c_i}^2{v_{i-1}}^2 - 8{c_i}^2 a_{t_{max}}\Delta_s \\
&\Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}}) \geq 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
&\Leftrightarrow (c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{(c_i+c_{i-1})^2{v_{i-1}}^2 - 8c_i a_{\omega_{max}}\Delta_s}.
\end{aligned}$$

A necessary condition for this inequality to hold is the lhs to be non-negative:

$$\begin{aligned}
&(c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq 0 \\
&\Leftrightarrow {v_{i-1}}^2 \geq -\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}.
\end{aligned}$$

We call this

$$precond1 = \left({v_{i-1}}^2 \geq -\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\right). \tag{B.9}$$

Assuming *precond1* to hold, we proceed with the original inequality by squaring both sides, note that the term $(c_{i-1}-c_i)^2 {v_{i-1}}^2$ immediately cancels out:

$$\begin{aligned}
&\frac{8(c_{i-1}+c_i)c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{c_{i-1}-c_i}+\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\geq -8c_i a_{\omega_{max}}\Delta_s\\
\Leftrightarrow\ &\frac{8(c_{i-1}+c_i)c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})+8(c_{i-1}-c_i)c_i a_{\omega_{max}}\Delta_s}{c_{i-1}-c_i}{v_{i-1}}^2\geq -\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2}\\
\Leftrightarrow\ &\left((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}\right){v_{i-1}}^2\geq -\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{c_{i-1}-c_i}.
\end{aligned}$$

As the rhs is negative, this would be tautology for $(c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}\geq 0$, for the complementary case $(c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}<0$ it translates to:

$$v_{i-1}\leq\sqrt{\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{\omega_{max}})}}.$$

A short derivation shows that it is always $(c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}<0$: So far, we assumed *v2hpos* and $\neg$*vmatzero* to hold. A consequence of this is:

$$\begin{aligned}
&\sqrt{\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}}>\sqrt{2a_{t_{max}}\Delta_s}\\
\Rightarrow\ &\frac{a_{\omega_{max}}}{c_{i-1}}>a_{t_{max}}\\
\Rightarrow\ &c_{i-1}a_{t_{max}}<a_{\omega_{max}}
\end{aligned}$$

with the case assumption $c_i<c_{i-1}$

$$\Rightarrow (c_{i-1}+c_i)a_{t_{max}}<2a_{\omega_{max}}.$$

Before concluding for this case we state some more intermediary results:

$$\left(exv1sv2s\wedge\neg vmatzero\wedge v2spos\wedge precond1\wedge\left(v_{i-1}\leq\sqrt{\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}}\right)\right)$$
$$\Rightarrow\ I_{a_\omega}\cap[v_{min|a_t},v_{max|a_t}]\neq\emptyset. \quad \text{(B.10)}$$

Analogously to the first possibility for intersection it also holds:

$$\left(exv1sv2s\wedge\neg vmatzero\wedge\right.$$
$$\left.\neg\left(v2spos\wedge precond1\wedge\left(v_{i-1}\leq\sqrt{\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}}\right)\right)\right)$$
$$\Rightarrow\ [0,v_2^*]\cap[v_{min|a_t},v_{max|a_t}]=\emptyset. \quad \text{(B.11)}$$

We can now conclude for this case:

$$\begin{aligned}
&\big((c_i>0)\wedge(c_{i-1}\geq 0)\wedge(c_i<c_{i-1})\big)\\
&\quad\Rightarrow\ \big(I_{a_\omega}\cap[v_{min|a_t},v_{max|a_t}]\neq\emptyset\Leftrightarrow\ (\neg exv1sv2s\vee(exv1sv2s\wedge(isect1\vee isect2\vee isect3)))\big), \quad \text{(B.12)}
\end{aligned}$$

where

$$
\begin{aligned}
\mathit{exv1sv2s} &= \left(v_{i-1} > \sqrt{\frac{8c_i a_{\omega_{max}} \Delta_s}{(c_i + c_{i-1})^2}} \right) \\
\mathit{isect1} &= \left(\left(v_{i-1} \leq \sqrt{\frac{4c_i \Delta_s (c_i a_{t_{max}} + a_{\omega_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)}} \right) \wedge \left(v_{i-1} \leq \sqrt{\frac{2\Delta_s (c_i a_{t_{max}} + a_{\omega_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} + (c_{i-1} + c_i) a_{t_{max}})}} \right) \right) \\
\mathit{isect2} &= (\mathit{vmatzero} \wedge \mathit{v2spos}) \\
\mathit{isect3} &= \left(\neg \mathit{vmatzero} \wedge \mathit{v2spos} \wedge \mathit{precond1} \wedge \left(v_{i-1} \leq \sqrt{\frac{2\Delta_s (c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} - (c_{i-1} + c_i) a_{t_{max}})}} \right) \right) \\
\mathit{vmatzero} &= \left(v_{i-1} \leq \sqrt{2 a_{t_{max}} \Delta_s} \right) \\
\mathit{v2spos} &= \left(v_{i-1} < \sqrt{\frac{2 a_{\omega_{max}} \Delta_s}{c_{i-1}}} \right) \\
\mathit{precond1} &= \left({v_{i-1}}^2 \geq -\frac{4c_i \Delta_s (c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)} \right).
\end{aligned}
$$

A quick repetition of the semantics:

isect1 true iff intersection occurs with $[v_1^*, v_1]$

isect2 true iff intersection occurs with $[0, v_2^*]$ and $v_{min|a_t} = 0$

isect3 true iff intersection occurs with $[0, v_2^*]$ and $v_{min|a_t} > 0$

As some of the conditions are lower bounds to v_{i-1} it cannot be taken for granted that once a suitable v_{i-1} has been found, decreasing it will never result in an empty intersection. We will now argue, however, that this is the case.

For the lower bound introduced by *exv1sv2s* this is obvious, as $\neg$*exv1sv2s* on its own is sufficient for a non-empty intersection. A closer look is needed for the lower bounds active in *isect3*:

We assume to have found a v_{i-1} that fulfills *isect3* and while decreasing v_{i-1}, $\neg$*vmatzero* is the first condition to become false. As can be verified, the condition *isect2* immediately becomes true and decreasing v_{i-1} is harmless from now on.

For the more complex case that *precond1* is the first condition to become false while decreasing v_{i-1}, it will now be shown that for this case *isect1* will always be true.

In case $\neg$*precond1* is true, it holds:

$$
\begin{aligned}
{v_{i-1}}^2 &< -\frac{4c_i \Delta_s (c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)} = \frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)} < \frac{4c_i \Delta_s (a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)} \\
\Rightarrow v_{i-1} &\leq \sqrt{\frac{4c_i \Delta_s (a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)}}.
\end{aligned}
$$

This means that the first part of *isect1* is fulfilled. For the truth of the second part we introduce a case distinction:

Subcase 1 $\frac{2\Delta_s (c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} - (c_{i-1} + c_i) a_{t_{max}})} \leq \frac{2\Delta_s (c_i a_{t_{max}} + a_{\omega_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} + (c_{i-1} + c_i) a_{t_{max}})}$**:** For *isect3* to have a possibility to hold, the lower bound introduced by *precond1* has to be lower than any upper bound, in particular it

has to hold:

$$-\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \leq \frac{2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})}.$$

Together with the subcase assumption the truth of the second part of *isect1* results.

Subcase 2 $\frac{2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})} > \frac{2\Delta_s(c_i a_{t_{max}} + a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})}$**:** We will first transform the subcase assumption to retrieve a statement that will be of use later on.

$$\begin{aligned}
&\frac{2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})} > \frac{2\Delta_s(c_i a_{t_{max}} + a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})} \\
\Leftrightarrow\ &\frac{(c_i a_{t_{max}} - a_{\omega_{max}})^2}{2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}}} > \frac{(c_i a_{t_{max}} + a_{\omega_{max}})^2}{2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}} \\
\Leftrightarrow\ &(c_i a_{t_{max}} - a_{\omega_{max}})^2(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}) > (c_i a_{t_{max}} + a_{\omega_{max}})^2(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}}) \\
\Leftrightarrow\ &-8c_i a_{t_{max}} a_{\omega_{max}}^2 + 2({c_i}^2 a_{t_{max}}^2 + a_{\omega_{max}}^2)(c_{i-1}+c_i)a_{t_{max}} > 0 \\
\Leftrightarrow\ &-4c_i a_{\omega_{max}}^2 + (c_{i-1}+c_i)({c_i}^2 a_{t_{max}}^2 + a_{\omega_{max}}^2) > 0.
\end{aligned}$$

With a last transformation step we can state the subcase assumption as

$${c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2 > (3c_i - c_{i-1})a_{\omega_{max}}^2. \tag{B.13}$$

For the fulfillment of the second part of *isect1* it is needed:

$$\begin{aligned}
&-\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \leq \frac{2\Delta_s(c_i a_{t_{max}} + a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})} \\
\Leftrightarrow\ &\frac{2c_i(a_{\omega_{max}} - c_i a_{t_{max}})}{c_{i-1}+c_i} \leq \frac{(c_i a_{t_{max}} + a_{\omega_{max}})^2}{2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}} \\
\Leftrightarrow\ &(2c_i a_{\omega_{max}} - 2{c_i}^2 a_{t_{max}})(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}) \leq ({c_i}^2 a_{t_{max}}^2 + 2c_i a_{t_{max}} a_{\omega_{max}} + a_{\omega_{max}}^2)(c_{i-1}+c_i) \\
\Leftrightarrow\ &4c_i a_{\omega_{max}}^2 - 4{c_i}^2 a_{t_{max}} a_{\omega_{max}} - 2{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2 \leq {c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2 + (c_{i-1}+c_i)a_{\omega_{max}}^2 \\
\Leftrightarrow\ &(4c_i - c_{i-1} - c_i)a_{\omega_{max}}^2 - 4{c_i}^2 a_{t_{max}} a_{\omega_{max}} + (-2{c_i}^2(c_{i-1}+c_i) - {c_i}^2(c_{i-1}+c_i))a_{t_{max}}^2 \leq 0 \\
\Leftrightarrow\ &\underbrace{(3c_i - c_{i-1})a_{\omega_{max}}^2 - 3{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2}_{\text{Eq. (B.13)} \Rightarrow \leq 0} \underbrace{-4{c_i}^2 a_{t_{max}} a_{\omega_{max}}}_{\leq 0} \leq 0.
\end{aligned}$$

As indicated by the braces, the subcase assumption leads to the lhs always being negative, which means that the second part of *isect1* is fulfilled.

We have shown for this case that once a suitable v_{i-1} has been found, decreasing it has no effect on the non-emptiness of the intersection.

B.1.1.3 Case 1.3, $c_i = c_{i-1}$

For $c_i = c_{i-1}$ it is $v_2 < 0$ and (if exists) $v_2^* < 0$ as can be seen in the respective definitions (Eqs. (3.32) and (3.33)). Furthermore v_1 and v_1^* degenerate according to

$$v_1 = \sqrt{{v_{i-1}}^2 + \frac{2a_{\omega_{max}}\Delta_s}{c_i}}, \quad v_1^* = \sqrt{{v_{i-1}}^2 - \frac{2a_{\omega_{max}}\Delta_s}{c_i}}.$$

Conviction that the intersection of I_{a_ω} and $[v_{min|a_t}, v_{max|a_t}]$ is never empty is now achieved through the definitions in Eqs. (3.30) and (3.29): One interval is always contained in the other one.

Therefore, we conclude for this case

$$\big((c_i > 0) \wedge (c_{i-1} \geq 0) \wedge (c_i = c_{i-1})\big) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \tag{B.14}$$

B.1.2 Case 2, $c_i < 0, c_{i-1} \leq 0$

This case has structural similarities to case 1: case 2.1 corresponds to case 1.2, and case 2.2 to case 1.1, respectively.

B.1.2.1 Case 2.1, $c_i > c_{i-1}$

Case assumptions lead to $c_{i-1} < 0$, $c_{i-1} - c_i < 0$ and to the unconditioned existence of v_1^* and v_2^*. Furthermore, $v_2^* > 0$ always holds.

Additionally it is always $v_1^* < 0$:

$$\begin{aligned}
& v_1^* < 0 \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1} - c_i)v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) < 0 \\
\Leftrightarrow\ & (c_{i-1} - c_i)v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}} > 0.
\end{aligned}$$

This is derived by:

$$\begin{aligned}
& (c_{i-1} - c_i)v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}} \\
& > (c_{i-1} - c_i)v_{i-1} + \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2} \\
& = (c_{i-1} - c_i)v_{i-1} + |c_i + c_{i-1}|v_{i-1} \\
& = (c_{i-1} - c_i + |c_i| + |c_{i-1}|)v_{i-1} \\
& = 2|c_i|v_{i-1} \\
& > 0.
\end{aligned}$$

Depending on the existence of v_1, v_2 it is now $I_{a_\omega} = [0, v_2^*]$ or $I_{a_\omega} = [0, v_1] \cup [v_2, v_2^*]$. For further discussion of interval overlapping we first derive $v_2^* \geq v_{min|a_t}$:

$$\begin{aligned}
& v_2^* \geq v_{min|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1} - c_i)v_{i-1} - \sqrt{(c_{i-1} + c_i)^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) \geq \sqrt{{v_{i-1}}^2 - 2a_{t_{max}}\Delta_s} \\
\Leftrightarrow\ & (c_{i-1} - c_i)^2 {v_{i-1}}^2 - 2(c_{i-1} - c_i)v_{i-1}\sqrt{\dots} + (c_{i-1} + c_i)^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}} \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2 a_{t_{max}}\Delta_s \\
\Leftrightarrow\ & 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}}) \geq 2(c_{i-1} - c_i)v_{i-1}\sqrt{\dots} \\
\Leftrightarrow\ & (c_{i-1} + c_i)v_{i-1} + \frac{8c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1} - c_i)v_{i-1}} \leq \sqrt{\dots}.
\end{aligned}$$

As the lhs is clearly negative due to case assumptions the last line always holds.

We state an intermediary result: the existence of a non-empty overlap as soon as v_1 and v_2 do not exist with different values.

$$(\neg exv1v2) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.15}$$

where

$$exv1v2 = \left(v_{i-1} > \sqrt{-\frac{8c_i a_{\omega_{max}} \Delta_s}{(c_i + c_{i-1})^2}} \right). \tag{B.16}$$

From now on *exv1v2* and thereby the existence and non-equality of v_1 and v_2 is assumed. With $I_{a_\omega} = [0, v_1] \cup [v_2, v_2^*]$ the first possibility for an overlap emerges for $v_2 \leq v_{max|a_t}$. Note that $v_2 > 0$ is guaranteed by case assumptions.

$$\begin{aligned}
& v_2 \leq v_{max|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i} \left((c_{i-1} - c_i) v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}}} \right) \leq \sqrt{{v_{i-1}}^2 + 2\Delta_s a_{t_{max}}}.
\end{aligned}$$

With both sides guaranteed to be positive, squaring becomes a equivalence transformation:

$$\begin{aligned}
& \Leftrightarrow (c_{i-1} - c_i)^2 {v_{i-1}}^2 - 2(c_{i-1} - c_i) v_{i-1} \sqrt{\dots} + (c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i \Delta_s a_{\omega_{max}} \leq 4{c_i}^2 {v_{i-1}}^2 + 8{c_i}^2 \Delta_s a_{t_{max}} \\
& \Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2) {v_{i-1}}^2 + 8c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}}) \leq 2(c_{i-1} - c_i) v_{i-1} \sqrt{\dots} \\
& \Leftrightarrow (c_{i-1} + c_i) v_{i-1} + \frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} - c_i) v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

For the inequality to be able to hold, its lhs must be non-negative. This is the case for:

$$\begin{aligned}
& (c_{i-1} + c_i) v_{i-1} + \frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} - c_i) v_{i-1}} \geq 0 \\
\Leftrightarrow\ & (c_{i-1} + c_i) {v_{i-1}}^2 \geq -\frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} - c_i)} \\
\Leftrightarrow\ & {v_{i-1}}^2 \leq -\frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} + c_i)(c_{i-1} - c_i)} \\
\Leftrightarrow\ & v_{i-1} \leq \sqrt{-\frac{4c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1} + c_i)(c_{i-1} - c_i)}}.
\end{aligned}$$

Assuming the lhs to be non-negative, we proceed with the original inequality by squaring both sides, again the term $(c_{i-1} + c_i)^2 {v_{i-1}}^2$ appears on both sides and therefore cancels out:

$$\begin{aligned}
& \frac{8c_i \Delta_s (c_{i-1} + c_i)(a_{\omega_{max}} - c_i a_{t_{max}})}{c_{i-1} - c_i} + \frac{16{c_i}^2 \Delta_s^2 (a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2 {v_{i-1}}^2} \geq 8c_i \Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & \frac{8c_i \Delta_s ((c_{i-1} + c_i)(a_{\omega_{max}} - c_i a_{t_{max}}) - (c_{i-1} - c_i) a_{\omega_{max}})}{c_{i-1} - c_i} {v_{i-1}}^2 \geq -\frac{16{c_i}^2 \Delta_s^2 (a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2} \\
\Leftrightarrow\ & \frac{2c_i a_{\omega_{max}} - c_i (c_{i-1} + c_i) a_{t_{max}}}{c_{i-1} - c_i} {v_{i-1}}^2 \leq -\frac{2c_i \Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2} \\
\Leftrightarrow\ & (2a_{\omega_{max}} - (c_{i-1} + c_i) a_{t_{max}}) {v_{i-1}}^2 \leq -\frac{2\Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})^2}{c_{i-1} - c_i} \\
\Leftrightarrow\ & v_{i-1} \leq \sqrt{\frac{-2\Delta_s (a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} - (c_{i-1} + c_i) a_{t_{max}})}}.
\end{aligned}$$

This leads to another intermediary result for this case:

$$\left(exv1v2 \wedge \left(v_{i-1} \leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}+c_i)(c_{i-1}-c_i)}}\right) \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})}}\right)\right) \Rightarrow I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \quad \text{(B.17)}$$

Note that also holds:

$$\left(exv1v2 \wedge \neg\left(\left(v_{i-1} \leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}+c_i)(c_{i-1}-c_i)}}\right) \wedge \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})}}\right)\right)\right) \Rightarrow [v_2, v_2^*] \cap [v_{min|a_t}, v_{max|a_t}] = \emptyset. \quad \text{(B.18)}$$

Now we proceed to the second possibility for an intersection: $v_1 \geq v_{min|a_t}$. For this to be possible, v_1 has to be non-negative:

$$\begin{aligned}
& v_1 \geq 0 \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \geq 0 \\
\Leftrightarrow\ & (c_{i-1}-c_i)v_{i-1} \leq -\sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}} \\
\Leftrightarrow\ & -(c_{i-1}-c_i)v_{i-1} \geq \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}
\end{aligned}$$

both sides are non-negative, squaring them

$$\begin{aligned}
\Leftrightarrow\ & (c_{i-1}-c_i)^2 {v_{i-1}}^2 \geq (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & -4c_{i-1}c_i {v_{i-1}}^2 \geq 8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & {v_{i-1}}^2 \leq -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\
\Leftrightarrow\ & v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}.
\end{aligned}$$

This condition will be named

$$v1pos = \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}\right) \quad \text{(B.19)}$$

and is assumed to hold from now on.

For decision on $v_1 \geq v_{min|a_t}$ there are two cases to be distinguished for $v_{min|a_t}$: $v_{min|a_t} = 0$ and $v_{min|a_t} > 0$.

In case $v_{min|a_t} = 0$, which means

$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}}\right), \quad \text{(B.20)}$$

we have a non-empty intersection as soon as v_1 exists with $v_1 \geq 0$. We state this as an intermediary result:

$$(exv1v2 \wedge vmatzero \wedge v1pos) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \tag{B.21}$$

We assume $v_{min|a_t} > 0$ (i.e., $\neg vmatzero$). For a non-empty intersection it is needed:

$$\begin{aligned}
& v_1 \geq v_{min|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}
\end{aligned}$$

both sides are assumed to be positive, they can be squared

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}} \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2\Delta_s a_{t_{max}} \\
& \Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}}) \geq -2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
& \Leftrightarrow\ -(c_{i-1}+c_i)v_{i-1} - \frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

We want to assume the lhs to be positive and translate this into a condition for v_{i-1}:

$$\begin{aligned}
& -(c_{i-1}+c_i)v_{i-1} - \frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq 0 \\
\Leftrightarrow\ & {v_{i-1}}^2 \geq -\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}
\end{aligned}$$

We call it

$$precond1 = \left({v_{i-1}}^2 \geq -\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\right). \tag{B.22}$$

Assuming *precond1* to hold we continue to process the original inequality by squaring both sides, as they now share signs. Note that the term $(c_{i-1}+c_i)^2 {v_{i-1}}^2$ immediately cancels out.

$$\begin{aligned}
& \frac{8c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})(c_{i-1}+c_i)}{c_{i-1}-c_i} + \frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)^2 {v_{i-1}}^2} \geq 8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & \frac{8c_i\Delta_s((a_{\omega_{max}} + c_i a_{t_{max}})(c_{i-1}+c_i) - (c_{i-1}-c_i)a_{\omega_{max}})}{c_{i-1}-c_i} \geq -\frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)^2 {v_{i-1}}^2} \\
\Leftrightarrow\ & \frac{{c_i}^2\Delta_s(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})}{c_{i-1}-c_i} \geq -\frac{2{c_i}^2\Delta_s^2(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)^2 {v_{i-1}}^2} \\
\Leftrightarrow\ & (2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}) \leq -\frac{2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i){v_{i-1}}^2} \\
\Leftrightarrow\ & (2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}){v_{i-1}}^2 \leq -\frac{2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{c_{i-1}-c_i}.
\end{aligned}$$

This inequality is a tautology for the case $(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}) \leq 0$ as the rhs is always non-negative. For the complementary case $(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}}) > 0$ it translates to:

$$v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})}}.$$

The case $(2a_{\omega_{max}} + (c_{i-1} + c_i)a_{t_{max}}) \leq 0$ can never occur in this context: We assumed *v1pos* and $\neg$*vmatzero* to hold, which has as a consequence:

$$\begin{aligned} & \sqrt{-\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}} > \sqrt{2a_{t_{max}}\Delta_s} \\ \Rightarrow & -\frac{a_{\omega_{max}}}{c_{i-1}} > a_{t_{max}} \\ \Rightarrow & a_{\omega_{max}} > -c_{i-1}a_{t_{max}} \\ \Rightarrow & a_{\omega_{max}} + c_{i-1}a_{t_{max}} > 0. \end{aligned}$$

The case assumption $c_i > c_{i-1}$ leads to

$$\Rightarrow 2a_{\omega_{max}} + (c_{i-1} + c_i)a_{t_{max}} > 0.$$

Prior to concluding for this case, we state some intermediary results:

$$\left(\textit{exv1v2} \wedge \neg \textit{vmatzero} \wedge \textit{v1pos} \wedge \textit{precond1} \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} + (c_{i-1} + c_i)a_{t_{max}})}} \right) \right) \Rightarrow I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \quad \text{(B.23)}$$

Note, that analogously to the first intersection possibility also this holds:

$$\left(\textit{exv1v2} \wedge \neg \textit{vmatzero} \wedge \neg \left(\textit{v1pos} \wedge \textit{precond1} \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)(2a_{\omega_{max}} + (c_{i-1} + c_i)a_{t_{max}})}} \right) \right) \right) \Rightarrow [0, v_1] \cap [v_{min|a_t}, v_{max|a_t}] = \emptyset. \quad \text{(B.24)}$$

Concluding for this case, we can state:

$$\begin{aligned} & \big((c_i < 0) \wedge (c_{i-1} \leq 0) \wedge (c_i > c_{i-1}) \big) \\ & \quad \Rightarrow \big(I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \Leftrightarrow (\neg \textit{exv1v2} \vee (\textit{exv1v2} \wedge (\textit{isect1} \vee \textit{isect2} \vee \textit{isect3}))) \big), \end{aligned} \quad \text{(B.25)}$$

where

$$
\begin{aligned}
\mathit{exv1v2} &= \left(v_{i-1} > \sqrt{-\frac{8c_i a_{\omega_{max}} \Delta_s}{(c_i + c_{i-1})^2}}\right) \\
\mathit{isect1} &= \left(\left(v_{i-1} \leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}+c_i)(c_{i-1}-c_i)}}\right) \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} - (c_{i-1}+c_i)a_{t_{max}})}}\right)\right) \\
\mathit{isect2} &= (\mathit{vmatzero} \wedge \mathit{v1pos}) \\
\mathit{isect3} &= \left(\neg \mathit{vmatzero} \wedge \mathit{v1pos} \wedge \mathit{precond1} \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}} + (c_{i-1}+c_i)a_{t_{max}})}}\right)\right) \\
\mathit{vmatzero} &= \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}}\right) \\
\mathit{v1pos} &= \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}\right) \\
\mathit{precond1} &= \left({v_{i-1}}^2 \geq -\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\right).
\end{aligned}
$$

The abbreviations' meanings are:

isect1 true iff intersection occurs with $[v_2, v_2^*]$

isect2 true iff intersection occurs with $[0, v_1]$ and $v_{min|a_t} = 0$

isect3 true iff intersection occurs with $[0, v_1]$ and $v_{min|a_t} > 0$

As some of the conditions are lower bounds to v_{i-1} it cannot be taken for granted that once a suitable v_{i-1} has been found, decreasing it will never result in an empty intersection. We will now argue, however, that this is the case.

For the lower bound introduced by *exv1v2* this is obvious, as $\neg$*exv1v2* on its own is sufficient for a non-empty intersection. More derivations are needed for the lower bounds active in *isect3*:

Assume we have found a v_{i-1} that fulfills *isect3* and while decreasing v_{i-1}, $\neg$*vmatzero* is the first condition to become false. The condition *isect2* immediately becomes true and decreasing v_{i-1} is harmless from now on.

For the more complex case that *precond1* is the first condition to become false while decreasing v_{i-1}, it will now be shown that for this case *isect1* will always be true.

For the case that $\neg$*precond1* holds, it is (also due to $c_i < 0$):

$$
\begin{aligned}
{v_{i-1}}^2 &< -\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} < -\frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \\
\Rightarrow v_{i-1} &\leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}.
\end{aligned}
$$

This guarantees the truth of the first part of *isect1*. For the second part we will distinguish two cases:

Subcase 1 $\frac{-2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})} \leq \frac{-2\Delta_s(a_{\omega_{max}}-c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}$**:** For *isect3* to be possible to hold the lower bound imposed by *precond1* has to be smaller than any active upper bound, in particular it

has to hold:

$$-\frac{4c_i\Delta_s(a_{\omega_{max}}+c_ia_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \leq \frac{-2\Delta_s(a_{\omega_{max}}+c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}.$$

Taking the subcase assumption into account, the fulfillment of the second part of *isect1* is obvious.

Subcase 2 $\frac{-2\Delta_s(a_{\omega_{max}}+c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})} > \frac{-2\Delta_s(a_{\omega_{max}}-c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}$**:** First, we will transform the subcase assumption for later use:

$$\begin{aligned}
&\frac{-2\Delta_s(a_{\omega_{max}}+c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})} > \frac{-2\Delta_s(a_{\omega_{max}}-c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}\\
\Leftrightarrow\ &\frac{(a_{\omega_{max}}+c_ia_{t_{max}})^2}{2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}}} > \frac{(a_{\omega_{max}}-c_ia_{t_{max}})^2}{2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}}}\\
\Leftrightarrow\ &(a_{\omega_{max}}+c_ia_{t_{max}})^2(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}}) > (a_{\omega_{max}}-c_ia_{t_{max}})^2(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})\\
\Leftrightarrow\ &8c_ia_{t_{max}}a_{\omega_{max}}^2-2(a_{\omega_{max}}^2+{c_i}^2a_{t_{max}}^2)(c_{i-1}+c_i)a_{t_{max}} > 0\\
\Leftrightarrow\ &4c_ia_{\omega_{max}}^2-(c_{i-1}+c_i)(a_{\omega_{max}}^2+{c_i}^2a_{t_{max}}^2) > 0.
\end{aligned}$$

A last step allows to state the subcase assumption as

$$(3c_i-c_{i-1})a_{\omega_{max}}^2 > {c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2. \tag{B.26}$$

For the truth of the second part of *isect1* we need:

$$\begin{aligned}
&-\frac{4c_i\Delta_s(a_{\omega_{max}}+c_ia_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \leq \frac{-2\Delta_s(a_{\omega_{max}}-c_ia_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}\\
\Leftrightarrow\ &\frac{2c_i(a_{\omega_{max}}+c_ia_{t_{max}})}{c_{i-1}+c_i} \leq \frac{(a_{\omega_{max}}-c_ia_{t_{max}})^2}{2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}}}\\
\Leftrightarrow\ &(2c_ia_{\omega_{max}}+2{c_i}^2a_{t_{max}})(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}}) \geq (a_{\omega_{max}}^2-2c_ia_{t_{max}}a_{\omega_{max}}+{c_i}^2a_{t_{max}}^2)(c_{i-1}+c_i)\\
\Leftrightarrow\ &4c_ia_{\omega_{max}}^2+4{c_i}^2a_{t_{max}}a_{\omega_{max}}-2{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2 \geq (c_{i-1}+c_i)a_{\omega_{max}}^2+{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2\\
\Leftrightarrow\ &(4c_i-c_{i-1}-c_i)a_{\omega_{max}}^2+4{c_i}^2a_{t_{max}}a_{\omega_{max}}-3{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2 \geq 0\\
\Leftrightarrow\ &\underbrace{(3c_i-c_{i-1})a_{\omega_{max}}^2-3{c_i}^2(c_{i-1}+c_i)a_{t_{max}}^2}_{\text{Eq. (B.26)}\Rightarrow\geq 0}+\underbrace{4{c_i}^2a_{t_{max}}a_{\omega_{max}}}_{\geq 0} \geq 0.
\end{aligned}$$

As the braces indicate, application of the subcase assumption leads to the lhs being always positive. This means that the second part of *isect1* is true.

A short summary: It has been shown that once an admissible v_{i-1} has been found for this case, it can be decreased without any effect on the non-emptiness of the intersection.

B.1.2.2 Case 2.2, $c_i < c_{i-1}$

Via $c_{i-1}-c_i > 0$ the case assumptions lead to the unconditioned existence of v_1^* and v_2^* with $v_1^* < 0$. Likewise, v_1 exists $\Rightarrow\ v_1 < 0$ follows directly from the definition of v_1 and the case assumptions.

The relevant intervals for intersection are now $[v_2, v_2^*]$ and $[0, v_2^*]$, respectively, depending on the existence of v_2.

We derive that $v_2 \leq v_{max|a_t}$: since $v_{max|a_t} > 0$ by definition (Eq. (3.30)), this is clear for $v_2 < 0$ and we can therefore assume $v_2 \geq 0$.

$$\begin{aligned}
& v_2 \leq v_{max|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \leq \sqrt{{v_{i-1}}^2 + 2\Delta_s a_{t_{max}}}
\end{aligned}$$

both sides are non-negative due to the above assumption, therefore they can be squared:

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 - 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}} \leq 4{c_i}^2 {v_{i-1}}^2 + 8{c_i}^2\Delta_s a_{t_{max}} \\
& \Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}}) \leq 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
& \Leftrightarrow (c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(a_{\omega_{max}} - c_i a_{t_{max}})}{(c_{i-1}-c_i)v_{i-1}} \leq \sqrt{\dots}.
\end{aligned}$$

As the lhs is never positive according to case assumptions, the inequality can easily be identified as a tautology. Therefore, $v_2 \leq v_{max|a_t}$ holds.

Combining all information gained so far, we receive a non-empty intersection as soon as $v_{min|a_t} \leq v_2^*$ holds. We will now derive the condition for this:

For the case that $v_{min|a_t} = 0$ it is always $v_{min|a_t} \leq v_2^*$ as $v_2^* > 0$:

$$\begin{aligned}
& v_2^* > 0 \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) > 0 \\
\Leftrightarrow\ & (c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}} < 0.
\end{aligned}$$

This can be derived as follows:

$$\begin{aligned}
& (c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}} \\
& < (c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2} \\
& = (c_{i-1}-c_i)v_{i-1} - |c_i+c_{i-1}|v_{i-1} \\
& = (-|c_{i-1}| + |c_i| - |c_i| - |c_{i-1}|)v_{i-1} \\
& = 2c_{i-1}v_{i-1} \\
& < 0.
\end{aligned}$$

We proceed to the more general case $v_{min|a_t} > 0$:

$$\begin{aligned}
& v_2^* \geq v_{min|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}
\end{aligned}$$

non-negative lhs and rhs allow squaring

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 - 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}} \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2\Delta_s a_{t_{max}} \\
& \Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}}) \geq 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
& \Leftrightarrow (c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

For this inequality to hold, the positiveness of the lhs is a vital precondition:

$$\begin{aligned}
&(c_{i-1}+c_i)v_{i-1}+\frac{4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}}\geq 0\\
\Leftrightarrow\ &(c_{i-1}+c_i){v_{i-1}}^2\geq -\frac{4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{c_{i-1}-c_i}\\
\Leftrightarrow\ &{v_{i-1}}^2\leq -\frac{4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\\
\Leftrightarrow\ &v_{i-1}\leq \sqrt{-\frac{4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}
\end{aligned}$$

Under the assumption of this condition to be fulfilled, we continue processing the original inequality by squaring it. Note that the term $(c_{i-1}+c_i)^2{v_{i-1}}^2$ cancels out immediately.

$$\begin{aligned}
&\frac{8(c_{i-1}+c_i)c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)}+\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\geq -8c_i\Delta_s a_{\omega_{max}}\\
\Leftrightarrow\ &\frac{8c_i\Delta_s((c_{i-1}+c_i)(c_i a_{t_{max}}-a_{\omega_{max}})+(c_{i-1}-c_i)a_{\omega_{max}})}{(c_{i-1}-c_i)}\geq -\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\\
\Leftrightarrow\ &\frac{{c_i}^2((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}})}{(c_{i-1}-c_i)}\geq -\frac{2{c_i}^2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2}\\
\Leftrightarrow\ &((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}){v_{i-1}}^2\geq -\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)}\\
\Leftrightarrow\ &{v_{i-1}}^2\leq \frac{-2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}})}\\
\Leftrightarrow\ &v_{i-1}\leq \sqrt{\frac{-2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}})}}.
\end{aligned}$$

We show that the precondition on v_{i-1} can be safely dropped, as values that fulfill the last inequality always fulfill that precondition:

$$\begin{aligned}
&-\frac{4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}\geq \frac{-2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}})}\\
\Leftrightarrow\ &\frac{2c_i}{c_{i-1}+c_i}\geq \frac{c_i a_{t_{max}}-a_{\omega_{max}}}{(c_{i-1}+c_i)a_{t_{max}}-2a_{\omega_{max}}}\\
\Leftrightarrow\ &\underbrace{\frac{2|c_i|}{|c_{i-1}|+|c_i|}}_{>1}\geq \underbrace{\frac{|c_i|a_{t_{max}}+a_{\omega_{max}}}{|c_{i-1}|a_{t_{max}}+a_{\omega_{max}}+|c_i|a_{t_{max}}+a_{\omega_{max}}}}_{<1}.
\end{aligned}$$

As a last consideration for this case it will be shown that the computed bound for the case $v_{min|a_t}>0$

always respects this condition, namely $v_{i-1} > \sqrt{2\Delta_s a_{t_{max}}}$:

$$\begin{aligned}
&\sqrt{\frac{-2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})}} > \sqrt{2\Delta_s a_{t_{max}}} \\
\Leftrightarrow\ &\frac{-(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})} > a_{t_{max}} \\
\Leftrightarrow\ &-(c_i a_{t_{max}} - a_{\omega_{max}})^2 < a_{t_{max}}(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}) \\
\Leftrightarrow\ &-{c_i}^2 a_{t_{max}}^2 + 2c_i a_{t_{max}} a_{\omega_{max}} - a_{\omega_{max}}^2 < a_{t_{max}}^2({c_{i-1}}^2 - {c_i}^2) - 2(c_{i-1}-c_i)a_{t_{max}}a_{\omega_{max}} \\
\Leftrightarrow\ &-{c_{i-1}}^2 a_{t_{max}}^2 + 2c_{i-1} a_{t_{max}} a_{\omega_{max}} - a_{\omega_{max}}^2 < 0.
\end{aligned}$$

The lhs is negative through the case assumptions.

Combining all knowledge gained we conclude for this case:

$$\begin{aligned}
&\big((c_i < 0) \wedge (c_{i-1} \leq 0) \wedge (c_i < c_{i-1})\big) \\
&\quad\Rightarrow\ \left(I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \Leftrightarrow\ v_{i-1} \leq \sqrt{-\frac{2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)\left((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}\right)}}\right).
\end{aligned} \tag{B.27}$$

B.1.2.3 Case 2.3, $c_i = c_{i-1}$

The definitions of v_1 and v_1^* (Eqs. (3.32) and (3.33)) reveal that $v_1 < 0$ (if exists) and $v_1^* < 0$ for the case $c_i = c_{i-1}$. They also show that v_2 and v_2^* degenerate according to:

$$v_2 = \sqrt{{v_{i-1}}^2 + \frac{2a_{\omega_{max}}\Delta_s}{c_i}}, \quad v_2^* = \sqrt{{v_{i-1}}^2 - \frac{2a_{\omega_{max}}\Delta_s}{c_i}}.$$

The intersection of I_{a_ω} and $[v_{min|a_t}, v_{max|a_t}]$ is never empty because one interval is always contained in the other one, as can be seen in the definitions in Eqs. (3.29) and (3.30).

Therefore, we conclude for this case

$$\big((c_i < 0) \wedge (c_{i-1} \leq 0) \wedge (c_i = c_{i-1})\big) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \tag{B.28}$$

B.1.3 Case 3, $c_i < 0, c_{i-1} > 0$

This case yields $c_{i-1} - c_i > 0$ and therefore its assumptions have $v_1^* < 0$ and $v_1 < 0$ (if exists) as a consequence (Definitions are in in Eqs. (3.32) and (3.33)). Furthermore, it is v_2 exists $\Rightarrow\ v_2 < 0$:

$$\begin{aligned}
&v_2 < 0 \\
\Leftrightarrow\ &\frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) < 0 \\
\Leftrightarrow\ &(c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}} > 0.
\end{aligned}$$

This is derived through:

$$\begin{aligned}
&(c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}} \\
&> (c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i + c_{i-1})^2 {v_{i-1}}^2} \\
&= (c_{i-1} - c_i - |c_i + c_{i-1}|)v_{i-1} \\
&= (|c_{i-1}| + |c_i| - |-|c_i| + |c_{i-1}||)v_{i-1} \\
&> 0.
\end{aligned}$$

The interval to intersect with $[v_{min|a_t}, v_{max|a_t}]$ therefore reduces to $[0, v_2^*]$. A first precondition for a non-empty intersection will be $v_2^* > 0$:

$$\begin{aligned}
& v_2^* > 0 \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) > 0 \\
\Leftrightarrow\ & (c_{i-1}-c_i)v_{i-1} < \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}
\end{aligned}$$

we take both sides to the power of two

$$\begin{aligned}
\Leftrightarrow\ & (c_{i-1}-c_i)^2 {v_{i-1}}^2 < (c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & -4c_i c_{i-1} {v_{i-1}}^2 < -8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ & {v_{i-1}}^2 < \frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\
\Leftrightarrow\ & v_{i-1} < \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}.
\end{aligned}$$

As before, we name this condition

$$v2spos = \left(v_{i-1} < \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}\right). \tag{B.29}$$

An intersection will occur as soon as $v_2^* \geq v_{min|a_t}$. For the case $v_{min|a_t} = 0$ the intersection is non-empty as soon as v_2^* is positive ($v2spos$):

$$(vmatzero \wedge v2spos) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.30}$$

where

$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}}\right). \tag{B.31}$$

For the case $v_{min|a_t} > 0$ ($\neg vmatzero$), the following has to hold:

$$\begin{aligned}
& v_2^* \geq v_{min|a_t} \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} - \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}
\end{aligned}$$

squaring an inequality which has sign-sharing sides is an equivalence operation

$$\begin{aligned}
& \Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 - 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} + (c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}} \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2\Delta_s a_{t_{max}} \\
& \Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}}) \geq 2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
& \Leftrightarrow (c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

For the inequality to hold the lhs has to be non-negative:

$$\begin{aligned}
& (c_{i-1}+c_i)v_{i-1} + \frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)v_{i-1}} > 0 \\
\Leftrightarrow\ & (c_{i-1}+c_i){v_{i-1}}^2 > -\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{c_{i-1}-c_i}.
\end{aligned}$$

We have to distinguish two cases here: $c_{i-1}+c_i \geq 0$ and $c_{i-1}+c_i < 0$. For the former the inequality always hold and for the latter it translates to:

$$\begin{aligned} {v_{i-1}}^2 &< -\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \\ \Leftrightarrow v_{i-1} &< \sqrt{-\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}. \end{aligned}$$

We state the condition in a unified, more formal way:

$$\textit{precond1} = \left((c_{i-1}+c_i \geq 0) \vee \left((c_{i-1}+c_i < 0) \wedge \left(v_{i-1} < \sqrt{-\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}\right)\right)\right). \tag{B.32}$$

Note that this could be simplified, but we resist to do that in order to explicitly state what is true for which case. Assuming now that the condition *precond1* is fulfilled and the lhs of the original inequality therefore is non-negative, we proceed by squaring it. Note that the term $(c_{i-1}+c_i)^2{v_{i-1}}^2$ immediately cancels out.

$$\begin{aligned} &\frac{8c_i\Delta_s(c_{i-1}+c_i)(c_i a_{t_{max}} - a_{\omega_{max}})}{c_{i-1}-c_i} + \frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2} \geq -8c_i\Delta_s a_{\omega_{max}} \\ \Leftrightarrow\ &\frac{8c_i\Delta_s((c_{i-1}+c_i)(c_i a_{t_{max}} - a_{\omega_{max}}) + (c_{i-1}-c_i)a_{\omega_{max}})}{c_{i-1}-c_i} \geq -\frac{16{c_i}^2\Delta_s^2(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)^2{v_{i-1}}^2} \\ \Leftrightarrow\ &({c_i}^2((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}))){v_{i-1}}^2 \geq -\frac{2{c_i}^2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{c_{i-1}-c_i} \\ \Leftrightarrow\ &((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}){v_{i-1}}^2 \geq -\frac{2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{c_{i-1}-c_i} \end{aligned}$$

As will be shown below, it is $((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}) < 0$

$$\begin{aligned} \Leftrightarrow {v_{i-1}}^2 &\leq \frac{-2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})} \\ \Leftrightarrow v_{i-1} &\leq \sqrt{\frac{-2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})}}. \end{aligned}$$

We claimed $((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}}) < 0$, which is a consequence of our assumptions $\neg \textit{vmatzero} \wedge \textit{v2spos}$. For both assumptions to hold, it has to be:

$$\begin{aligned} 2\Delta_s a_{t_{max}} &< \frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\ \Rightarrow c_{i-1}a_{t_{max}} &< a_{\omega_{max}} \end{aligned}$$

With $c_i < 0$ by case assumption

$$\Rightarrow (c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}} < 0.$$

Combining the two possibilities for an intersection, we finally conclude for this case:

$$\Big((c_i < 0) \wedge (c_{i-1} > 0)\Big) \Rightarrow \Bigg(I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \Leftrightarrow \Bigg(v2spos \wedge \Bigg(vmatzero \vee \Bigg(\neg vmatzero \wedge precond1 \wedge \Bigg(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})}}\Bigg)\Bigg)\Bigg)\Bigg)\Bigg), \quad \text{(B.33)}$$

where

$$v2spos = \left(v_{i-1} < \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}\right)$$
$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}}\right)$$
$$precond1 = \left((c_{i-1}+c_i \geq 0) \vee \left((c_{i-1}+c_i < 0) \wedge \left(v_{i-1} < \sqrt{-\frac{4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}\right)\right)\right).$$

B.1.4 Case 4, $c_i > 0, c_{i-1} < 0$

This case bears some structural analogy to case 3. The case assumptions directly lead to $c_{i-1} - c_i < 0$, $v_2 < 0$ and $v_2^* < 0$ (if exists). Furthermore, it can be shown that v_1^* exists $\Rightarrow\ v_1^* < 0$:

$$\begin{aligned}
v_1^* &= \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 - 8c_i\Delta_s a_{\omega_{max}}}\right) \\
&< \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2}\right) \\
&= \frac{1}{2|c_i|}\left((-|c_{i-1}| - |c_i| + ||c_i| - |c_{i-1}||)v_{i-1}\right) \\
&= \frac{1}{2|c_i|}\left((-(|c_{i-1}| + |c_i|) + ||c_i| - |c_{i-1}||)v_{i-1}\right) \\
&< 0.
\end{aligned}$$

The interval remaining for intersection with $[v_{min|a_t}, v_{max|a_t}]$ therefore reduces to $[0, v_1]$. For a non-empty intersection we demand v_1 to be positive:

$$\begin{aligned}
& v_1 \geq 0 \\
\Leftrightarrow\ & \frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}\right) \geq 0 \\
\Leftrightarrow\ & -(c_{i-1}-c_i)v_{i-1} \leq \sqrt{(c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}}.
\end{aligned}$$

As both sides of the inequality are non-negative, we can square it:

$$\begin{aligned}
&\Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 \leq (c_i+c_{i-1})^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}} \\
&\Leftrightarrow -4c_{i-1}c_i {v_{i-1}}^2 \leq 8c_i\Delta_s a_{\omega_{max}} \\
&\Leftrightarrow {v_{i-1}}^2 \leq -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\
&\Leftrightarrow v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}.
\end{aligned}$$

We will call this condition

$$v1pos = \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}} \right). \tag{B.34}$$

A non-empty intersection will occur as soon as $v_1 \geq v_{min|a_t}$. For the case $v_{min|a_t} = 0$ this is the case as soon as *v1pos* holds:

$$(vmatzero \wedge v1pos) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.35}$$

where

$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}} \right). \tag{B.36}$$

We now consider the case $v_{min|a_t} > 0$ (i.e., $\neg vmatzero$), assuming *v1pos* to hold:

$$\begin{aligned}
&v_1 \geq v_{min|a_t} \\
\Leftrightarrow\ &\frac{1}{2c_i}\left((c_{i-1}-c_i)v_{i-1} + \sqrt{(c_{i-1}+c_i)^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}}} \right) \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}.
\end{aligned}$$

Both sides are non-negative, so they can be squared:

$$\begin{aligned}
&\Leftrightarrow (c_{i-1}-c_i)^2 {v_{i-1}}^2 + 2(c_{i-1}-c_i)\sqrt{\dots} + (c_{i-1}+c_i)^2 {v_{i-1}}^2 + 8c_i\Delta_s a_{\omega_{max}} \geq 4{c_i}^2 {v_{i-1}}^2 - 8{c_i}^2\Delta_s a_{t_{max}} \\
&\Leftrightarrow 2({c_{i-1}}^2 - {c_i}^2){v_{i-1}}^2 + 8c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}}) \geq -2(c_{i-1}-c_i)v_{i-1}\sqrt{\dots} \\
&\Leftrightarrow -(c_{i-1}+c_i)v_{i-1} - \frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq \sqrt{\dots}.
\end{aligned}$$

For the inequality to hold the lhs has to be non-negative:

$$\begin{aligned}
&-(c_{i-1}+c_i)v_{i-1} - \frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)v_{i-1}} \geq 0 \\
\Leftrightarrow\ &-(c_{i-1}+c_i){v_{i-1}}^2 \geq \frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{c_{i-1}-c_i}.
\end{aligned}$$

As the rhs is negative, this is a tautology for $c_{i-1} + c_i \leq 0$. For the complementary case $c_{i-1} + c_i > 0$ it translates to

$$\begin{aligned}
{v_{i-1}}^2 &\leq -\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \\
\Leftrightarrow v_{i-1} &\leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}.
\end{aligned}$$

We restate this precondition as

$$precond1 = \left((c_{i-1} + c_i \leq 0) \vee \left((c_{i-1} + c_i > 0) \wedge \left(v_{i-1} \leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)}}\right)\right)\right). \tag{B.37}$$

Assuming *precond1* to hold, we can now continue with the original inequality by squaring it. Note that the term $(c_i + c_{i-1})^2 {v_{i-1}}^2$ cancels out immediately:

$$\begin{aligned}
&\frac{8(c_{i-1} + c_i)c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1} - c_i)} + \frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2 {v_{i-1}}^2} \geq 8c_i\Delta_s a_{\omega_{max}} \\
\Leftrightarrow\ &\frac{8c_i\Delta_s((c_{i-1} + c_i)(a_{\omega_{max}} + c_i a_{t_{max}}) - (c_{i-1} - c_i)a_{\omega_{max}})}{(c_{i-1} - c_i)} \geq -\frac{16{c_i}^2\Delta_s^2(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2 {v_{i-1}}^2} \\
\Leftrightarrow\ &\frac{(c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}}}{(c_{i-1} - c_i)} \geq -\frac{2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)^2 {v_{i-1}}^2} \\
\Leftrightarrow\ &((c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}}){v_{i-1}}^2 \leq -\frac{2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{c_{i-1} - c_i}
\end{aligned}$$

Due to $(c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}} > 0$ (see below for derivation):

$$\begin{aligned}
&\Leftrightarrow {v_{i-1}}^2 \leq \frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)((c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}})} \\
&\Leftrightarrow v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)((c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}})}}.
\end{aligned}$$

The claim $(c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}} > 0$ can be justified as a consequence of the previous assumptions $\neg vmatzero \wedge v1pos$. For both of them to hold, it has to be

$$\begin{aligned}
2\Delta_s a_{t_{max}} &< -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\
\Rightarrow\ -c_{i-1}a_{t_{max}} &< a_{\omega_{max}} \\
\Rightarrow\ c_{i-1}a_{t_{max}} + a_{\omega_{max}} &> 0
\end{aligned}$$

With $c_i > 0$ by case assumption we then retrieve

$$\Rightarrow (c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}} > 0.$$

Summarizing the two possibilities for an intersection, we conclude:

$$\begin{aligned}
\Big((c_i > 0) \wedge (c_{i-1} < 0)\Big) \Rightarrow\ &\Big(I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \\
&\Leftrightarrow \Big(v1pos \wedge \Big(vmatzero \vee \Big(\neg vmatzero \wedge precond1 \\
&\wedge \Big(v_{i-1} \leq \sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1} - c_i)((c_{i-1} + c_i)a_{t_{max}} + 2a_{\omega_{max}})}}\Big)\Big)\Big)\Big)\Big),
\end{aligned} \tag{B.38}$$

where

$$v1pos = \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}} \right)$$
$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}} \right)$$
$$precond1 = \left((c_{i-1} + c_i \leq 0) \vee \left((c_{i-1} + c_i > 0) \wedge \left(v_{i-1} \leq \sqrt{-\frac{4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1} - c_i)(c_{i-1} + c_i)}} \right) \right) \right).$$

B.1.5 Case 5, $c_i = 0, c_{i-1} \neq 0$

This case treats the condition $c_i = 0$, recall that the interval to be intersected with $[v_{min|a_t}, v_{max|a_t}]$ now has the borders $\hat{v}_1$ and $\hat{v}_2$ which are defined by Eq. (3.37).

B.1.5.1 Case 5.1, $c_{i-1} > 0$

For this case the interval to be intersected with $[v_{min|a_t}, v_{max|a_t}]$ simplifies to $[0, \hat{v}_2]$ as the lower interval border $\hat{v}_1$ can be verified to be negative by the case assumptions.

For a non-empty intersection we demand the upper bound $\hat{v}_2$ to be non-negative:

$$\begin{aligned} & \hat{v}_2 \geq 0 \\ \Leftrightarrow\ & \frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1} \geq 0 \\ \Leftrightarrow\ & 2\Delta_s a_{\omega_{max}} \geq c_{i-1} {v_{i-1}}^2 \\ \Leftrightarrow\ & v_{i-1} \leq \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}. \end{aligned}$$

We call this condition

$$v2hpos = \left(v_{i-1} \leq \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}} \right). \tag{B.39}$$

In order to retrieve a non-empty intersection $\hat{v}_2 \geq v_{min|a_t}$ has to hold. In case $v_{min|a_t} = 0$ this is guaranteed by *v2hpos*:

$$(vmatzero \wedge v2hpos) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.40}$$

where

$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}} \right). \tag{B.41}$$

Assuming now $v_{min|a_t} > 0$ (i.e., $\neg vmatzero$) we more closely examine the condition

$$\begin{aligned} & \hat{v}_2 \geq v_{min|a_t} \\ \Leftrightarrow\ & \frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1} \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}. \end{aligned}$$

Due to our assumptions both sides are non-negative, so they can be squared:

$$\begin{aligned}
&\Leftrightarrow \frac{4\Delta_s^2 a_{\omega_{max}}^2}{{c_{i-1}}^2 {v_{i-1}}^2} - \frac{4\Delta_s a_{\omega_{max}}}{c_{i-1}} + {v_{i-1}}^2 \geq {v_{i-1}}^2 - 2\Delta_s a_{t_{max}} \\
&\Leftrightarrow \frac{4\Delta_s^2 a_{\omega_{max}}^2}{{c_{i-1}}^2} + (2\Delta_s a_{t_{max}} - \frac{4\Delta_s a_{\omega_{max}}}{c_{i-1}}){v_{i-1}}^2 + \geq 0 \\
&\Leftrightarrow (2\Delta_s a_{t_{max}} {c_{i-1}}^2 - 4\Delta_s a_{\omega_{max}} c_{i-1}){v_{i-1}}^2 + \geq -4\Delta_s^2 a_{\omega_{max}}^2 \\
&\Leftrightarrow (c_{i-1} a_{t_{max}} - 2a_{\omega_{max}}) c_{i-1} {v_{i-1}}^2 + \geq -2\Delta_s a_{\omega_{max}}^2.
\end{aligned}$$

As it is $c_{i-1} a_{t_{max}} - 2a_{\omega_{max}} < 0$ (derivation to follow)

$$\begin{aligned}
&\Leftrightarrow {v_{i-1}}^2 \leq \frac{-2\Delta_s a_{\omega_{max}}^2}{(c_{i-1} a_{t_{max}} - 2a_{\omega_{max}}) c_{i-1}} \\
&\Leftrightarrow v_{i-1} \leq \sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{(c_{i-1} a_{t_{max}} - 2a_{\omega_{max}}) c_{i-1}}}.
\end{aligned}$$

The derivation of $c_{i-1} a_{t_{max}} - 2a_{\omega_{max}} < 0$ is obtained through the previous assumptions $\neg vmatzero \wedge v1hpos$. In case both are true, it holds

$$\begin{aligned}
2\Delta_s a_{t_{max}} &< -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\
\Rightarrow\ -c_{i-1} a_{t_{max}} &< a_{\omega_{max}} \\
\Rightarrow c_{i-1} a_{t_{max}} + a_{\omega_{max}} &> 0 \\
\Rightarrow c_{i-1} a_{t_{max}} + 2a_{\omega_{max}} &> 0.
\end{aligned}$$

Combining all knowledge gained so far, we can conclude for this case:

$$\begin{aligned}
&\Big((c_i = 0) \wedge (c_{i-1} > 0)\Big) \Rightarrow \Bigg(I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \\
&\qquad \Leftrightarrow \Bigg(v2hpos \wedge \Bigg(vmatzero \vee \Bigg(\neg vmatzero \wedge \Bigg(v_{i-1} \leq \sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{(c_{i-1} a_{t_{max}} - 2a_{\omega_{max}}) c_{i-1}}}\Bigg)\Bigg)\Bigg)\Bigg)\Bigg),
\end{aligned} \tag{B.42}$$

where

$$\begin{aligned}
v2hpos &= \left(v_{i-1} \leq \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}\right) \\
vmatzero &= \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}}\right).
\end{aligned}$$

B.1.5.2 Case 5.2, $c_{i-1} < 0$

For this case $\hat{v}_2$ is negative and hence the interval to be intersected with $[sv_{min|a_t}, v_{max|a_t}]$ reduces to $[0, \hat{v}_1]$. For the intersection to be non-empty, we demand $\hat{v}_1$ to be positive:

$$\begin{aligned} & \hat{v}_1 \geq 0 \\ \Leftrightarrow & -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1} \geq 0 \\ \Leftrightarrow & -2\Delta_s a_{\omega_{max}} \leq c_{i-1} {v_{i-1}}^2 \\ \Leftrightarrow & v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}. \end{aligned}$$

This will be abbreviated by

$$v1hpos = \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}} \right). \tag{B.43}$$

For the intersection with $[v_{min|a_t}, v_{max|a_t}]$ to be non-empty, $\hat{v}_1 \geq v_{min|a_t}$ has to hold. For the case $v_{min|a_t} = 0$ this is true, as soon as *v1hpos* is fulfilled:

$$(v1hpos \wedge vmatzero) \Rightarrow I_{a\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset, \tag{B.44}$$

with

$$vmatzero = \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}} \right). \tag{B.45}$$

For $v_{min|a_t} > 0$ (i.e., $\neg vmatzero$) the condition for a non-empty intersection is derived as follows:

$$\begin{aligned} & \hat{v}_1 \geq v_{min|a_t} \\ \Leftrightarrow & -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1} v_{i-1}} - v_{i-1} \geq \sqrt{{v_{i-1}}^2 - 2\Delta_s a_{t_{max}}}. \end{aligned}$$

Both sides of the inequality share signs, so they can be squared:

$$\begin{aligned} & \Leftrightarrow \frac{4\Delta_s^2 a_{\omega_{max}}^2}{{c_{i-1}}^2 {v_{i-1}}^2} + \frac{4\Delta_s a_{\omega_{max}}}{c_{i-1}} + {v_{i-1}}^2 \geq {v_{i-1}}^2 - 2\Delta_s a_{t_{max}} \\ & \Leftrightarrow \frac{2\Delta_s a_{\omega_{max}}^2}{{c_{i-1}}^2 {v_{i-1}}^2} + \frac{2a_{\omega_{max}}}{c_{i-1}} \geq -a_{t_{max}} \\ & \Leftrightarrow 2\Delta_s a_{\omega_{max}}^2 + (2a_{\omega_{max}} c_{i-1} + {c_{i-1}}^2 a_{t_{max}}) {v_{i-1}}^2 \geq 0 \\ & \Leftrightarrow (2a_{\omega_{max}} + c_{i-1} a_{t_{max}}) c_{i-1} {v_{i-1}}^2 \geq -2\Delta_s a_{\omega_{max}}^2. \end{aligned}$$

Claiming $2a_{\omega_{max}} + c_{i-1} a_{t_{max}} > 0$ (proof below), we retrieve

$$\begin{aligned} & \Leftrightarrow {v_{i-1}}^2 \leq \frac{-2\Delta_s a_{\omega_{max}}^2}{(2a_{\omega_{max}} + c_{i-1} a_{t_{max}}) c_{i-1}} \\ & \Leftrightarrow v_{i-1} \leq \sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{c_{i-1}(2a_{\omega_{max}} + c_{i-1} a_{t_{max}})}}. \end{aligned}$$

The previously made assumptions $\neg vmatzero \wedge v1hpos$ lead to the above claimed $2a_{\omega_{max}} + c_{i-1}a_{t_{max}} > 0$. For the truth of both it is required:

$$\begin{aligned} 2\Delta_s a_{t_{max}} &< -\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}} \\ \Rightarrow\ -c_{i-1}a_{t_{max}} &< a_{\omega_{max}} \\ \Rightarrow c_{i-1}a_{t_{max}} + a_{\omega_{max}} &> 0 \\ \Rightarrow c_{i-1}a_{t_{max}} + 2a_{\omega_{max}} &> 0. \end{aligned}$$

Concluding for this case we summarize the conditions for a non-empty intersection:

$$\begin{aligned} \Big((c_i = 0) \wedge (c_{i-1} < 0)\Big) \Rightarrow \Bigg(& I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset \\ & \Leftrightarrow \left(v1hpos \wedge \left(vmatzero \vee \left(\neg vmatzero \right.\right.\right. \right. \\ & \left.\left.\left.\left. \wedge \left(v_{i-1} \leq \sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{c_{i-1}(2a_{\omega_{max}} + c_{i-1}a_{t_{max}})}} \right) \right) \right) \right) \Bigg), \end{aligned} \tag{B.46}$$

where

$$\begin{aligned} v1hpos &= \left(v_{i-1} \leq \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}} \right) \\ vmatzero &= \left(v_{i-1} \leq \sqrt{2\Delta_s a_{t_{max}}} \right). \end{aligned}$$

B.1.6 Case 6, $c_i = c_{i-1} = 0$

Similarly to cases 1.3 and 2.3 the intersection with $[v_{min|a_t}, v_{max|a_t}]$ is always non-empty. This is due to the fact that $I_{a_\omega} = \mathbb{R}_0^+$ for this case (see Eq. (3.31)). The conclusion for this case therefore is:

$$\big((c_i = 0) \wedge (c_{i-1} = 0)\big) \Rightarrow\ I_{a_\omega} \cap [v_{min|a_t}, v_{max|a_t}] \neq \emptyset. \tag{B.47}$$

We have now derived the biggest bound to v_{i-1} that guarantees a non-empty intersection of $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω} for all smaller values. This constraint can be treated as an isolated constraint and is accounted for in the first phase of velocity profile generation. Note that it is also crucial to ensure that v_i lies within the now guaranteed overlap. To do this, we apply the same method backwards, assuming to move from p_i to p_{i-1}. This is possible due to our symmetry assumptions that imply identical absolute values for accelerational and decelerational constraints.

B.2 Pseudo Code

This section presents a pseudo code algorithm that aggregates the results of the previous section. It computes an upper bound for v_{i-1} that guarantees overlap of $[v_{min|a_t}, v_{max|a_t}]$ and I_{a_ω} at p_i for all smaller values. Note that for implementation the algorithm can be optimized by delaying the application of the square root at various locations. To maintain coherence with the derivations this has not been done here.

if $c_i > 0 \wedge c_{i-1} \geq 0$ **then** // Case 1
  if $c_i > c_{i-1}$ **then** // Case 1.1
    thresh $\leftarrow \sqrt{\frac{2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(a_{t_{max}}(c_i+c_{i-1})+2a_{\omega_{max}})(c_i-c_{i-1})}}$;
  if $c_i < c_{i-1}$ **then** // Case 1.2
    thresh1 $\leftarrow \sqrt{\frac{8c_i a_{\omega_{max}}\Delta_s}{(c_{i-1}+c_i)^2}}$;
    tmp1 $\leftarrow \sqrt{\frac{4c_i\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})}{(c_{i-1}-c_i)^2}}$;
    tmp2 $\leftarrow \sqrt{\frac{2\Delta_s(c_i a_{t_{max}}+a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}}$;
    thresh_tmp1 $\leftarrow$ min(tmp1,tmp2);
    thresh_tmp2 $\leftarrow$ min($\sqrt{\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}}, \sqrt{2a_{t_{max}}\Delta_s}$);
    thresh_tmp3 $\leftarrow -\infty$;
    tmp $\leftarrow$ min($\frac{2a_{\omega_{max}}\Delta_s}{c_{i-1}}$, $\frac{2\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}$);
    if tmp $> \frac{-4c_i\Delta_s(c_i a_{t_{max}}-a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \wedge$ tmp $> 2a_{t_{max}}\Delta_s$ **then**
      thresh_tmp3 $\leftarrow \sqrt{\text{tmp}}$
    thresh $\leftarrow$ max(max(thresh1,thresh_tmp1),max(thresh_tmp2,thresh_tmp3))
  if $c_i = c_{i-1}$ **then** // Case 1.3
    thresh $\leftarrow \infty$
if $c_i < 0 \wedge c_{i-1} \leq 0$ **then** // Case 2
  if $c_i > c_{i-1}$ **then** // Case 2.1
    thresh1 $\leftarrow \sqrt{\frac{-8c_i a_{\omega_{max}}\Delta_s}{(c_{i-1}+c_i)^2}}$;
    tmp1 $\leftarrow \sqrt{\frac{-4c_i\Delta_s(a_{\omega_{max}}-c_i a_{t_{max}})}{(c_{i-1}+c_i)(c_{i-1}-c_i)}}$;
    tmp2 $\leftarrow \sqrt{\frac{-2\Delta_s(a_{\omega_{max}}-c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}-(c_{i-1}+c_i)a_{t_{max}})}}$;
    thresh_tmp1 $\leftarrow$ min(tmp1,tmp2);
    thresh_tmp2 $\leftarrow$ min($\sqrt{\frac{-2a_{\omega_{max}}\Delta_s}{c_{i-1}}}, \sqrt{2a_{t_{max}}\Delta_s}$);
    thresh_tmp3 $\leftarrow -\infty$;
    tmp $\leftarrow$ min($\frac{-2a_{\omega_{max}}\Delta_s}{c_{i-1}}$, $\frac{-2\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})^2}{(c_{i-1}-c_i)(2a_{\omega_{max}}+(c_{i-1}+c_i)a_{t_{max}})}$);
    if tmp $> \frac{-4c_i\Delta_s(a_{\omega_{max}}+c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)} \wedge$ tmp $> 2a_{t_{max}}\Delta_s$ **then**
      thresh_tmp3 $\leftarrow \sqrt{\text{tmp}}$
    thresh $\leftarrow$ max(max(thresh1,thresh_tmp1),max(thresh_tmp2,thresh_tmp3))
  if $c_i < c_{i-1}$ **then** // Case 2.2
    thresh $\leftarrow \sqrt{\frac{-2\Delta_s(a_{\omega_{max}}-c_i a_{t_{max}})^2}{(c_{i-1}-c_i)((c_i+c_{i-1})a_{t_{max}}-2a_{\omega_{max}})}}$
  if $c_i = c_{i-1}$ **then** // Case 2.3
    thresh $\leftarrow \infty$

if $c_i < 0 \wedge c_{i-1} > 0$ **then** // Case 3
    vtwostarpos $\leftarrow \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}$;
    precond $\leftarrow \infty$;
    if $c_{i-1} + c_i < 0$ **then**
        precond $\leftarrow \sqrt{\frac{-4c_i\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}$;
    thresh_tmp $\leftarrow$ min(precond, $\sqrt{\frac{-2\Delta_s(c_i a_{t_{max}} - a_{\omega_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} - 2a_{\omega_{max}})}}$);
    thresh_tmp $\leftarrow$ max(thresh_tmp, $\sqrt{2\Delta_s a_{t_{max}}}$);
    thresh $\leftarrow$ min(thresh_tmp, vtwostarpos)
if $c_i > 0 \wedge c_{i-1} < 0$ **then** // Case 4
    vonestarpos $\leftarrow \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}$;
    precond $\leftarrow \infty$;
    if $c_{i-1} + c_i > 0$ **then**
        precond $\leftarrow \sqrt{\frac{-4c_i\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})}{(c_{i-1}-c_i)(c_{i-1}+c_i)}}$;
    thresh_tmp $\leftarrow$ min(precond, $\sqrt{\frac{-2\Delta_s(a_{\omega_{max}} + c_i a_{t_{max}})^2}{(c_{i-1}-c_i)((c_{i-1}+c_i)a_{t_{max}} + 2a_{\omega_{max}})}}$);
    thresh_tmp $\leftarrow$ max(thresh_tmp, $\sqrt{2\Delta_s a_{t_{max}}}$);
    thresh $\leftarrow$ min(thresh_tmp, vonestarpos)
if $c_i = 0$ **then** // Case 5
    if $c_{i-1} > 0$ **then** // Case 5.1
        vtwohatpos $\leftarrow \sqrt{\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}$;
        thresh_tmp $\leftarrow$ max($\sqrt{2\Delta_s a_{t_{max}}}$, $\sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{c_{i-1}(c_{i-1}a_{t_{max}} - 2a_{\omega_{max}})}}$);
        thresh $\leftarrow$ min(vtwohatpos, thresh_tmp);
    if $c_{i-1} < 0$ **then** // Case 5.2
        vonehatpos $\leftarrow \sqrt{-\frac{2\Delta_s a_{\omega_{max}}}{c_{i-1}}}$;
        thresh_tmp $\leftarrow$ max($\sqrt{2\Delta_s a_{t_{max}}}$, $\sqrt{\frac{-2\Delta_s a_{\omega_{max}}^2}{c_{i-1}(c_{i-1}a_{t_{max}} + 2a_{\omega_{max}})}}$);
        thresh $\leftarrow$ min(vonehatpos, thresh_tmp);
if $c_i = 0 \wedge c_{i-1} = 0$ **then** // Case 6
    thresh $\leftarrow \infty$;
return thresh;

C Sobel Operator for Gradients

For waypoint translation we chose to use a coordinate system derived from the gradient of the distance map. Thereby the axes of the coordinate system correspond to changing the waypoint's obstacle distance and its lateral displacement with respect to the closest obstacle. To obtain the the distance map's gradient at a location on the map, we employ the Sobel Operator.

The Sobel Operator is a well known and commonly used method for 2D gradient computation on discrete data. It computes the gradient by averaging and differencing in a window around the desired coordinate, estimating a separate value for the x- and the y-component of the gradient. An introduction can be found in [5, Chapt. 7.3], the original Sobel Operator is introduced in [5, pp. 271-2].

To increase robustness, we use a 5x5 window instead of the original 3x3 one. The following equation displays the weights we use to compute the x-component S_x and the y-component S_y of the gradient:

$$S_x = \begin{pmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{pmatrix}, \qquad S_y = \begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}. \quad \text{(C.1)}$$

For our application the gradient magnitude is not important, so we can skip normalization and directly compute the angle $\alpha \in [0, 2\pi)$ of the gradient as

$$\alpha = \begin{cases} \arctan\left(\frac{S_y}{S_x}\right) & S_y \geq 0 \\ \arctan\left(\frac{S_y}{S_x}\right) + \pi & \text{else.} \end{cases} \quad \text{(C.2)}$$