Robust Wind-Aware Path Optimization on Board Small Fixed-Wing UAVs

Towards a More Robust Path Planning in Wind Using Trochoids and Clothoids

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master Thesis

Robust Wind-Aware Path Optimization On Board Small Fixed-Wing UAVs

Towards A More Robust Path Planning In Wind Using Trochoids and Clothoids

Autumn Term 2021

Supervised by:
Thomas Stastny
Sebastian Verling

Author:
Thomas Bucher

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

ROBUST WIND-AWARE PATH OPTIMIZATION ON BOARD SMALL FIXED-WING UAVs

Authored by (in block letters):

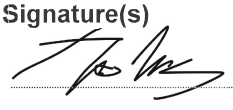
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
BUCHER	THOMAS

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Bern, 09.01.2022	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Preface

This work was written as a master thesis as part of the master program "Robotics, System and Control" at ETH Zurich. The research interest for this work was expressed by the Fixed-Wing Group of the Autonomous System Laboratory at ETH Zurich and was already initially covered by a semester project under the same name. Based on the outcome of the semester project, further research in the field was intended, especially the approach to bring the findings into application.

A special thanks goes to the two supervisors of the project, Thomas Stastny and Sebastian Verling, who supported the work with their broad knowledge in the field and with the outcomes of their previously conducted studies on similar research topics.

Abstract

When controlling UAVs in wind, especially for curved parts of trajectories, large path tracking deviations may be observed. This leads to two main problems: Areas with obstacles and no-fly zones may be violated or the final pose of the curved trajectory may be not reached exactly, degrading collected data for e.g., survey missions. Drone operators typically rely on the controller to attenuate for such wind disturbances during path following tasks. However little effort is spent on determining whether a designed path to follow is actually trackable in the current wind conditions, often leading to large track deviations. This work tries to tackle this problem with the approach to incorporate the wind estimation in the path planning stage.

The approach developed in this work accounts for steady wind in a horizontal 2D setup. Since computational efficiency is key for on-board reoptimization of current flight plans, lightweight root-finding algorithms are used to determine suitable path solutions. With the help of the geometric shapes of trochoids and shifted clothoids, turning segments that account for the wind conditions are generated. With a three segment approach for a point-to-point coordination, the algorithm computes the time-optimal path between an initial and final pose.

To enable UAVs to track the proposed paths, two path sampling and respective path following algorithms with two main focuses are proposed. One algorithm focuses on pre-sampling of the path which can be adapted to any arbitrary curved path for UAVs. The second approach focuses on computational efficiency when used on a microcontroller and hence describes the geometrical shape of the path with a minimal set of parameters. To show the ability to implement the algorithms in state-of-the-art autopilot software, the latter approach is implemented in PX4. With extensive Software in the Loop simulations with Gazebo the performance of the path following in different wind conditions was accessed. Results show that trochoidal paths provide a suitable choice when solely time-optimal navigation between two poses is of main interest. Clothoidal paths on the other hand provide better tracking capability along the path due to the modeling of the UAV roll dynamics.

Based on the outcome of this work, the developed software framework can be implemented on-board a UAV to validate the performance with real flight data. Further, the work offers a base to extend the path generation also for 3D application, which is for example useful for emergency landings in high winds. Moreover the path sampling and path following approaches can be tested for alternative arbitrary curves proposed by different path planning strategies for fixed-wing UAVs.

Symbols & Abbreviations

Symbols

α	Tangent angle at point in path, Expression for $\frac{g}{V_a}$
δ	Orientation of turning segment
Δ	Change
κ	Path Curvature
$\mathcal{H}$	Hamiltonian
ϕ	Bank Angle
λ	Co-States in Hamiltonian
σ	Linear interpolation parameter
$\bar{\phi}$	Maximum bank angle
$\dot{\phi}$	Bank angle rate
$\dot{\bar{\phi}}$	Maximum bank angle rate
ψ	Heading angle (w.r.t. North)
ω	Turn rate, Waypoint
A	Transition point between first and second segment
B	Transition point between second and third segment
C	Constant
e	Signed track error
e_{seg}	Closest distance to a segment
f	Objective function for root-Finding
H	Bisector plane
J	Cost function, Jacobian matrix
V_a	Vehicle airspeed
V_ω	Wind speed
k	Shift parameter for the last turning segment, Scaling factor robust trochoid
m	Shift parameter for initial and final turning segment
N	Number of solver iterations
n	Bisector plane vector
P_{cl}	Closest point on path
P_{UAV}	UAV position
P_s	Sector origin point
q	Segment vector
R	Turning radius
R_0	Minimum turning radius

r	Acceptance radius
T	Total flight time, Flight time for the last segment (depending on path type)
t	Time
$\hat{t}$	Tangent
t_a	Flight time of the first segment
t_b	Flight time of the third segment
t_{int}	Integration time constant for clothoid segments
u	Control input
$\bar{u}$	Control input limit
x	North axis
y	East axis
z	Limitation parameter of Newton-Rapshon solver update step

Acronyms and Abbreviations

AIAA	American Institute of Aeronautics and Astronautics Journal
ETH	Eidgenössische Technische Hochschule
FTIW	Flight Time in Wind
GCS	Ground Control Station
IC	Initial Condition
ICUAS	International Conference on Unmanned Aircraft Systems
MBA	Maximum Bank Angle
PD	Proportional–Derivative
PID	Proportional–Integral–Derivative
SAR	Scan and Rescue
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VRP	Vehicle Routing Problem
WSL	Windows Subsystem for Linux

Abbreviations for Path Types

C	Curved
S	Straight
CSC	Curved-Straight-Curved Path Type
CCC	Curved-Curved-Curved Path Type
L	Maximum-Rate Turn to the left
R	Maximum-Rate Turn to the right
S	Straight Segment
RSR	Right-Straight-Right Path Type
RSL	Right-Straight-Left Path Type
LSR	Left-Straight-Right Path Type
LSL	Left-Straight-Left Path Type
RLR	Right-Left-Right Path Type
LRL	Left-Right-Left Path Type

Chapter 1

Introduction

This work is based on [1] and hence the background as well as some of the introduction overlap. Due to that, certain parts of this chapter are adapted from [1].

1.1 Background

The use of unmanned aerial vehicles (UAVs) has seen rapid development in recent years. Two of the main research interests are search and rescue- (SAR) or general survey missions, for example in poorly accessible areas. For this application, small fixed-wing and/or convertible VTOL platforms (e.g., tail sitter UAVs like Wingtra[1]) are of particular interest, since they have much greater range than multi-copter platforms.

Due to the small and lightweight design of such fixed-wing UAVs or VTOL platforms, some challenges arise, the sensitivity to wind being one of them. Especially, scanning missions are often carried out in environmental conditions that can involve large winds and turbulences. In figure 1.1 the behavior of a UAV for a simple waypoint-based turn segment between *Waypoint 1* and *Waypoint 2* during a scanning-mission in wind is shown. Due to the wind, the UAV is not able to follow the desired path properly leading to two major problems. The first being areas with obstacles or no-fly zones, which could be crossed when deviating from the planned path. Furthermore also a change of terrain in such an area could be a severe problem. The exact tracking of *Waypoint 2* is particularly of great interest for the situation when returning to a scanning segment at a desired heading and position. If one or both are not reached, the collected mapping or sensing data quality may decrease.

[1] Wingtra - VTOL drones for mapping and surveying: https://wingtra.com/

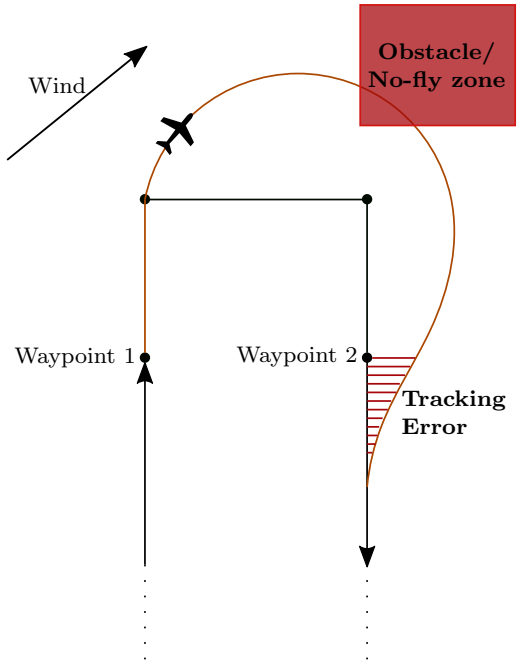


Figure 1.1: The behaviour of a UAV for a simple waypoint turning segment in wind.

To be able to continue the mission plan without a major deviation, the wind conditions have to be taken into consideration when controlling the UAV. This robustness to wind disturbances on board small fixed-wing UAVs is currently approached on the guidance level, indicated by *Path Following* in the simplified flow chart of a UAV control structure in figure 1.2. However, the mission plans remain ground-relative without consideration of curvature constraints induced by the wind speed or direction. Although wind-aware guidance logic is able to decrease deviation from the planned mission path segment, appropriately adjusted paths taking the vehicle constraints and the current wind conditions into account would further decrease the track deviations, which results in the UAV being able to reach the desired position with higher certainty.

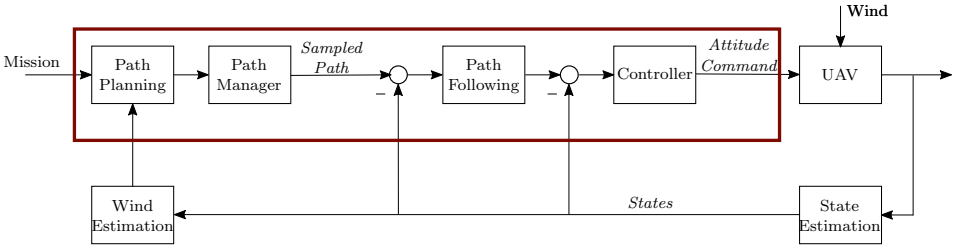


Figure 1.2: Simplified diagram of the general guidance, navigation and control structure of a UAV.

1.2 Objective

Based on the given situation, explained in section 1.1, the aim of [1] was to incorporate wind estimations, based on on-board and/or ground control station (GCS) sensors, already in the path planning procedure. This leads to better trackable paths under the current wind conditions. The subsystem of interest is marked in red in figure 1.2. To further increase the robustness of such an approach, the uncertainty in wind speed and heading due to gusts, the turn rate constraints of the aircraft, as well as the guidance level controller stability properties along the designed paths were considered.

For the optimization in wind, trochoids (arcs with instantaneous curvature change shifted by wind) or clothoids (arcs with curvature rate constraints shifted by wind) were used. For these geometrical shapes, the time-optimal path combination can be found with simple numerical methods for root-finding. In order to find the optimal path between an initial and final pose of the UAV in the given wind situation, a framework was proposed which allows to calculate all path variations with trochoidal as well as clothoidal path segments. To be able to run re-optimizations of the current flight plan on-board when the wind situation changes significantly, the framework was implemented in a suitable way to be run on a microcrontroller on-board the UAV. Ideally, this framework would be able to be run on board a small microcontroller or a companion computer on-board the UAV.

Based on the outcome of [1] the goal is to extend the implemented framework with further variations of the given geometrical approaches to have a complete set of paths to evaluate the time-optimal path between the initial and final pose of the UAV. Given the time-optimal path for a certain situation can be evaluated, the goal then is to pass the path to the guidance controller, indicated with the *Path Following* and *Controller* in figure 1.2, to allow the UAV to follow the proposed path. Most current autopilot software for drones such as PX4[2] only allow to follow straight lines or constant circular paths (loiters). This and the fact that for the path types generated by the framework no analytical closest point formulation exists, leads to the need for a new path following approach. The proposed approach has to be computationally efficient since the according values have to be calculated in every control loop during the flight of the UAV.

To show that the generated paths of the framework can be followed by a UAV, the path following approach should be implemented in PX4 which offers a working autopilot software stack. For simulations then Software in the Loop (SiL) tests should be used to show the functionality of the proposed approach and allow a comparison with existing path following techniques. Finally, flight tests with a small fixed-wing UAV could then conclude on the question if the proposed approach is a suitable technique to improve the navigation of fixed-wing UAVs in wind conditions.

1.3 Assumptions

As a basis for the consideration in this work, the assumptions and simplifications used throughout this work are established in the following.

The problem setup for the path planning is a point-to-point coordination between two poses in 2D. The definition of a setup is hence the position and the heading for the initial as well as for the final position. These are marked red in figure 1.3

[2] PX4 - Open Source Autopilot for Drones: https://px4.io/

showing an example path in the 2 coordinate system. For the path planning procedure constant wind is assumed, which should be tackled with a certain amount of robustness to uncertainties in the wind estimation as described in section 1.2. The prevailing wind is generally assumed towards north if not otherwise specified. Since the point-to-point coordination is calculated for the 2D case, only horizontal wind is considered. This wind component is generally assumed to be of higher magnitude and spatial extent when comparing to the vertical wind component. For the considerations in this work, we do not take wind speed into account, that exceeds the airspeed. In comparison to the nominal part of the wind speed, the uncertain part is small and only slowly changing. The uncertainty of the wind estimation is assumed in both, magnitude and angle. All vectors and other parameters are given with respect to an earth-fixed local Cartesian North-East coordinate system. In this system, North is referred by x and East with y as indicated in figure 1.3.

The UAV is assumed to fly coordinated turns, meaning turns with zero sideslip angle such that the *heading* is aligned with the air-path track *angle* at all times. For the path planning approach, segment-wise stationary flight conditions are assumed, meaning that changes of actuator position and resulting aerodynamic forces are only required to compensate for path deviations. Moreover, it is assumed that the flight control system already enforces the UAV not to stall or exceed the maximum load factor during turns.

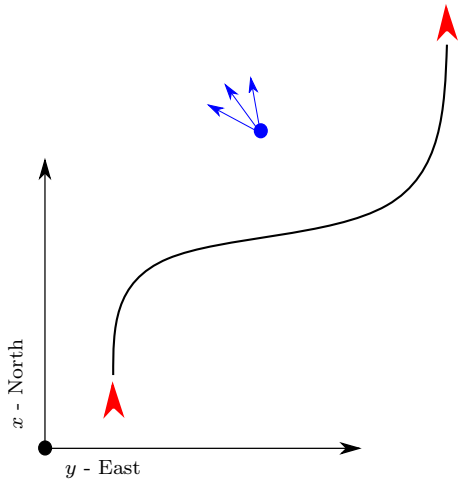


Figure 1.3: Point-to-point coordination in a 2D NE-coordinate system. Initial and final pose indicated in red, varying wind conditions in blue.

As a summary and for reference later on in the work, the framework conditions are summarized in table 1.1.

Condition	**Explanation**
Point-to-point Coordination	Coordination from initial to final pose only in 2D. The third dimension is omitted.
Coordinated Turns	For the turns no sideslip is assumed.
Coordinate System	Earth-fixed local 2D Cartesian North-East (NE) coordinate system. As x-axis the North-axis and as y-axis the East-axis is defined.
Environment	No terrain, obstacles or no-fly zones are assumed.
Wind	Due to the 2D problem setup, only the horizontal wind is considered. The wind speed does not exceed the airspeed.
Wind Uncertainty	Only a small part of the nominal wind estimation is assumed to be uncertain and slowly varying. Uncertainty in the magnitude and angle estimation of the wind.
UAV Behaviour	Parameters for small fixed-wing UAVs are assumed. The flight control system already enforces the UAV not to stall or exceed the maximum load factor during turns. Segment-wise stationary flight conditions.

Table 1.1: Summary of framework conditions.

1.4 Research Questions

The objective of this work is to find alternative path types for path planning that allow to take the prevailing wind condition into account. Therefore a new path following approach has to be proposed, which can be implemented in state-of-the-art UAV autopilot systems. The overall approach will then be evaluated in software and hardware tests. Therefore during this work, the goal is to answer the following key questions:

- Is there a significant difference in computation time for the two different path types and their respective variations?
- Which path types are most likely to lead to minimal time in certain setups?
- What is the behavior in strong winds (wind speed close to vehicle speed)?
- What are the different ways to follow the generated paths and how do the approaches differ?
- Are there other path types which could be used with the proposed path following approach?

- What is the behavior of the UAV dynamics for the new path types in different setups? How do they compare with currently implemented path planning approaches?
- How does the new overall approach to navigate in wind perform in different scenarios in software and in hardware tests?
- How do we account for uncertainties in the wind estimation? What is possible beyond conservative performance limits?

1.5 Structure of Work

In the following chapters, progressively a solution for robust wind-aware path optimization is built and tested. In chapter 2 the current literature is reviewed and the most relevant work is presented. In chapter 3 all mathematical basics are built, to be able to derive a suitable path planning framework in wind. Chapter 4 presents the extension of the path planning framework presented in [1] by a additional path type and the results of the complete framework are discussed. To enable state-of-the-art UAV guidance controllers to follow the proposed paths, different path sampling and following approaches are shown in chapter 5. To show the ability to implement the proposed approach in an open source autopilot in chapter 6 the extension of the PX4 software stack is shown. Furthermore the results for various simulations in this setup are presented and compared. Chapter 7 builds the end with the conclusion and suggestions for further advances based on the outcome of this work.

Chapter 2

Literature Research

In this chapter, the research done in the past as well as state-of-the-art approaches in the field of interest of this work are presented. The work covers in subsequent chapters various topics for which the literature research can be found here grouped in thematic subsections. If needed, a short introduction to the subtopic is given and the most important keywords used in the literature research are stated. Below you find a brief overview of the topics of the subsection as well as the motivation for the specific fields of research in this work.

Over the last years, research and commercial interest in UAVs gained massively. Papers and books are addressing various topics, reaching from design, launching systems to controller theory for UAVs. In this work, we are interested in approaches that increase the performance of UAVs in wind conditions as well as approaches to follow such arbitrary paths. The literature research in [1] mainly focused on general path planning for UAVs in wind with a focus on approaches using trochidal and clothoidal path segments. In this work path planning is still of interest, but the focal point is moved towards the implementation of mathematically demanding path types as for example CCC clothoidal paths or alternative advances such as Bézier curves. Both directions would allow for a wider comparison of different approaches to plan paths in wind and might bring advantages for the use in path following. Said advantages might be the better adaption of the path to the UAV dynamics or the possibility to use the path type in an optimization approach.

The path planning in [1] and in this work mostly focus on a 2D horizontal setup to plan and test paths. This is a fair assumption for testing the general improvement of the path planning and also the path following approaches. Implemented in a UAV acting in the 3D world also the vertical dimension becomes of interest. Whereas for lawnmower missions, executed mostly during steady height flight, this is not of significance, for landing or starting phases in high wind this is of main interest. Hence a suitable extension for the vertical dimension has to be found which still allows to use the optimized path to horizontal wind and account for vertical disturbances if necessary.

To be able to follow the generated paths, state-of-the-art guidance laws used to control fixed-wing UAVs have to be considered. The currently used approaches and their limitations are hereby of main interest to determine the path following advance to follow the wind-robust paths. Based on this field of research also the question arises how the generated paths can be represented that they can be used by state-of-the-art guidance controllers and hence also be deployed to the field. Unlike following a straight line or a loiter, for most of the proposed path types the calcu-

lation of relevant path following parameters does not have a closed-form solution. To be able to follow such paths, these calculations might be approximated or the path has to be sampled in a way that these parameters can be easily approximated. This brings in the need for an adapted path following approach which handles calculations of the relevant parameters as well as for example waypoint switching in a way that is suitable to use with small-fixed wing UAVs.
The most important keywords to find the literature presented in this section are:

Optimal path in Wind · Fixed-Wing UAV · Trochoids · Clothoids · Clothoid approximation · Path planning · Wind estimation uncertainty · Trajectory Planning

2.1 Path Planning in 2D

For an extensive review on the general path planning for small fixed-wing UAVs in- and excluding the consideration of wind, see [1]. In said work the literature review mainly focuses on the general approach to incorporate wind in path planning, the specific geometrical approach to incorporate wind in path planning with the use of trochidal and clothoidal paths and specific use cases where wind robust paths are already used, for example emergency landing scenarios.

In this work, the focus lies on further approaches to plan paths with consideration of the wind conditions. Hence the goal is to move from slightly augmented Dubins paths towards approaches that better incorporate the UAVs dynamics, which are simple to compute or are more suitable to use in optimization procedures.

2.1.1 Clothoidal Paths

Unlike trochoids, clothoids allow to incorporate the UAV roll dynamics in the path planning procedure [1]. Whereas a trochoidal segment assumes full bank angle from the starting point on, clothoidal segments have the property of a linearly increasing curvature. This effect can then in path planning be adapted to the specific UAV roll dynamics the path is planned for. Hence clothoids do not only allow to use the maximum bank angle of the UAV but also the bank angle change rate (roll rate). This allows to plan paths that are closely adapted to the UAVs physical capabilities. Although with this additional tuning parameter, the mathematical definition becomes more enhanced. This comes from the fact that for a trochoidal path segment only the state of being in full bank angle has to be considered, for clothoids two additional aircraft roll states have to be introduced. The two states are linearly transitioning to full bank angle and back to zero bank angle. The mathematical definition of the position on a clothoidal segment is built by Fresnel-Integrals. Hence to calculate the position there is no closed-form solution which makes numerical integration necessary. To be able to compute the solution of a path planning approach using clothoids in an efficient way, there might be a need for an approximation or a different approach. Further to mention is that the clothoidal paths used in path planning with wind do not resemble the standard form of clothoids. If a uniform constant wind field is incorporated in path planning, the behavior of the wind is added to the clothoid. Hence the implemented form for path planning is a shifted version of the basic mathematical definition of clothoids. This might lead to challenges or additional effort when using an approximation proposed in a different field of research.

The most important keywords to find the literature presented in this section are:

Optimal path in Wind · Fixed-Wing UAV · Clothoids · Clothoid approximation · Path planning · Wind estimation uncertainty · Trajectory Planning

[2] gives a well summarized overview towards state-of-the-art approaches that are used in path planning. It surfs well for the purpose to give a general overview of what literature has shown over the recent years and gives an outlook to future approaches that may get interesting for the use in UAV path planning. It indicates topics such as cooperative calculation techniques, cloud computing to optimize current flight paths or the advances in neuronal networks that have been done. Although [2] provides a concise summary, the proposed approaches are not suitable for the use in the main focus of this work being the generation of simple path planning on-board a UAV due to the extent of the computational efforts. Moving more towards the main focus of clothoid paths, literature does not show any specific approaches to use the basic clothoidal form in path planning. For implementation, the focus is more moved towards approximations that resemble the basic behavior of clothoid paths but show a more suitable form for implementation.

A first approach in this direction, not specifically for the use of path planning, is given in [3]. The work focuses on the fact that due to the irreducible Fresnel integrals in the clothoid formulation, the evaluation of such expressions is computationally expensive. The main motivation here is taken from the mathematical definition of clothoids which are incompatible with the polynomial / rational representations used in computer aided geometric design. Therefore an approach is shown that allows to approximate the behavior of a clothoid satisfying a prescribed tolerance using Pythagorean-hodrograph (PH) curves as polynomial approximations. A framework is proposed which then approximates a clothoidal path based on end points, tangents and curvature. Therefore a PH curve of degree 7 is involving the iterative solution of a system of five algebraic equations in which five real unknowns have to be solved for. This approach is chosen to approximate finite segment of the clothoid curve due to the fact that clothoids are transcendental curves for which it is impossible to parametric a whole segment exactly by a polynomial or rational function. The computational extents of the involved calculations are still high and due to the area of use presented in the work, no specific approach for the use in path planning can be concluded.

[4] proposes a comprehensive approach for path planning. Motivated by the goal to generate C^2 continuous paths based on the physical limitation parameters imposed by the UAV dynamics, an extensive path planning as well as a suitable controller are proposed. To be able to generate C^2 continuous, time-optimal paths in a UAV navigation setup, the clothoidal behavior is approximated by cubic splines. To find the time-optimal path in a given setting, the classical Dubins approach in 3D is used where then the continuous curvature paths are generated based on the the method presented in [5] using pseudo-parametrized algebraic splines. The paper then proposes a control law for the paths generated which is implemented on PX4 as an on-board autopilot.

Altough in UAV path planning plenty of spacial curves such as Bézier, B-spline, NURBS, or other parametric curves commonly are used, [6] presents an approach that follows the implementation of a clothoid curve. This mainly due to the fact that said spacial curves are intuitive for attitude planning since they are mostly very sensitive to parameter initialization (e.x. control points and weights). Furthermore the basic form of those curves does not allow to incorporate vehicle constraints and/or require optimization procedures, which mostly are not feasible for real-time applications. The work presents therefore an three-dimensional (3D) path plan-

ning approach which is based on the approximation presented in [7] which exactly matches the definition of a clothoid. In [6] this approach is then extended and adapted to be used in UAV path planning where a combination of a vertical and horizontal planar clothoid approximation is used.

Coming from a general approach for path planning [8] presents a real-time approximation of clothoids with bounded errors. Unlike the approaches presented so far here, the computational efficiency for the calculation of clothoids is reached with a set of sampled clothoid paths in a look-up table. Here only a basic clothoid is stored in a look-up table and through suitable transformations the general clothoid is then generated.

[9] presents a comprehensive approach for smooth 3D path planning for non-holomic UAVs. Therefore a control scheme is proposed consisting of a local planner and kinematic control. In the local planning part, 3D clothoids approximated by Rational Bézier curves are used and in the second stage a path following approach for the generated paths is shown. Due to the fact that for close approximations of clothoid such as the one used in [6] it is still necessary to use numerical integration for calculation, in [9] the approximation with Bézier curves is chosen due to the analytical representation of a path. Here the approach shows how to fit the control points and the weights of the Bézier curve to resemble the main proprieties of a clothoidal curve in 3D.

The work presented in this subsection shows how to generate clothoidal paths either in a 2D or 3D setup. Based on the fact that the exact computation of clothoid paths is computationally expensive due to the numerical integration involved, various approximation methods are proposed. Although none of the approaches presented here specifically incorporates wind on the path planning level.

2.1.2 Bézier Paths

As indicated in Chapter 2.1.1 are Bézier curves used as a possible approach to approximate the behavior of clothoid segments and is also used in other applications involving path planning. A Bézier curve is hereby a special case of a spline curve, in which the curve has only one polynomial component in piecewise [9]. The curves can be generated in 2D or 3D setup by defining $(n + 1)$ control points with the Bézier curve then being the interpolation between these points. For such a setup the Bézier curve can be represented by a polynomial expression of order n. The polynomial expression is then expressed in terms of Bernstein polynomials of degree n. A detailed mathematical description can be found in [9].

The main advantage of Bézier curves over clothoids is that they have a closed-form solution due to their polynomial representation and that they have scaling as well as rotation proprieties [10]. The latter property brings the ability to fit a Bézier curve to any desired behavior such as that of clothoidal curves with relatively low effort. On the other hand, the main disadvantages of Bézier curves used in path planning for UAVs is that the physical limitations of the vehicle can not be directly incorporated in the mathematical description as it is possible for example for trochoids or clothoids. Hence Bézier curves become very sensitive in tuning of the control points and their respective weighting (for Rational Bézier). Moreover this approach lacks of a basic intuition for the generated path.

Nonetheless a quick overview of path planning approaches using Bézier curves is given, mainly motivated by the simple computation and the ability to be used in

optimization approaches which may become interesting in a global path planning prospective. The most important keywords to find the literature presented in this section are:

Fixed-Wing UAV · Bézier curves · Polynomial curves · Path planning · Optimization · Continuous-curvature - C1 / C2 continuity

Various approaches using Bézier curves due to its ability to approximate clothoids have been shown section 2.1.1. Such works being [11], [9] or [6]. [12] shows a locally-adjustable, continuous-curvature, bounded path-planning algorithm for fixed-wing UAVs. For small-fixed wing UAVs curvature continuous paths are better trackable then other path types. To ensure this, a path planning framework is proposed to generate feasible paths that satisfy curvature continuity and upper bound constraints. Based on that Bézier curves built the ideal approach to plan a path in such a setup.
[13] proposes an global path planning approach. By modeling the path planning as a single objective optimization problem that utilizes a receding horizon approach, where the path is constrained to avoid obstacle collision and account for flight aerodynamics constraints. Hereby Bézier curves are used to model the UAVs path. Bézier curves were here explicitly chosen due to their simplicity, quick computation of curvature and reduced number of design variables.

[14] shows a genetic algorithm to plan paths in 3D under the constraints of minimum curvature, minimum torsion and maximum climb (or dive) angle imposed by the UAV kinematics. Four control points are used to ensure that the generated Bézier curve meets the initial and final pose in 3D. The remaining control points of the definition of the Bézier curves are free and can be used to define the optimal path between the two poses. Similarly in [15] a methodology for generating smooth feasible paths for UAV in 3D space for fixed-wing UAV is presented. To ensure the same constraints as in [14] the initial and final poses are connected with a seventh-order Bézier curves which then also indirectly insures the smoothness of the vehicle acceleration profile. For the computation of these curves an optimization problem is imposed, for which an algorithm with fast convergences to the final result is imposed.

Apart from literature for fixed-wing UAVs also in research for different types of UAVs the use of Bézier curves can be found, for example for multirotor UAVs in [10]. Furthermore also for the use with autonomous ground robots, Bézier curves have been studies such as in [16].

Bézier curves show good properties to use in optimization frameworks. None of the proposed approaches incorporates wind in the path planning procedure. The effort to extend such approaches to the use of wind conditions for path planning has not been studied in literature and hence has to be evaluated. Also the necessary degree of the Bézier curves used widely varies in path planning and has to be adjusted for the specific use.

2.2 Path Planning in 3D

Up to this point only path planning in a planar 2D setup was considered. Hence also for the wind only the horizontal component was taken into account. This is a fair assumption for use cases such as level-flight in survey missions. Although fixed-wing UAV act in a 3D environment and hence also the vertical component is of interest specifically for use cases such as emergency landings. Hence in this section

an overview of the literature on extension to 3D or general approaches of 3D path planning is given. A special focus is here given to approaches that incorporate wind and might be useful to extend the 2D approach proposed in [1]. Below a concise overview of the most important recent work in the field is given. In appendix A a detailed overview of the literature is given in tabular form to compare the different approaches used in the respective works. Due to the extensive literature review in the field of path planning in wind for fixed-wing in [1] and this work a proposal for a survey paper in this field of research was developed. The proposal showing an outline and structure of a possible survey paper can be found in appendix B.

The most important keywords to find the literature presented in this section are:

Optimal path in Wind · Fixed-Wing UAV · Clothoids · Clothoid approximation · Path planning · Wind estimation uncertainty · Trajectory Planning

For the 2D setup the goal was to find the time-optimal path for given conditions between an initial and a final pose. A common basic concept to ensure this in robotics are Dubins paths. With the combination of full-turn and straight segments, the time-optimal path can be found. This approach is widely used for constraint vehicles such as ground robots or fixed-wing UAVs. For the setup in 3D the same approach can be used which is also known as Dubins airplane and was initially studied in [17] and in [18] a detailed approach to implement this approach for fixed-wing UAVs is given. Here the path planning is divided into two 2D path planning problems which then are combined to a 3D path. The planar 2D path planning approach was already widely studied. The work gives an approach to further find a solution in the vertical dimension. Here three different cases of altitude difference between initial and final poses are established across literature. This mainly because of the fact that the time optimal path in the horizontal plane is not long enough to ensure that the UAV can reach the goal altitude in this time due to the maximal climb- / sink-rate of the UAV. For the vertical dimension the approach uses simple glider slopes. This is why for high altitude difference of the poses the use of helix was introduced. This was initially studied in [17] for constrained path planning in 3D for fixed-wing UAVs.

Other than assuming a simple climb or sink rate for the vertical dimension, in [19] also for this dimension the Dubins car behaviour was used. The final trajectory hence resembles a Dubins path in 3D. The vertical path is hereby also limited by the constraints of maximal climb rate as well as the direction of the climb (ascent / descent). The paper mainly focuses on the use of such a local path planning approach for a global planner also involving obstacle avoidance for static and dynamic objects.

Two similar approaches are presented in [20] and [21]. Also here the general idea of the Dubins airplane is used and the 3D problem is decoupled into two 2D problems. In [20] the goal is to find cost-efficient 3D paths that satisfy the maximum allowed curvature and the pitch angle of the UAV. For both problems, Dubins curves with a closed-form solution are used. To couple the two paths to a 3D path a local optimisation to find a cost-efficient solution is proposed. Furthermore the paper claims that fewer turns are involved in the final path than other approaches in the field have shown. Similar to the latter approach, in [21] another approach is presented. For the vertical behavior of the path here a simple glider slope is assumed. Furthermore this approach also presents an algorithm for path tracking which resembles the line-of-sight guidance approach. Under specified assumptions this approach guarantees that the track error, both in position and in attitude, asymptotically tends to zero. Furthermore the robustness to wind influence is tested using the Dryden

model[1].

Also [22] shows an approach to generate 3D paths based on a decoupled approach. On the horizontal plane the path is planned by a Dubins path and on the vertical plane by a Helix curve also if the initial and final pose are close. Then a smoothing algorithm is proposed to generate a 3D path which represents a shorter and smoother path. It is shown that the proposed method is able to effectively plan paths under different initial conditions and achieve real-time computation. Specifically accounting for the use case of emergency landing, [23] suggests another decoupled approach which specifically accounts for the minimum safe altitude for an emergency landing.

All the latter approaches have in common that a decoupling or a superposition approach of a certain kind is used. Several works also show different approaches coming from the idea to impose a pure 3D problem. In [24] such an approach is shown. The path planning is here based on finding a suitable 3D representation of a CSC path similar to the approach in 2D. Hence the method is based on 3D geometry and a path of minimal length can be generated faster as with the use of an iterative method. For a fairly large altitude difference where the maximal allowed pitch angle of the UAV has to be exceed, this approach fails. Here a numerical method such as multiple shooting to obtain an optimal path is suggested.

A few papers also incorporate wind disturbances in the path planning. In [25] a vector field approach is used to account for constant wind disturbances. Therefore two vector fields are developed, one is the tangent vector field based on path tangent vector and coordinate transformation, and the other is the combined vector field based on the combination of a conservative vector field and a solenoidal vector field. [26] presents a comprehensive approach for real-time path planning with online 3D wind field prediction in complex terrain. Therefore a 3D wind field prediction method is proposed which can be run in real-time on-board a UAV. The wind field is then used to generate a path based on the Dubins airplane model.

Most of the shown literature uses a decoupled path planning approach in 2D to calculate a 3D path for fixed-wing UAVs in a computationally simple way. Only a few of the approaches incorporate wind in the path planning, most of them deal with wind disturbances with a low-level controller. For those accounting for wind, only the wind in the horizontal plane is considered. This is due to the fact that this wind component is considered to have more influence on the UAV flight behavior. Furthermore the vertical wind components mostly consist of gusts instead of a constant wind field. To represent the vertical wind component computationally heavy wind field estimators have to be used. Here most of the papers rely on tackling the wind disturbance with the low-level guidance controller. As a suitable extension to the 2D horizontal path planning presented in [1] an extension with a decoupling approach as shown in [20], [21] or [22] is suggested.

2.3 Path Sampling

The literature does not show any specific approaches to sample trochidal, clothoidal or Bézier curves for the use with fixed-wing UAVs. A sampling approach is needed to sample the generated paths in a suitable way that state-of-the-art guidance controllers are able to follow such a planned path. Due to this, in this work a new

[1] The Dryden wind turbulence model is a mathematical model of continuous gusts.

sampling approach is presented which is suitable for the paths used here and allows to easily calculate the necessary parameters for path following with a guidance controller.

2.4 Guidance laws

Since the generated paths have to be followed by state-of-the-art guidance controllers, here the short literature review on this field of reasearch done in [1] is shown.

Initial guidance laws for UAVs were done with linear approaches such as PD or PID controller. [27] introduces the L_1-guidance law with the nonlinear path following law that has been adapted from the pure pursuit-based method. The main advantage of this method lies in the formulation of the point to track on the path, the error formulation and the calculation of the acceleration are more suitable for use with UAVs and allows to track the desired trajectory. Another main advantage, especially concerning the behavior in wind, is the fact that the vehicle ground speed is used for acceleration computations. The paper also proves the stability of the guidance law under different speeds and under saturated lateral acceleration as is also shown on UAV test flights in wind. Following the basis of this guidance law, more sophisticated approaches were developed. The authors of [28] show such an approach with the L^+2-guidance law. This approach mainly improves the L_1-guidance law in the following weaknesses: Lag due to the dynamic response of bank angle situation if the cross-track error is larger than look-ahead distance or modification of look-ahead distance, so the transient response is independent of the ground speed. Another key part of this method is the implementation of the roll dynamics of the UAV, which can lead to instability as shown with a root locus analysis. Moreover, to avoid an undefined heading if the intersection between the look-ahead distance and the desired path does not exist, a policy is followed to smoothly track the desired path in such a situation. Another extension for the well-tested L_1-guidance law can be found in [29]. This work focuses especially on the drawbacks that the guidance law brings for small slow-flying fixed-wing UAVs being the handling of high winds and small loiter radii.

Most of the guidance laws have the ability to track both circles and lines, which make up the vast majority of a typical fixed-wing vehicle's flight plan. This requires arbitrarily generated paths to be sampled accordingly such that the UAV is able to track them.

Chapter 3

Mathematical Basics

In this chapter the mathematical basics for this work are defined based on [1]. This includes the vehicle model, handling of the coordinate system and mathematical basics for clohtoids and trochoids in a path planning setup used throughout this work. Furthermore the optimality formulation for time-optimal path planning as well as the following path orientation combinations are introduced. The explicit mathematical formulations for the path planning cases for all types of trochoid paths (CSC/CCC) and for most of the clothoid cases (CSC) can be found in [1] and and will not be repeated here.

3.1 Vehicle Model

A simple particle model in a 2D setup is used. The model is described by the following equations:

$$\dot{x} = V_a \cos\psi(t) + V_w \tag{3.1}$$
$$\dot{y} = V_a \sin\psi(t) \tag{3.2}$$
$$\dot{\psi} = \alpha\phi(t) \tag{3.3}$$
$$\dot{\phi} = u(t) \tag{3.4}$$

Here $x(t)$ and $y(t)$ represent the inertial position of the UAV in the horizontal North-East plane as can also be seen in figure 3.1. Hereby $x(t)$ denotes the position in direction of North and respective $y(t)$ the position in direction of East. $\psi(t)$ is the heading angle of the aircraft measured from North. V_a is the constant airspeed and V_w is the constant wind speed. V_w is assumed to be aligned with the x-axis for most cases. This does not hold for cases where uncertainty of the wind is simulated, since the uncertainty also affects the heading of the wind leading to wind heading, that are not aligned with the x-axis. If this case occurs, the wind heading is expressed by ψ_ω. Generally $V_\omega < V_a$ is assumed. When this inequality is violated, some regions in the state space might get infeasible and hence feasibility for arbitrary initial and final conditions can not be ensured [30]. With the model described by equation (3.1) - equation (3.4) the turn rate of the UAV can not be changed instantaneously. This is only used for the calculation of clothoids, since trochoids assume instantaneous turns. Hence for the calculations with trochoids equation (3.4) can be replaced with:

$$\phi = u(t) \tag{3.5}$$

For clothoids in equilibrium turning flight, the heading rate of change can be expressed as a function of bank angle $\phi(t)$

$$\dot{\psi}(t) = \frac{g}{V_a} \tan \phi(t) \tag{3.6}$$

For normal UAV path planning, where the bank angle can be assumed to be small ($\phi(t) < 30°$), the relationship equation (3.6) can be approximated with the simple expression

$$\dot{\psi}(t) \approx \frac{g}{V_a} \phi(t) = \alpha \phi(t) \tag{3.7}$$

As can be seen from equation (3.4) and equation (3.5) either the rate of bank angle or the bank angle directly can be used as an input variable. Due to the aircraft dynamics, actuator limits and structural load limits on the wings, the control signal is constrained to the set

$$u \in [-\bar{u}, \bar{u}] \tag{3.8}$$

Since for time-optimal path solutions only straight segments or maximum turning segments are assumed, the set can be restricted to

$$u \in \{-\bar{u}, 0, \bar{u}\} \tag{3.9}$$

Here the slightly overloaded notation has to be highlighted again, since $u(t)$ is used as bank angle for trochoids and as rate of bank angle for clothoids.

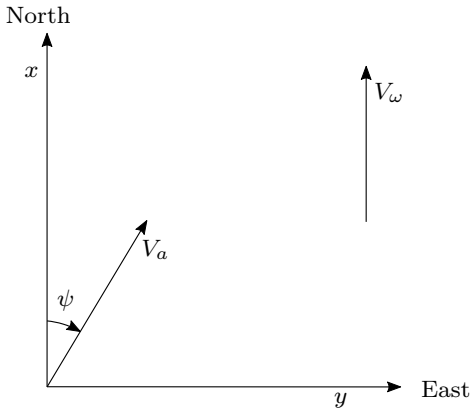


Figure 3.1: UAV particle model in 2D setup.

In this work, for the parameters of the vehicle model and environmental conditions, the following values in table 3.1 can be assumed as nominal, hence are used when no specific value is mentioned.

Parameter	Value
Airspeed V_a	$20\frac{m}{s}$
Wind Speed V_ω	$5\frac{m}{s}$
Wind Heading ψ_ω	$0°$
Max. bank angle $\bar{\phi}$	$\pm 30°$
Max. rate of bank angle $\bar{\dot{\phi}}$	$0.3\frac{rad}{s}$
Nom./Max. uncertainty in wind speed V_ω	$\pm 1\frac{m}{s}$ / $\pm 2\frac{m}{s}$
Nom./Max. uncertainty in wind heading ψ_ω	$\pm 5°$ / $\pm 10°$

Table 3.1: Nominal values for vehicle and environmental parameters.

3.2 Trochoids

Trochoids are used for path planning in wind due to a simple geometric argument. This relies on the observation that circular UAV paths in the air-relative frame correspond to trochoidal paths in the inertial frame. [31]. The trochoidal frame is aligned with the wind heading ψ_ω. Hence the trochoidal coordinates can be found by

$$\begin{pmatrix} x_t(t) \\ y_t(t) \end{pmatrix} = \begin{pmatrix} \cos\psi_\omega & \sin\psi_\omega \\ -\sin\psi_\omega & \cos\psi_\omega \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{3.10}$$

Since for the nominal case we assume $\psi_\omega = 0$ the two coordinate frames align and $x_t(t) = x(t)$ as well as $y_t(t) = y(t)$. Due to the completeness and the use for simulations of uncertainty in wind estimation, the equations are still derived with respect to the trochoidal frame. The kinematic equation can therefore be described by

$$\dot{x}_t(t) = V_a \cos(\psi(t) - \psi_\omega) + V_\omega \tag{3.11}$$

$$\dot{y}_t(t) = V_a \sin(\psi(t) - \psi_\omega) \tag{3.12}$$

$$\dot{\psi}(t) = u(t) \tag{3.13}$$

In the case of a turn at a constant maximum rate $\omega \in \{-\bar{u}, \bar{u}\}$, the equation can be stated as

$$\dot{x}_t(t) = V_a \cos(\delta\omega t + \psi_t) + V_\omega \tag{3.14}$$

$$\dot{y}_t(t) = V_a \sin(\delta\omega t + \psi_t) \tag{3.15}$$

where $\psi_t = \phi(0) - \phi_\omega$ and $\delta \in \{-1, 1\}$ account for the direction of the turn. To find a point on the trochoid path following formulation can be used

$$x_t(t) = \frac{V_a}{\delta\omega} \sin(\delta\omega t + \phi_t) + V_\omega t + x_{t_0} \tag{3.16}$$

$$y_t(t) = -\frac{V_a}{\delta\omega} \cos(\delta\omega t + \phi_t) + y_{t_0} \tag{3.17}$$

where x_{t_0} and y_{t_0} is the starting position of the segment at $t = 0$.

3.3 Clothoids

The geometrical shape which throughout this work will be referred to as clothoids are actually a shifted version of the basic definition of clothoids. The definition of so-called shifted clothoids used here can be found in equation (3.18) and equation (3.19). To be able to model a constant wind field in 2D for path planning the normal clothoid representation was extended by such a wind field. This can be seen in equation (3.18) where the term of the wind speed V_ω was introduced to model a wind field towards north.

$$x_1(t) = \int_0^t (V_a \cos(\psi_1(\tau) + V_\omega) d\tau \tag{3.18}$$

$$y_1(t) = \int_0^t (V_a \sin(\psi_1(\tau))) d\tau \tag{3.19}$$

Since for clothoids roll dynamics are considered for turns, two different cases can be obtained: The case where the maximum bank angle is reached or it is not reached for a certain segment. The equations will be derived for both cases separately to establish the suitable expressions for $\psi(t)$ in equation (3.18) and equation (3.19).

Case 1: The maximum bank angle is reached
For this case the turn rate is saturated and the UAV continues the maximum rate turn for a certain time. After that time, the turn rate is decreased until the UAV reaches straight flight again. The turn is initiated with the maximum bank angle rate

$$\psi(t) = \bar{u}t \tag{3.20}$$

The time until the maximum bank angle is reached is defined by $t_1 = \bar{\phi}/\bar{u} > 0$. Hence, the bank angle for the whole turn can be stated by

$$\phi(t) = \begin{cases} \bar{u}t & t \in [0, t_1] \\ \bar{u}t_1 & t \in [t_1, \bar{t} - t_1] \\ -\bar{u}(t - \bar{t}) & t \in [\bar{t} - t_1, \bar{t}] \end{cases} \tag{3.21}$$

From this, the heading angle follows by

$$\psi(t) = \begin{cases} \alpha\bar{u}\frac{t^2}{2} + \psi_0 & t \in [0, t_1] \\ \alpha\bar{u}t_1 t + \psi_0 + C_1 & t \in [t_1, \bar{t} - t_1] \\ \alpha\bar{u}(t\bar{t} - \frac{t^2}{2}) + \psi_0 + C_1 & t \in [\bar{t} - t_1, \bar{t}] \end{cases} \tag{3.22}$$

The constants C_1 and C_2 from equation (5.56) can be found from the continuity conditions at $t = t_1$ and $t = \bar{t} - t_1$

$$C_1 = -\alpha\bar{u}\frac{t_1^2}{2} \tag{3.23}$$

$$C_2 = -\alpha\frac{\bar{u}}{2}(2t_1^2 + \bar{t}^2 - 2\bar{t}t_1) \tag{3.24}$$

This leads to the following heading angle at the end of the turn, hence when $t = \bar{t}$

$$\psi(\bar{t}) = -\alpha\bar{u}t_1^2 + \psi_0 + \alpha\bar{u}t_1\bar{t} \tag{3.25}$$

Case 2: The maximum bank angle is not reached
For this case, the UAV will start to come back to straight level flight before the maximum bank angle is reached. The bank angle for the entire turn is defined as

$$\phi(t) = \begin{cases} \bar{u}t & t \in [0, \bar{t}/2] \\ -\bar{u}(t - \bar{t}) & t \in [\bar{t}/2, \bar{t}] \end{cases} \tag{3.26}$$

From this, the heading angle follows by

$$\psi(t) = \begin{cases} \alpha\bar{u}\frac{t^2}{2} + \psi_0 & t \in [0, \bar{t}/2] \\ \alpha\bar{u}(t\bar{t} - \frac{t^2}{2}) + \psi_0 + C & t \in [\bar{t}/2, \bar{t}] \end{cases} \tag{3.27}$$

In addition for this case, the constant C can be found from the continuity condition at $t = \bar{t}/2$

$$C = -\alpha\bar{u}\frac{\bar{t}^2}{4} \tag{3.28}$$

In figure 3.2 for *Case 1* and *Case 2* an example trajectory is shown.

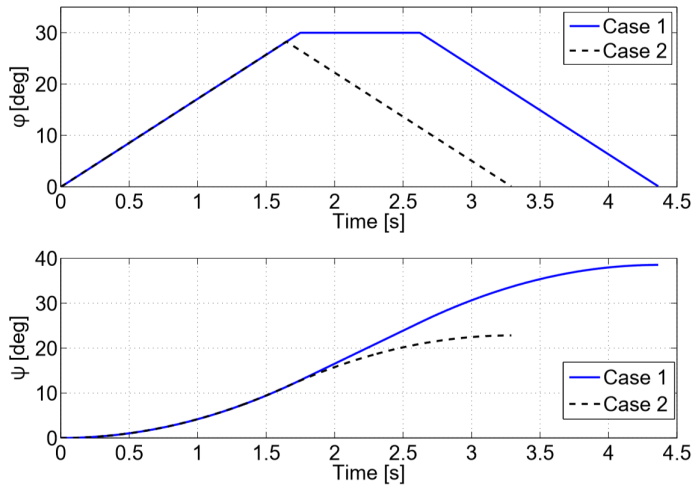


Figure 3.2: Bank angle $\phi(t)$ and heading angle $\psi(t)$ for *Case 1* and *Case 2* of clothoid turns [32].

3.4 Path Planning Setup

In section 3.1 the properties of the UAV model are explained and in section 3.2 - section 3.3 the basics of two different geometric shapes used for path planning in wind are derived. Based on this, a setup for a point-to-point path planning used in this work is established. Point-to-point means that each initial and final pose in the planar space is defined. The pose consists of the position as well as the heading in the following way

$$x(0) = x_0, \quad y(0) = y_0, \quad \psi(0) = \psi_0 \tag{3.29}$$

$$x(T) = x_f, \quad y(T) = y_f, \quad \psi(T) = \psi_f \tag{3.30}$$

with T being the entire flight time of the trajectory. For use cases, such as surveys and many others, the goal is to find the time-optimal path between such two poses.

Hence the goal is to find a control input $u^*(t)$ such that the UAV arrives at the desired final pose in minimal time. To find such a control input, the goal is to minimize the cost function

$$J = \int_0^T dt = T \tag{3.31}$$

for the kinematic equations equation (3.1) - equation (3.4) of the respective model and the symmetric control limits

$$-\bar{u} \leq u(t) \leq \bar{u} \tag{3.32}$$

Finding minimum-length paths for paths with bounded curvature was studied intensively by Dubins [33] and was later extended to find minimum-time paths for constant-speed mobile robots with bounded turn rates, creating the expression *Dubins's car* [34]. The extension to minimum-time path planning for UAVs in this work relies on the same principle.

The necessary conditions for the time optimality are here derived for trochoids, but hold similarly for the setup with clothoids. Assuming alignment with the trochoidal frame, following [35] we can state the Hamiltonian for the time-optimal control problem

$$\mathcal{H} = 1 + \lambda_1(V_a \cos(\psi(t) + V_\omega) + \lambda_2 V_a \sin\psi(t) + \lambda_3 u \tag{3.33}$$

where $u \in \mathcal{U}$ and $\mathcal{U} = [-\bar{u}, \bar{u}]$ are the controls at hand. For the co-states holds

$$\dot{\lambda}_1 = 0, \quad \dot{\lambda}_2 = 0, \quad \dot{\lambda}_3 = \lambda_1 V_a \sin\psi(t) - \lambda_2 V_a \cos\psi(t) \tag{3.34}$$

According to the minimum principle along an optimal trajectory, the following has to hold

$$\mathcal{H}(x^*, u^*, \lambda^*, t) \leq \mathcal{H}(x^*, u, \lambda^*, t) \tag{3.35}$$

for all $u \in \mathcal{U}$ and $0 \leq t \leq T$. Since λ_1 and λ_2 are constant, the case for $\lambda_3 \neq 0$ and $\lambda_3 = 0$ can be analyzed. First $\lambda_3 \neq 0$ is assumed. To satisfy Pontryagin's minimum principle, one needs $u = -\text{sign}(\lambda_3)\bar{u}$ a maximum effort, which is a maximum-rate turn to the left or right. Secondly, we consider $\lambda_3 = 0$. Since λ_1 and λ_2 are constant, the vector $\lambda = [\lambda_1, \lambda_2, \lambda_3]^T$ is a constant vector. As implied by the Pontryagin's minimum principle λ has to be nonzero. Therefore, by the third equation it can be implied that

$$\lambda || \begin{pmatrix} \cos\psi(t) \\ \sin\psi(t) \\ 0 \end{pmatrix} \tag{3.36}$$

This leads to the fact that $\psi(t)$ has to be constant, which only holds for straight paths. From this derivation it can be assumed that the time-optimal path only consists of turns at maximum rate and straight segments. Hence, for the setup used in this work, the paths consist of straight paths and trochoid or clothoid turns respectively. If zero wind $V_\omega = 0$ is assumed, the turn segments simplify to circular segments, which leads to the classical Dubins assumption.

The combination of segments leading to the time-optimal can be chosen arbitrarily, hence the number of segments to combine can be increased arbitrarily. Due to the aspect that the re-optimization of the paths should be computed with small computational effort, we restrict the number in this work to 3 consecutive segments. The optimality of such a setup is proven in [35] and [32]. To handle the different path

combinations and orientations, a notation for this work is introduced. The general types of path segments are noted by:

C - Curved segment
S - Straight segment

Hence, for 3 segments there exist CSC- and CCC-paths (due to the fact that two consecutive straight segments do not lead to a usable path type). Adding the directionality of the turn segments to state specific path combinations, the following notation is used:

L - Maximum-rate turn to the left
R - Maximum-rate turn to the right
S - Straight segment

Therefore, 6 possible combinations can be found:

CSC - LSL, RSR, LSR, RSL
CCC - LRL, RLR

The orientation of a left (L) and right (R) turn is defined with respect to the setup in figure 3.1.
Not all path types exist for all possible combinations of initial and final poses [35]. To find the time-optimal path for a certain setup, the total flight time T for all feasible path types has to be compared. In figure 3.3 a setup is shown, where all six introduced path types exist. The setup shows zero wind conditions $V_\omega = 0$ and hence has circular turn segments as assumed in the classical Dubins case. The Figure is only used to show the orientation of the different path type combinations.

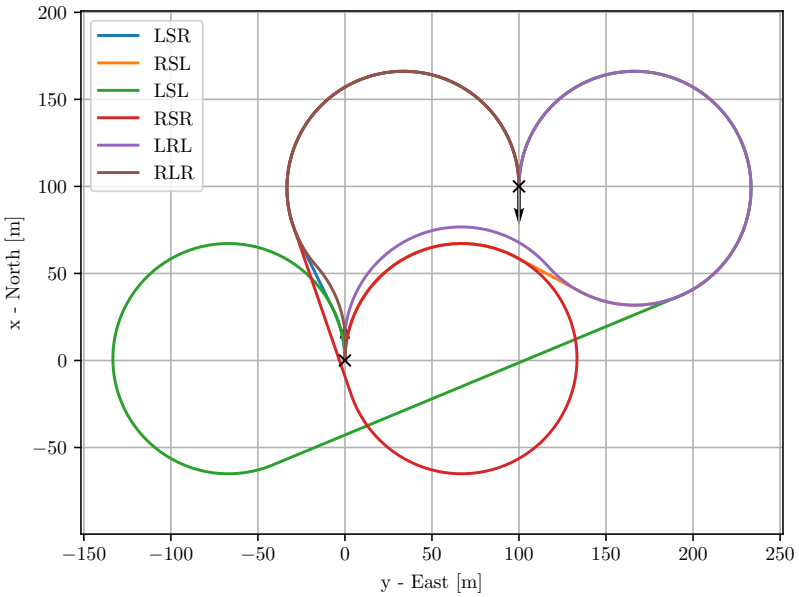


Figure 3.3: Path types for zero wind conditions $V_\omega = 0$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 100m$, $y_f = 100m$ and $\psi_f = 180°$.

Chapter 4

Path Planning

Based on the mathematical setup given in chapter 3 here the path planning setup is shown. Hereby the goal is to plan the time-optimal path based on the mission plan as well as the current wind estimation as shown in figure 4.1 in form of a flowchart. The mission plan is defined as the initial and final pose given in the 2D setup. The current wind direction is given by the magnitude V_ω and the heading ψ_ω. These are the parameters which change for a certain path planning scenario. Fixed parameters such as kinematic limitations of the UAV are considered for all path planning cases.

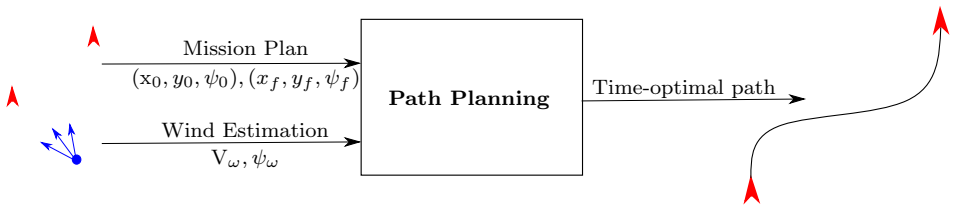


Figure 4.1: Flowchart of the path planning setup to generate time-optimal paths based on the mission plan and current wind estimations.

There are various ways to find a time-optimal path in a 2D setup. [1] presents a three-segment approach based on the principle of Dubins paths with the use of trochoid and clothoid geometry was implemented. Hence the path planning is used to find a suitable segment orientation and path length to generate the time-optimal path. Therefore the candidates for the time-optimal path consist of six possible combinations of orientations:

$$\text{Path Types} = \{\text{RSR, RSL, LSR, LSL, RLR, LRL}\}$$

To be able to calculate all possible paths between the initial and final pose the framework implemented in [1] is used. In figure 4.2 an overview of the framework can be seen and in appendix C a short overview of the software structure of the framework is given. The gray part, resembling CCC clothoid paths, has not been implemented so far. In this chapter this last approach for the framework is derived and implemented. Further documentation on all other derivations of the framework can be found in [1]. Once the whole framework is implemented the path planning is tested for different conditions and its performance is analyzed.

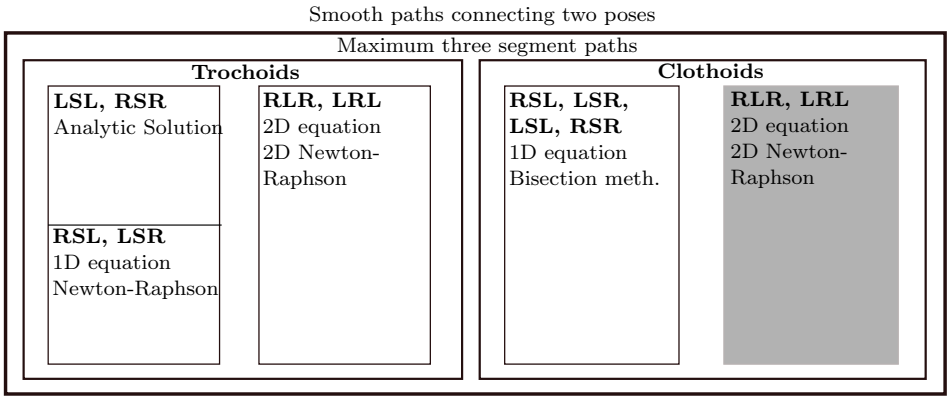


Figure 4.2: Overview of the path planning framework using trochoids and clothoids to find time-optimal three-segment approaches.

4.1 Path Type CCC Clothoid

The CCC case for clothoid paths is the mathematically most demanding approach in the framework and the reason the approach was not derived and implemented so far. During this section this will be done in detail. The main challenge in deriving and implementing the CCC form for clothoid paths comes from two points, one being the fact that the position on a clothoid path is defined by Fresnel Integrals which have no closed-form solution. The second point is unlike for trochoids where only one type of turning segment exists (assuming maximal bank angle from start) for trochoids always the two cases for reaching and not reaching the maximal bank angle have to be assumed (see section 3.3). This leads to the fact that for the combination of three segments all possible combinations have to be checked for a time-optimal path. The combination of these two facts imposes a challenge which is covered in the following subsections.

4.1.1 Mathematical definition and transition conditions

The basic setup of path planning with a three-segment approach was established in Section 3.4 and is in detail derived in [1]. Due to it extensive use throughout this section, the definition for the three curved clothoid segments (C) and the transition conditions between the single curves for the path planning setup are repeated.
Based on the clothoid definition shown in Section 2.1.1 the statement for the three segments can be made. Since the segments are consecutive, the starting point of every segment is expressed with x_{i0}, y_{i0}. Therefore for the first clothoid ($t \in [0, t_a]$) holds:

$$x_1(t) = \int_0^t (V_a \cos(\psi_1(\tau)) + V_\omega) d\tau + x_{10} \tag{4.1}$$

$$y_1(t) = \int_0^t (V_a \sin(\psi_1(\tau))) d\tau + y_{10} \tag{4.2}$$

For the second clothoid ($t \in [0, t_b]$) holds:

$$x_2(t) = \int_0^t (V_a \cos(\psi_2(\tau)) + V_\omega) d\tau + x_{20} \tag{4.3}$$

$$y_2(t) = \int_0^t (V_a \sin(\psi_2(\tau))) d\tau + y_{20} \tag{4.4}$$

For the third clothoid ($t \in [0, T]$) holds:

$$x_3(t) = \int_0^t (V_a \cos(\psi_3(\tau)) + V_\omega) d\tau + x_{30} \tag{4.5}$$

$$y_3(t) = \int_0^t (V_a \sin(\psi_3(\tau))) d\tau + y_{30} \tag{4.6}$$

Since only the CCC case (RLR, LRL) is of interest, the directions of the turns are defined as follows:

$$\delta_1 = \delta_3 \tag{4.7}$$

$$\delta_2 = -\delta_1 \tag{4.8}$$

The linear turning behaviour is based on $\psi_i(t)$. The function increases the bank angle linearly until the maximum is reached and then decreases the bank angle again to end up with zero bank angle at the end of each segment. For this setup two behaviours can be distinguished: The case where the maximum bank angle is reached and the case where not (see case 1 and case 2 in section 2.1.1).

To make sure that a consecutive path with three clothoids segments can be planned, certain transition conditions have to be implied. The first conditions can be taken from the initial conditions. Since point-to-point coordination is of interest, the initial and final poses have to be reached:

$$[x_1(t), y_1(t), \psi_1(t)]^T|_{t=0} = [x_0, y_0, \psi_0]^T \tag{4.9}$$

$$[x_3(t), y_3(t), \psi_3(t)]^T|_{t=T} = [x_f, y_f, \psi_f]^T \tag{4.10}$$

To ensure continuity between the single clothoid segments, the continuity conditions for the poses are taken into account:

$$x_1(t_A) = x_2(0)\ ,\ y_1(t_A) = y_2(0)\ ,\ \psi_1(t_A) = \psi_2(0) \tag{4.11}$$

$$x_2(t_B) = x_3(0)\ ,\ y_2(t_B) = y_3(0)\ ,\ \psi_2(t_B) = \psi_3(0) \tag{4.12}$$

The transition conditions on the heading angle $\psi_i(t)$ in (4.11) and (4.12) imply that no discontinuities occur at the transition point. This implies that for the transition points holds that:

$$\frac{d}{dt}\psi(t) = 0 \tag{4.13}$$

Additionally the continuity conditions for the velocity at the transition points could be stated, but these can be reduced to the continuity of the heading $\psi_i(t)$ stated in the equations above.

4.1.2 Derivation

The derivations shown here hold for the case that all three curved segments (C) reach the maximum bank angle (see case 1 in Section 3.3). The derivation holds similarly for all other combinations of clothoid segment types. The necessary adaptions for the other combinations are shown in the Section 4.1.3 and also are explicitly derived in Appendix D. For the other variation case the heading angle $\psi_i(t)$ of the respective segment i has to be changed if the maximum bank angle is not reached. For these cases the heading angle is described by (5.58) and hence also the transition conditions for the heading angle (4.11) and (4.12) have to be adjusted.

Using (4.9) the equations for the first clothoid segment (5.54) and (4.2) can be restated as:

$$x_1(t) = \int_0^t (V_a \cos(\psi_1(\tau)) + V_\omega) d\tau + x_0 \tag{4.14}$$

$$y_1(t) = \int_0^t (V_a \sin(\psi_1(\tau))) d\tau + y_0 \tag{4.15}$$

Moreover the initial starting heading of $\psi_1(\tau)$ can be set to the initial heading:

$$\psi_1(0) = \psi_0 \tag{4.16}$$

Using (4.10) the equations for the third clothoid segment (4.5) and (4.6) can be restated as:

$$x_{30} = x_f - \int_0^T (V_a \cos(\psi_3(\tau)) + V_\omega) d\tau \tag{4.17}$$

$$y_{30} = y_f - \int_0^T (V_a \sin(\psi_3(\tau))) d\tau \tag{4.18}$$

Moreover the heading of the third clothoid $\psi_3(\tau)$ can be set to the final heading at $t = T$:

$$\psi_3(T) = \psi_f \tag{4.19}$$

At this point the specific formulations for $\psi_i(t)$ have to be used for the derivation. Stating the explicit solution for $\psi_3(T)$ (4.19) can be reformulated as:

$$\psi_f = \psi_3(T) = -\alpha\delta_3\bar{u}t_1^2 + \psi_{30} + \alpha\delta_3\bar{u}t_1T \tag{4.20}$$

The heading continuity conditions from (4.11) and (4.12) can be defined as[1]:

$$\psi_1(t_A) = \psi_2(0) = \psi_{20} \tag{4.21}$$

$$\psi_2(t_B) = \psi_3(0) = \psi_{30} \tag{4.22}$$

From (4.16) and (4.21) follows:

$$\psi_{20} = -\alpha\delta_1\bar{u}t_1^2 + \psi_0 + \alpha\delta_1\bar{u}t_1t_A \tag{4.23}$$

Similarly follows from (4.19) and (4.22):

$$\psi_{30} = -\alpha\delta_2\bar{u}t_1^2 + \psi_{20} + \alpha\delta_2\bar{u}t_1t_B \tag{4.24}$$

From (4.23) and (4.24) it follows:

$$\psi_{30} = -\alpha\delta_2\bar{u}t_1^2 + -\alpha\delta_1\bar{u}t_1^2 + \psi_0 + \alpha\delta_1\bar{u}t_1t_A + \alpha\delta_2\bar{u}t_1t_B \tag{4.25}$$

Since for CCC path types (4.8) holds, (4.25) can be simplified to:

$$\psi_{30} = \psi_0 + \alpha\delta_1\bar{u}t_1t_A + \alpha\delta_2\bar{u}t_1t_B \tag{4.26}$$

Using (4.20) the equation can be simplified in such a way that it only depends on the unknowns t_A, t_B and T:

$$\psi_f + \alpha\delta_3\bar{u}t_1^2 - \alpha\delta_3\bar{u}t_1T = \psi_0 + \alpha\delta_1\bar{u}t_1t_A + \alpha\delta_2\bar{u}t_1t_B \tag{4.27}$$

[1] For the CSC cases here the heading continuity conditions are stated as for example as $\psi_1(t_A) = \psi_2(0) + 2k\pi, \quad k \in \mathbb{Z}$. For CCC cases here the variation with k is omitted as in [35].

Rearranging (4.27) to express $t_B = f(t_A, T)$:

$$t_B = \frac{\psi_f - \psi_0}{\alpha\delta_2\bar{u}t_1} + \frac{\delta_3}{\delta_2}t_1 - \frac{\delta_3}{\delta_2}T - \frac{\delta_1}{\delta_2}t_A \tag{4.28}$$

Using (4.7) and (4.8) the equation simplifies to:

$$t_B = \frac{\psi_f - \psi_0}{\alpha\delta_2\bar{u}t_1} - t_1 + T + t_A \tag{4.29}$$

Now the positional information for x and y of the continuity conditions in (4.11) and (4.12) are used to state two additional equations depending on the unknown parameters t_A, t_B and T. From the stated equations for the two transition points, the following equations can be stated:

$$\begin{pmatrix} x_{20} \\ y_{20} \end{pmatrix} = \begin{pmatrix} \int_0^{t_A}(V_a\cos(\psi_1(\tau)) + V_\omega)d\tau + x_0 \\ \int_0^{t_A}(V_a\sin(\psi_1(\tau)))d\tau + y_0 \end{pmatrix} \tag{4.30}$$

$$\begin{pmatrix} x_{30} \\ y_{30} \end{pmatrix} = \begin{pmatrix} \int_0^{t_B}(V_a\cos(\psi_2(\tau)) + V_\omega)d\tau + x_{20} \\ \int_0^{t_B}(V_a\sin(\psi_2(\tau)))d\tau + y_{20} \end{pmatrix} \tag{4.31}$$

Replacing x_{30} and y_{30} in the equation above by the expressions in (4.17) and (4.18) the equation (4.31) becomes:

$$\begin{pmatrix} x_f - \int_0^{T}(V_a\cos(\psi_3(\tau)) + V_\omega)d\tau \\ y_f - \int_0^{T}(V_a\sin(\psi_3(\tau)))d\tau \end{pmatrix} = \begin{pmatrix} \int_0^{t_B}(V_a\cos(\psi_2(\tau)) + V_\omega)d\tau + x_{20} \\ \int_0^{t_B}(V_a\sin(\psi_2(\tau)))d\tau + y_{20} \end{pmatrix} \tag{4.32}$$

Using the expressions for x_{20} and y_{20} from (4.30) and (4.32) two transcendental equations for the unknown can be obtained:

$$\mathbf{f}(t_A, t_B, T) = \begin{pmatrix} f_1(t_A, t_B, T) \\ f_2(t_A, t_B, T) \end{pmatrix} = \begin{pmatrix} -\int_0^{t_A}(V_a\cos(\psi_1(\tau)) + V_\omega)d\tau - x_0 + x_f - \int_0^{T}(V_a\cos(\psi_3(\tau)) + V_\omega)d\tau - \int_0^{t_B}(V_a\cos(\psi_2(\tau)) + V_\omega)d\tau \\ -\int_0^{t_A}(V_a\sin(\psi_1(\tau)))d\tau - y_0 + y_f - \int_0^{T}(V_a\sin(\psi_3(\tau)))d\tau - \int_0^{t_B}(V_a\sin(\psi_2(\tau)))d\tau \end{pmatrix} = 0 \quad (4.33)$$

Using (4.29) to replace the unknown t_B the equations reduces to two transcendental equations $\mathbf{f}(t_A, T)$ with the two unknowns t_A and T in the following manner (Due to readability not the whole equation $\mathbf{f}(t_A, T) = 0$ is written out here, but an example for $f_1(t_A, T)$):

$$f_1(t_A, T) = ... \int_0^{t_B} ... = ... \int_0^{\frac{\psi_f - \psi_0}{\alpha\delta_2 \bar{u} t_1} - t_1 + T + t_A} ... \quad (4.34)$$

With a suitable root-finding algorithm the roots of the equation can be found. If the second-order Newton-Raphson method is chosen, the following mapping can be defined:

$$g(t_A, T) = \begin{pmatrix} t_A \\ T \end{pmatrix} - \mathbf{J}^{-1}(t_A, T)\mathbf{f}(t_A, T) \quad (4.35)$$

where

$$\mathbf{J}(t_A, T) = \begin{bmatrix} \frac{\partial f_1}{\partial t_a}(t_A, T) & \frac{\partial f_1}{\partial T}(t_A, T) \\ \frac{\partial f_2}{\partial t_A}(t_A, T) & \frac{\partial f_2}{\partial T}(t_A, T) \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (4.36)$$

is the Jacobian matrix. Also here holds; if the initial guess $(t_{a_0}, T_0)^T$ is close enough to the true solution, then the mapping defined by

$$\begin{pmatrix} t_{a_{i+1}} \\ T_{i+1} \end{pmatrix} = \mathbf{g}(t_{A_i}, T_i) \quad (4.37)$$

converges to a root. Important here: If the Jacobian becomes singular, the root-finding algorithm may be started with a different initial guess.

The entries in the Jacobian $a - d$ can be built in the following way[2]:

$$
\begin{aligned}
a = \frac{\partial f_1}{\partial t_a}(t_A, T) &= \frac{\partial}{\partial t_A}(-\int_0^{t_A}(V_a\cos(\psi_1(\tau)) + V_\omega)d\tau - x_0 + x_f - \int_0^{T}(V_a\cos(\psi_3(\tau)) + V_\omega)d\tau - \int_0^{t_B}(V_a\cos(\psi_2(\tau)) + V_\omega)d\tau) \\
&= -V_a\cos\left(\psi_1(t_A, t_A, T)\right) - V_a\cos\left(\psi_2\left(T - t_1 + t_A - \frac{\psi_0}{\alpha\delta_2 t_1\bar{u}} + \frac{\psi_f}{\alpha\delta_2 t_1\bar{u}}, t_A, T\right)\right) + V_a\int_0^{t_A}\sin\left(\psi_1(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_1(t, t_A, T)\,dt \\
&\quad + V_a\int_0^{T-t_1+t_A-\frac{\psi_0}{\alpha\delta_2 t_1\bar{u}}+\frac{\psi_f}{\alpha\delta_2 t_1\bar{u}}}\sin\left(\psi_2(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_2(t, t_A, T)\,dt + V_a\int_0^{T}\sin\left(\psi_3(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_3(t, t_A, T)\,dt - 2V_\omega
\end{aligned}
\tag{4.38}
$$

$$
\begin{aligned}
b = \frac{\partial f_1}{\partial T}(t_A, T) &= \frac{\partial}{\partial T}(-\int_0^{t_A}(V_a\cos(\psi_1(\tau)) + V_\omega)d\tau - x_0 + x_f - \int_0^{T}(V_a\cos(\psi_3(\tau)) + V_\omega)d\tau - \int_0^{t_B}(V_a\cos(\psi_2(\tau)) + V_\omega)d\tau) \\
&= -V_a\cos\left(\psi_2\left(T - t_1 + t_A - \frac{\psi_0}{\alpha\delta_2 t_1\bar{u}} + \frac{\psi_f}{\alpha\delta_2 t_1\bar{u}}, t_A, T\right)\right) - V_a\cos\left(\psi_3(T, t_A, T)\right) + V_a\int_0^{t_A}\sin\left(\psi_1(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_1(t, t_A, T)\,dt + \\
&\quad V_a\int_0^{T-t_1+t_A-\frac{\psi_0}{\alpha\delta_2 t_1\bar{u}}+\frac{\psi_f}{\alpha\delta_2 t_1\bar{u}}}\sin\left(\psi_2(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_2(t, t_A, T)\,dt + V_a\int_0^{T}\sin\left(\psi_3(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_3(t, t_A, T)\,dt - 2V_\omega
\end{aligned}
\tag{4.39}
$$

[2]Here the following two derivation rules are used: $\frac{\partial}{\partial t}(\int_0^t f(x)dx) = f(t)$, $\frac{\partial}{\partial t}(\int_0^{g(t)} f(x)dx) = f(g(t))g'(t)$ and $\frac{d}{dx}(\int_{a(x)}^{b(x)} dt) = f(x, b(x))\frac{d}{dx}b(x) - f(x, a(x))\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{\partial}{\partial x}f(x, t)dt$ (Leibniz integral rule)

$$c = \frac{\partial f_2}{\partial t_A}(t_A, T) = \frac{\partial}{\partial t_A}(\int_0^{t_A}(V_a \sin(\psi_1(\tau)))d\tau + y_0 + y_f - \int_0^{T}(V_a \sin(\psi_3(\tau)))d\tau - \int_0^{t_B}(V_a \sin(\psi_2(\tau)))d\tau)$$

$$= -V_a \left(\sin\left(\psi_1(t_A, t_A, T)\right) + \sin\left(\psi_2\left(\frac{\alpha\delta_2 t_1 \bar{u}\left(T - t_1 + t_A\right) - \psi_0 + \psi_f}{\alpha\delta_2 t_1 \bar{u}}, t_A, T\right)\right) + \int_0^{t_A} \cos\left(\psi_1(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_1(t, t_A, T)\, dt + \right.$$

$$\left. \int_0^{\frac{\alpha\delta_2 t_1 \bar{u}\left(T - t_1 + t_A\right) - \psi_0 + \psi_f}{\alpha\delta_2 t_1 \bar{u}}} \cos\left(\psi_2(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_2(t, t_A, T)\, dt + \int_0^{T} \cos\left(\psi_3(t, t_A, T)\right)\frac{\partial}{\partial t_A}\psi_3(t, t_A, T)\, dt \right. \quad (4.40)$$

$$d = \frac{\partial f_2}{\partial T}(t_A, T) = \frac{\partial}{\partial T}(\int_0^{t_A}(V_a \sin(\psi_1(\tau)))d\tau + y_0 + y_f - \int_0^{T}(V_a \sin(\psi_3(\tau)))d\tau - \int_0^{t_B}(V_a \sin(\psi_2(\tau)))d\tau)$$

$$= -V_a \left(\sin\left(\psi_2\left(\frac{\alpha\delta_2 t_1 \bar{u}\left(T - t_1 + t_A\right) - \psi_0 + \psi_f}{\alpha\delta_2 t_1 \bar{u}}, t_A, T\right)\right) + \sin\left(\psi_3(T, t_A, T)\right) + \int_0^{t_A} \cos\left(\psi_1(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_1(t, t_A, T)\, dt + \right.$$

$$\left. \int_0^{\frac{\alpha\delta_2 t_1 \bar{u}\left(T - t_1 + t_A\right) - \psi_0 + \psi_f}{\alpha\delta_2 t_1 \bar{u}}} \cos\left(\psi_2(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_2(t, t_A, T)\, dt + \int_0^{T} \cos\left(\psi_3(t, t_A, T)\right)\frac{\partial}{\partial T}\psi_3(t, t_A, T)\, dt \right. \quad (4.41)$$

For (4.38) - (4.41) have the following common derivative expressions w.r.t. t_A as well as T:

$$\frac{\partial}{\partial t_A}\psi_1(t, t_A, T)\, dt = 0 \tag{4.42}$$

$$\frac{\partial}{\partial t_A}\psi_2(t, t_A, T)\, dt = \alpha\delta_1 t_1 \bar{u} \tag{4.43}$$

$$\frac{\partial}{\partial t_A}\psi_3(t, t_A, T)\, dt = \alpha\delta_1 t_1 \bar{u} + \alpha\delta_2 t_1 \bar{u} \tag{4.44}$$

$$\frac{\partial}{\partial T}\psi_1(t, t_A, T)\, dt = 0 \tag{4.45}$$

$$\frac{\partial}{\partial T}\psi_2(t, t_A, T)\, dt = 0 \tag{4.46}$$

$$\frac{\partial}{\partial T}\psi_3(t, t_A, T)\, dt = \alpha\delta_2 t_1 \bar{u} \tag{4.47}$$

4.1.3 Implementation

When implementing root-fining methods in 2D, the bisection method gets significantly more complex than in 1D. Thus for the use of 2D root-finding the second-order Newton-Raphson method is preferred in most cases. One of the main differences between the algorithms is that for the Newton-Raphson approach the first derivative of the transcendental equations is needed. Therefore the implementation of the Jacobian $J(t_A, T)$ as stated in (4.36). The implementation of the inverse of the Jacobian $J^{-1}(t_A, T)$ for the necessary calculation in (4.35) can be implemented as following:

$$J^{-1}(t_A, T) = \frac{1}{\det(J(t_A, T))} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{4.48}$$

This leads to an implementation of (4.35) in the following way:

$$\begin{bmatrix} t_{Anew} \\ T_{new} \end{bmatrix} = \begin{bmatrix} t_{Aold} \\ T_{old} \end{bmatrix} - \begin{bmatrix} \frac{d}{\det(J(t_A,T))} f_1(t_A, T) - \frac{c}{(\det(J(t_A,T))} f_2(t_A, T) \\ -\frac{b}{\det(J(t_A,T))} f_1(t_A, T) + \frac{a}{(\det(J(t_A,T))} f_2(t_A, T) \end{bmatrix} \tag{4.49}$$

Through the iterative process of root-finding with the second-order Newton-Raphson method, the iteration is termined by a convergence criteria with e_{min} being the convergence tolerance:

$$(t_{Anew} - t_{Aold})^2 + (T_{new} - T_{old})^2 < e_{min} \tag{4.50}$$

A specific value for e_{min} is assumed to be defined around:

$$e_{min} = 0.001 \tag{4.51}$$

If no convergence to this value can be seen, after N steps the iteration terminates with the statement that no root was found ($N = 50$ for the examples in the rest of this section).

The section above explains the general implementation procedure of the 2D root-finding problem used here. As explained in chapter 4 one of the main challenges imposed by this problem is that there are two different cases for the clothoid segments used here: The case where the maximum bank angle is reached (R) and the case where it is not reached (NR). To find the time-optimal path all the different variations for the segments have to be checked. This leads to $2^3 = 8$ combinations for a three segment path planning approach as can be seen in table 4.1.

1. Segment	**2. Segment**	**3. Segment**
R	R	R
NR	R	R
R	NR	R
NR	NR	R
R	R	NR
NR	R	NR
R	NR	NR
NR	NR	NR

Table 4.1: Possible combinations for the different clothoid segments

For the different variations also the specific transition conditions shown in Section 4.1.1 change. To be able to implement all the different variations in Appendix D for all combinations the specific adaptions for implementation are shown.

4.1.4 Convergence Analysis

For the path planning approach for CCC clothoid paths, the use of the Jacobian for the Newton-Rapshon approach is a key part of the mathematical model. To verify the entries (4.38) - (4.41) the implementation of the explicit solution is compared to the numerical solution for each entry. In figure 4.3 and figure 4.4 the results for the entries a and c are shown. Here the value for the parameter T is fixed, since the derivatives are taken with respect to t_A. Respectively figure 4.5 and figure 4.6 shows the results for b and d for a fixed value for the parameter t_A. Apart from small spikes in the numerical solutions, coming from numerical inaccuracies in the calculated values for f_1 and f_2, all explicit solutions show a convergence to the numerical solutions. This proofs the correct derivation of the entries in section 4.1.2.

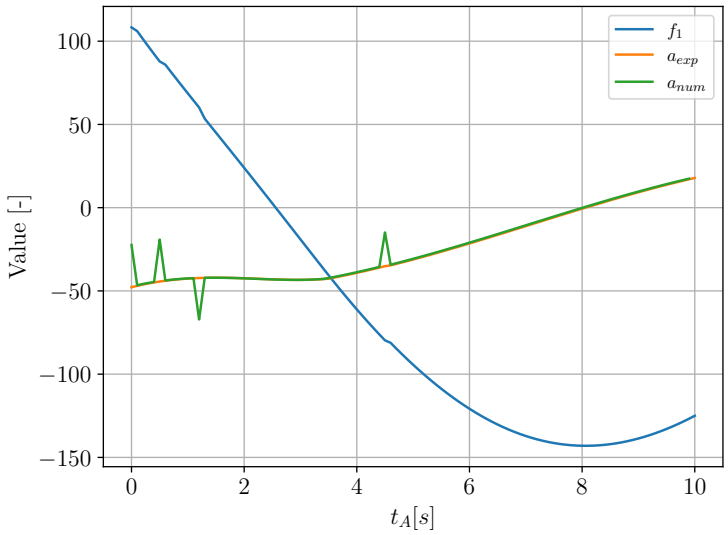


Figure 4.3: Result for Jacobian entry a for the setup $t_A = [0, 10s]$ and $T = 4s$ for $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 300m$, $y_f = 300m$ and $\psi_f = 0°$

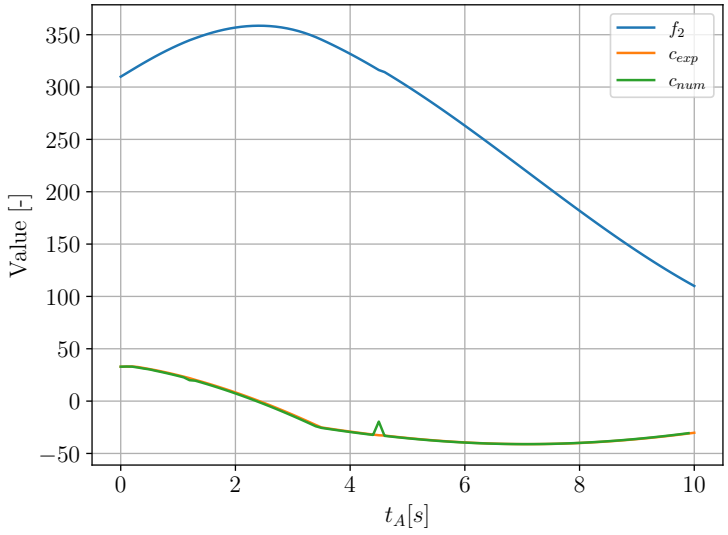


Figure 4.4: Results for Jacobian entry c for the same setup as Figure 4.3

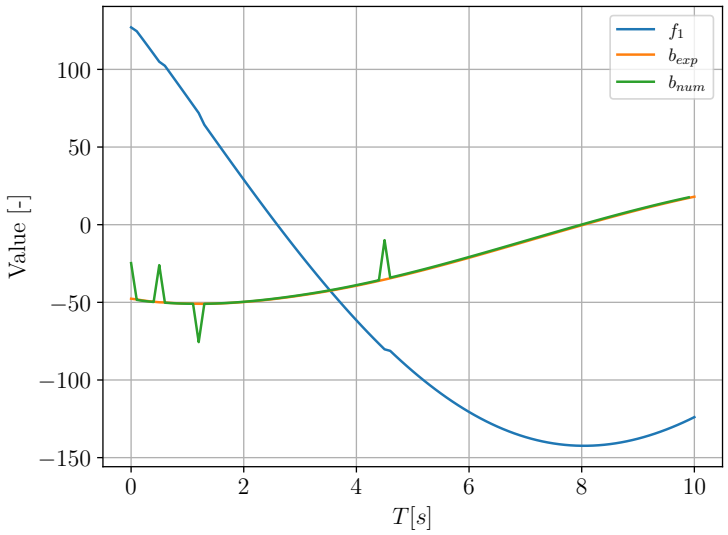


Figure 4.5: Results for Jacobian entry b for the setup $T = [0, 10s]$ and $t_A = 4s$ for $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 300m$, $y_f = 300m$ and $\psi_f = 0°$

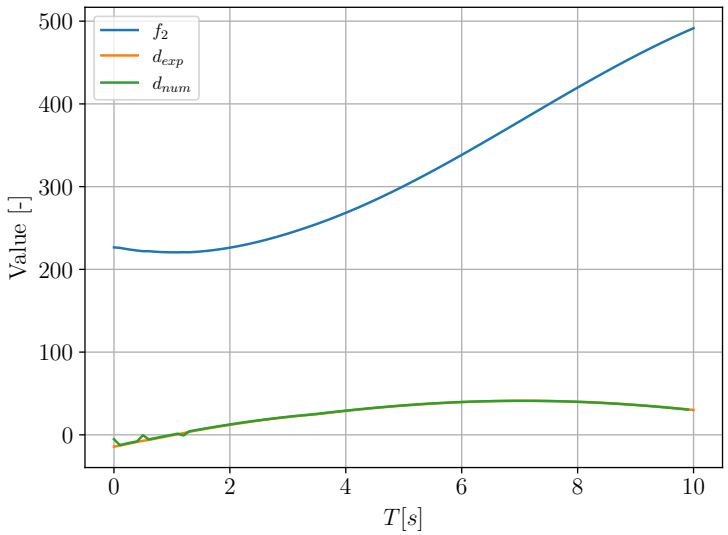


Figure 4.6: Results for Jacobian entry d for the same setup as Figure 4.5

As described in section 4.1.3 for the solving process of the point-to-point path planning problem a 2D Newton-Rapshon Solver is used. To test the implementation a reference path for a point-to-point setup is defined:

$$\begin{aligned} x_0 &= 0m \\ y_0 &= 0m \\ \psi_0 &= 0° \\ x_f &= 381.542m \\ y_f &= 233.744m \\ \psi_f &= 0° \\ V_a &= 20m/s \\ V_\omega &= 5m/s \\ t_A &= 12s \\ T &= 5s \end{aligned}$$

To define the reference path shown above, a path with the defined values for the unknown parameters of the solver t_A and T was built. The resulting final pose (x_f, y_f and ψ_f) is taken as the final pose for the solving process. Hence the solver should converge to the values $t_A = 12s$ and $T = 5s$ which build the ground truth.

To test the behaviour of the solver, the parameter z in the update step (4.49) is introduced as shown in (4.52). In the standard form of the Newton-Raphson solver $z = 1.0$ holds. If z is decreased, the update step is decreased, preventing the value from overshooting. On the other hand a decrease also might increase the number of steps to converge to the correct solution.

$$\begin{bmatrix} t_{Anew} \\ T_{new} \end{bmatrix} = \begin{bmatrix} t_{Aold} \\ T_{old} \end{bmatrix} - z \begin{bmatrix} \frac{d}{\det(J(t_A,T))} f_1(t_A,T) - \frac{c}{(\det(J(t_A,T))} f_2(t_A,T) \\ -\frac{b}{\det(J(t_A,T))} f_1(t_A,T) + \frac{a}{(\det(J(t_A,T))} f_2(t_A,T) \end{bmatrix} \tag{4.52}$$

For the simulations that will be shown below, the initial conditions for t_A and T are chosen from an even grid with a spacing of $\{3s, 6s, 9s, 12s, 15s\}$ for both parameters. Some of the initial conditions lay within areas where the transcendental equations (4.33) show a large gradient with respect to t_A and T as can be seen in figure 4.7. This leads to large values in the Jacobian (4.36) and hence to a large change in the update step (4.49). Initial conditions that have a small value for t_A and T tend to run in to negative values for the unknown parameters which does not lead to correct solutions. Therefore the solver is stopped for cases t_A or T turn negative during the solving process and the solver is started with the next initial condition.

Figure 4.8 - 4.10 show the results of the convergence for the initial conditions mentioned above. For the different plots the parameter z was chosen as $z \in \{1.0, 0.5, 0.1\}$. For $z = 1.0$, which resembles the normal form of the Newton-Raphson formulation, the closest IC converged to the correct solution (marked with a cross). Due to the fact that in this case the step size of the original formulation is not altered, the number of steps for a convergence can be kept low. On the other hand only the closest IC converges to the correct solution. Figure 4.9 and figure 4.10 reduce the step size to $z = 0.5$ and $z = 0.1$. It can be seen that a larger number of ICs converge to the correct solution, however the number of steps to converge increases significantly. This leads to a larger computation time which is not beneficial for a time-efficient use in the C++ framework. If a convergence of the closest

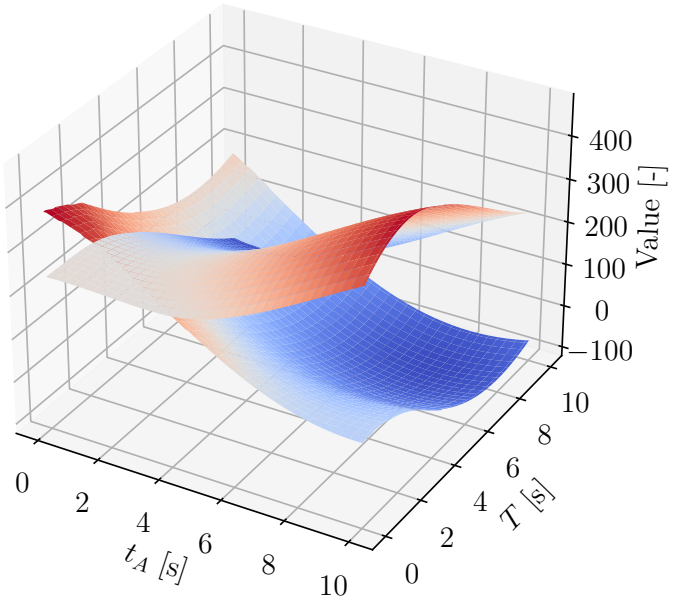


Figure 4.7: Solutions for f_1 and f_2 for $T, t_A \in 0, 10s$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$

IC with $z = 1.0$ is not possible, instead of decreasing z it might be more beneficial to increase the density of the grid due to the computation extend.

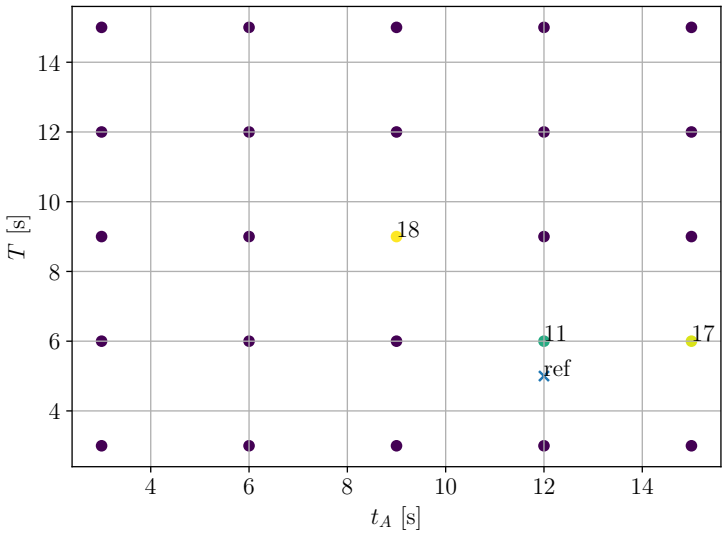


Figure 4.8: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown. $z = 1.0$ (normal Newton-Raphson form). $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

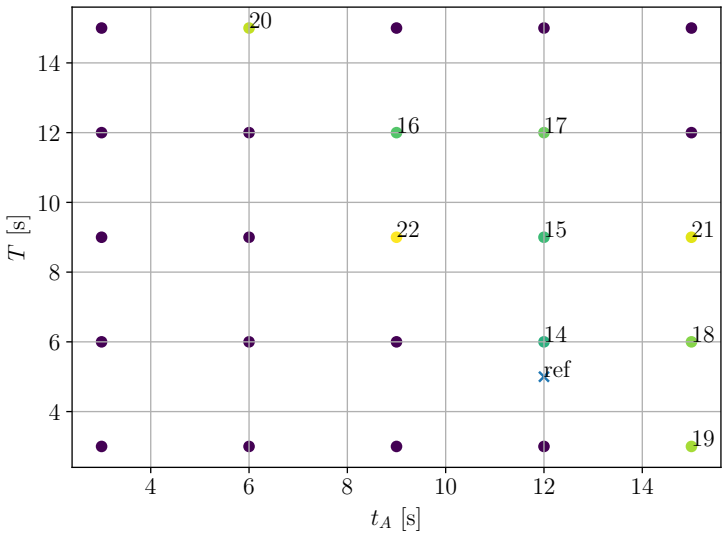


Figure 4.9: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown. $z = 0.5$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

As mentioned above, for some ICs the first update step in the solver might be relatively large, causing a large overshoot towards negative values or large values that diverge from the correct solution. Figure 4.12 shows the behaviour of the solver if the update step of the solver in (4.49) is limited to $\pm 2\frac{s}{step}$. This prevents

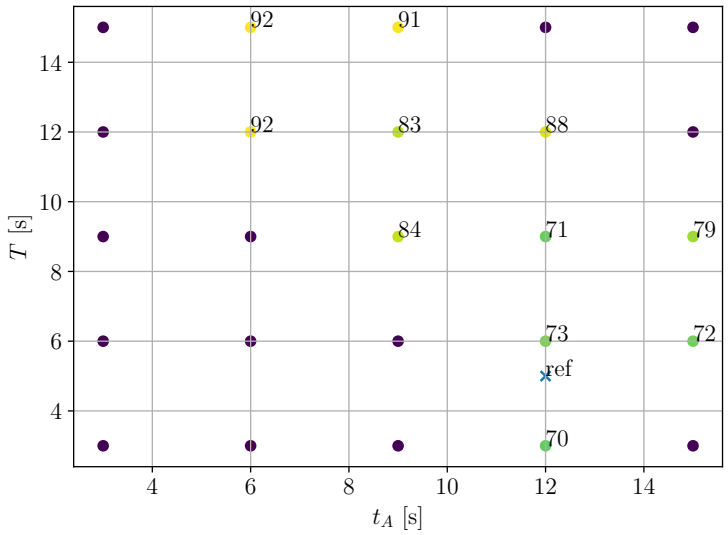


Figure 4.10: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown. $z = 0.1$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

areas with high derivatives to cause an overshoot of the solver. The figure shows the solutions for $z = 1.0$ hence the normal Newton-Rapshon formulation. It can be seen that a new IC converges to the correct solution and that apart from the closest solution, the number of steps until convergence has decreased. This is due to the large overshoot for the first update step that is now decreased due to the maximal allowed update per solver step. This behaviour can be seen for the IC $t_A = 15s$ and $T = 6s$ in figure 4.13. Hence for certain cases it might be beneficial to implement such a limitation to have more IC converge to the correct solution as well as decrease the number of steps needed by the solver. Although it has to be mentioned that certain ICs do not converge anymore to the correct solution after a limitation of the update step. This can be seen with the IC $t_A = 9s$ / $T = 9s$. For the case with no limitation, the IC converges within 18 steps of the solver. For the case with the limitation no convergence towards the solver can be reached. This is due to the fact that the limitation especially limits the initial steps of the solver leads to the fact that the solutions for t_A and T hit a periodic state for which (4.50) never is valid. This can be seen in figure 4.14 where for said IC the solutions of t_A and T for the the different solver steps are shown. Hence a possible implementation of an update step limitation has to be chosen with care.

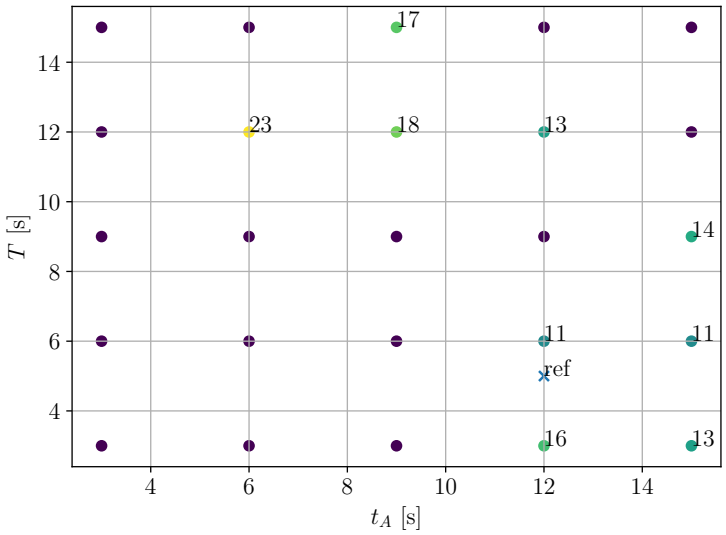


Figure 4.11: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown. $z = 1.0$ and limitation of the solver update for t_A and T to $\pm 2 \frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

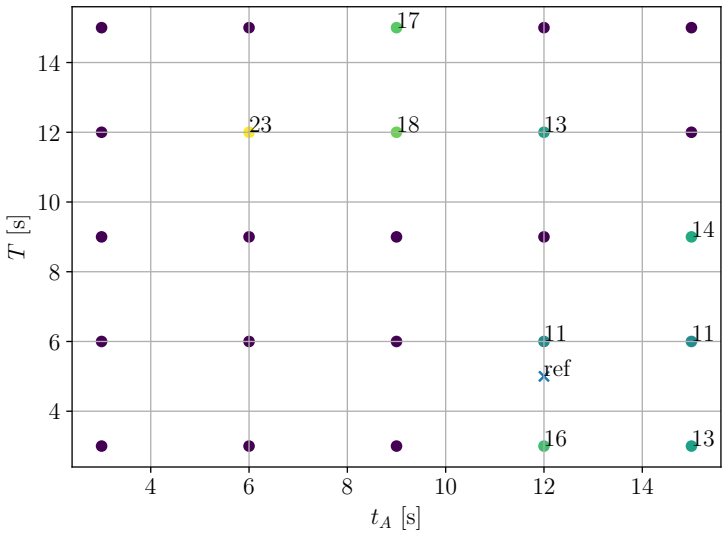


Figure 4.12: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown. $z = 1.0$ and limitation of the solver update for t_A and T to $\pm 2 \frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

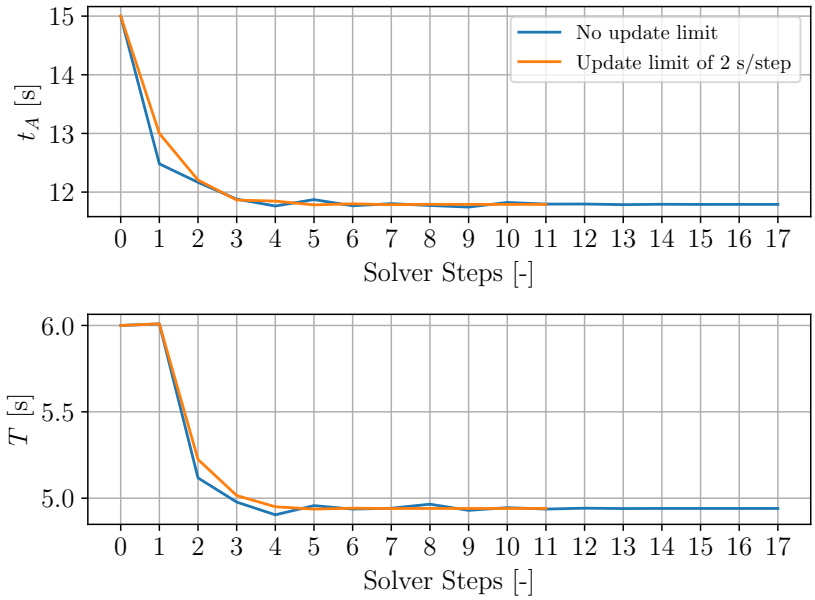


Figure 4.13: Solver solutions for t_A and T. Initial Condition $t_A = 15s$ and $T = 6s$. Comparison of the case for no limitation and limitation of the solver update for t_A and T to $\pm 2 \frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

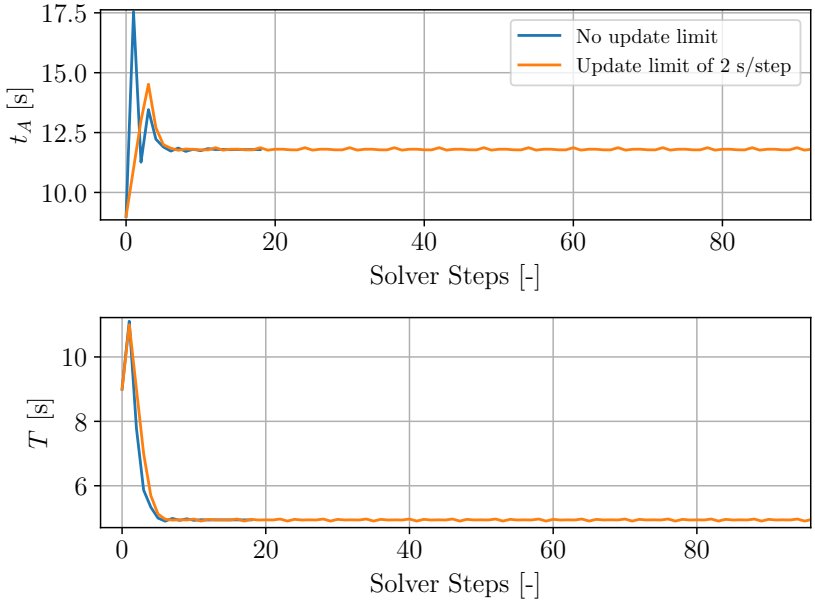


Figure 4.14: Solver solutions for t_A and T. Initial Condition $t_A = 9s$ and $T = 9s$. Comparison of the case for no limitation and limitation of the solver update for t_A and T to $\pm 2 \frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

As mentioned above the introduction of a limitation of the update step in the solver can lead to the fact that more ICs converge to the correct solution. The question arises if this behaviour is desirable as long as at least one IC always converges to the correct solution. To answer this question, the fact that the computation time of the solver is key for the implementation on-board of an UAV has to be kept in mind.

For the 25 ICs in the test case only a few converge to the correct solution. Hence the computation times for the ICs which do not lead to a correct solution play an important role. In figure 4.17 also the number of solver steps for the ICs which do not lead to a correct solution are shown. It can be seen that for the case with no limitation of the solver update steps in figure 4.15 some of these ICs only take a few computations before the solver stops computation for this specific IC. This does not happen for the case with a limitation in figure 4.16 where all ICs that not converge need 50 steps ($N = 50$ here), being the maximal allowed iterations of the solver as shown in (4.50). This phenomena comes from the fact that ICs that are initialized far from the correct solution lead to large solver update steps. This leads to the fact that a lot of those ICs violate the limitations for t_A or T during the solver updates which leads to a determination of the calculation for the IC ($0 < t_A/T < 30$). A specific case for said phenomena can be seen in figure 4.18.

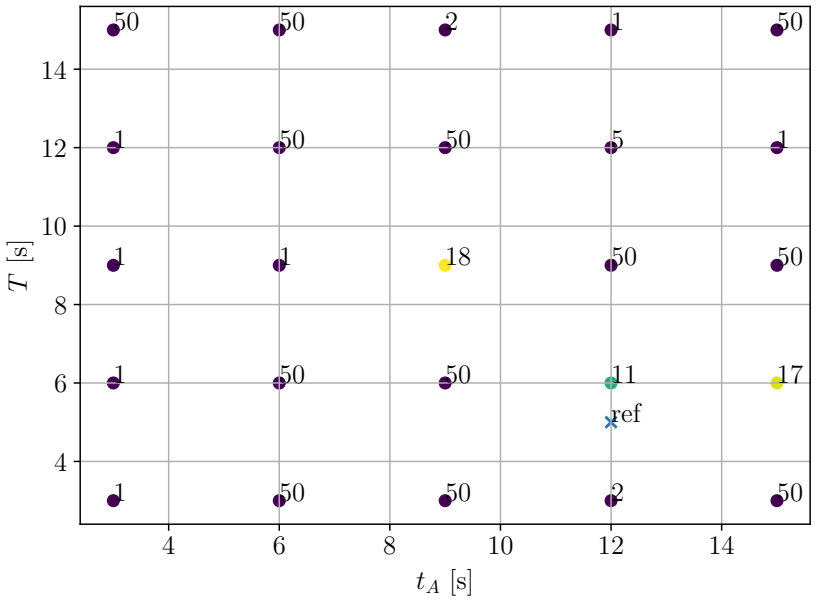


Figure 4.15: Case with no limitation of solver update steps

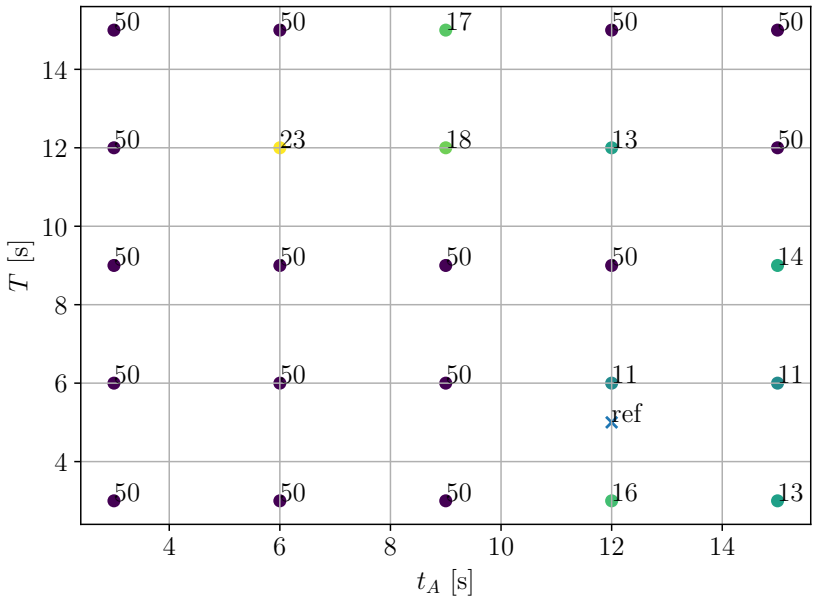


Figure 4.16: Case with limitation of the solver update step by $\pm 2 \frac{s}{step}$

Figure 4.17: Number of steps of the solver for the different initial conditions. All ICs which not converged to a solution are marked in purple.

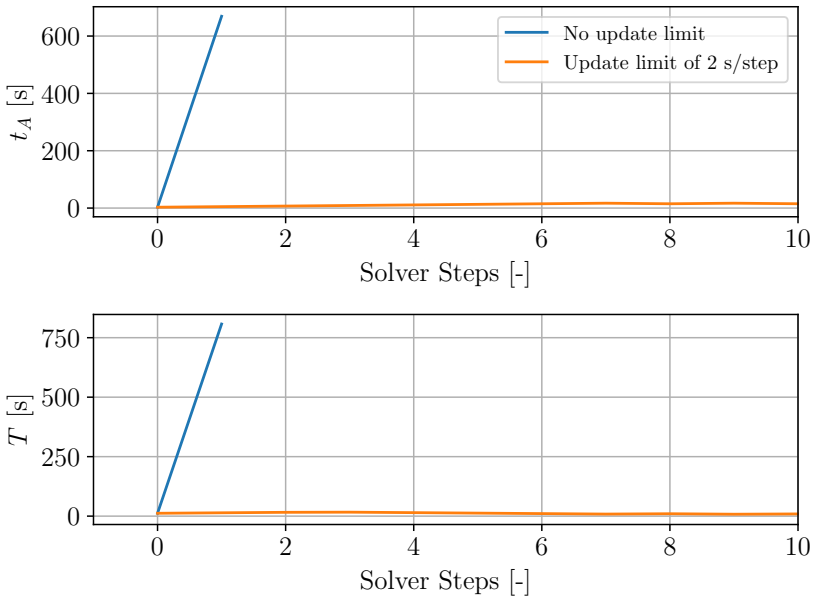


Figure 4.18: Solver solutions for t_A and T. Initial Condition $t_A = 3s$ and $T = 12s$. Comparison of the case for no limitation and limitation of the solver update for t_A and T to $\pm 2 \frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

Finally a closer look at the explicit solutions and their convergence behavior for t_A and T is taken. Figure 4.19 shows the paths for all ICs of figure 4.12 that converged to a solution. It can be seen that all ICs converge to the same solution being the time optimal CCC clothoid point-to-point coordination for this setup. Furthermore displayed in the plot is the reference path that was built in a forward geometry way. The small divergence of the found solutions to the reference comes from the numerical difference in the integrals in the forward geometry way and in the solver.

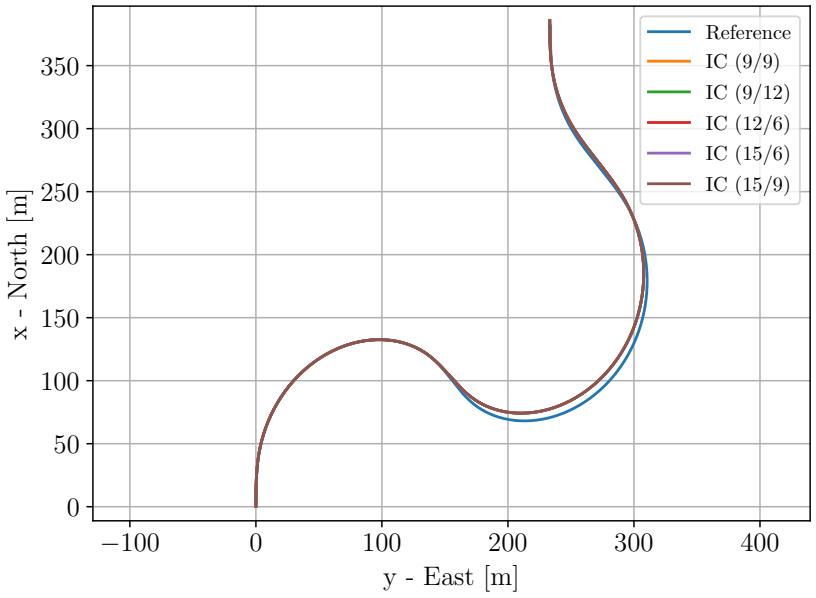


Figure 4.19: Solution paths for the converged IC (t_A / T) of figure 4.12. Reference path of used setup for the solver. $z = 1.0$ and limitation of the solver update for t_A and T to $\pm 2\frac{s}{step}$. $V_a = 20m/s$ and $V_\omega = 5m/s$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 381.542m$, $y_f = 233.744m$ and $\psi_f = 0°$.

It is not always guaranteed that all converging ICs converge to the same solution. Often there are various solutions for a specific CCC path type that fulfill the point-to-point coordination. In figure 4.20 for test case 2, all solution paths the solver converges to are shown. The different solutions which the solver converges to can also be seen in the overview of the behaviour of all ICs in figure 4.21. In the path planning framework a newly found solution is always compared to the previously found solution. The goal of the C++ framework is to find the time-minimal path solution. Hence the solutions are compared based on their total flight time.

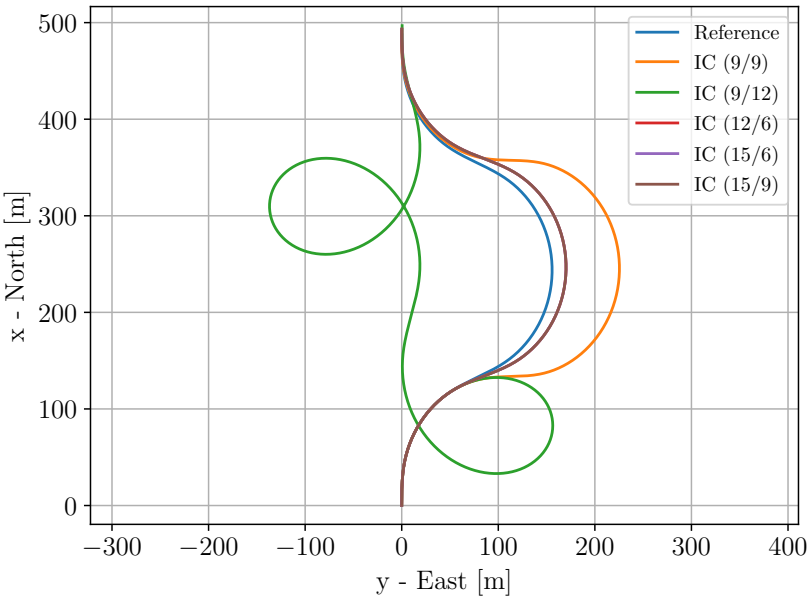


Figure 4.20: Paths for the converged IC (t_A / T) of test case 3 (see appendix E)

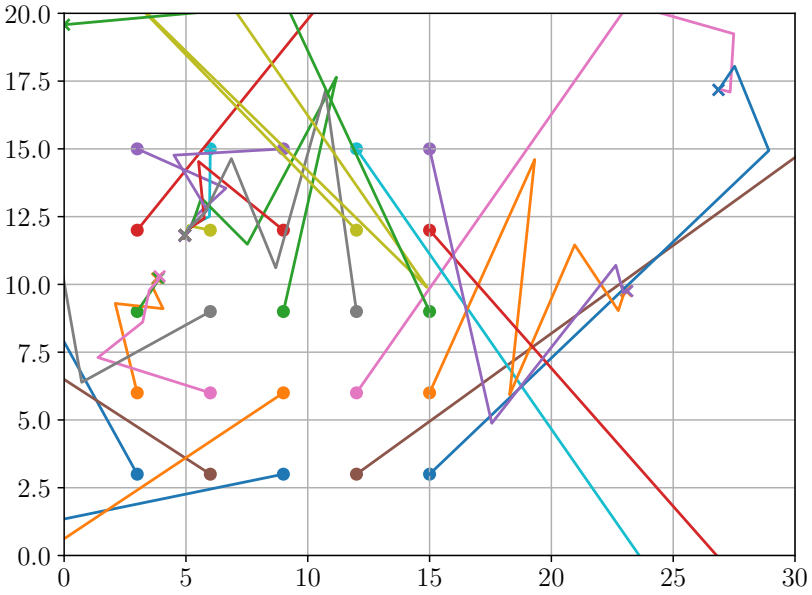


Figure 4.21: Behaviour of all ICs for test case 2. The initial conditions are marked with a point and the end point of the respective IC is marked with a cross in the same color. The four solutions the different ICs converge to can be seen. The x-axis represents $t_A[s]$ and the y-axis respectively $T[s]$.

As described in section 4.1.3 there are other forms of the CCC clothoid path type apart from the form derived so far. The derivation of the other remaining forms is shown in appendix D. The alternative forms all contain at least one segment for

which the maximum bank angle is not reached and hence the following holds:

$$t_A < 2t_1 \vee t_B < 2t_1 \vee T < 2t_1 \tag{4.53}$$

Where t_1 is the time until the maximum bank angle is reached. In the test setup used here this boundary is:

$$2t_1 = 3.49s \tag{4.54}$$

With the solver shown above it is not possible to converge to a solution where one of the segment times is below this boundary. With the implementation of the alternative forms described in the appendix D this is possible.

First a result for the form NR-R-R is shown. In figure 4.22 it can be seen that one solution found is $t_A = 2.5s$ and $T = 5s$. Hence it holds:

$$t_A = 2.5s < 2 * t_1 = 3.49s \tag{4.55}$$

Therefore the found solution is of the type NR-R-R. The described boundary for both parameters is also highlighted in the figure 4.22 with dashed lines.

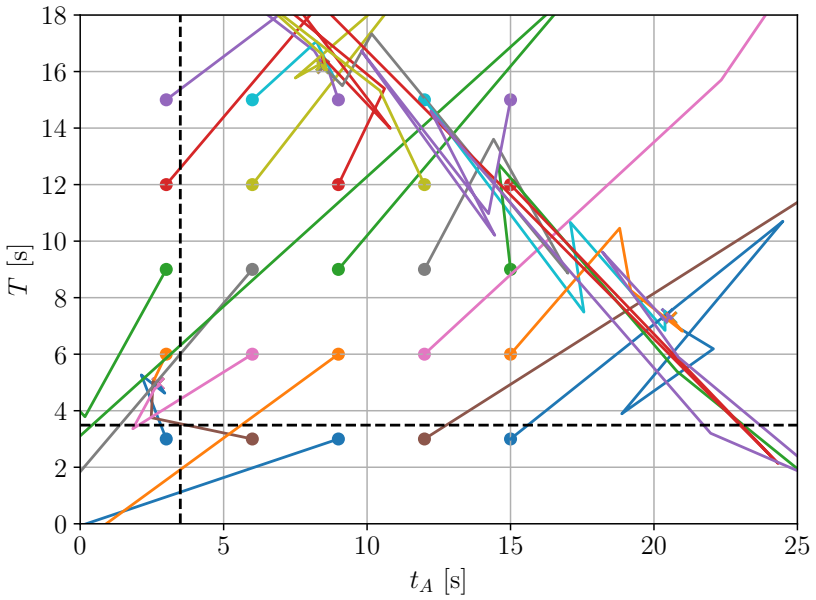


Figure 4.22: Behaviour of all ICs during the steps of the solver. The initial conditions are marked with a point and the end point of the respective IC is marked with a cross in the same color. The boundaries for the non-reaching case (NR) is marked with dashed lines. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 307.025m$, $y_f = -60.0063m$ and $\psi_f = 0°$.

In the same form as in figure 4.23 also an example for the form NR-R-NR is shown. The derivation for this form can be found in appendix D.5. As can be seen in the figure one of the solution that the solver converges is $t_A = T = 3.1s$. Hence for this case holds:

$$t_A < 2t_1 \wedge T < 2t_1 \tag{4.56}$$

Hence the first and last segment do not reach the maximum bank angle. For the middle segment, it is assumed that $t_B >= 2t_1$ and hence this segment reaches the maximum bank angle. From this fact, the form NR-R-NR follows.

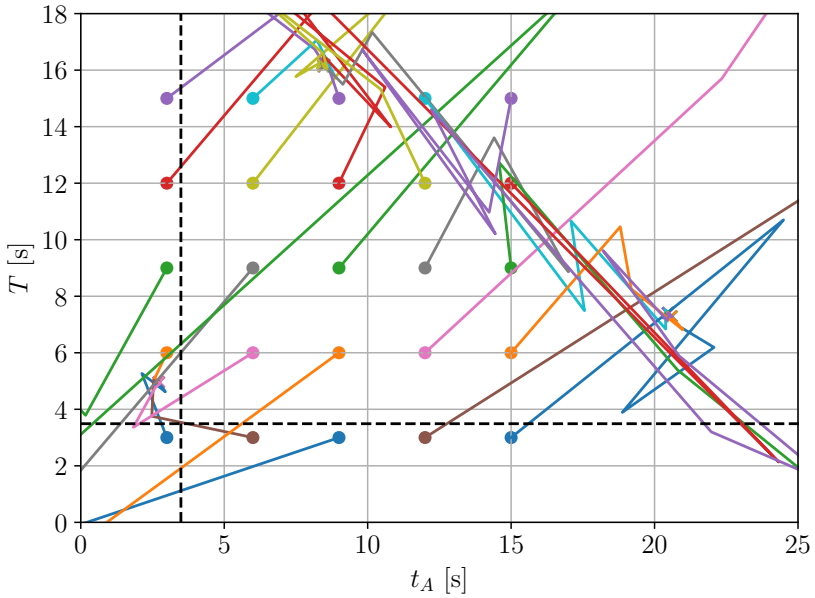


Figure 4.23: Behaviour of all ICs during the steps of the solver. The initial conditions are marked with a point and the end point of the respective IC is marked with a cross in the same color. The boundaries for the non-reaching case (NR) is marked with dashed lines. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 260.68m$, $y_f = 0.06m$ and $\psi_f = 0°$.

Regarding this, the adequate choice of the grid of initial conditions is important. The 8 forms of the CCC clothoids which are described in detail in appendix D are distributed as shown in figure 4.24 where the dashed lines indicate the bound of the NR to the R case for the two parameters. Hence the dashed lines are drawn at t_1. 1 to 8 in figure 4.24 refer to the 8 forms of the CCC clothoids implemented. In each sector of the plot two forms are mentioned since one refers to the form where the middle section reaches the maximal bank angle, the other one to the form where it is not reached. To find all possible solutions it is important that in every segment at least one initial condition is placed. Only if this condition is fulfilled, the solver can find the time-optimal path within all forms of the CCC case.

In this section only one specific test case was analyzed. Appendix E contains further test cases that give further insight into the behavior of the solver.

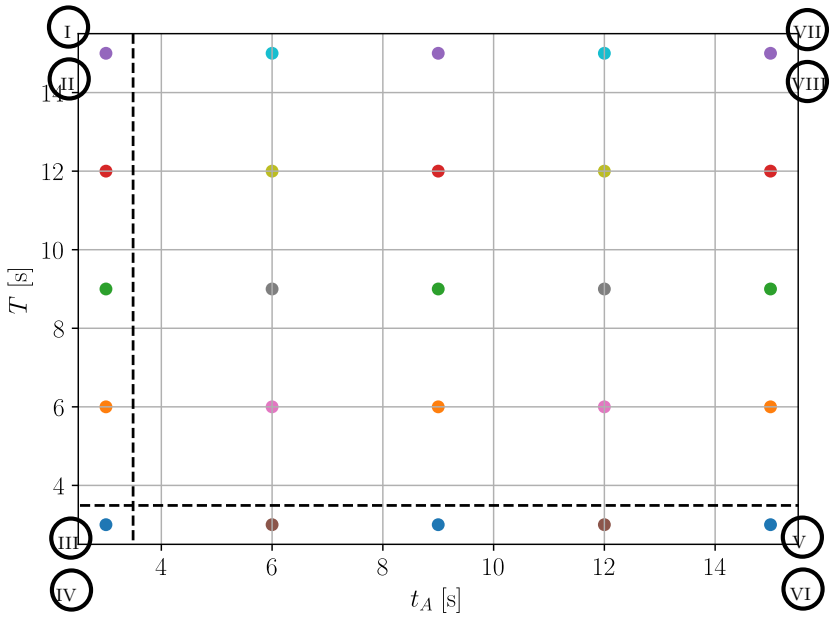


Figure 4.24: Distribution of all forms of CCC clothoids described in appendix D in a IC grid for t_A and T with the values $\{3s, 6s, 9s, 12s, 15s\}$. The boundaries for the non-reaching case (NR) is marked with dashed lines.

4.2 Results Path Planning Framework

With the derivation and implementation shown in section 4.1 the framework presented in figure 4.2 is completed. Hence the framework is able to search for feasible solutions for the given initial and final pose within the 6 possible path types (4 x CSC / 2 x CCC) both for clothoids and trochoids. As described in [1] and in section 4.1 are certain solutions not feasible due to non time-optimal behavior and not only because of physical infeasibility. Hence if a path type for example has a time-optimal solution that includes a turn of more than 2π in one segment, the path will not be shown as feasible option, since always another path type exists which does not show a full encirclement and hence is more time-optimal. Such an output of feasible paths by the path planning framework for a given setup can be seen in figure 4.25. Here can be seen that for the trochoid solutions more path types converge to a feasible solution. This comes from the fact that for clothoid solutions a turn to the left in the first segments leads to a full encirclement and hence does not lead to a feasible solution. Out of all the feasible solutions either for clothoids or trochoids the path planning framework searches then for the time-optimal path type. This path is then presents the time-optimal connection for the given poses in the present wind conditions. Figure 4.26 shows a similar setup for even higher wind conditions as a comparison.

Ending up with a time-optimal path for a given setup, the goal is now to allow state-of-the-art UAVs to follow the proposed paths. This will be extensively discussed in chapter 5.

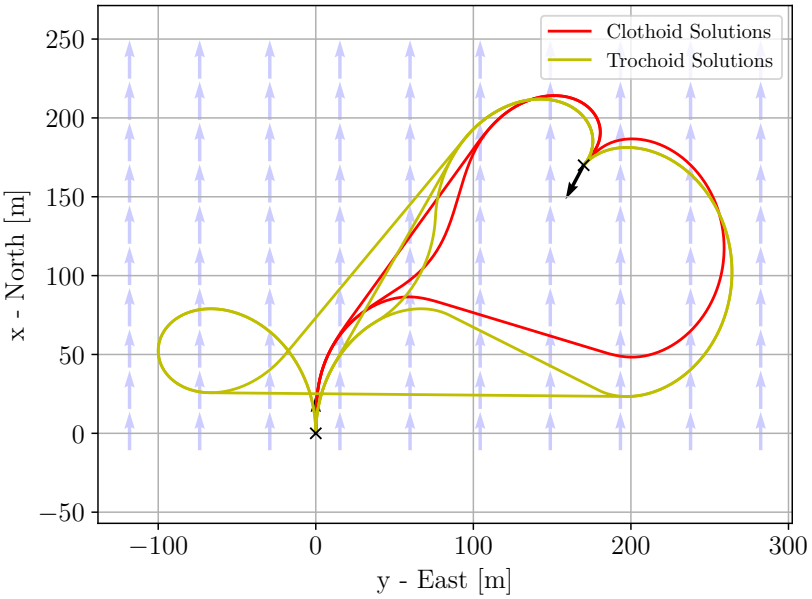


Figure 4.25: All feasible solutions calculated by the path planning framework. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Arbitrary final pose. Wind speed $V_\omega = 5\frac{m}{s}$ and heading $\psi_\omega = 0°$.

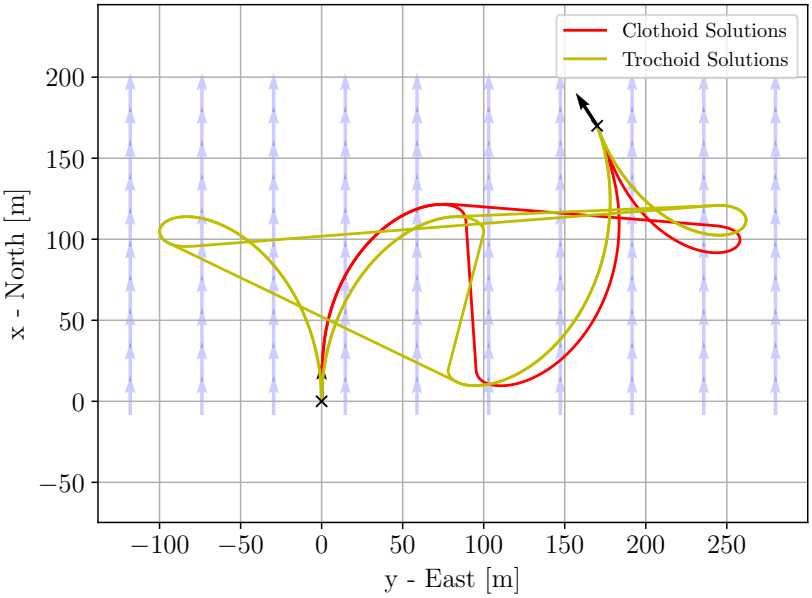


Figure 4.26: All feasible solutions calculated by the path planning framework. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Arbitrary final pose. Wind speed $V_\omega = 10\frac{m}{s}$ and heading $\psi_\omega = 0°$.

Chapter 5

Path Following

To be able to use the generated paths of the framework presented in chapter 4 with the state-of-the-art guidance controller, the paths have to be sampled or represented in a certain way. Hence the main focus in this chapter is the interconnection between the path planning framework and the guidance controller as indicated in figure 5.1.

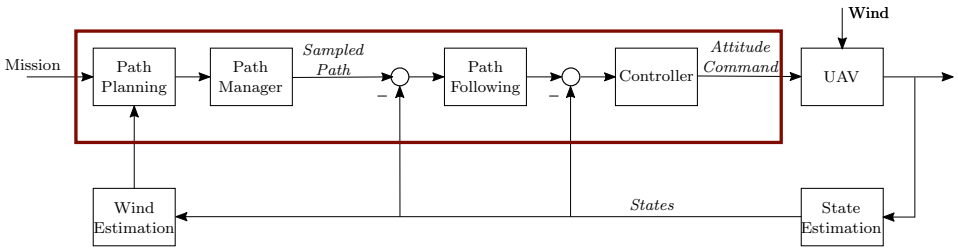


Figure 5.1: Flow chart overview of the guidance and control structure of a UAV. The connection between path planning and path following is highlighted in red.

A detailed overview of the parameters used for the path planning and the sampled path as input to the controller is given in figure 5.2. Here the mission is given by

$$\text{Mission} = [x_0, y_0, \psi_0, x_f, y_f, \psi_f, V_a] \tag{5.1}$$

consisting of the initial and final pose as well as the airspeed V_a. The wind estimation is given by

$$\text{Wind Estimation} = [V_\omega, \psi_\omega] \tag{5.2}$$

consisting of the wind speed V_ω and wind orientation ψ_ω. The sampled path which is handed over to the controller can be for example given by[1]

$$\text{Sampled Path} = N \times [x, y, \kappa] \tag{5.3}$$

consisting of the triplet of position x and y as well as the curvature at the respective position κ. N is the number of triplets which is needed to accurately represent the sampled path. The accuracy here is defined by the positional error between the representation of the path by the sampled triplet and the path generated by the path planning algorithm as a ground truth.
Path following then evaluates, based on the current state of the UAV, the required parameters for the controller input

$$\text{Control Input} = [e, \hat{t}, \kappa] \tag{5.4}$$

[1] Later in this chapter different ways to represent the path are discussed. This representation is here used to show the general workflow of the path following in guidance and control of UAVs.

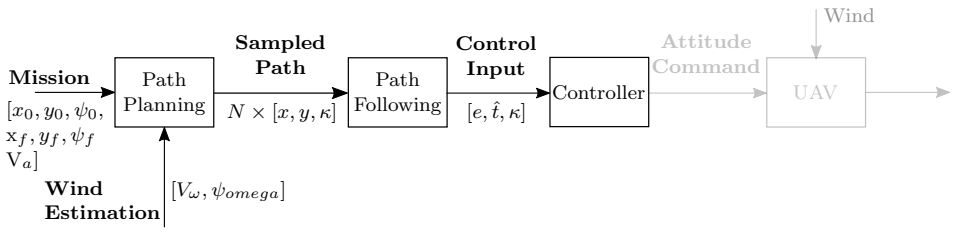


Figure 5.2: Overview of the solution approaches for all different path types for trochoids and clothoids.

with e being the track error between the UAV and the closest point on the path, $\hat{t}$ the tangent at the closest point and κ the curvature at the closest point. The definition and orientation of these parameters is shown in figure 5.3. Furthermore other variables are established which will be used throughout this chapter. P_{UAV} being the current position of the UAV and P_{cl} the closest point on the arbitrary path based on which then the control parameters are calculated. $\hat{n}$ defines the path normal at P_{cl} oriented towards the center of the circle built with κ^{-1}.

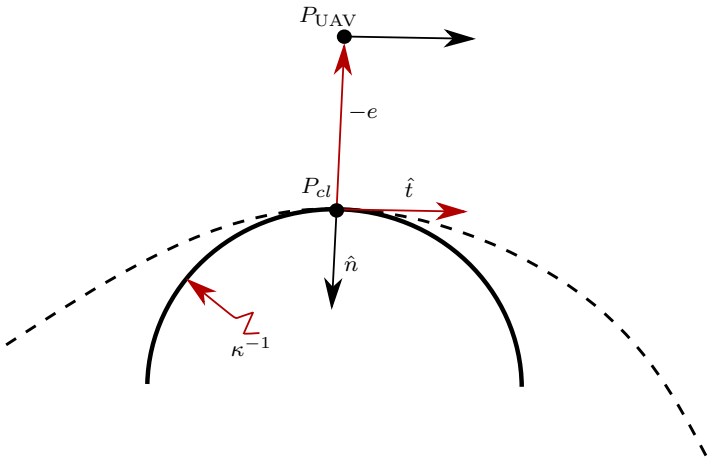


Figure 5.3: Definition and orientation of the relevant path following parameters shown for an arbitrary path and UAV position. The control input parameters are highlighted in red.

Based on the parameters the guidance controller evaluates the attitude commands to navigate the UAV towards the path. For all references and later on also for simulations throughout this work the NPFG controller presented in [36] is used. This controller is taken as a representation of a state-of-the-art guidance controller, similar derivations and results shown also hold for different controllers using the same control parameters[2].

[2]For example for the classical L_1 guidance controller.

The calculation of the control inputs for the proposed paths in this work imposes a set of challenges when used for guidance with UAVs. State-of-the-art autopilot software such as PX4 generally use simple shapes when planning paths. Such shapes are often lines or loiters (or helix in a 3D setup) as illustrated in figure 5.4. Such geometrical shapes can simply be described by a few parameters as shown in figure 5.4 in red. However, more importantly, for these shapes a closed-form solution to calculate the closest distance to an arbitrary UAV position exists. This is essential for a fast calculation of the controller inputs stated above. Both issues, the geometrical representation of the path as well as the calculation of the controller inputs, impose a challenge for trochoid or clohtoid segments. In the sections below, both issues are discussed in detail for the proposed path types.

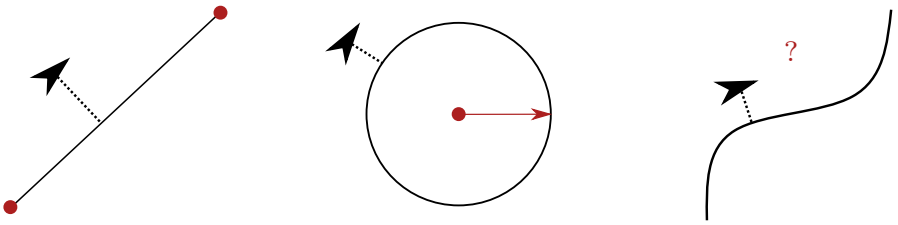


Figure 5.4: Path following for different segments of the generated path. As example an arbitrary CSC path type is shown.

To calculate the necessary controller inputs the path following has to calculate the respective parameters. The path following is adapted to the type of the respective segment. In figure 5.4 an example for a CSC path type is shown. For the first and last segment, a path following has to be developed to follow the path accurately. For the straight segment (S) a classical waypoint following approach is chosen.

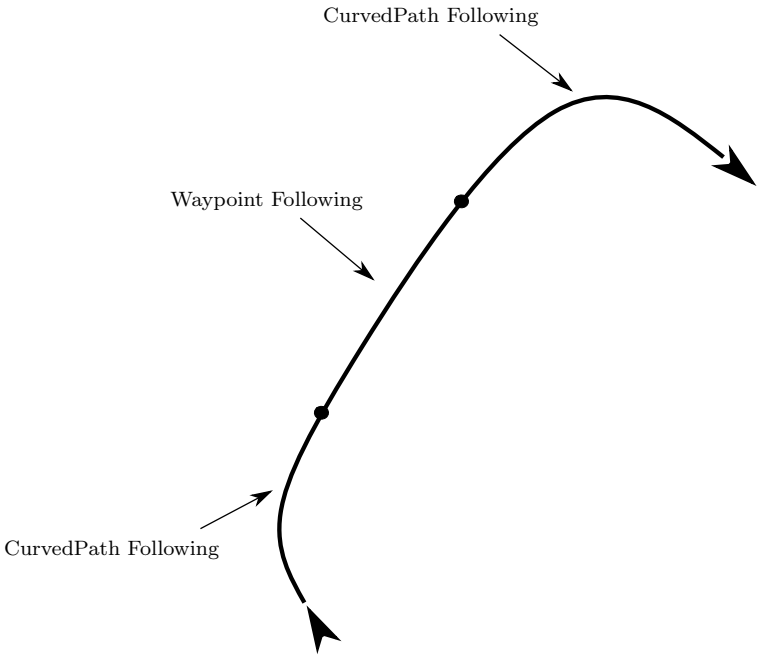


Figure 5.5: Path following for different segments of the generated path. As example an arbitrary CSC path type is shown.

As mentioned above, for the implementation the number of parameters to store is a limiting factor. For the waypoint following approach for a straight segment this does not impose a major problem since here only

$$\#\text{parameters} = Nn([x, y]) = 2n([x, y]) = 4 \tag{5.5}$$

parameters are needed.

For curved segments, built for example with trochoids or clothoids, the situation is different. Here the number of sampled points N has to be high enough to approximate the path. Here two approaches are proposed, which both are illustrated in figure 5.6 below:

- Pre-calculate all sampled points N and store them (either RAM or accessible on SD card)
- Store all necessary parameters to online calculate the currently necessary parameters of the path

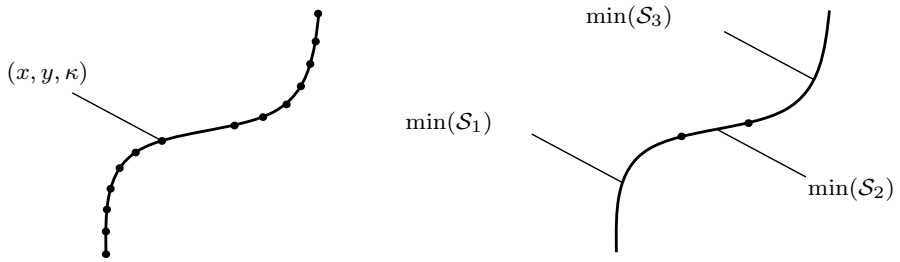


Figure 5.6: Sketches showing the pre-calculation approach on the left and the online calculation approach on the right side.

If all parameters are calculated beforehand, the parameter scale with

$$\#\text{parameters} = Nn([x, y, \kappa]) = 3N \tag{5.6}$$

If the first method is chosen then a general approach has to determine the behavior of the path between the sampled points. Here the way to determine this behavior between sampled points mainly influences the quality of the approximation. Although generally for such an approach the number of samples N determines the maximal error of the approximation by

$$\min_{\forall p \in P} |p_{approx}(p) - p_{exact}| \leq \epsilon \tag{5.7}$$

To have a sufficient approximation based on the use case, the error has to be lower than a certain maximal value ϵ. Here also a key factor is to sample the path with higher density in areas with high curvature since the highest errors occur in such areas. Nevertheless this first method has one main advantage; it can be applied to any sampled arbitrary curved path of any shape. Hence here no adjustments have to be made if the method is used with trochoids, clothoids or any other arbitrary path type as long as the path can be sampled with the triplet of x, y and κ.

The second method presents an approach which is optimized to the currently used path type. Here the exact geometrical representation of the path type are implemented and can be evaluated at every point along the path. This provides a considerable difference to the first approach where the behavior between the sampled points has to be estimated. Hence to calculate the closest point of the UAV

to the path with this method, the exact values of the respective path type can be used. Thus it is for example possible to directly calculate the current value for $\hat{t}$ or κ. For this second method the main question that arises is how many parameters are minimally required to represent a segment of the respective path type.

The two methods shortly introduced above both have desirable proprieties to generate a suitable path following approach for the path types used in this work. Hence for both methods an approach is proposed in section 5.1 and section 5.3.

5.1 Pre-calculation approach

The goal of the path sampling is to generate as few samples as possible to be able to follow the planned path. This is due to the fact that the RAM to store the current information on the μC is very limited.
For the approach to interpolate the behaviour between the different sampled points of a path the following approaches may be considered:

- Linear segment interconnection
- Mean curvature interpolation
- Linear interpolation

5.1.1 Linear segment interconnection

The approach to interconnect the sampled points with linear segments is the simplest approach. With increasing N the position error can be reduced to a minimum. Since a high number of sampled points N is not beneficial for the implementation as well as with this approach the curvature κ per sampled point is not used. Hence this approach is not beneficial for use here. The latter reason would lead to the fact that the UAV always tries to follow an infinite straight line due to the lack of curvature information which leads to a non-beneficial tracking behavior of the path.

5.1.2 Mean curvature interpolation

The two other proposed approaches involve a more sophisticated approximation of the generated path and use the curvature information sampled in the triplet $[x, y, \kappa]$. In figure 5.7 the general approach is shown. If for two sampled points p_i and p_{i+1} their respective curvature is $\kappa_i \neq 0 \vee \kappa_{i+1} \neq 0$ then the two points are interconnected with an arc which is built based on the curvature information of the two points. In such a way it is possible to approximate the generated path with the use of only a few sampled points. The main interest here are the control inputs that are afterwards handed over to the controller (highlighted in red in figure 5.7).

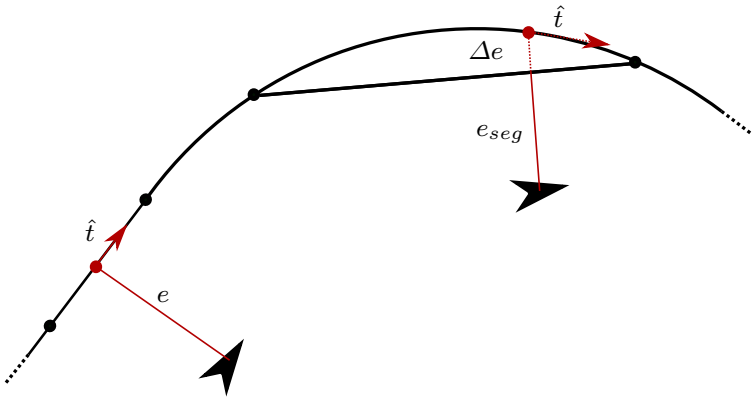


Figure 5.7: General approach for path sampling using the curvature information of the sampled points. Highlighting of the track error, closest-point and path normal which are the main inputs to the controller when following the path.

With the approach of interpolation by mean curvature, the two sampled points are interconnected with a circular path with

$$\kappa_{mean} = \frac{\kappa_i + \kappa_{i+1}}{2} \tag{5.8}$$

This approach provides a simple approximation and leads to low positional errors. In figure 5.8 the approximation is shown in comparison to the original path for two encirclements of trochoid segment. The error plot in figure 5.9 shows the positional error of the approximated path. It can be seen that the maximal error peak always appears in the middle of a segment approximation since the approximation with the mean curvature diverges the most from the original path for this part. Furthermore the highest positional errors occur for segments with high curvature as stated in the section above. Overall, the absolute error is relatively low. Although the major downside of the approach is that the curvature of the approximated path jumps discretely at the sampling points due to the change of the curvature reevaluated with (5.8). This behaviour does not influence the approximation of the closest point P_{cl} or track error e for the controller inputs but leads to discrete jumps in the approximation of the curvature κ and hence the tangent $\hat{t}$. The behaviour of the curvature over all segments of the approximated path can be seen in figure 5.10. This can result in an odd behaviour of the controller tracking such a path. Hence this approach is not suitable to use with a guidance controller.

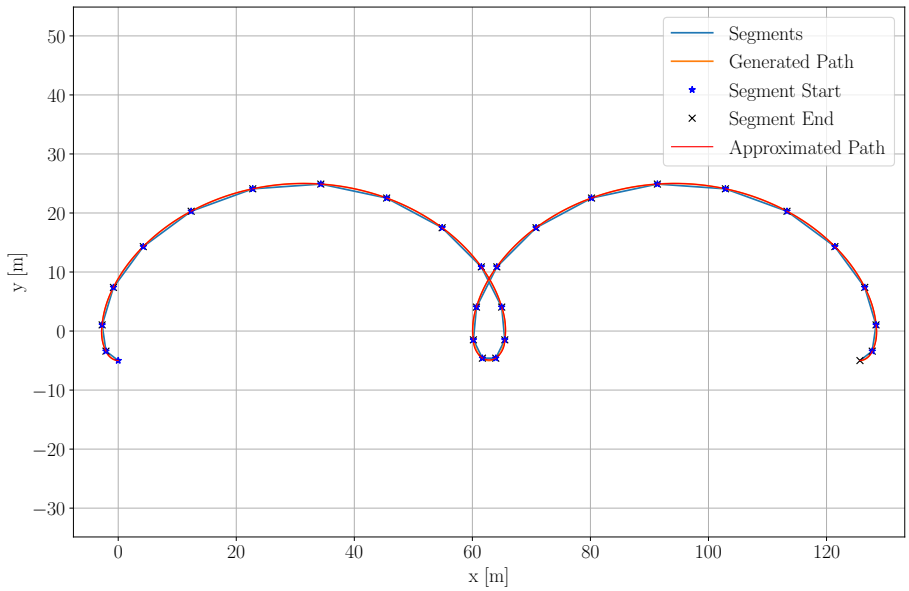


Figure 5.8: Double trochoidal turn segment approximated with mean curvature of the sampled points.

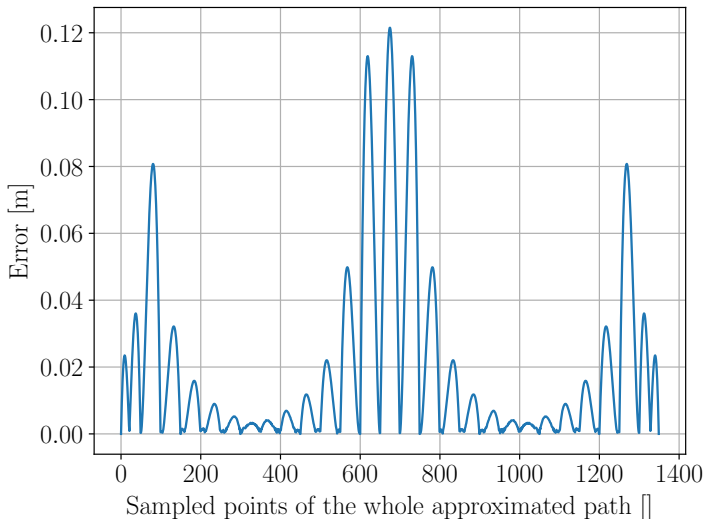


Figure 5.9: Error of the approximated path with the mean curvature approach in comparison to the original path generated by the C++ framework.

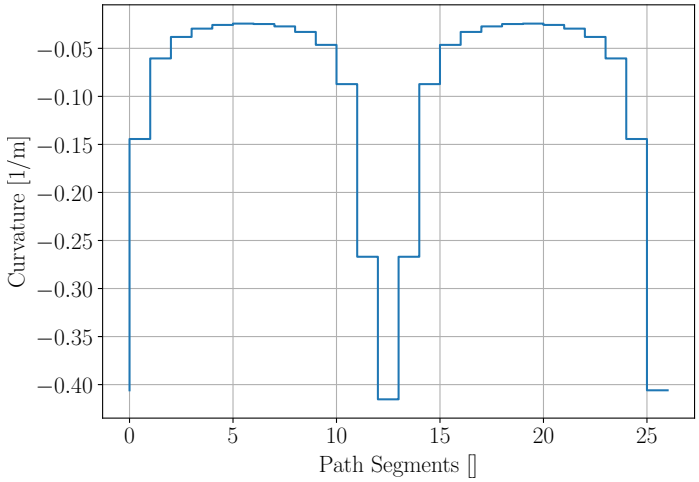


Figure 5.10: Curvature approximation of the mean curvature approach over all segments of the approximated path.

5.1.3 Linear interpolation

Hence a more sophisticated approximation has to be found which provides a continuous behaviour of the curvature κ and hence also the tangent $\hat{t}$. This can be done with linear interpolation of the circular segments built with the curvature of the sampled points building the segment. The overview of such an approach is shown in figure 5.11. Based on σ, determined by the closest point to the UAV position on the segment, the delta track error Δe_σ and $\hat{t_\sigma}$ for the the two circular segments with κ_i and κ_{i+1} are evaluated (displayed in orange and blue in figure 5.11). Based on σ then the linear interpolation of the controller inputs is evaluated

$$\Delta e_\sigma = (1-\sigma)\Delta e_i(\sigma) + \sigma \Delta e_{i+1}(\sigma) \tag{5.9}$$

$$\hat{t}_\sigma = (1-\sigma)\hat{t}_i(\sigma) + \sigma \hat{t_{i+1}}(\sigma) \tag{5.10}$$

$$\kappa_\sigma = (1-\sigma)\kappa_i + \sigma \kappa_{i+1} \tag{5.11}$$

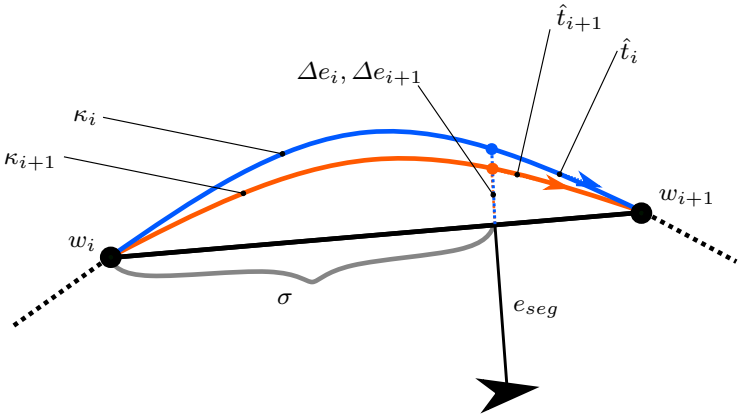


Figure 5.11: Linear interpolation between two sampled points with curvature κ_i and κ_{i+1} based on the segment distance σ.

For the linear interpolation approach shown above first Δe and $\hat{t}$ for the respective values of the curvature κ_i and κ_{i+1} have to be calculated to be used in (5.9) - (5.10). In figure 5.12 the overview of such a circular segment spanned by a specific curvature defined by one of the two points is shown. Based on the curvature, the radius

$$R = \frac{1}{|\kappa|} \tag{5.12}$$

is defined. The segment length is given by a and the distance to the closest point of the UAV to the segment is given by σ. Note the slight abuse of the parameter σ. For the linear interpolation $\sigma \in [0,1]$ but here σ is used as the actual distance. θ is the angle spanning the circular segment between p_i and p_{i+1} and is defined by

$$\theta = 2\arcsin(\frac{a}{2R}) \tag{5.13}$$

The distance r is then defined as

$$r = R\cos(\frac{1}{2}\theta) \tag{5.14}$$

Based on these parameters Δe is given by

$$\Delta e(\sigma) = \sqrt{R^2 + (\frac{a}{2} - \sigma)^2} - r \tag{5.15}$$

The calculation of Δe does not depend on the orientation of the circular segment based on the two waypoints p_i and p_{i+1}. On the other hand for the calculation of $\hat{t}$ the orientation is relevant. Here the case for a general circular segment spanned in the first quadrant is derived. When implementing the approach for all segments, the respective angles have to be adjusted. First the middle point of the segment is defined

$$x_m = \frac{x_i + x_{i+1}}{2} \tag{5.16}$$

$$y_m = \frac{y_i + y_{i+1}}{2} \tag{5.17}$$

The distance in x and y direction between the two points is given by

$$\Delta x = x_{i+1} - x_i \tag{5.18}$$

$$\Delta y = y_{i+1} - y_i \tag{5.19}$$

Based on the curvature κ the middle point spanning the circular segment is

$$x_{circ} = \begin{cases} x_m - \frac{r\Delta y}{a} & \kappa \geq 0 \\ x_m + \frac{r\Delta y}{a} & \kappa < 0 \end{cases} \tag{5.20}$$

$$y_{circ} = \begin{cases} y_m + \frac{r\Delta x}{a} & \kappa \geq 0 \\ y_m - \frac{r\Delta x}{a} & \kappa < 0 \end{cases} \tag{5.21}$$

Based on the xy-coordinate system the angle between the x-axis and the the side of the circular segment with p_i is evaluated[3]

$$\beta_i = \arccos(\frac{|x_i - x_{circ}|}{R}) \tag{5.22}$$

To evaluate the tangent at the correct point of the circular segment, t is defined as

$$t = r + \Delta e \tag{5.23}$$

Hence angle α is then defined as

$$\alpha = \arccos(\frac{t}{R}) \tag{5.24}$$

Based on that the absolute angle of the circular segment that spans the line to the closest point $\hat{p}$ ca be evaluated

$$\phi = \begin{cases} \beta + (\frac{\theta}{2} - \alpha) & \sigma > \frac{a}{2} \\ \beta + (\frac{\theta}{2} + \alpha) & \sigma \leq \frac{a}{2} \end{cases} \tag{5.25}$$

Based on the angle ϕ then the tangent at the closest point is

$$\hat{t} = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix} \tag{5.26}$$

Using the definition for ϕ in (5.25) and the following definitions of the parameters $\hat{t}$ can be stated as

$$\hat{t}(\sigma) = \begin{bmatrix} \cos(\arccos(\frac{|x_i - x_{circ}|}{R}) + (\frac{\theta}{2} \pm \arccos(\frac{r + \Delta e(\sigma)}{R}))) \\ \sin(\arccos(\frac{|x_i - x_{circ}|}{R}) + (\frac{\theta}{2} \pm \arccos(\frac{r + \Delta e(\sigma)}{R}))) \end{bmatrix} \tag{5.27}$$

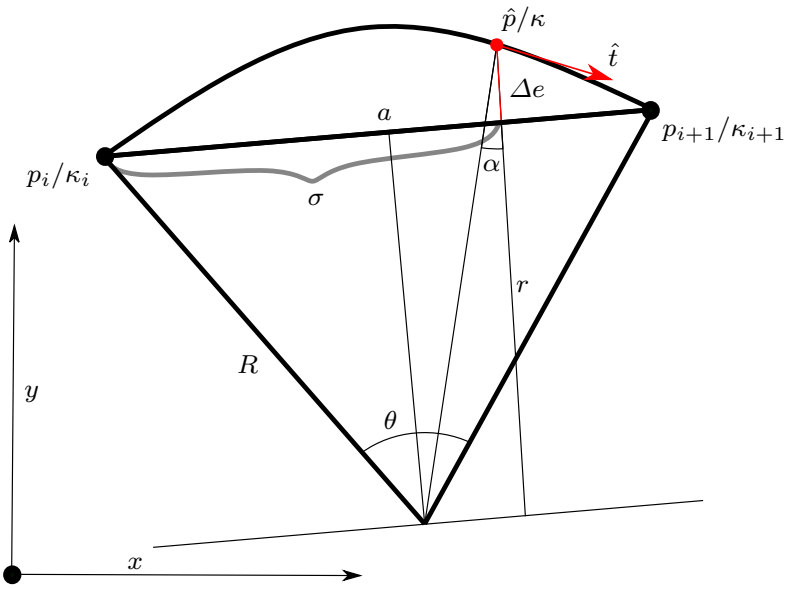


Figure 5.12: Circular segment connecting the sampled points p_i and p_{i+1} with curvature κ_i or κ_{i+1}. Focus on the calculation of the control inputs Δe and $\hat{t}$ for the curvature of one of the two waypoints.

Ending up with the full definition of the linear interpolation approach, only one last step has to be done. To calculate the track error e as input for the controller the track error to the closest point on the segment, which is denoted with r in figure 5.12, e_{seg} has to be added

$$e = r + \Delta e_\sigma = e_{seg} + \Delta e_\sigma \approx |P_{\text{UAV}} - P_{cl}| \tag{5.28}$$

With the approach of linear interpolation, the error of the approximated path to the path generated by the framework behaves similar to the track error with the mean curvature approach. The behaviour of the error is shown in figure 5.13.

As intended the behaviour of the curvature of the approximated path is continuous as can be seen in figure 5.14. This is the effect of the positional continuity when the approach switches to the next segment. Due to this continuity the parameters e and κ are continuous along the sampled path, which is beneficial for the tracking behaviour of the guidance controller. Although the position is continuous when switching the segments at the sampling point, the orientation is not exactly continuous. This leads to the fact that a small discrete jump in the tangent occurs when the segment is switched. For this approach taken here, the latter behavior is taken as a suitable approximation for the continuity of of the tangent $\hat{t}$. The approximation is described in detail in appendix F.

[3]Based on the quadrant of the circle where the side is spanned in this angle has to be adapted in the implementation.

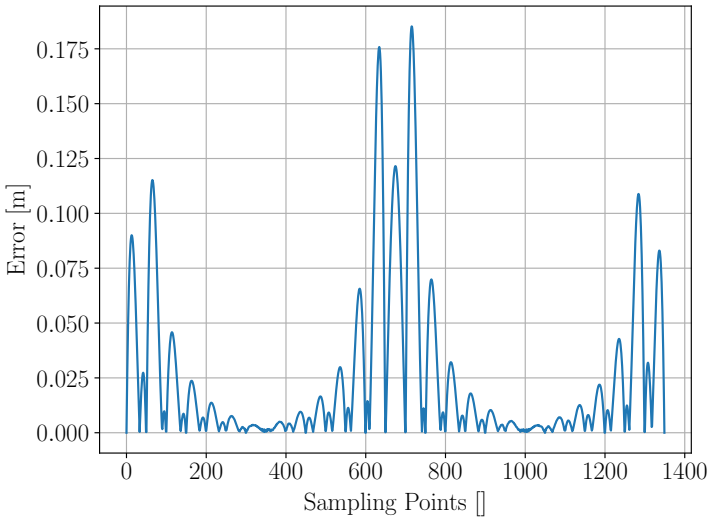


Figure 5.13: Error of the approximated path with the linear interpolation approach in comparison to the original path generated by the C++ framework.

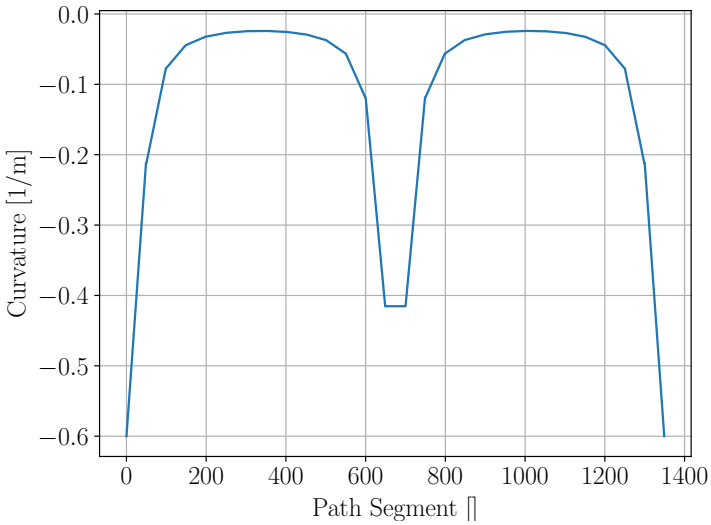


Figure 5.14: Curvature approximation of the linear interpolation approach over all segments of the approximated path.

Finally the limitations of the approach presented here have to be discussed. The parameter determining the limitation is the inter-waypoint sampling time Δt. This parameter determines the time-spacing with which the points p_i are sampled. Hence a smaller parameter Δt increases the performance of the approximation of the path, but on the other hand the number of sampling points and hence data to store increases. Therefore a trade-off has to be made to evaluate the suitable value for Δt to reach the desired performance.
Although there is an upper limit for Δt which must not be exceeded. Based on the curvature of the path and the wind direction, the spacial distance between the

sampled points varies. To be able to use the approach described in figure 5.12 it has to be possible to span a circular segment between the two points for all segments. Hence based on the notation in figure 5.12 it has to hold

$$\frac{1}{2}||p_{i+1} - p_i|| \leq \min\left(\frac{1}{\kappa_i}, \frac{1}{\kappa_{i+1}}\right) \tag{5.29}$$

If this equation is not fulfilled, it is for at least one of the radii based on the curvature of the sampled points not possible to span a circular segment and the approach fails. This situation occurs only for relatively high Δt and occurs first for segments in areas with high curvature due to the fact that the high curvature leads to small radii since $R = \frac{1}{\kappa} \to 0$.

5.2 Waypoint switching

On the mission level a suitable approach to switch between waypoints for path following has to be implemented which is suitable for the approach presented in section 5.1.3. For this purpose different approaches can be used and these are compared in this section.

Acceptance radius

Most autopilots for fixed-wing UAVs use the approach of an acceptance radius because of its simplicity. For this method the transition from one waypoint to another is made when the UAV enters a predefined sphere with the radius r

$$|p_{\mathrm{UAV}}(t) - w_{i+1}| \leq r \tag{5.30}$$

where $p_{\mathrm{UAV}}(t)$ is the current position of the UAV, w_{i+1} the coordinates of the next waypoint and r the radius of the sphere. In figure 5.15 an example for this waypoint switching method is shown. Here the switching to the next waypoint already occurs outside of the segment area[4]. In this area the normal path following logic shown in the section above can not be applied and another path following policy has to be used. This topic is covered is specifically addressed in the approach shown in section 5.2.

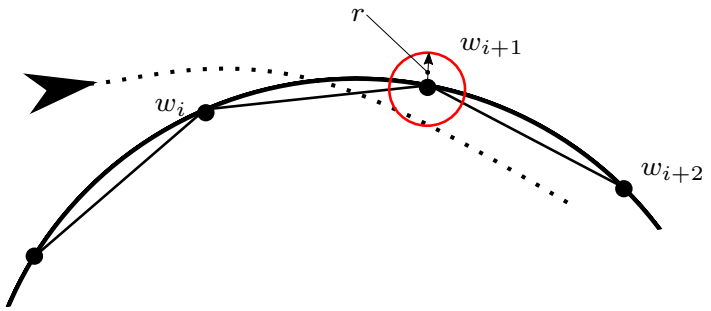


Figure 5.15: Waypoint switching with an acceptance radius r shown for the waypoint w_{i+1}

As intended by (5.30) for the acceptance radius r a fixed value can be used. Instead of using a fixed acceptance radius in implementations often an adaptive acceptance radius based on the parameters of the path following logic and the UAV ground speed is used. This is the case for the implementation in PX4. For the navigation of FW the acceptance radius is defined by the L1 distance determined by[5]

$$L_{1_{\mathrm{distance}}} = \frac{1}{\pi} L_{1_{\mathrm{damping}}} L_{1_{\mathrm{period}}} ||\boldsymbol{v}_{xy_{\mathrm{ground}}}|| \tag{5.31}$$

A similar formulation can be useful for the path following used here. There the formulation should be based on one or more parameters from the following list: current path curvature κ, inter-waypoint sampling distance Δt, UAV ground speed $||\boldsymbol{v}_{xy_{\mathrm{ground}}}||$ and tuning parameters of the NPFG controller.

[4]The segment area is defined as the area spanned by the perpendicular lines at the points spanning the segment.

[5]The use of the acceptance radius in the L_1 guidance controller in PX4: https://docs.px4.io/v1.11/en/flight_modes/mission.html

Regardless of the used formulation for the acceptance radius, the method holds generally several drawbacks:

- The waypoint switching occurs in the last segment area which leads to a behaviour that does not present the planned path inside some parts of the sphere.
- If the radius of the sphere is too large, the accuracy of the flight can be reduced greatly.
- If the radius of the sphere is too small, the UAV can never reach the waypoint and the waypoint switching will not be made.

Bisector plane

Another common and simple approach is to use a switching plane instead of a sphere. If the UAV crosses a defined plane H the waypoint is switched to the next one. Since the plane H is infinite the UAV will always switch to the next waypoint. An example of this approach is shown in figure 5.16. One of the main advantages of this method is that the accuracy of the flight near the waypoint does only depend on the accuracy of the path following of the UAV and not on the waypoint switching method itself.

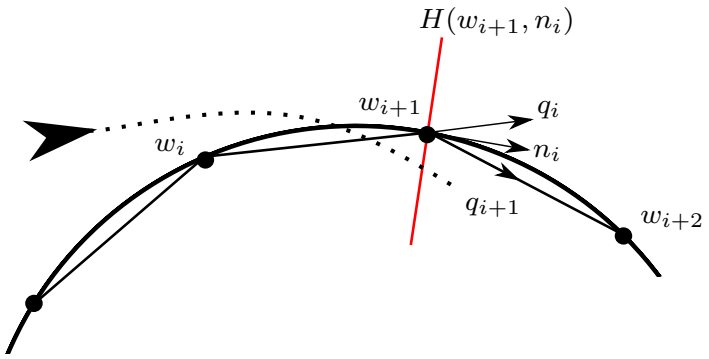


Figure 5.16: Waypoint switching with bisector plane shown for the waypoint w_{i+1}

To use this method the plane H has to be defined mathematically based on the last, current and next waypoint. These waypoints are represented by w_i, w_{i+1} and w_{i+2} and are used to calculate the unit vectors $\boldsymbol{q}$

$$\boldsymbol{q_i} = \frac{w_{i+1} - w_i}{|w_{i+1} - w_i|} \tag{5.32}$$

$$\boldsymbol{q}_{i+1} = \frac{w_{i+2} - w_{i+1}}{|w_{i+2} - w_{i+1}|} \tag{5.33}$$

Here $\boldsymbol{q_i}$ defines the current flight leg and $\boldsymbol{q}_{i+1}$ the next flight leg. Next the vector $\boldsymbol{n}$ representing the bisector has to be defined

$$\boldsymbol{n_i} = \frac{\boldsymbol{q_i} + \boldsymbol{q}_{i+1}}{|\boldsymbol{q_i} + \boldsymbol{q}_{i+1}|} \tag{5.34}$$

Using the orientation of the plane given by $\boldsymbol{n}$ and one point on the line spanning the plane given by w_{i+1} the switching can be stated as

$$p_{\mathbf{UAV}}(t) \in H(w_{i+1}, \boldsymbol{n_i}) \tag{5.35}$$

Hence the switching is done when the UAV is on the plane or beyond it.

Sector switching

Here the control input parameters $\hat{t}$, κ and e are stated for different regions during control. To ensure a continuous behaviour of the controller and hence the attitude commands, the definitions of the control parameters have to be continuous in every point during the path following. Ideally they are also smooth generating attitude commands the UAV is able to follow. The case here is shown for waypoint switching based on the perpendicular line at the end of a segment which is established in figure 5.17.

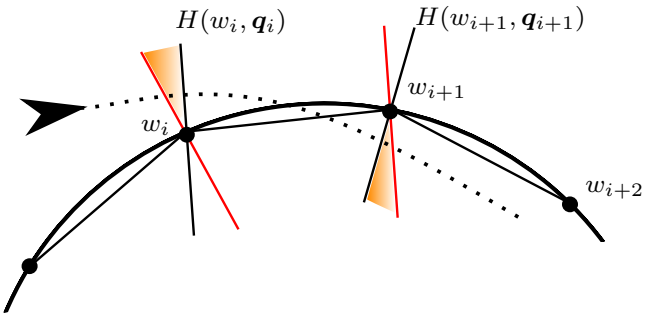


Figure 5.17: Waypoint switching based on the segment normal highlighted in red. Yellow area marking shows area where not only the linear interpolation can be used. Dotted line shows a UAV path which hits all three different areas that exist between the two waypoint switching lines (red) for a segment.

The control parameters are built in the following way based on the section the UAV currently is in:

$$e(\sigma, p_{\text{UAV}}) = \begin{cases} e_{segment} + (1-\sigma)\Delta e_i(\sigma) + \sigma \Delta e_{i+1}(\sigma), & p_{\text{UAV}} \notin H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ e_{w_{i+1}}, & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ (1-\sigma_\alpha)e_i(\sigma_i) + \sigma_\alpha e_{i+1}(\sigma_{i+1}), & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \in H(w_{i+1}, \boldsymbol{q}_{i+1}) \end{cases}$$

$$\kappa(\sigma, p_{\text{UAV}}) = \begin{cases} (1-\sigma)\kappa_i(\sigma) + \sigma \kappa_{i+1}(\sigma), & p_{\text{UAV}} \notin H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ (1-\sigma)\kappa_i(\sigma) + \sigma \kappa_{i+1}(\sigma)|_{\sigma=1} = \kappa_{i+1}, & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ (1-\sigma_\alpha)e_i(\sigma_i) + \sigma_\alpha e_{i+1}(\sigma_{i+1}), & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \in H(w_{i+1}, \boldsymbol{q}_{i+1}) \end{cases}$$

$$\hat{t}(\sigma, p_{\text{UAV}}) = \begin{cases} (1-\sigma)\hat{t}_i(\sigma) + \sigma \hat{t_{i+1}}(\sigma), & p_{\text{UAV}} \notin H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ (1-\sigma)\hat{t}_i(\sigma) + \sigma \hat{t_{i+1}}(\sigma)|_{\sigma=1} = \hat{t}_{i+1}, & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \notin H(w_{i+1}, \boldsymbol{q}_{i+1}) \\ (1-\sigma_\alpha)e_i(\sigma_i) + \sigma_\alpha e_{i+1}(\sigma_{i+1}), & p_{\text{UAV}} \in H(w_{i+1}, \boldsymbol{q}_i) \wedge \in H(w_{i+1}, \boldsymbol{q}_{i+1}) \end{cases}$$

To guarantee a continuous behaviour of the control parameters in the inner cone, linear interpolation between the two boundary values is used. The linear interpolation is based on the angle spanned by the UAV position due to the fact that the

angle is a continuous parameter in the cone. Based on the angle the parameter σ_α is defined, as can be seen in figure 5.18.

Due to the continuous and exact behavior according to the planned path the waypoint switching presented last is the most suitable to use with the pre-calculation approach.

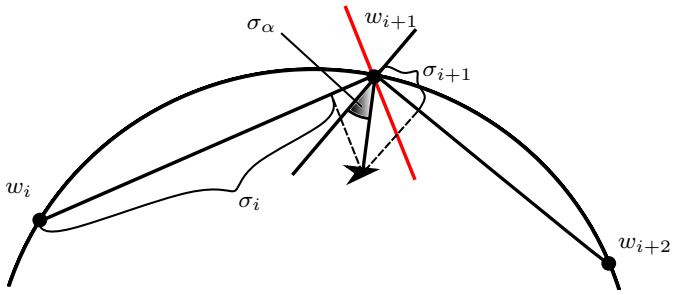


Figure 5.18: Linear interpolation in the inner cone section based on the parameter σ_α.

5.3 Online-calculation approach

The second approach introduced in the beginning of chapter 5 is specifically adjusted to the path type used and does not show a general approach for the navigation of arbitrary curved paths. In this section the approach is shown for the path types used in this work and with Bézier curves an outlook of the use with other path types is given.

As mentioned for the approach shown in section 5.1 the number of parameters to store to navigate through a curved segment is a key aspect considered for the application here. With the online-calculation approach the main question is how many parameters have to be stored to uniquely describe a segment of the respective path type. The number of parameter is hence defined by the specific expressions for the position, the tangent and the curvature throughout the path. Hence this minimal number of parameters are than handed over to the path following logic instead of pre-calculated triplets describing the path. This adjustment is shown in the figure Figure 5.19. Here $\mathcal{S}$ are all parameters describing a segment of a specific path type. From these parameters only the minimal representation to uniquely define the segment is then used for the path following.

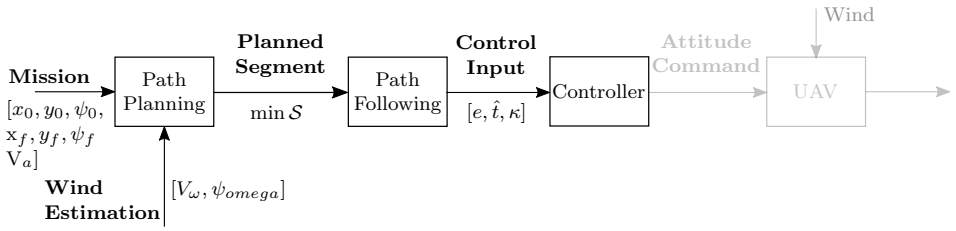


Figure 5.19: Flow chart for controlling an UAV with the online-calculation approach.

Comparing to the approach shown in section 5.1 less parameters have to be stored for the current approach if

$$\min(\mathcal{S}) < 3N \tag{5.36}$$

To conclude if the online-calculation approach is beneficial, the definitions of the different geometrical path types have to be considered to define the minimal set of parameters $\min(\mathcal{S})$ to be stored. Therefore in the next section the definitions for trochoids and clothoids in the path planning setup as used throughout this work are considered. Furthermore the procedure for e general Bézier curve, and hence not specifically for a UAV path planning setup, is shown. The latter type is shown due to the use across literature highlighted in section 2.1.2.

5.3.1 Minimal segment representation

Trochoids

The necessary waypoints along the path are here calculated when a waypoint is reached and the next waypoint has to be calculated for the path following. Therefore the equations below describe the expressions for the position, tangent (hence the speed) and curvature along the whole path. Additional to the trochoid form

presented in section 3.2 here the heading of the wind is not assumed to be $\psi_\omega = 0$:

$$x(t) = \frac{V_a}{\delta_1 \omega} \sin(\delta_1 \omega t + \psi_0) + V_\omega \sin(\psi_\omega) t + x_0 \tag{5.37}$$

$$y(t) = -\frac{V_a}{\delta_1 \omega} \cos(\delta_1 \omega t + \psi_0) + V_\omega \cos(\psi_\omega) t + y_0 \tag{5.38}$$

$$\frac{dx}{dt}(t) = V_a \cos(\delta_1 \omega t + \psi_0) + V_\omega \sin(\psi_\omega) \tag{5.39}$$

$$\frac{dy}{dt}(t) = V_a \sin(\delta_1 \omega t + \psi_0) + V_\omega \cos(\psi_\omega) \tag{5.40}$$

$$\hat{t}(t) = \begin{bmatrix} \frac{dx}{dt}(t) \\ \frac{dy}{dt}(t) \end{bmatrix} \tag{5.41}$$

$$\kappa(t) = \frac{V_a \delta_1 \omega \left(V_a + V_\omega \sin\left(\delta_1 \omega t + \psi_0 + \psi_\omega\right)\right)}{\left(V_a^2 + 2 V_a V_\omega \sin\left(\delta_1 \omega t + \psi_0 + \psi_\omega\right) + V_\omega^2\right)^{1.5}} \tag{5.42}$$

For this setup the following parameters have to be stored:

$$x_0, y_0\text{: Initial position} \tag{5.43}$$

$$V_a, \psi_0\text{: Airspeed, Initial heading} \tag{5.44}$$

$$V_\omega, \psi_\omega\text{: Windspeed, Wind heading} \tag{5.45}$$

$$\delta_1 \omega\text{: Oriented turn rate} \tag{5.46}$$

$$\Delta t\text{: Time increment for sampling} \tag{5.47}$$

$$T\text{: Total time for segment} \tag{5.48}$$

The setup for a trochoid segment hence ends up with 9 parameters for an arbitrary number of sampled points N. To increase the number of sampled points, here only the time increment for sampling Δt has to be decreased. Here the parameters V_a and V_ω are not taken from the current state of the UAV, but the values used in the path planning stage are stored. This due to the fact that the point-to-point path planning was done with the stated parameters and the use of current values would then not end up at the desired final pose.

Clothoids

Moreover the case for clothoids is derived in detail here since it imposes some further challenges. The calculation for the desired values for a clothoidal segment is:

$$x(t) = \int_0^t \left(V_a \cos\left(\psi(\tau)\right) + V_\omega \cos\left(\psi_\omega\right)\right) d\tau + x_0 \tag{5.49}$$

$$y(t) = \int_0^t \left(V_a \sin\left(\psi(\tau)\right) + V_\omega \sin\left(\psi_\omega\right)\right) d\tau + y_0 \tag{5.50}$$

$$\frac{dx}{dt} = V_a \cos\left(\psi(t)\right) + V_\omega \cos\left(\psi_\omega\right) \tag{5.51}$$

$$\frac{dy}{dt} = V_a \sin\left(\psi(t)\right) + V_\omega \sin\left(\psi_\omega\right) \tag{5.52}$$

$$\hat{t}(t) = \begin{bmatrix} \frac{dx}{dt}(t) \\ \frac{dy}{dt}(t) \end{bmatrix} \tag{5.53}$$

$$\kappa = \frac{V_a \left(V_a + V_\omega \cos\left(\psi_\omega - \psi(t)\right)\right) \frac{d}{dt}\psi(t)}{\left(V_a^2 + 2 V_a V_\omega \cos\left(\psi_\omega - \psi(t)\right) + V_\omega^2\right)^{1.5}} \tag{5.54}$$

For the case the maximal bank angle is reached $\psi(t)$ is defined as

$$\psi(t) = \begin{cases} \alpha\bar{u}\frac{t^2}{2} + \psi_0 & t \in [0, t_1] \\ \alpha\bar{u}t_1 t + \psi_0 - \alpha\bar{u}\frac{t_1^2}{2} & t \in [t_1, \bar{t} - t_1] \\ \alpha\bar{u}(t\bar{t} - \frac{t^2}{2}) + \psi_0 - \alpha\frac{\bar{u}}{2}(2t_1^2 + \bar{t}^2 - 2\bar{t}t_1) & t \in [\bar{t} - t_1, \bar{t}] \end{cases} \tag{5.55}$$

Therefore $\frac{d}{dt}\psi(t)$ is defined as

$$\frac{d}{dt}\psi(t) = \begin{cases} \alpha\bar{u}t & t \in [0, t_1] \\ \alpha\bar{u}t_1 & t \in [t_1, \bar{t} - t_1] \\ \alpha\bar{u}(\bar{t} - t) & t \in [\bar{t} - t_1, \bar{t}] \end{cases} \tag{5.56}$$

For the case the maximal bank angle is not reached $\psi(t)$ is given by

$$\psi(t) = \begin{cases} \alpha\bar{u}\frac{t^2}{2} + \psi_0 & t \in [0, \bar{t}/2] \\ \alpha\bar{u}(t\bar{t} - \frac{t^2}{2}) + \psi_0 - \alpha\bar{u}\frac{\bar{t}^2}{4} & t \in [\bar{t}/2, \bar{t}] \end{cases} \tag{5.57}$$

Therefore $\frac{d}{dt}\psi(t)$ is defined as

$$\frac{d}{dt}\psi(t) = \begin{cases} \alpha\bar{u}t & t \in [0, \bar{t}/2] \\ \alpha\bar{u}(\bar{t} - t) & t \in [\bar{t}/2, \bar{t}] \end{cases} \tag{5.58}$$

To cover both cases stated above, the following set of parameters have to be stored to uniquely represent the clothoidal segment:

$$x_0, y_0\text{: Initial position} \tag{5.59}$$
$$V_a, \psi_0\text{: Airspeed, Initial heading} \tag{5.60}$$
$$V_\omega, \psi_\omega\text{: Windspeed, Wind heading} \tag{5.61}$$
$$\alpha\bar{u}\text{: Oriented maximum heading change rate} \tag{5.62}$$
$$t_1\text{: Time to reach maximum bank angle} \tag{5.63}$$
$$\Delta t\text{: Time increment for sampling} \tag{5.64}$$
$$T\text{: Total time for segment} \tag{5.65}$$

Compared to a trochoidal segment here one additional parameter is needed which still leads to the conclusion that for a segment with more than three waypoints, the online calculation of the waypoints is beneficial based on the necessary amount of parameters to store. Tough for the online calculation of a clothoid segment, another problem appears. The expressions describing the x and y position of a clothoidal segment given in (5.49) and (5.50) representing Fresnel Integrals. These integrals do not have a closed-form solution which leads to the consequence that numerical integration has to be used to calculate the next waypoint online. Although only the incremental change to the next waypoint has to be calculated, still n expressions have to be evaluated per positional information (hence x and y position)

$$n = 2 * \frac{\Delta t}{t_{\mathbf{int}}} \tag{5.66}$$

with Δt being the time increment for sampling of the waypoints and t_{int} the increment for the numerical integration of the Fresnel Integrals. This has to be done both for x and y coordinates of the waypoint and hence the number of evaluations is doubled. The solutions for the tangent and curvature along the path can be expressed analytically, hence do not introduce additional complexity compared to the trochoidal segment.

Arbitrary Paths

So far only clothoidal and trochodial path segments were considered to be used for the described path following logic. Since the explicit geometrical form of any path can be implemented, it can be applied to arbitrary curved paths. This is here shown with the use of a Bézier curve. Important to mention here: the mathematical description and examples shown here should not show ideal paths for UAVs considering the dynamics of the UAV or navigation in windy conditions. Only the proof of concept for an additional path type which might be beneficial for UAV path planning is shown.
Bézier curves of different order can be used to describe a path in 2D. The example here is restricted to the fourth order due to the ability to form similar paths as has been shown with the three-segment CSC approach throughout this work. Mathematically speaking a Bézier curve of fourth order creates a CSC path with the straight segment (S) collapse in to one point being the only vertex the curve has. An example of such a Bézier path can be seen in figure 5.20.

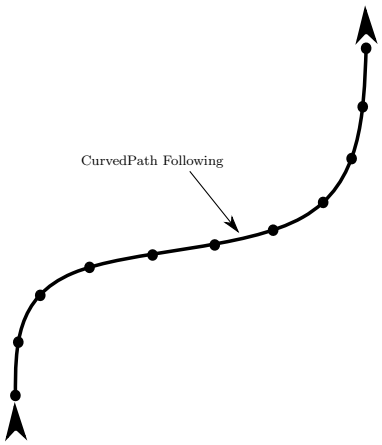


Figure 5.20: Example of a sampled Bézier curve of third order.

Since such a path is not subdivided in segments as CSC and CCC paths, the whole path can be followed with the Curved Path following logic as one segment. The approach also works when the curvature changes sign throughout the path or the curvature is close to zero as can be seen later on.

To mathematically describe the necessary parameters of a Bézier curve of fourth order, the following expressions have to be evaluated:

$$x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t)t^2 x_2 + t^3 x_3 \tag{5.67}$$

$$y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t)t^2 y_2 + t^3 y_3 \tag{5.68}$$

$$\frac{dx}{dt} = -3t^2 x_2 + 3t^2 x_3 + 3t x_1(2t-2) + 2t x_2(3-3t) - 3x_0(1-t)^2 + 3x_1(1-t)^2 \tag{5.69}$$

$$\frac{dy}{dt} = -3t^2 y_2 + 3t^2 y_3 + 6t y_1(t-1) - 6t y_2(t-1) - 3y_0(t-1)^2 + 3y_1(t-1)^2 \tag{5.70}$$

$$\hat{t}(t) = \begin{bmatrix} \frac{dx}{dt}(t) \\ \frac{dy}{dt}(t) \end{bmatrix} \tag{5.71}$$

$$\kappa(t) = \kappa_{\text{Bezier}}(t) \tag{5.72}$$

The given formulation in (5.67) and (5.68) is defined based on 4 control points

$$p = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{5.73}$$

The coordinates of the four points $p_i = [p_x, p_y]$ with $i \in [0, 3]$ are then used to build the formulation of the Bezier curve. Of the four control points, p_0 builds the start position and p_3 the final position. p_1 and p_2 determine the curvature of the path. To evaluate the position along the path $t \in [0, 1]$ is chosen. As stated above, the described path type here does not describe a path planning setup for point-to-point coordination. Due to the extent of the formulation of the curvature $\kappa(t)$ the expression for (5.72) can be found in (G.1).

For completeness also here the parameters necessary to uniquely define the path are stated for the use in online waypoint coordination. Important to state here: This path type defines a whole trajectory between two points. For trochoid and clothoid segments above, the stated parameters have to be stored for every curved segment (C) of the respective path. Hence here the stated parameters define the entire path:

$$x_0, x_1, x_2, x_3\text{: x-positions of the control points} \tag{5.74}$$

$$y_0, y_1, y_2, y_3\text{: y-positions of the control points} \tag{5.75}$$

$$\Delta t\text{: Time increment for sampling} \tag{5.76}$$

5.3.2 Path following policy

This approach is driven by the goal to formulate a simple formulation for the control parameter that is valid inside of the whole segment and does not have different policies. Also this approach relies on the fact that the waypoints for the respective type are calculated online, which has proven to use lower storage for most cases in the last section. In the first path following approach shown in section 5.1 the triplet (x, y, κ) of N sampled points was assumed to be the starting point of the algorithm. Using online waypoint calculation also other values can be calculated temporally per evaluation cycle and do not have to be stored to uniquely define the path. Hence here the exact calculation of the position, tangent and the curvature is an interesting approach. Throughout this section only the derivation for trochoid segments is shown but can be adapted similarly for the use with clothoid or Bézier curves with the use of the mathematical definitions seen in the section above.

In figure 5.21 the setup of this approach is shown. Two achieve the goal of not having different policies inside one segment area here the bisector plane switching (see section 5.2) is used. To have a continuous behavior a path following policy is built for the circular sector spanned by the last bisector plane $\boldsymbol{n}_{i-1}$ and the next $\boldsymbol{n}_i$ which are used for waypoint switching. Two define the sector the last $\boldsymbol{q}_{i-1}$, the current $\boldsymbol{q}_i$ and next segment $\boldsymbol{q}_{i+1}$ are defined as

$$\boldsymbol{q}_{i-1} = \frac{w_i - w_{i-1}}{|w_i - w_{i-1}|} \tag{5.77}$$

$$\boldsymbol{q}_i = \frac{w_{i+1} - w_i}{|w_{i+1} - w_i|} \tag{5.78}$$

$$\boldsymbol{q}_{i+1} = \frac{w_{i+2} - w_{i+1}}{|w_{i+2} - w_{i+1}|} \tag{5.79}$$

Based on the segments the two bisector vectors $\boldsymbol{n}_{i-1}$ and $\boldsymbol{n}_i$ can be defined

$$\boldsymbol{n}_{i-1} = \frac{\boldsymbol{q}_{i-1} + \boldsymbol{q}_i}{|\boldsymbol{q}_{i-1} - \boldsymbol{q}_i|} \tag{5.80}$$

$$\boldsymbol{n_i} = \frac{\boldsymbol{q_i} + \boldsymbol{q}_{i+1}}{|\boldsymbol{q_i} - \boldsymbol{q}_{i+1}|} \tag{5.81}$$

Based on this, waypoint switching to the next segment is defined as the following for the case shown in figure 5.21

$$p_{\mathbf{UAV}}(t) \in H(w_{i+1}, R_{\frac{\pi}{2}}\boldsymbol{n_i}) \tag{5.82}$$

$R_{\frac{\pi}{2}}$ here defines a rotation by 90° in counter-clockwise direction. The point from which the section is spanned P_s is then defined as

$$P_s = (w_i + a\boldsymbol{n}_{i-1}) \cap (w_{i+1} + b\boldsymbol{n}_i) \tag{5.83}$$

Based on this point P_s every position of the vehicle P_{UAV} is then scaled to the segment $\boldsymbol{q}_i$. With this approach a continuous behavior of the control parameters is ensured in and at the borders of the sector. For this projection the point P_σ is defined as

$$P_\sigma = (P_s + a\overline{P_s P_{\mathrm{UAV}}}) \cap (w_i + b\boldsymbol{q}_i) \tag{5.84}$$

With this formulation it is always possible to find a projection of the UAV position on either side of the path (shown in figure 5.21 in transparent and bolt). Based on the the position of P_σ then the percentile distance σ on the segment is then built as

$$\sigma = \frac{\overline{w_i w_{i+1}}}{\overline{w_i P_s}} \tag{5.85}$$

Based on σ now the parameters of the original trochoid curve should be evaluated online. Therefore the approximated time of the segment t_{app} has to be evaluated. The waypoints w_i are discredited with the distance inter-sampling distance Δt. Hence the waypoint w_i is evaluated at

$$t_{w_i} = i\Delta t \tag{5.86}$$

To built the approximated time of the current position, we state the following formulation as a sufficient approximation[6]

$$t_{app}(w_i, \sigma) = t_{w_i} + t_{sigma} = i\Delta t + \sigma\Delta t \tag{5.87}$$

Based on the approximated time t_{app} now all necessary parameters are built to calculate the necessary path following parameters. The position of the closest point on the path P_{cl} is given by

$$P_{cl}(t_{app}) = \begin{bmatrix} x(t_{app}) \\ y(t_{app}) \end{bmatrix} = \begin{bmatrix} \frac{V_a}{\delta_1 \omega} \sin(\delta_1 \omega t_{app} + \psi_0) + V_\omega t_{app} + x_0 \\ -\frac{V_a}{\delta_1 \omega} \cos(\delta_1 \omega t_{app} + \psi_0) + y_0 \end{bmatrix} \tag{5.88}$$

The tangent is given by

$$\hat{t}_{cl}(t_{app}) = \begin{bmatrix} \frac{dx}{dt}(t_{app}) \\ \frac{dy}{dt}(t_{app}) \end{bmatrix} = \begin{bmatrix} V_a \cos(\delta_1 \omega t_{app} + \psi_0) + V_\omega \\ V_a \sin(\delta_1 \omega t_{app} + \psi_0) \end{bmatrix} \tag{5.89}$$

The curvature at the closest point P_{cl} is then given by

$$\kappa(t_{app}) = \frac{V_a \delta_1 \omega \left(V_a + V_\omega \cos\left(\delta_1 \omega t_{app} + \psi_0\right)\right)}{\left(V_a^2 + 2V_a V_\omega \cos\left(\delta_1 \omega t_{app} + \psi_0\right) + V_\omega^2\right)^{1.5}} \tag{5.90}$$

To be able to sign the track error e later on, the path normal at the closest point on the path has to be defined. Based on (5.89) and (5.90) the path normal $\hat{n}$ can be defined as:

$$\hat{n}_{cl}(t_{app}) = \begin{bmatrix} -\mathrm{sign}(\kappa(t_{app}))\frac{dy}{dt}(t_{app}) \\ -\mathrm{sign}(\kappa(t_{app}))\frac{dx}{dt}(t_{app}) \end{bmatrix} \tag{5.91}$$

Based on the path following calculation above the necessary control inputs for the controller can be calculated per calculation cycle for

$$P_{\mathrm{UAV}} \in H(w_i, R_{\frac{\pi}{2}}\boldsymbol{n}_{i-1}) \ \wedge P_{\mathrm{UAV}} \notin H(w_{i+1}, R_{\frac{\pi}{2}}\boldsymbol{n}_i) \tag{5.92}$$

as

$$e = \mathrm{sign}(\hat{n}_{cl} \cdot \overline{P_{cl}P_{\mathrm{UAV}}})\overline{P_{cl}P_{\mathrm{UAV}}} \tag{5.93}$$

$$\kappa = \kappa(t_{app}) \tag{5.94}$$

$$\hat{t} = \hat{t}(t_{app}) \tag{5.95}$$

[6]The approximation used here is shown in appendix H

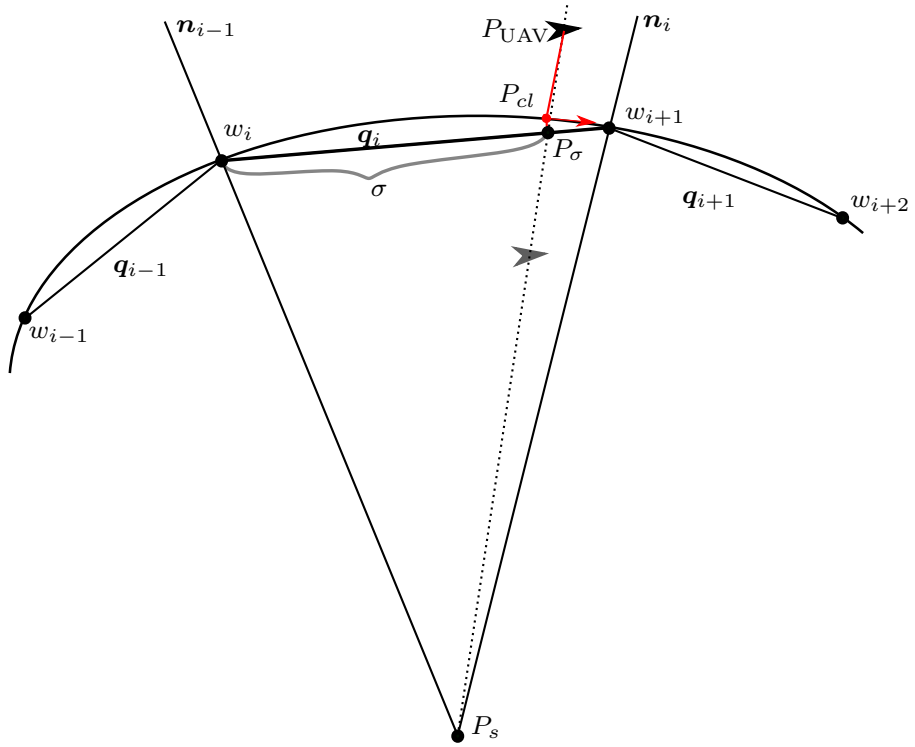


Figure 5.21: Setup of the scaled sigma approach. Sector which is used for waypoint switching is limited by the two bisector vectors $\boldsymbol{q}_{i-1}$ and $\boldsymbol{q}_i$. In red all the relevant control parameter values are shown for an example vehicle position.

Chapter 6

Simulations

In chapter 5 the simulations shown were implemented in a separate testbed to analyze the parameters for the different path following approaches and hence state a proof of concept. In order now to use the approaches in a autopilot setup, the implementation on such a system is shown. Therefore the approach shown in section 5.3 for trochoid segments is implemented in PX4. To test the implementation SIL[1] simulation with the simulation environment of Gazebo[2] is used. Finally the performance of the implementation is shown for various scenarios and use cases.

6.1 Implementation

In this section a brief overview of the software architecture and the specific adjustments done to implement the approach shown in section 5.3 for trochoids is shown. A brief overview of the high-level software architecture is given in figure 6.1. A more detailed overview of the software architecture and its full documentation can be found in [37].

As shown in figure 6.1 the flight plan is defined on the ground control station with QGroundControl[3]. Here the trochoid sections are planned and then the necessary parameters are assembled to a MAVLink[4] message. With this message the information is sent to the drone where it is then processed by the PX4 Flight Control. Here the online calculation approach and the respective path following policy presented in section 5.3 are implemented in the position controller. As mentioned throughout this work, the position controller used here is the NPFG controller described in [36] and implemented on the fixed-wing PX4 branch of the ASL[5].

The whole implementation from path planning in QGroundControl to the behavior of the position controller can then be tested with SIL in Gazebo for different setups

[1] Software in the Loop: Simulators allow PX4 flight code to control a computer modeled vehicle in a simulated "world". Interaction with this vehicle just as with a real vehicle, using QGroundControl, an offboard API, or a radio controller/gamepad is possible.

[2] Gazebo is a robotic simulation tool which allows to implement a variety of robotic systems to test. Gazebo is one of the recommended simulation environments by PX4: http://gazebosim.org/

[3] Full flight control and mission planning for MAVLink enabled drones: http://qgroundcontrol.com/

[4] Lightweight messaging protocol for communication with drones and between on-borad drone components: https://mavlink.io/en/

[5] NPFG controller implemented in the respective develop branch: https://github.com/ethz-asl/ethzasl_fw_px4/tree/develop.

such as varying wind conditions[6]. In section 6.1.1 - 6.1.2 the necessary adjustments to the different subsystems are described in more detail to show the main workflow of the new setup. Smaller adjustments are not covered here and can be directly seen in the documented code implementations.

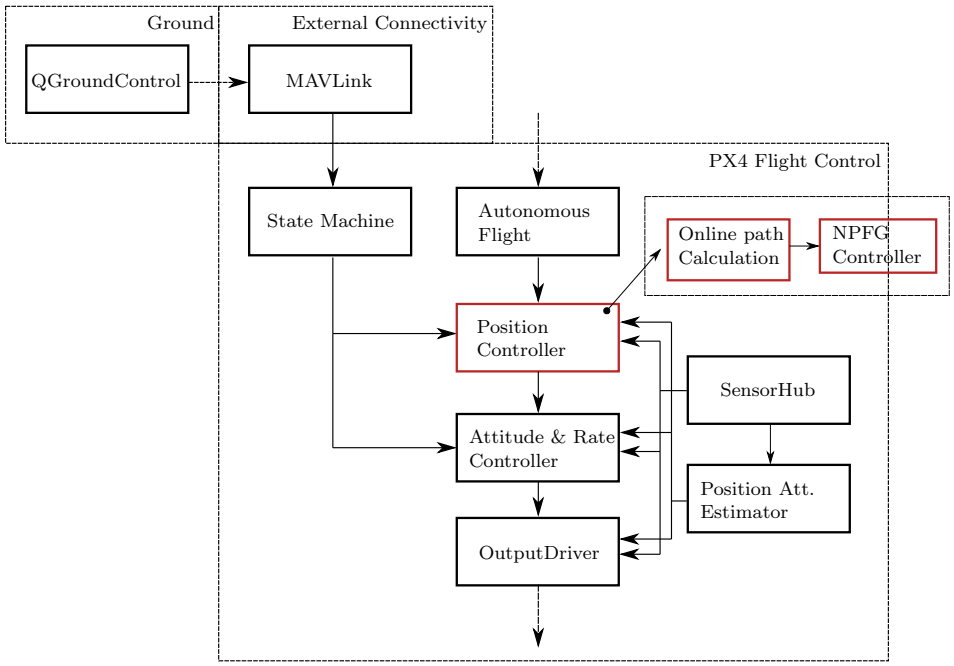


Figure 6.1: Overview of high-level software architecture of PX4.

6.1.1 QGroundControl & MAVLink

For mission planning QGroundControl is used. In this setup different maneuvers such as waypoint or loiter navigation can be chosen. For such maneuvers all necessary parameters have to be defined and are then uploaded to the UAV via MAVLink. To extend this setup with the possibility to navigate the UAV with a different maneuver, the setup has to be extended with a new waypoint type being either of trochoid or clothoid form. Therefore a single segment of the proposed path is defined as a waypoint as shown in figure 6.2.

[6]PX4 provides the ability to do different SIL simulations with a setup in Gazebo: https://docs.px4.io/master/en/simulation/gazebo.html

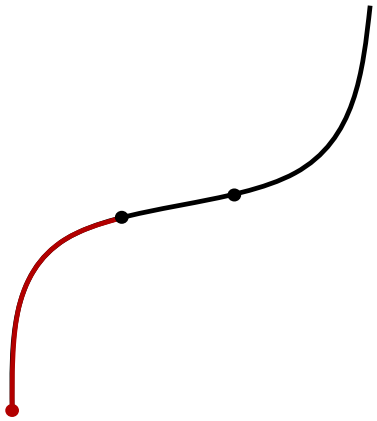


Figure 6.2: Single segment of the time-optimal path defined as waypoint type in QGroundControl.

To be able to send the parameters defining a trochoid segment to the UAV as shown in figure 6.2 a suitable MAVLink message for the transfer has to be defined. The message has to be able to transmit all necessary parameters defining a trochoid segment as shown in section 5.3.1. It is assumed to store the inter-waypoint sampling distance Δt as a fixed parameter since this parameter does not define the shape of the trochoid segment.

For the transmission of parameters the message `MAV_CMD_WAYPOINT_USER_1 (31000)`[7] is used. The message will be filled with the following parameters (3 pre-defined and 4 user-defined 32bit values)[8]:

Param(:Label)	**Description**	**Stored Value (32bit)**
1	Total Time	T [s]
2	Wind speed and angle	V_ω [m/s] / ψ_ω [°]
3	Airspeed and initial heading	V_a [m/s] / ψ_0 [°]
4	Oriented turn rate	$\delta\omega$ [rad/s]
5	Latitude	x_0
6	Longitude	y_0
7	Altitude	Altitude (MSL) [m]

With the implementation of the new waypoint type, a three-segment path approach of a CCC trochoid path is presented as shown in figure 6.3. Here all the values can be entered manually and be adjusted.

[7] https://mavlink.io/en/messages/common.html#MAV_CMD_WAYPOINT_USER_1

[8] Possible forms for the implementation of concatenated values of parameters 2 and 3 are shown in appendix I

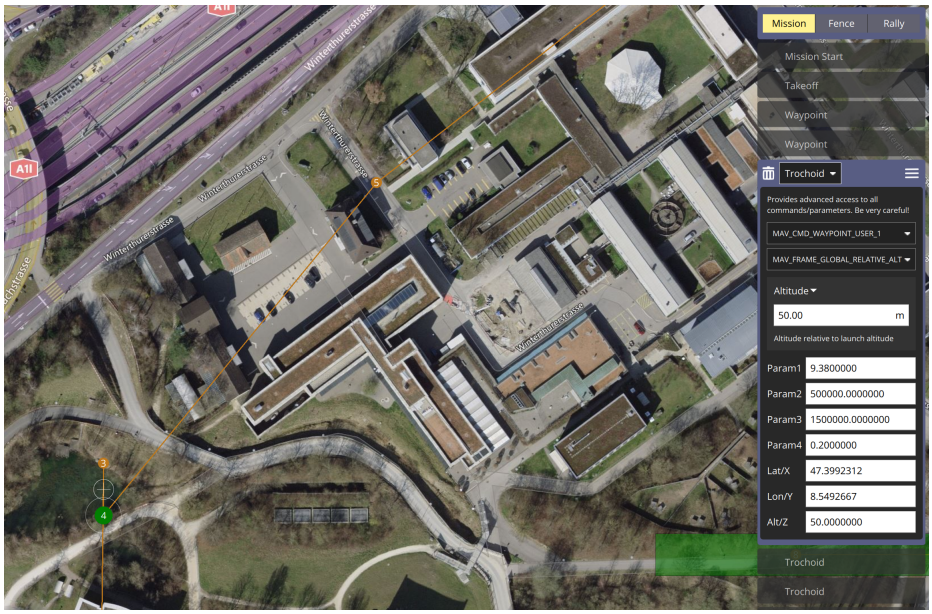


Figure 6.3: Example mission with a three segment trochoid path (CCC) represented by three separate trochoid waypoints in QGroundControl.

6.1.2 PX4

On-board the UAV the MAVLink message described in section 6.1.1 is sent to the PX4 Flight Control. As soon as a trochoid element is active in the mission plan, the respective parameters are extracted. The parameters are extracted in a way that they are accessible in the required form in the position controller[9].

In figure 6.4 the further implementations are shown. The parameters of the MAVLink message are used to calculate the current waypoints as shown in section 5.3.1. Based on the calculated waypoints the policy presented in section 5.3.2 can be used to calculate the current controller inputs at every controller cycle.

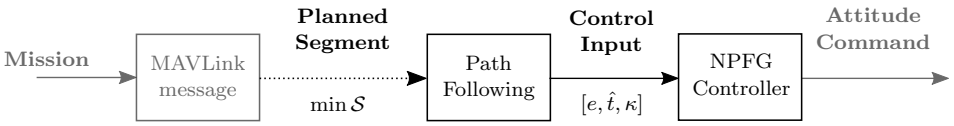


Figure 6.4: Implementation of the online calculation approach in the position controller of PX4.

6.2 Simulation Results

To test the performance of the implementations above, Software in the Loop (SIL) with Gazebo is used. PX4 allows for SIL[10], where the flight stack is run on computer and a physical simulation engine is used, in the case here Gazebo. With this setup, the whole implementation shown in figure 6.1 can be tested. As a physical simulation engine, Gazebo allows to simulate for different environmental conditions

[9]The population of the values through the different parts of the flight control and the hijacked values can be found in appendix J.

[10]Simulation with PX4: https://docs.px4.io/master/en/simulation/

such as wind which is of particular interest for the simulations considered throughout this work.

After simulations the recorded data can be extracted from the flight log file and being postprocessed with a suitable data analysis environment [11]. In the following subsections the results of various simulations are shown to highlight the performance of the implemented approach.

6.2.1 Path Following

The first simulation scenario investigates on the general path following ability of the implemented online-calculation approach both for clothoid and trochoid paths. Here these two main questions are answered which were imposed in the beginning of chapter 5:

- Is the UAV able to follow the proposed path in the given wind conditions?
- Is the calculation of the controller input continuous and to a certain degree smooth to ensure trackable attitude commands?

In order to show the performance of the implemented path sampling and path following approaches demanding path types are presented here. By demanding paths CCC path types with a lot of turning movement in all directions relative to the wind are referred to.

In figure 6.5 the trajectory simulation results for such a CCC trochoid path type is shown. In orange the planned path which is generated by the path planning framework is indicated. Along the path all waypoints which are calculated online on demand are shown in black. These waypoints are used to calculate the current control inputs as derived in section 5.3.2. The performance of the UAV in the simulation is then highlighted in blue.

In figure 6.5 it can be seen that the path following performance is fairly well in most parts along the path such for example the final pose can be reached closely. On the other hand for certain parts of the path the tracking performs not as expected which leads to large tracking errors. This is the case for the parts of the segments where the UAV starts to turn against the wind such as can be seen during the second segment. Due to the same reason, but also due to the jump in curvature for the transition between the second and the third segment, a major tracking error occurs. The refereed jump in curvature can also be seen in figure 6.6 on the bottom. This effect is due to the full bank assumption throughout the whole segment for the trochoid path type which leads to not exactly trackable paths due to the UAV roll dynamics. This issue was already covered in [1] extensively.

[11] The results shown in this work are post-processed with `pyulog` in `Python`: https://docs.px4.io/master/en/log/flight_log_analysis.html

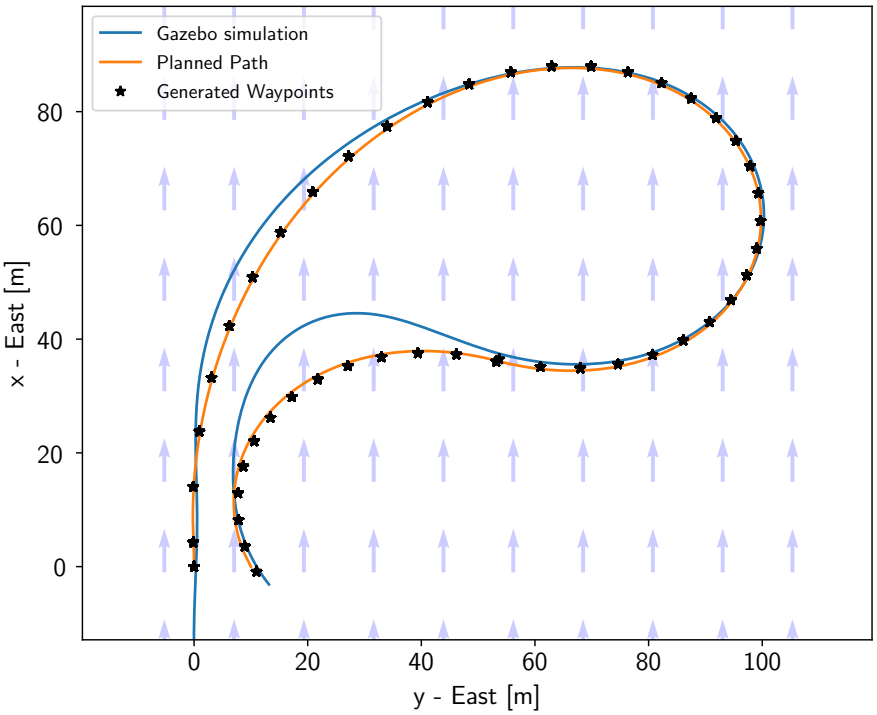


Figure 6.5: Simulation results of the online-calculation approach with a CCC trochoid path type. $V_a = 15m/s$, $V_\omega = 5m/s$ and $\psi_\omega = 0°$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Inter-sampling distance $\Delta t = 0.5s$.

To answer the second question stated in the beginning of this subsection the calculations of the controller inputs are considered. In figure 6.6 the controller inputs are displayed for the simulation shown in figure 6.5. As can be seen, all parameters have a continuous behavior throughout a segment. The only jumps that occur are in the exact moment, when the mission plan hands over the parameters for the next segment. This can be seen fairly well in the path curvature plot on the bottom. Here the two jumps, where the curvature value switches sign, refer to the transition point from the first to the second and respective from the second to the third segment. As the results in figure 6.5 show do this jumps not populate in to odd behaviors of the attitude of the UAV.

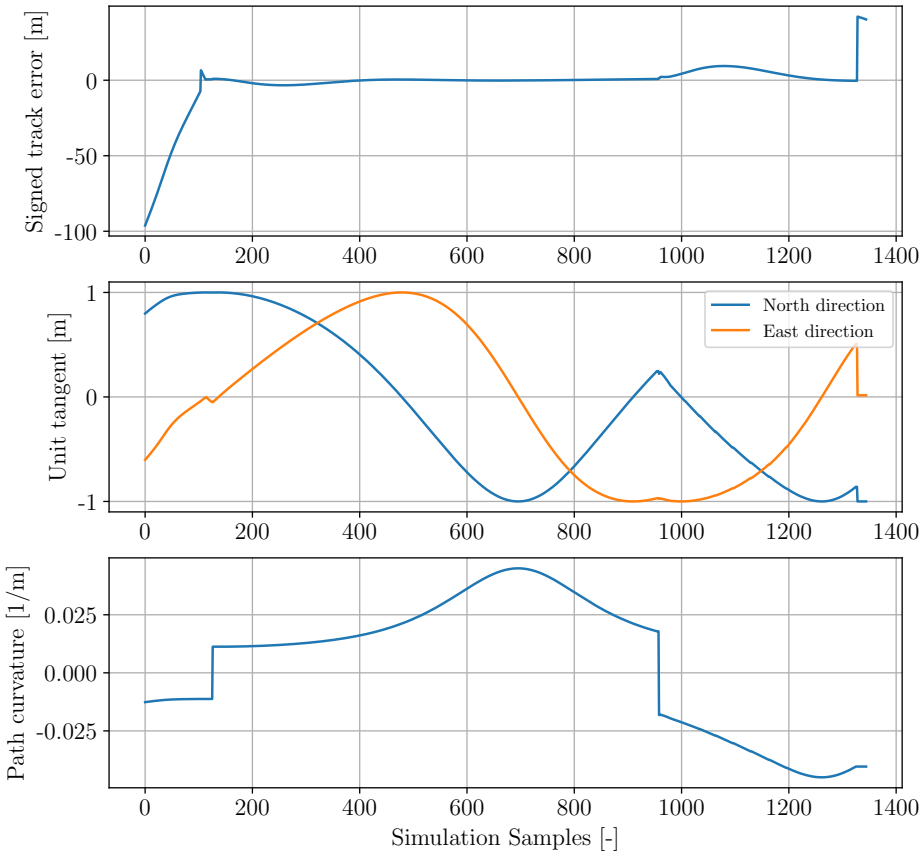


Figure 6.6: The controller input parameters e, $\hat{t}$ and κ for the simulation shown in figure 6.5. The parameters are calculated according to the policy presented in section 5.3.2.

In a similar fashion as above for a CCC trochoid path type here the results for a clothoid path of the same path type are shown. As can be seen in figure 6.7 the path type proposed with clothoid segments is not as time-optimal as the path built with trochoids as has been stated by [1]. This is based on the linear changing curvature which the clothoid geometry accounts for. Due to modeling of the roll dynamic effect of the UAV by path planning, the overall track error majorly decreased compared to the trochoidal path shown in figure 6.5. As for the trochoidal path, also here the largest deviations occur when the UAV is starting an maneuver towards the wind as can be seen after the two transition points between the segments. As stated

in section 5.3 for clothoid segments the integration of the position has to be done due to the geometrical definition of the clothoid form. The integration time here is chosen fairly low in a way that the waypoints exactly match the proposed path. Although such a precise calculation can lead to high computational effort which may cause problems depending on the frequency the position controller is run on.

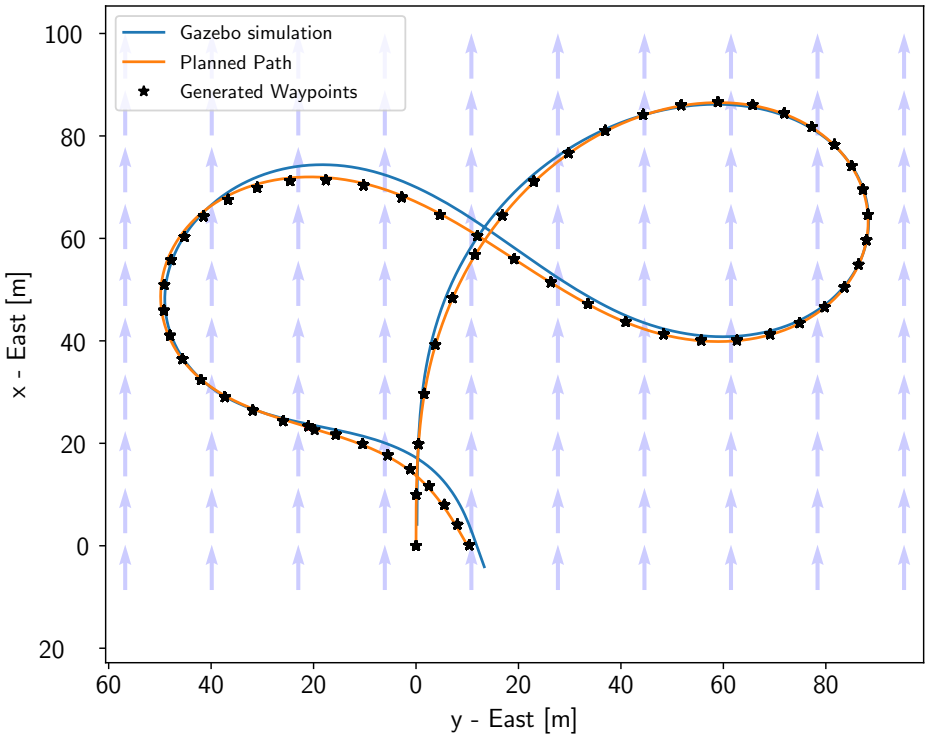


Figure 6.7: Simulation results of the online-calculation approach with a CCC clothoid path type. $V_a = 15m/s$, $V_\omega = 5m/s$ and $\psi_\omega = 0°$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Inter-sampling distance $\Delta t = 0.5s$ and integration time constant $t_{int} = 0.01s$.

As seen for the trochoid path also here the controller inputs have a continuous and mostly smooth behavior as can be seen in figure 6.8. Also here the only jumps in the parameter calculation occur when a segment is switched. Although here the jumps are mostly smaller due to the linear changing curvature which has no discrete jumps anymore. Especially this propriety makes an UAV more likely to accurately follow a clothoidal instead of a trochoidal path.

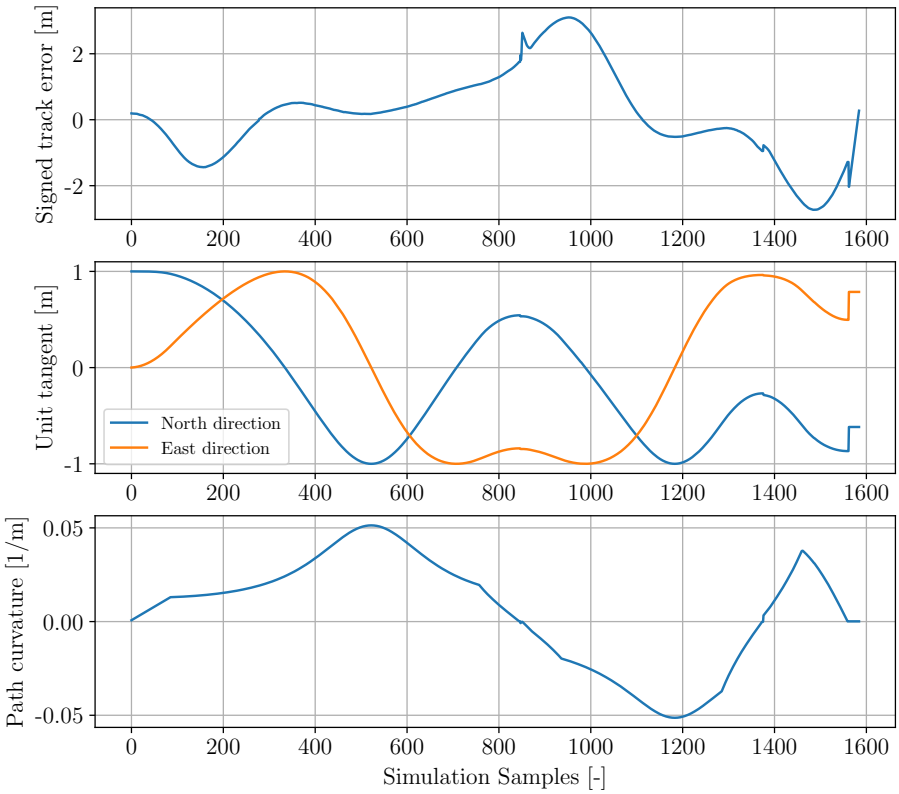


Figure 6.8: The controller input parameters e, $\hat{t}$ and κ for the simulation shown in figure 6.7. The parameters are calculated according to the policy presented in section 5.3.2.

To sum up what the simulations in this subsection have shown, two main conclusions can be made. If a time-optimal navigation between two poses is of main interest and not the exact tracking of the path between the two poses, then a trochidal path should be chosen. A trochoidal path proposes at all times the more time optimal path for the same path type than a clothoidal path. A specific use case where this might be the case is for a turn segment in a scanning mission. But on the other hand, if the exact tracking of the path is of main interest, as it may be the case for emergency landings or certain parts of a survey mission, then a clothoidal path should be chosen.

6.2.2 Robustness

The paths generated by the path planning framework are based on the current estimation of the wind as well as the heading of the wind. These estimations are

prone to uncertainty. Hence the generated paths by the framework should also be feasible if the real wind conditions diverge to a certain degree from the estimations on-board the UAV. To show the robustness against such uncertainties, the nominal wind conditions were altered for both the heading and the magnitude of the wind. The variations of these two parameters were limited by:

$$\Delta V_\omega = \pm 2 \frac{m}{s} \tag{6.1}$$

$$\Delta \psi_\omega = \pm 10° \tag{6.2}$$

Inside this boundaries the simulation conditions were altered arbitrarily. In figure 6.9 the results for these different simulation conditions are shown for the proposed path in figure 6.7. As can be seen in the figure also for varying wind conditions the path is fairly well trackable by the UAV. The largest track deviations occur in the regions as described in the subsection above. Although it has to be mentioned here: For ideal robustness against uncertainties in wind estimations, clothoidal paths are more beneficial than trochoidal paths due to their better modeling of the roll dynamics of the UAV.

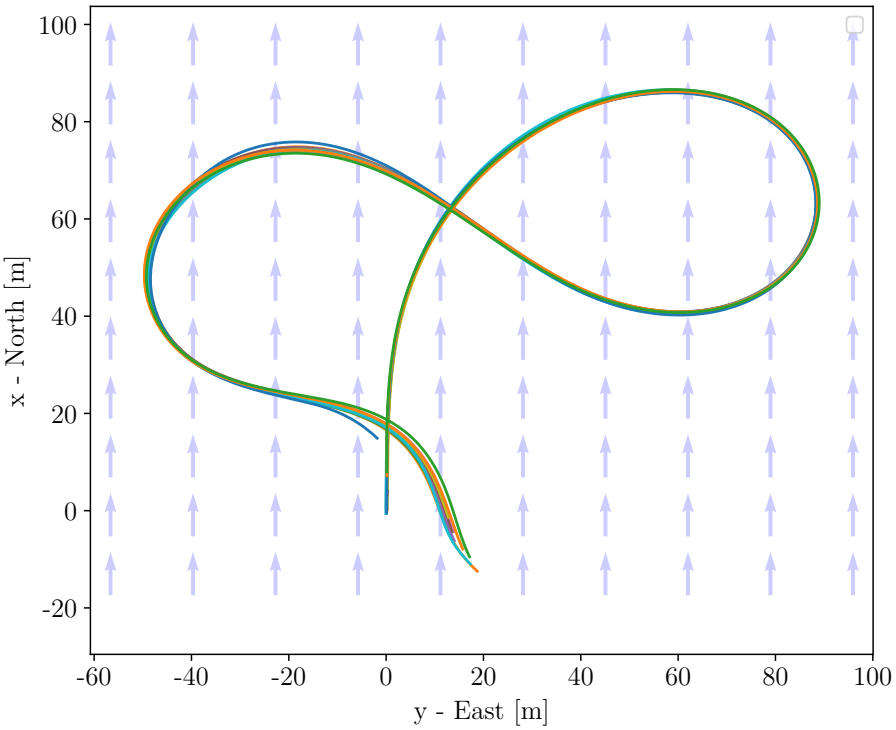


Figure 6.9: Results for simulations with varying wind magnitude and heading with the same setup as in figure 6.7. Variation in wind magnitude $\Delta V_\omega = \pm 2 \frac{m}{s}$ and in heading $\Delta \psi_\omega = \pm 10°$.

6.2.3 Use Case: Turning segments

In this subsection one specific example where the proposed paths can be used for optimization in UAV navigation is shown. One of the most common maneuvers for UAV is a turning segment. Such a maneuver occurs for example in survey missions at the boarder of the survey area on both sides as can be seen in figure 6.10. Here a turning maneuver by 180° is executed to reenter the survey area.

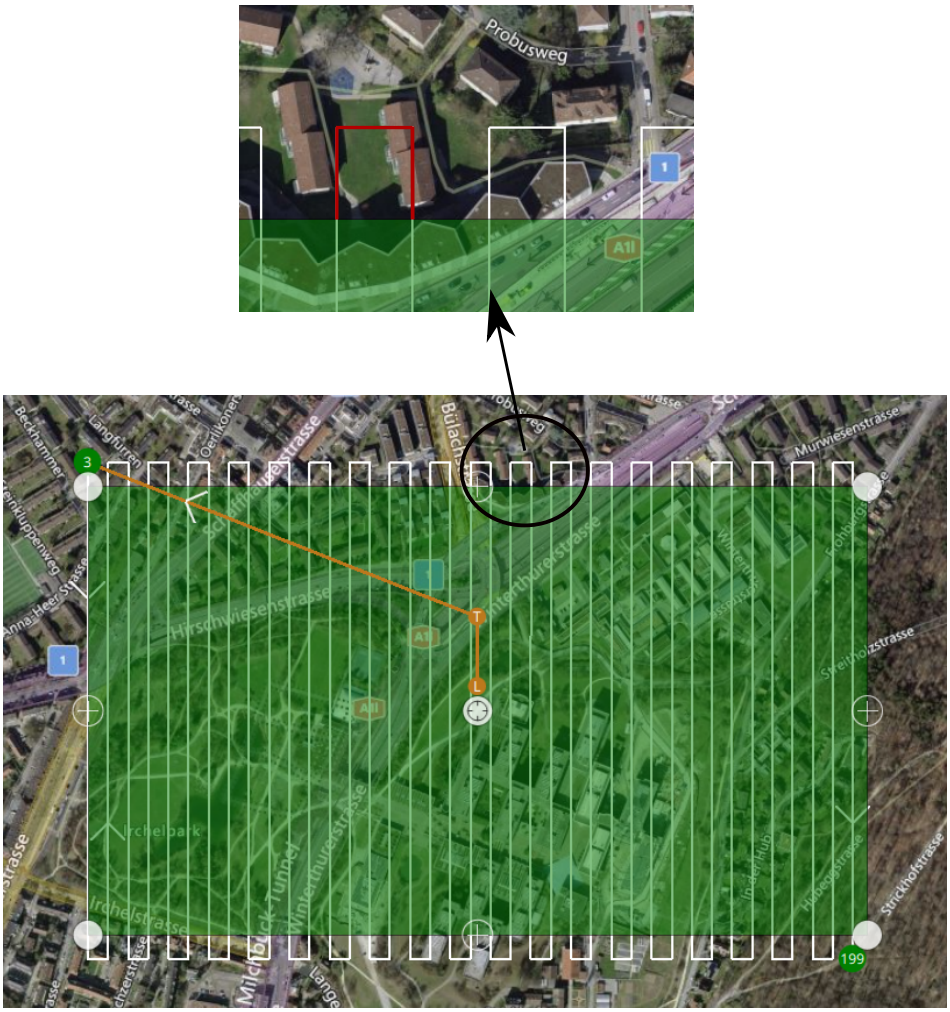


Figure 6.10: Survey mission planned in QGroundControl as a use case for a turning segment. One specific 180° turning segment is highlighted in red.

For such turning segments in survey missions planned in QGroundControl a simple approach is taken: Outside of the survey area, two additional waypoints are introduced which then represent the turning segment. This setup is also refereed to with figure 1.1 which was used for as introduction of this work. With this approach the UAV tries first to move towards the first and then to the second waypoint before returning back to the survey area. The behavior of such a turning segment can be seen in figure 6.11 where the simulation results in wind are shown. The simulation results for the waypoint following approach behave fairly well since the acceptance radius used to switch to the next waypoint is adapted to the current setup by the L_1 distance described in equation (5.31). Although the main problem with this approach is that beforehand the path of the UAV can not be predicted and is solely defined by the controller. Hence the controller only knows how to ideally reach the next waypoint but does not take into account the whole turning segment already at the beginning.

For the same set of turning points shown in figure 6.11 the path planning framework was used to generate a time-optimal trochoid path between the two poses. With this approach the whole point-to-point coordination of the turning segment is

considered and ideally shaped according to the current wind conditions. As can be seen in the figure the UAV is able to follow the proposed CSC trochoid path with a relatively low overall track error and reaches the final pose exactly. The latter is of main interest since the position has to be exact when reentering the survey area to ensure the quality of the surveyed data. Also the proposed path shows a more time-optimal path to navigate between the two turning points. Due to the number of turning segments on both ends of the survey area, the overall survey time can be decreased significantly. Furthermore this approach has another main advantage; the path to be flown is already known in advance. This is ideal if not only the time-optimal navigation between two poses is of relevance but also the performance of the tracking the path due to obstacles nearby. With the waypoint navigation approach such obstacles can also be taken into account but since the exact path is not known beforehand the safety margins have to be kept quite high.

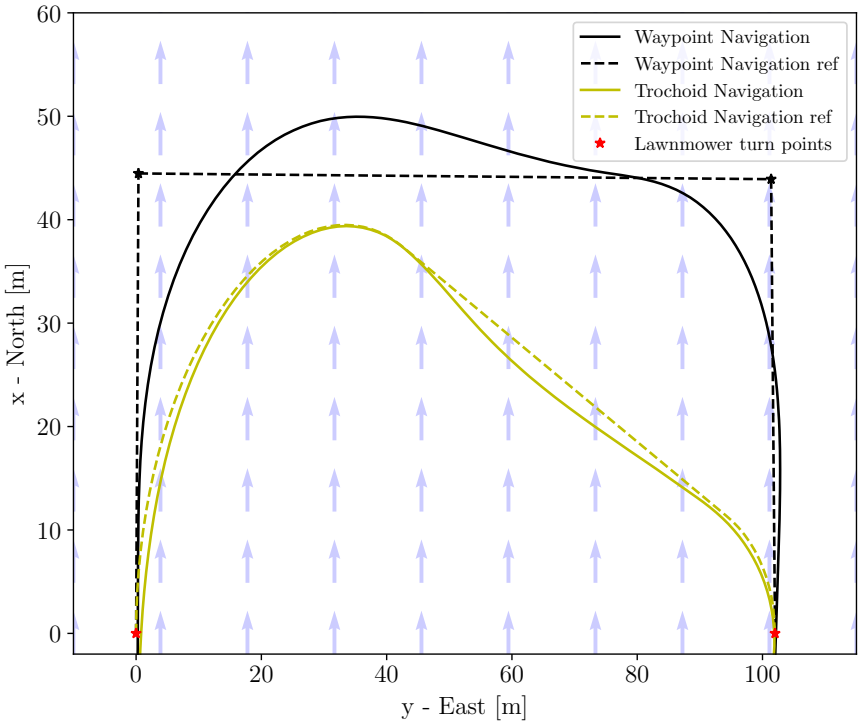


Figure 6.11: Comparison between the currently implemented waypoint navgation and the time-optimal trochoid path for a turning segment. The inner-distance between the lawnmower lines is $102m$. Airspeed is $V_a = 15\frac{m}{s}$, wind speed $V_\omega = 5\frac{m}{s}$ and wind heading $\psi_\omega = 0°$

In figure 6.11 the simulation was conducted with wind speed $V_\omega = 5\frac{m}{s}$. To show the performance for higher wind speeds which are closer to the airspeed of $V_a = 15\frac{m}{s}$ in figure 6.12 the wind speed is increased to $V_\omega = 10\frac{m}{s}$. Also for this simulation the the CSC trochoid path types shows good tracking behavior, reaches the final turn pose exactly and is more time-optimal than the waypoint navigation approach. Based on the results for robustness to uncertainty in the wind conditions shown in section 6.2.2 and the results shown here, trochoid paths generated by the path planning framework propose a time-optimal and robust path for turning segments in survey missions. As stated in section 6.2.2 if the tracking behavior between the two turning poses is of major interest due to nearby obstacles, instead of trochoid

paths, also clothoid paths can be used for better tracking behavior.

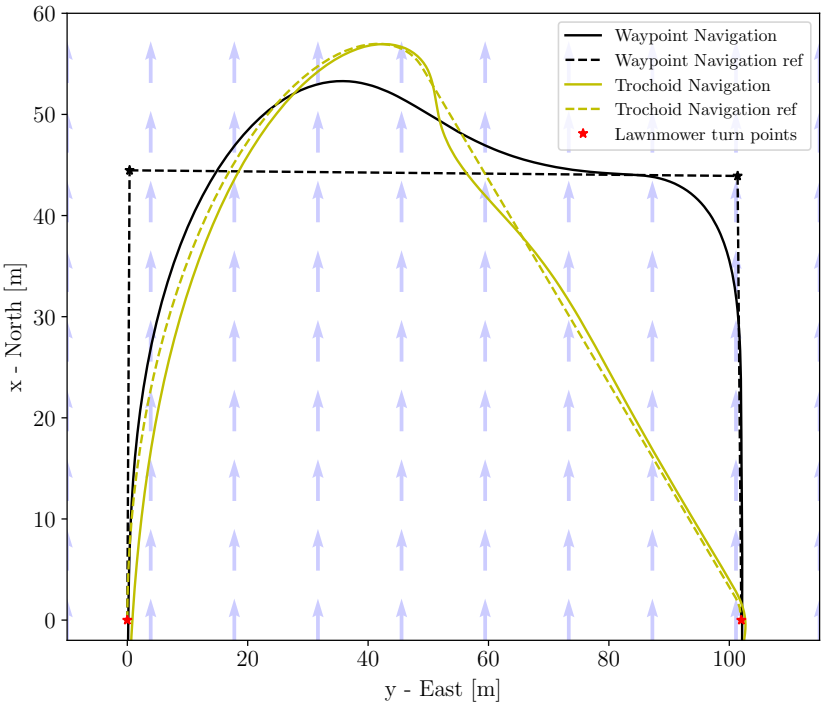


Figure 6.12: Comparison between the currently implemented waypoint navigation and the time-optimal trochoid path for a turning segment. The inner-distance between the lawnmower lines is $102m$. Airspeed is $V_a = 15\frac{m}{s}$, wind speed $V_\omega = 10\frac{m}{s}$ and wind heading $\psi_\omega = 0°$

Chapter 7

Outlook

7.1 Conclusion

With the results shown in this work as well as in [1] several conclusions can be drawn. Firstly a path planning framework was proposed which computes time-optimal paths for point-to-point coordination in 2D. The framework is written in a way that it can be deployed to an embedded system, hence a microcontroller on-board a UAV. This allows the UAV to do re-optimization of the current mission plans based on the updated wind conditions. A special focus was set on the implementation of the CCC path type for clothoids which is not yet described in literature. The approach presented here allows to solve for this path type in a simple but still computationally efficient way.

To be able to navigate along the proposed paths which are not standard in UAV mission planning and navigation two path sampling and respective path following approaches are presented. The pre-calculation approach tends to use more memory to represent the geometrical shape of the path but on the other hand, the respective path following approach can be used for all kinds of path types with the same logic. Hence the performance of the approach was not only shown for trochoid and clothoid paths but also for Bézier curves of 4^{th} order as an example. The latter path type was chosen for the comparison due to the similar behavior as the proposed three segment paths of the path planning framework. Furthermore Bézier curves are used across UAV path planning in literature (see section 2.1.2), allow for low computational operations in the path following policy due to their representation by polynomial expressions and hence are also suitable to use in optimization frameworks.
The second path sampling and following approach shows storage-wise a more suitable approach to use on-board a microcontroller. The performance of this approach was shown by implementing it in the PX4 software stack to show the ability to use it in state-of-the-art open source autopilot software. Through SIL simulation the implementation was tested and the tracking behavior of the UAV in different wind conditions examined.

With the approaches presented in this work, it is possible to improve the flight plans for small fixed-wing UAVs in wind in a robust manner. The improvements can be grouped in two main categories. The first improvement is faster paths to navigate in wind from point-to-point in 2D. The proposed paths guarantee a more time-optimal path than currently implemented approaches as shown for turning segments of survey missions in section 6.2.3. Furthermore the feasibility of paths

in wind is improved. Whereas current guidance controllers fully rely on tackling the wind effect on a low-controller level once wind occurs, the approach here pre-calculates the path in the current wind conditions and accesses its feasibility. Thus the ideal path is known pre-flight and is not solely defined based on the controller performance. This is furthermore beneficial when used with an obstacle avoidance policy on a higher level of the guidance and control structure of the UAV.

7.2 Further Procedure

The results in this work propose a point-to-point coordination for UAVs in 2D which is time-optimal and robust in the given wind conditions. The developed software was embedded and tested on a local system throughout the whole process. Based on the generated software and the results in this work, the following further procedure in this field of research may be suggested:

Flight validation
The online-calculation approach was implemented in the PX4 software stack. For this setup extensive SIL simulations with Gazebo have been done which show the performance of the implementation also for alternating conditions. Such a test setup already provides a sophisticated test environment for a high-level controller implementation. This is due to the fact that the general logic of the new implementation can be tested quite well and no low-level controllers such as the attitude controller are altered. Nevertheless real flight tests would give a better insight on the performance of small fixed wing UAVs for the proposed path types. Therefore the current implementations of the PX4 software stack have to be built on a microcontroller on-board of a UAV and tests can be conducted. The following questions arise once real flight tests are done:

- Do the assumptions used in the path planning tool resemble the UAVs roll dynamics well?
- How is the performance of the implemented online waypoint calculation and the respective path following approach?
- Is there a major difference between the real flight tests and the results shown in SIL with Gazebo?

Extension to 3D
For survey missions landing is a critical part in the navigation of a UAV. Unlike for the level flight part of the survey mission, here the vertical position of the UAV does not stay constant. Hence the extension to the third dimension could make the current path generation also appropriate for the use for landings or even more specifically for emergency landings, where mostly high winds are present. Here the presented approach could provide a time-optimal and more importantly feasible path for the current wind conditions in the horizontal plane. To extend to 3D the decoupled approach for the vertical plane is proposed. This is due to the reason that unlike for the horizontal plane, the wind conditions in the vertical plane are not constant and mostly consist of gusts. Since it is hard to account for such wind conditions here the wind should not be considered in the path planning stage and be left to tackle on a low-controller level once deviations due to wind occur. With a superposition of a vertical path planning approach, the time-optimal setup can be extended to 3D. Suitable extensions to 3D are extensively discussed in section 2.2 and shown in a tabular form in appendix A.

Extension to other path types
Both path following approaches presented in this work can also be used for different path types. A specific example was given in section 5.3.1 with an example of a Bézier curve of 4^{th} order. This path type allows for simpler calculations due its polynomial representation. Extensive tests of such path types might give a good insight if the path following policy can also be used for paths generated with different path planning or optimization frameworks for UAV paths. Here especially the advantages of such path types over clothoids and trochoids should be addressed.

Trochoid adjustment
Trochoids provide a simple and computationally efficient way to calculate paths for UAV in wind. One major disadvantage is the geometrical representation of trochoids. For each segment from the first moment of the segment on trochoids assume full bank angle. This leads to jumps in the curvature of paths generated with trochoid segments (see figure 6.6). Although small fixed-wing UAVs have fast roll dynamics such an assumption does not represent the UAVs capabilities. Hence in [1] the principle of robust trochoids was introduced. The basic idea and variations of robust trochoids can be found in appendix K. An other approach for the trochoidal paths proposed by the path planning framework of this work may be a smoother transition between the segments implemented in the path following logic. Since the next segment is already known from the mission plan, the convergence between the two segments at the transition point can be done smoothly. This would then reduce the non-trackable effect of discrete jumps in the controller inputs such as the current curvature of the path.

Wind Speed exceeding Air Speed
In this work only cases are considered for which the wind speed V_ω is smaller than the airspeed V_a. Cases for which this may be exceeded are emergency landing cases. A close analysis of the procedure for such cases based on the outcome of this work may be valuable.

Introducing Heuristics
For certain setups, it does not make sense to calculate some path types since they will be suboptimal or even infeasible. Based on heuristics, calculated from the setup parameters of the point-to-point coordination, some path types may be disregarded. This would lead to the advantage that these path types do not have to be considered for the whole process of the root-finding and hence the computation time can be decreased, which is key for the efficient implementation on a microcontroller. Also for low wind cases $V_\omega \leqslant 1\frac{m}{s}$ the nominal Dubins path calculation might be used, since the differences to the path types calculated in this work are not significant and the Dubins approach is computationally less expensive.

Bibliography

[1] Bucher Thomas, "Semester Project: Robust Wind-Aware Path Optimization On Board Small Fixed-Wing UAVs," Mar. 2021.

[2] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, Jan. 2020.

[3] R. T. Farouki, F. Pelosi, and M. L. Sampoli, "Approximation of monotone clothoid segments by degree 7 Pythagorean–hodograph curves," *Journal of Computational and Applied Mathematics*, vol. 382, p. 113110, Jan. 2021.

[4] J. Stephan, S. Notter, O. Pfeifle, F. Pinchetti, and W. Fichter, "Spline Trajectory Planning and Guidance for Fixed-Wing Drones," in *AIAA Scitech 2020 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020.

[5] F. Pinchetti, A. Joos, and W. Fichter, "Efficient continuous curvature path generation with pseudo-parametrized algebraic splines," *CEAS Aeronautical Journal*, vol. 9, no. 4, pp. 557–570, Dec. 2018.

[6] V. Girbés, G. Vanegas, and L. Armesto, "Clothoid-Based Three-Dimensional Curve for Attitude Planning," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1886–1898, Aug. 2019.

[7] G. Harary and A. Tal, "3D Euler spirals for 3D curve completion," *Computational Geometry*, vol. 45, no. 3, pp. 115–126, Apr. 2012.

[8] M. Brezak and I. Petrovic, "Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, Apr. 2014.

[9] G. Vanegas, F. Samaniego, V. Girbes, L. Armesto, and S. Garcia-Nieto, "Smooth 3D path planning for non-holonomic UAVs," in *2018 7th International Conference on Systems and Control (ICSC)*. Valencia: IEEE, Oct. 2018, pp. 1–6.

[10] N. Montes, M. C. Mora, and J. Tornero, "Trajectory Generation based on Rational Bezier Curves as Clothoids," in *2007 IEEE Intelligent Vehicles Symposium*. Istanbul, Turkey: IEEE, Jun. 2007, pp. 505–510, iSSN: 1931-0587.

[11] F. Samaniego, J. Sanchis, S. Garcia-Nieto, and R. Simarro, "Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs," *Electronics*, vol. 9, no. 1, p. 51, Dec. 2019.

[12] X. Wang, P. Jiang, D. Li, and T. Sun, "Curvature Continuous and Bounded Path Planning for Fixed-Wing UAVs," *Sensors*, vol. 17, no. 9, p. 2155, Sep. 2017.

[13] B. T. Ingersoll, J. K. Ingersoll, P. DeFranco, and A. Ning, "UAV Path-Planning using Bezier Curves and a Receding Horizon Approach," in *AIAA Modeling and Simulation Technologies Conference*. Washington, D.C.: American Institute of Aeronautics and Astronautics, Jun. 2016.

[14] D. G. Macharet, A. A. Neto, and M. F. M. Campos, "Feasible UAV Path Planning Using Genetic Algorithms and Bézier Curves," in *Advances in Artificial Intelligence – SBIA 2010*, A. C. da Rocha Costa, R. M. Vicari, and F. Tonidandel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6404, pp. 223–232, series Title: Lecture Notes in Computer Science.

[15] A. A. Neto, D. G. Macharet, and M. F. M. Campos, "Feasible path planning for fixed-wing UAVs using seventh order Bézier curves," *Journal of the Brazilian Computer Society*, vol. 19, no. 2, pp. 193–203, Jun. 2013.

[16] J.-w. Choi, R. Curry, and G. Elkaim, "Path Planning Based on Bezier Curve for Autonomous Ground Vehicles," in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. San Francisco, California, USA: IEEE, Oct. 2008, pp. 158–166.

[17] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *2007 46th IEEE Conference on Decision and Control*. New Orleans, LA, USA: IEEE, 2007, pp. 2379–2384.

[18] M. Owen, R. W. Beard, and T. W. McLain, "Implementing Dubins Airplane Paths on Fixed-Wing UAVs*," in *Handbook of Unmanned Aerial Vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Dordrecht: Springer Netherlands, 2015, pp. 1677–1701.

[19] V. Darbari, S. Gupta, and O. P. Verma, "Dynamic motion planning for aerial surveillance on a fixed-wing UAV," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL, USA: IEEE, Jun. 2017, pp. 488–497.

[20] P. Vana, A. Alves Neto, J. Faigl, and D. G. Macharet, "Minimal 3D Dubins Path with Bounded Curvature and Pitch Angle," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 8497–8503.

[21] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti, "Path Generation and Tracking in 3-D for UAVs," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 980–988, Jul. 2009.

[22] Yu Wang, Shuo Wang, Min Tan, Chao Zhou, and Qingping Wei, "Real-Time Dynamic Dubins-Helix Method for 3-D Trajectory Smoothing," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 730–736, Mar. 2015.

[23] P. Váňa, J. Sláma, and J. Faigl, "Surveillance planning with safe emergency landing guarantee for fixed-wing aircraft," *Robotics and Autonomous Systems*, vol. 133, p. 103644, Nov. 2020.

[24] S. Hota and D. Ghose, "Optimal path planning for an aerial vehicle in 3D space," in *49th IEEE Conference on Decision and Control (CDC)*. Atlanta, GA, USA: IEEE, Dec. 2010, pp. 4902–4907.

[25] Y. Liang, Y. Jia, J. Du, and J. Zhang, "Vector field guidance for three-dimensional curved path following with fixed-wing UAVs," in *2015 American Control Conference (ACC)*. Chicago, IL, USA: IEEE, Jul. 2015, pp. 1187–1192.

[26] P. Oettershagen, F. Achermann, B. Müller, D. Schneider, and R. Siegwart, "Towards Fully Environment-Aware UAVs: Real-Time Path Planning with Online 3D Wind Field Prediction in Complex Terrain," *arXiv:1712.03608 [cs]*, Dec. 2017, arXiv: 1712.03608.

[27] S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov Stability of a Nonlinear Path Following Guidance Method," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1718–1728, Nov. 2007.

[28] R. Curry, M. Lizarraga, B. Mairs, and G. H. Elkaim, "L+2, an improved line of sight guidance law for UAVs," in *2013 American Control Conference*. Washington, DC: IEEE, Jun. 2013, pp. 1–6.

[29] T. Stastny and R. Siegwart, "On Flying Backwards: Preventing Run-away of Small, Low-speed, Fixed-wing UAVs in Strong Winds," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 5198–5205.

[30] L. Techy, "Flight Vehicle Control and Aerobiological Sampling Applications," Nov. 2009, accepted: 2014-03-14T20:18:47Z Publisher: Virginia Tech.

[31] R. Rysdyk, "Course and Heading Changes in Significant Wind," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1168–1171, Jul. 2007.

[32] L. Techy, C. A. Woolsey, and K. A. Morgansen, "Planar path planning for flight vehicles in wind with turn rate and acceleration bounds," in *2010 IEEE International Conference on Robotics and Automation*. Anchorage, AK: IEEE, May 2010, pp. 3240–3245.

[33] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, p. 497, Jul. 1957.

[34] J.-D. Boissonnat, A. Cerezo, and J. Leblond, "Shortest paths of bounded curvature in the plane," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*. Nice, France: IEEE Comput. Soc. Press, 1992, pp. 2315–2320.

[35] L. Techy and C. A. Woolsey, "Minimum-Time Path Planning for Unmanned Aerial Vehicles in Steady Uniform Winds," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 6, pp. 1736–1746, Nov. 2009.

[36] Thomas Stastny, "Stability Considerations for Wind-Aware Path Following Guidance," Jan. 2021.

[37] "PX4 Architectural Overview | PX4 User Guide."

[38] M. Shanmugavel, A. Tsourdos, R. Zbikowski, and B. White, "3D Path Planning for Multiple UAVs Using Pythagorean Hodograph Curves," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. Hilton Head, South Carolina: American Institute of Aeronautics and Astronautics, Aug. 2007.

[39] S. Schopferer and T. Pfeifer, "Performance-aware flight path planning for unmanned aircraft in uniform wind fields," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. Denver, CO, USA: IEEE, Jun. 2015, pp. 1138–1147.

[40] A. A. Neto, D. G. Macharet, and M. F. M. Campos, "3D path planning with continuous bounded curvature and pitch angle profiles using 7th order curves," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany: IEEE, Sep. 2015, pp. 4923–4928.

[41] S. Benders, A. Wenz, and T. A. Johansen, "Adaptive Path Planning for Unmanned Aircraft Using In-flight Wind Velocity Estimation," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dallas, TX: IEEE, Jun. 2018, pp. 483–492.

[42] S. Benders, "Reconfigurable Path Planning for Fixed-wing Unmanned Aircraft Using Free-Space Roadmaps," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dallas, TX: IEEE, Jun. 2018, pp. 891–898.

[43] S. Benders, S. Schopferer, and A. Nawrath, "In-flight Kinematic Model Parameter Estimation and Adaptive Path Planning for Unmanned Aircraft," in *AIAA Scitech 2020 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020.

[44] A. Wolek and C. Woolsey, "Disturbance rejection in Dubins path planning," in *2012 American Control Conference (ACC)*. Montreal, QC: IEEE, Jun. 2012, pp. 4873–4878.

[45] ——, "Feasible Dubins Paths in Presence of Unknown, Unsteady Velocity Disturbances," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 782–787, Apr. 2015.

[46] S. Schopferer, J. S. Lorenz, A. Keipour, and S. Scherer, "Path Planning for Unmanned Fixed-Wing Aircraft in Uncertain Wind Conditions Using Trochoids," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dallas, TX: IEEE, Jun. 2018, pp. 503–512.

Appendix A

Tabular literature overview for path planning in 3D

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Time-optimal Paths for a Dubins airplane [17]	2007	Extension of the 2D Dubins approach to altitude. Glider slopes with unsaturated or saturated altitude velocity or / and helix.	no	Circular (Dubins)	Turn rate, angle of climb	no	no	Altitude difference is classified as low, medium or high	
Path Generation and Tracking in 3-D for UAVs [21]	2009	Dubins approach in horizontal space, approach with 3 subpaths for extension to 3D	no	Circular (Dubins)	Bounded curvature and angle of climb	Circa UAV (UAV Model), Dryden model (turbulences),	no	Path tracking approach with wind robustness	

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Optimal Path Planning for an Aerial Vehicle in 3D Space [24]	2010	Construction of CSC path in 3D by using a geometric approach	no	Arbitrary shape	Turn rate	not mentioned	no		Generate flyable paths for high pitch angles, real-time implementation
3D Path Planning for Multiple UAVs Using Pythagorean Hodograph Curves [38]	2012	Use spatial PH curves to produce paths with continuous curvature profiles in 3D	no	Spatial PH curves	Turn rate, angle of climb	MATLAB (Simulation Tool)	no	Path length not directly considered in the optimization, two phases of planning (flyable path and optimization)	
Implementing Dubins Airplane Paths on Fixed-Wing UAVs [18]	2014	3D Dubins airplane approach	no	Circular (Dubins)	Turn rate, angle of climb	6-DOF model in MATLAB / Simulink	[1]	Cases are defined to be low altitude, medium altitude and high altitude, Approach for a path manager is shown, initial and final pitch angles are not taken into account	tackle the decoupling of airspeed and flight-path angle

[1] GitHub-Repo: https://github.com/ntnu-arl/DubinsAirplane

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Performance-Aware flight path planning for unmanned aircraft in uniform wind fields [39]	2015	Superposition of 2D trochoids (horizontal plane) and 2D double integrator dynamics (vertical plane)	const. horizontal wind	Trochoids	Bank angle, inclination angle	Prometheus (UAV Model), 4D roadmap (path planning framework)	no		Implementation of the influence of airspeed changes in one dimension to the other dimension, better account for roll dynamics
Vector Field Guidance for Three-Dimensional Curved Path Following with Fixed-Wing UAVs [25]	2015	Combined vector field approach	const. horizontal wind	Arbitrary shape	Turn rate, angle of climb	not mentioned	no		

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Real-Time Dynamic Dubins-Helix Method for 3-D Trajectory Smoothing [22]	2015	Smoothing approach for superposition of 2D Dubins approach in the horizontal plane and smoothed glider slope in the vertical plane. Adding helix to meet altitude constraints.	no	Circular (Dubins)	Turn rate, angle of climb	not mentioned	no		
3D Path Planning with Continuous Bounded Curvature and Pitch Angle Profiles Using 7th Order Curves [40]	2015	Continous Bézier curves of 7th order in 3D	no	Bézier Curve	Turn rate, angle of climb	not mentioned	no	Comparison to other data from similar approaches, gives good approximations to CSC Dubins curves but not to CCC curves	Better results for short distance scenarios, extension to 3D routing scenario

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Dynamic Motion Planning for Aerial Surveillance with a Fixed-Wing UAV [19]	2017	Superposition of Dubins approach in both planes	no	Circular (Dubins)	Turn rate, angle of climb	Sig Rascal (UAV Model) in JBSim. Simulation of sensors, autopilot and control systems with ArduPilot Platform.	no	Works in a partially known and updated urban environment.	Integration of sensors to impart perception ability
Towards Fully Environment-Aware UAVs: Real-Time Path Planning with Online 3D Wind Field Prediction in Complex Terrain [26]	2017	Maximum 6 segement approach in 3D. Dubins path in horizontal plane and simple climbing angle calculation in vertical plane.	TV 3D wind fields	Shifted Dubins version	Turn rate, angle of climb	not mentioned	no		Better wind field estimation/data,
Smooth 3D path planning for non-holonomic UAVs [9]	2018	3D clothoids Approx. by Rational Bézier Curves of order 13 optimized with a gradient-descent algorithm	no	Approx. Clothoids	Turn rate, angle of climb	Flight Gear 2018 (flight tool), MATLAB (mathematical software)	no	Additional kinematic control approach	Machine learning to generate a family of RB curves approximation unitary 3D clothoids in a efficient way

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
Adaptive Path Planning for Unmanned Aircraft Using In-flight Wind Velocity Estimation [41]	2018	Superposition of 2D trochoids (horizontal plane) and 2D 4th degree polynomial dynamics (vertical plane)	const. horizontal wind	Trochoids	Turn rate, angle of climb	Skywalker X8 (UAV model), MATLAB / Simulink (mathematical software)	no	Turbulence simulation with Dyren wind model, gust estimation for saftey margins,	Metric for safety distance as a function of estimated turbulence, use of time-varying wind and weather forecast, flight test
Re-configurable Path Planning for Fixed-wing Unmanned Aircraft Using Free-Space Roadmaps [42]	2018	Superposition of 2D trochoids (horizontal plane) and 2D 4th degree polynomial dynamics (vertical plane)	const. horizontal wind	Trochoids	Turn rate, angle of climb	Prometheus and Explorer (UAV Models),	no	Only considers CSC paths, bank angle smoothing in post-processing	Improve non-uniform sampling, different geometries than trochoids, re-planning due to parameters changes during flight

Paper	Year	Approach	Wind	Geometry	UAV Limitations	Implement.	Repo	Comment	Further Work
In-flight Kinematic Parameter Estimation and Adaptive Path Planning for Unmanned Aircraft [43]	2020	Global path planner with trochoid approach to wind in horizontal plane	const. horizontal wind	Trochoids	Turn rate, angle of climb	ArduPlane autopilot and flight simulation using the JSBSim dynamic flight simulation. Most implementations in C++ and use the ROS framework	no	Addresses mainly to parameter estimation during flight. Approach to trigger a replanning of the current path based on estimated parameters	Real-world tests, improve triggering of replanning, better decoupling of the estimation of flight parameters
Minimal 3D Dubins Path with Bounded Curvature and Pitch Angle [20]	2020	Decoupling approach with decoupled Dubins approach in both planes	no	Circular (Dubins)	Bounded curvature and angle of climb	Simulation in Julia 1.2	[2]	Comparison to Real-time Dynamic Dubins-Helix (RDDH) approach, local iterative optimization of initial path, lower and upper bound on optimal path length	Other curvature parametrization (due to abrupt lateral acceleration)

[2]GitHub-Repo: https://github.com/comrob/Dubins3D.jl

Appendix B

Survey Paper Proposal

Due to the extent of the literature considered in [1] and in this work, a possible survey publication for path planning in wind for fixed-wing UAVs was proposed. The proposal on the two following pages gives an overview of a general outline of the paper, a possible structure of the publication, and suggests a suitable journal targeting the field of research.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Paper Proposal

V0.1 - 07.07.2021

Administrative Information

Topic:	Survey on wind-aware path planning for fixed-wing UAVs
Type:	Survey paper
Title:	TBD
Journal:	Journal of Guidance, Control, and Dynamics
Journal Guidelines:	Author Guidelines of JGCD
Student:	Thomas Bucher (thbucher@student.ethz.ch)
Supervisors:	Thomas Stastny (thomas.stastny@mavt.ethz.ch, Sebastian Verling (sebastian.verling@wingtra.com)
Workload:	2 weeks
Draft revision:	TBD
Submission:	TBD

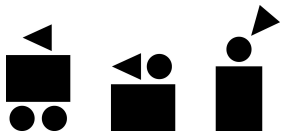
Outline

The paper gives a broad survey on path planning in wind for fixed-wing UAVs.

Current approaches to tackle wind are done on low-level controllers. This still might lead to large deviation, mainly because the low-level controller only reacts to deviations when they appear. This leads to large path deviation which might lead to violation of no-fly zones or collisions with objects. Also for survey missions with UAV which mostly need high accuracy to be able to use the recorded data large deviations decrease the quality. To tackle this problem, already at the path planning stage the wind can be taken into account to prevent large deviations from the planned path. Exactly this is the main focus of the paper.

First the mathematical basics of the topic are established. Then a broad literature review on 2D path planning in the horizontal plane is presented. The different parts in this section will be addressing different subtopics in the field such as path planning without wind, with steady wind or with uncertainties. Afterwards a specific insight into clothoids and trochoids will be given.

In a next part, the strategies to extend the wind-aware path planning to 3D are presented. Here the main focus lays on the different strategies to extend the planning to the vertical dimension. The literature review here is kept concise and is shown in a tabular form.



As last part of the paper literature addressing use cases such as emergency landing and lawnmower paths for survey missions is presented. Here the main focus lays on 2D applications.

Structure

In table 1 the main structure of the paper and the extent of the single parts is shown.

Part	**Topic**	**Description**
1	Problem Outline	Difference of tackling wind in low-level control and path planning. Establish main focus of the paper. Show possible problems with path deviations for UAVs.
2	Mathematical Background	Establish UAV model and path planning approach. Keep short, additional parts to appendix or other literature.
3	Literature without wind (2D)	
4	Literature with steady wind (2D)	
5	Literature with wind uncertainties (2D)	
6	Literature on clothoids / trochoids (2D)	
7	Extension to 3D	
8	Use cases	
9	Conclusion and further research interest	
10	Appendix	

Table 1: Structural overview of the paper

Appendix C

Software Framework

Due to one of the main objectives of this work, to be able to do re-optimization onboard of UAV, computational efficiency is key. Due to this reason and the goal to implement the software on a microcontroller on the UAV, C++ is a suitable environment for single core performance. Since during this work, only the implementation in a test framework and not already the implementation to a microcontroller is of main interest, a suitable test environment has to be set up. Due to the easy handling and functional graphical output capabilities, this part of the framework is realized in Python.

The general framework setup will here only be discussed briefly to give a general overview. A graphical overview of the software framework can be found in Figure C.1. A more detailed documentation of the different software parts can be found on the project repository[1] or directly in the code files. As stated above, the main functionalities of the framework are implemented in C++. To test the functionalities in Python with a suitable graphical interface, pybind11[2] was used to create Python bindings of the C++ code.
To implement the core part of the software in C++, the functionalities are split in different classes in an object-oriented way. The *Path* object stores the path types, path variables such as t_a, t_b or T, waypoint sampling of the path $x(t)$, $y(t)$, $\psi(t)$ and other properties, which can be assigned to a path. Such an object is the output for every feasible path type that is obtained by the solver. In the *Problem* object, all variable parameters for the path planning problem can be defined, such as initial and final position (x_0 / x_f, y_0 / y_f, ψ_0 / ψ_f), vehicle parameters ($\bar{\phi}$, $\dot{\bar{\phi}}$, V_a) and environmental parameters (V_ω, ψ_ω). This object builds the basis for the *Solver* object. There are two separate solvers implemented, one for the solutions with trochoids and one with clothoids for the turning segment. As a result of either of the solver, an overview of the results, the indication of the time-optimal path type and a *Path* object of all feasible paths are accessible. In *Utils* and *Math* functions are located, that are used in the core functionalities, but have to be imported separately or functions that are defined outside of the main classes. The basic structure and part of the implementation of the trochoid solver rely on an existing implementation on GitHub[3].

[1] Project Repository on GitHub (Access only if permission): https://github.com/thomasbuchersw/uavpathplaner.
[2] pybind11 repo on GitHub: https://pybind11.readthedocs.io/en/stable/index.html.
[3] ConvectedDubins by arturwolek: https://github.com/robotics-uncc/ConvectedDubins.

With pybind11 the binding definitions are located in a single file and the functions can be used in Python simply by importing the created package. In Python a wide variety of setups can be tested in different files, reaching from testing single solver setups towards dynamic simulation of an UAV for the created paths.

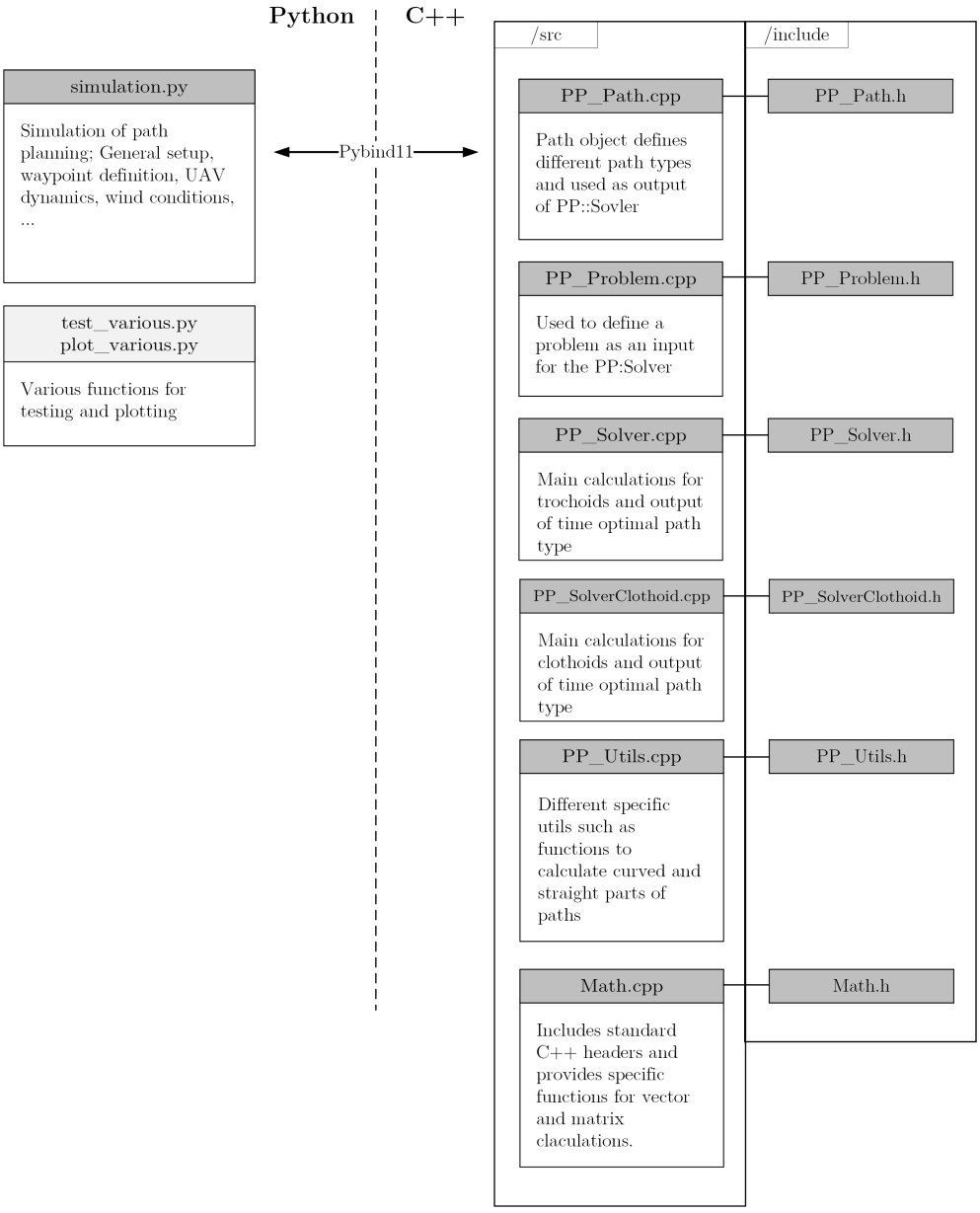


Figure C.1: Overview of the implementation framework consisting of a python and a C++ part.

Appendix D

Derivation alternative CCC clothoid forms

As described in section 4 and shown in table 4.1 there are other variations of the CCC clothoid. The basic structure of the derivation follows the form which is shown in section 4.1.2 for the R-R-R case. With the reaching (R) and non-reaching (NR) of the maximal bank angle for the single segments of the path the expressions for the heading $\psi_i(t)$ and the heading angle at the transition points ψ_{i0} change as can be seen in section 3.3. This changes and the following derivation of the different forms are shown in the upcoming sections. Therefore only the explicit changes to the form shown in section 4.1.2 are shown instead of rewriting the whole structure. In table D.1 all the variations for the CCC clothoid path type are shown and the reference to the respective derivation is given.

1. Segment	**2. Segment**	**3. Segment**	**Derivation**
R	R	R	Section 4.1.2
NR	R	R	Section D.1
R	NR	R	Section D.2
NR	NR	R	Section D.3
R	R	NR	Section D.4
NR	R	NR	Section D.5
R	NR	NR	Section D.6
NR	NR	NR	Section D.7

Table D.1: Possible combinations for the different clothoid segments

D.1 NR-R-R

For this case the first segment changes to the case where the maximum bank angle is not reached. Hence $\psi_1(t)$ changes to (5.58). Hence also the heading ψ_{20} at the first transition point changes:

$$\psi_{20} = \alpha\delta_1\bar{u}\frac{t_A^2}{4} + \psi_0 \tag{D.1}$$

Using ψ_{20} in (4.27) t_B becomes:

$$t_B = \frac{t_A^2}{4t_1} - \frac{\psi_f - \psi_0}{\alpha\delta_2\bar{u}t_1} + T \tag{D.2}$$

D.2 R-NR-R

Here $\psi_2(t)$ changes to (5.58) hence the heading for the transition point changes:

$$\psi_{30} = \alpha\delta_2\bar{u}\frac{t_B^2}{4} + \psi_{20} \tag{D.3}$$

Based on this t_B is[1]:

$$t_B = 2\sqrt{|Tt_1 - 2t_1^2 + t_1t_A + \frac{\psi_f - \psi_0}{\alpha\delta_2\bar{u}}|} \tag{D.4}$$

D.3 NR-NR-R

Here $\psi_1(t)$ and $\psi_2(t)$ change to (5.58) hence the heading for the transition points change:

$$\psi_{20} = \alpha\delta_1\bar{u}\frac{t_A^2}{4} + \psi_0 \tag{D.5}$$

$$\psi_{30} = \alpha\delta_2\bar{u}\frac{t_B^2}{4} + \psi_{20} \tag{D.6}$$

Based on this t_B is[1]:

$$t_B = \sqrt{|4Tt_1 + t_A^2 - 4t_1^2\frac{4(\psi_f - \psi_0)}{\alpha\delta_2\bar{u}}|} \tag{D.7}$$

D.4 R-R-NR

Here $\psi_3(t)$ changes to (5.58) hence the heading for the transition point changes:

$$\psi_f = \alpha\delta_3\bar{u}\frac{T^2}{4} + \psi_{30} \tag{D.8}$$

Based on this t_B is:

$$t_B = \frac{T^2}{4t_1} + t_A + \frac{\psi_f - \psi_0}{\alpha\delta_2 t_1\bar{u}} \tag{D.9}$$

D.5 NR-R-NR

Here $\psi_1(t)$ and $\psi_3(t)$ change to (5.58) hence the heading for the transition points change:

$$\psi_{20} = \alpha\delta_1\bar{u}\frac{t_A^2}{4} + \psi_0 \tag{D.10}$$

$$\psi_f = \alpha\delta_3\bar{u}\frac{T^2}{4} + \psi_{30} \tag{D.11}$$

Based on this, t_B is:

$$t_B = t_1 + \frac{T^2}{4t_1} + \frac{t_A^2}{4t_1} + \frac{\psi_f - \psi_0}{\alpha\delta_2 t_1\bar{u}} \tag{D.12}$$

[1]Negative solutions were omitted directly.

D.6 R-NR-NR

Here $\psi_2(t)$ and $\psi_3(t)$ change to (5.58) hence the heading for the transition points change:

$$\psi_{30} = \alpha\delta_2\bar{u}\frac{t_B^2}{4} + \psi_{20} \tag{D.13}$$

$$\psi_f = \alpha\delta_3\bar{u}\frac{T^2}{4} + \psi_{30} \tag{D.14}$$

Based on this t_B is[1]:

$$t_B = \sqrt{|T^2 - 4t_1^2 + 4t_1t_A + \frac{4(\psi_f - \psi_0)}{\alpha\delta_2\bar{u}}|} \tag{D.15}$$

D.7 NR-NR-NR

Here $\psi_1(t)$ - $\psi_3(t)$ change to (5.58) hence the heading for the transition points change:

$$\psi_{20} = \alpha\delta_1\bar{u}\frac{t_A^2}{4} + \psi_0 \tag{D.16}$$

$$\psi_{30} = \alpha\delta_2\bar{u}\frac{t_B^2}{4} + \psi_{20} \tag{D.17}$$

$$\psi_f = \alpha\delta_3\bar{u}\frac{T^2}{4} + \psi_{30} \tag{D.18}$$

Based on this t_B is[1]:

$$t_B = \sqrt{|T^2 + t_A^2\frac{4(\psi_f - \psi_0)}{\alpha\delta_2\bar{u}}|} \tag{D.19}$$

Appendix E

Test Cases CCC Solver

E.1 Case 2

$$
\begin{aligned}
x_0 &= 0m \\
y_0 &= 0m \\
\psi_0 &= 0^\circ \\
x_f &= 381.686m \\
y_f &= -233.629m \\
\psi_f &= 0^\circ
\end{aligned}
\qquad
\begin{aligned}
V_a &= 20m/s \\
V_\omega &= 5m/s \\
t_A &= 5s \\
T &= 12s
\end{aligned}
$$

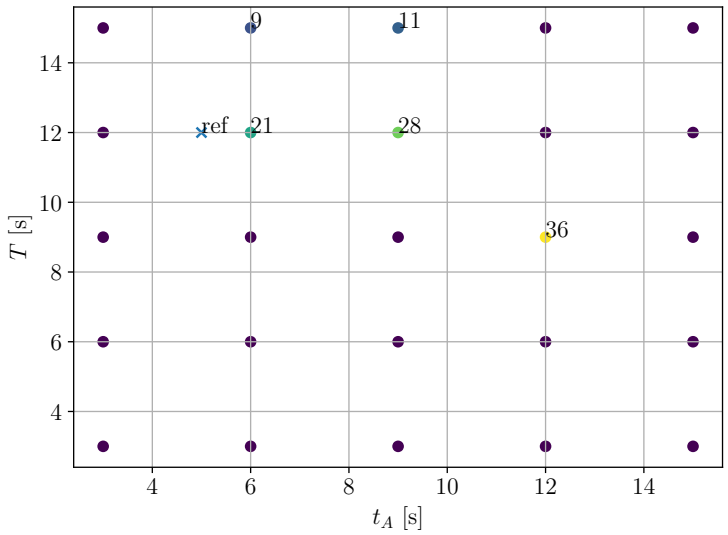


Figure E.1: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown.

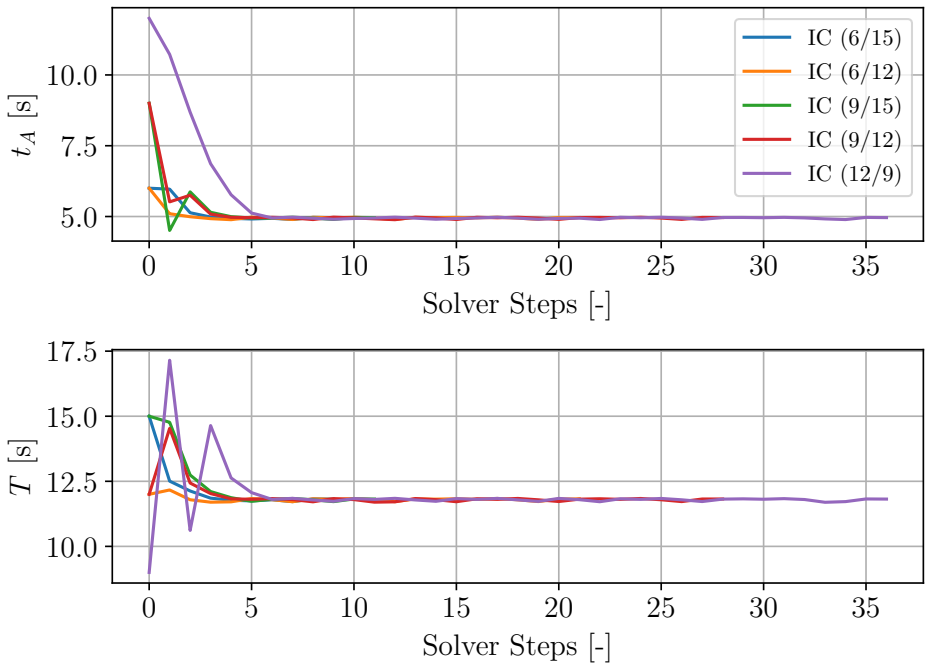


Figure E.2: Behaviour of all converging ICs for test case 2.

E.2 Case 3

$$
\begin{aligned}
x_0 &= 0m \\
y_0 &= 0m \\
\psi_0 &= 0° \\
x_f &= 487.377m \\
y_f &= 0.0911375m \\
\psi_f &= 0°
\end{aligned}
\qquad
\begin{aligned}
V_a &= 20m/s \\
V_\omega &= 5m/s \\
t_A &= 7s \\
T &= 7s
\end{aligned}
$$

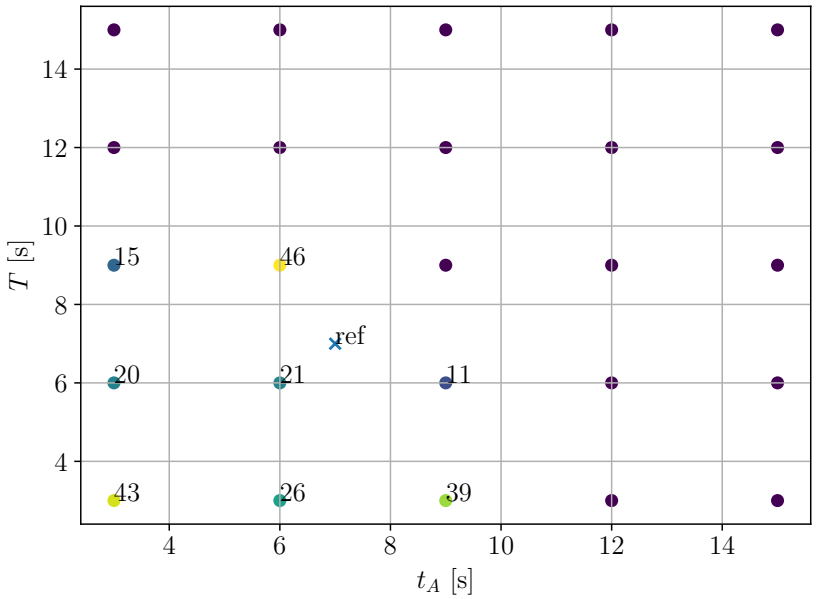


Figure E.3: Convergence of the solver for the different initial conditions for t_A and T. If the initial condition converged the number of solver steps for the convergence are shown.

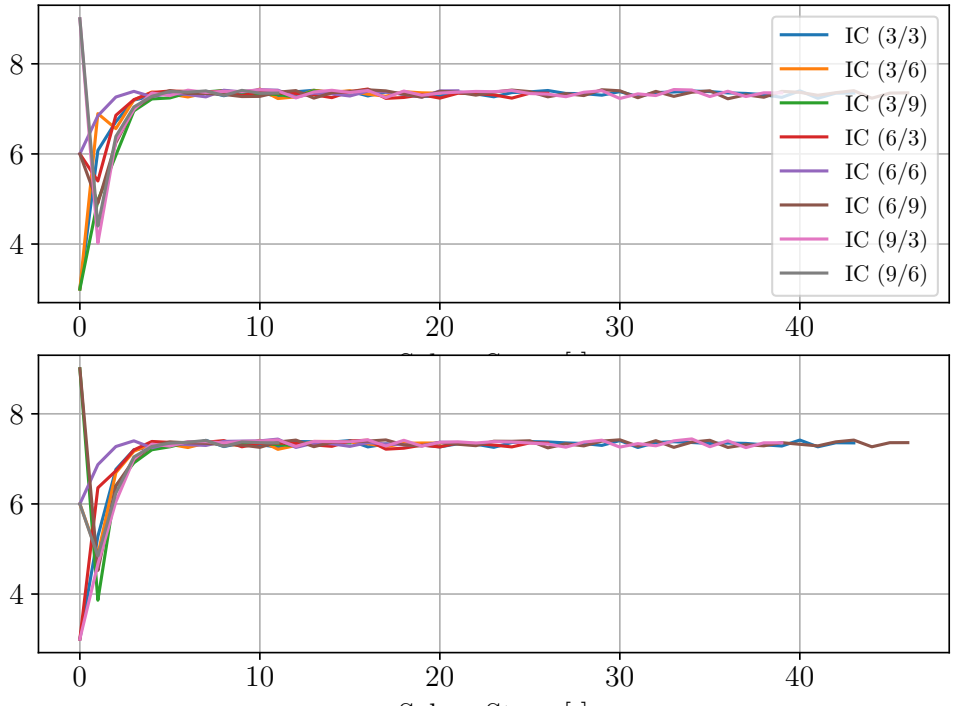


Figure E.4: Behaviour of all converging ICs for test case 3.

Appendix F

Approximation used in linear interpolation approach

The goal for the control inputs e, κ and $\hat{t}$ in path following is that their behaviour is smooth, but still the calculation for these parameters should be simple since their computation has to be done every at every controller loop. This leads to a certain approximation that is used for the approach presented in section 5.1.3. For the assumptions taken for this method, at every sampled point the positional continuity is ensured. The continuity is not valid for the orientation. This leads from the fact that simple circular segments are used to approximate the curve between two sampled points. In figure F.1 the setup for a situation for a switch between two segments is illustrated. As can be seen the last tangent of the first segment as well as the first tangent of the next segment are calculated bot based on a circular segment spanned with

$$R_{i+1} = \frac{1}{\kappa_{i+1}} \tag{F.1}$$

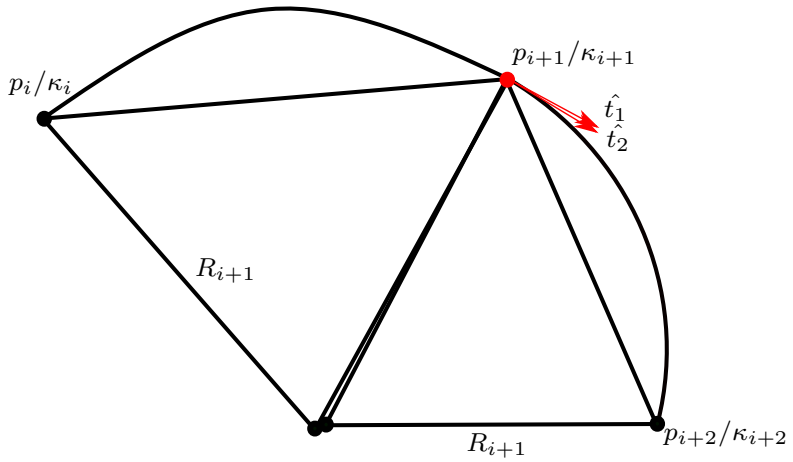


Figure F.1: Setup to calculate the tangent with the pre-sampling approach for the last point of a segment and for the first point of the next segment. For both segment a circular segment is spanned with R_{i+1} interconnecting the two sampled points of the segment.

The segments are always spanned in a way that they interconnect the two points of the segments. This leads to the fact that the middle point of the two segments

does not coincide. Based on thsi fact the orientation of the tangent changes once the segment is switched as can be seen by the two tangents $\hat{t}_1$ and $\hat{t}_2$. This leads to a small discrete jump in the control parameter $\hat{t}$. This jump is assumed to be small enough that no discrete jumps are generated in the attitude command of the UAV.

An obvious reason would be to store the tangent for every sampled point. For this setup the sampled information per point would consist of the set

$$\{x, y, \kappa, \hat{t}_x, \hat{t}_y\} \tag{F.2}$$

Hence five values would have to be stored per sampled point leading to

$$n_{add} = 2N \tag{F.3}$$

additional double values to store per segment. This is not beneficial if the storage amount should be kept low and hence such a setup was not pursued further in this work.

Appendix G

Curvature of Bézier curve

The curvature of a Bézier curve defined by (5.67) and (5.68) can be defined as

$$\kappa_{\text{Bézier}}(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t) + \dot{y}(t))^{\frac{3}{2}}} = \frac{f(x,y,t)}{g(x,y,t)} \tag{G.1}$$

where

$$\begin{aligned} f(x,y,t) = 0.67(&-(tx_1 - 2tx_2 + tx_3 - x_0(t-1) + 2x_1(t-1) - x_2(t-1)) \\ &(-t^2y_2 + t^2y_3 + 2ty_1(t-1) - 2ty_2(t-1) - y_0(t-1)^2 + y_1(t-1)^2) + (ty_1 - 2ty_2 + ty_3 - y_0(t-1) \\ &+ 2y_1(t-1) - y_2(t-1))(-t^2x_2 + t^2x_3 + 2tx_1(t-1) - 2tx_2(t-1) - x_0(t-1)^2 + x_1(t-1)^2)) \end{aligned}$$

$$\begin{aligned} g(x,y,t) = ((&t^2x_2 - t^2x_3 - 2tx_1(t-1) + 2tx_2(t-1) + x_0(t-1)^2 - x_1(t-1)^2)^2 + (t^2y_2 - t^2y_3 - 2ty_1 \\ &(t-1) + 2ty_2(t-1) + y_0(t-1)^2 - y_1(t-1)^2)^2)^{1.5} \end{aligned}$$

Appendix H

Approximation used in scaled sigma approach

For the scaled sigma approach in section 5.3.2 an approximation to simplify the calculations in the path following cycle is used. To illustrate the approximation in figure H.1 is shown as a section of figure 5.21. In this approach the position of the UAV P_{UAV} is scaled to the segment q_i resulting in the intersection point P. For the calculation of the approximated time for the closest point on the path to the segment, the following calculation is used

$$t_{app}(w_i, \sigma) = t_{w_i} + t_{sigma} = i\Delta t + \sigma\Delta t \tag{H.1}$$

This simplification does assume the point P_{app} to be the closest point on the path to the UAV. Hence the point is perpendicular to the segment q_i at the point P and does not exactly represent the actual closest point P_{cl}. The points P_{cl} and P_{app} only coincide for

$$\sigma = \{0, 0.5, 1\} \tag{H.2}$$

and hence are only for these positions exact. For all other values of a small error occurs when calculating the closest point on the path. In this work this error is neglected and it is assumed

$$P_{cl} = P_{app} \tag{H.3}$$

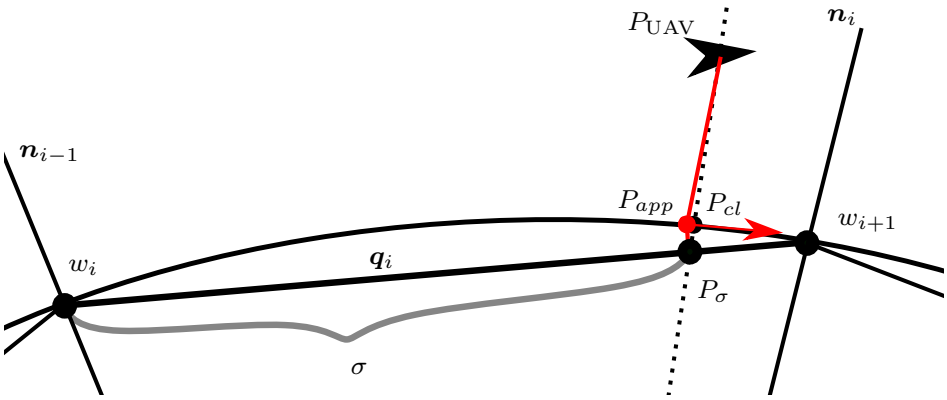


Figure H.1: Illustration of the approximation used in the scaled sigma approach shown in figure 5.21.

Appendix I

Concatenation of values in MAVLink message

Since not enough user-defined values are available for parameters 2 and 3 in section 6.1.1 two parameters are stored in the 32bit value. To implement this two methods are proposed:

Float digit separation

Two ranges for the digits of the `float` value are reserved for the two values. To built the concatenated value addition is used and to decouple division as well as the modulo operator are used. An example to build a concatenated value is shown here:

$$V_a = 15.3\frac{m}{s} \tag{I.1}$$

$$\psi_0 = 135.1^\circ \tag{I.2}$$

$$\text{Out}_{\text{float}} = 100000V_a + 10\psi_0 = 153|1351 \tag{I.3}$$

Union concatenation

Through the use of the C++ concept of `Unions` the 32bit value can be divided in to two 16bit values. This approach allows maximal precision to both values and also negative values can be displayed. Therefore both values that should be concatenated are casted into a `uint16_t`. To allow for maximal precision with an integer value are multiplied before casted (example for V_a and ψ_0). An example of such a use of `Unions` is shown in listing I.1

$$V_a = 15.3\frac{m}{s} = 1530\frac{cm}{s} \tag{I.4}$$

$$\psi_0 = 135.1^\circ = 13510^{c\circ} \tag{I.5}$$

Since QGroundControl only allows for a fixed number of decimal numbers, the second approach is not suitable, since here small numbers might be built. Hence the first approach is used for the implementation.

Listing I.1: Example of coupling and decoupling with unions

```
#include <stdio.h>
#include <stdint.h>

union u_parameter_t
{
    struct {
        uint16_t u16_heading;        // heading in centi-degree
        uint16_t u16_magnitude;    // magnitude in cm/s
    };

    float f32_encoded;
};

int main()
{
    // Encoding
    float f32_heading = 157.8f;
    float f32_magnitude = 16.8f;

    union u_parameter_t u_parameter = {};

    // conversion to 100 times value and cast to integer
    // TODO: Ensure these are positive values
    u_parameter.u16_magnitude = (uint16_t)(f32_magnitude*100);
    u_parameter.u16_heading = (uint16_t)(f32_heading*100);

    printf("value as float: %7.7e \n", u_parameter.f32_encoded);
    printf("values as int: %i %i\n", u_parameter.u16_magnitude,
        u_parameter.u16_heading);


    // Decoding
    float f32_param_to_decode = 5.3881194e-35f;
    union u_parameter_t u_parameter2 = {};
    u_parameter2.f32_encoded = f32_param_to_decode;
    printf("decoded values as int: %i %i\n",
        u_parameter2.u16_magnitude, u_parameter2.u16_heading);
}
```

Appendix J

Parameter Population in PX4 Position Controller

To be able to access the desired parameters defining a trochoid segment in the respective controller, the parameters have to be populated in the existing software architecture. Therefore multiple values are hijacked:

Parameter	**MAVL msg**	**Mission Item**	**PS struct**
Total time T	1	`time_inside`	`vx`
Wind speed V_ω and angle ψ_ω	2	`acceptance_radius`	`vy`, `yawspeed`
Airspeed V_a and initial heading ψ_0	3	`loiter_radius`	`vz`, `loiter_radius`
Oriented turn rate $\delta\omega$	4	`yaw`	`yaw`

The only fixed parameter Δt is added to the `mission_block` as parameter `_param_npfg_sampling_time`.

Appendix K

Robust Trochoids

General introduction to the principle and variations of robust trochoids presented in [1]:

One problem with trochoids, which has been pointed out in this work already is the assumption of an instant maximum bank angle in a turning segment. This introduces a continuous jump in the curvature, when a turn segment is obtained, as illustrated in figure K.1 for the case of *Nominal Trochoids*. Due to the roll dynamics of a fixed-wing UAV, such a path might introduce large tracking deviations especially in high wind conditions. A better approximation for the roll dynamics has been introduced with clothoids, which choose the curvature according to the maximum turning rate. However, clothoids provide a conservative way of path calculation in wind and the computation time is expected to be slightly higher than for trochoids.

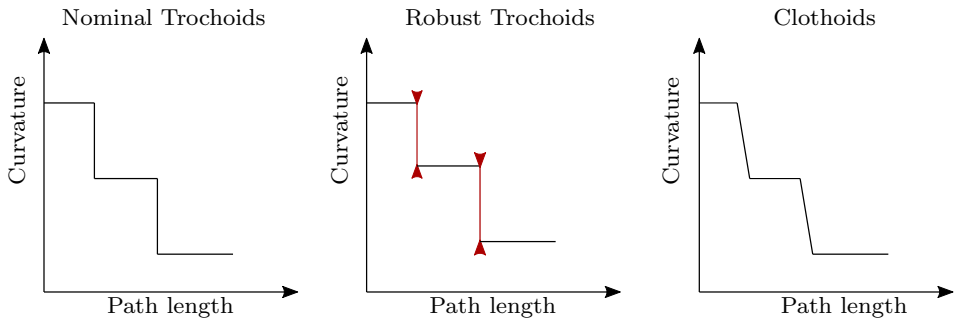


Figure K.1: Behaviour of curvature transition for paths generated with Nominal Trochoids, Robust Trochoids and Clothoids.

Hence, an adjusted version of the trochoid turning segment is introduced, which accounts more for the roll dynamics. *Robust Trochoids* tackle the problem of continuous jumps in curvature by increasing the minimum turning radius used for the path calculation:

$$R_{rob} = kR_{nom}, \quad k > 1.0 \tag{K.1}$$

R_{nom} being the minimum turning radius for the nominal trochoid and R_{rob} the minimum turning radius for the robust trochoid. With such an adjustment, the jumps have a smaller size than for nominal trochoids, as indicated in red in figure K.1.

To illustrate the behavior of the different path types introduced up to this point, in figure K.2 all three path types in comparison to the Dubins case for no wind are shown. For the robust trochoid $k = 1.2$ is assumed.
The choice of the parameter k is key for the performance of the robust trochoid. The simplest approach is to choose k in a conservative way as a constant. Such approaches and their behavior compared to the nominal chase are shown for a Dubins set up in [44]

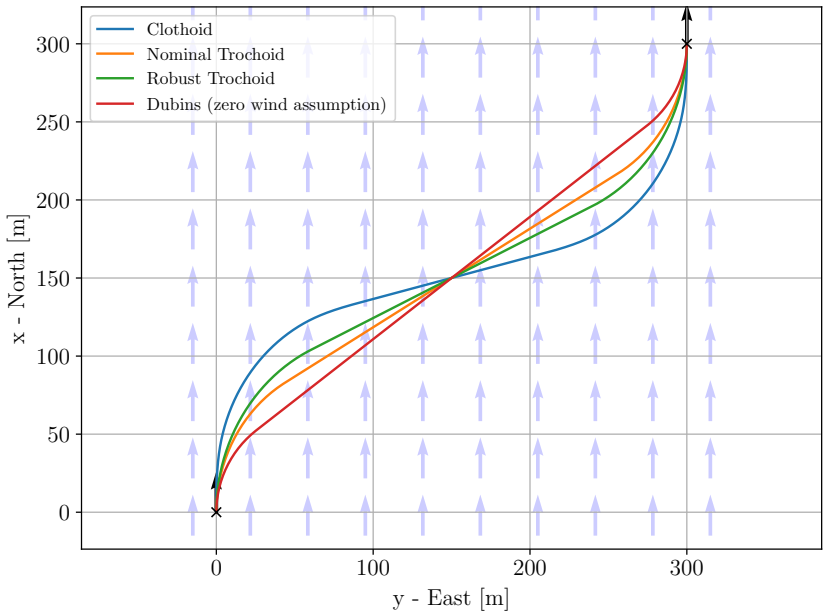


Figure K.2: Comparison of different turning segment approaches for $V_a = 20m/s$, $V_\omega = 5m/s$ and $\psi_\omega = 0°$. Initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. Final pose $x_f = 300m$, $y_f = 300m$ and $\psi_f = 0°$. $k = 1.2$ for robust trochoid turning path.

and [45]. Such an approach shows a good behaviour, especially regarding the tracking accuracy. Nevertheless, the conservative parameters are chosen irrespective of the current environmental parameters such as airspeed V_ω or wind heading ψ_ω. Therefore the performance of these approaches is drastically decreased.

[46] introduces an alternative approach that is not solely relying on conservative performance limits. Therefore, an unknown wind component is introduced to the nominal wind formulation. The unknown component is defined by its magnitude ΔV_ω and the deviation from the reference wind direction $\Delta\psi_\omega$. The adjusted wind vector takes the form

$$\boldsymbol{v}_\omega = V_\omega \begin{pmatrix} \cos(\psi_\omega) \\ \sin(\psi_\omega) \end{pmatrix} + \Delta V_\omega \begin{pmatrix} \cos(\psi_\omega + \Delta\psi_\omega) \\ \sin(\psi_\omega + \Delta\psi_\omega) \end{pmatrix} \tag{K.2}$$

The goal is now to plan the trochoid according to the known wind components, in a way that it is still trackable and safe for possible unknown deviations. One approach is to introduce turn rate constraints. With this approach, the flight controller may be able to use the performance margin to fully or partially compensate for the additional wind component. [46] also introduces a measure to account for the safety distance to certain objects or no-fly zones, that will not be discussed in this work.

Given the curvature of the path in the plane

$$\kappa = \frac{1}{R} = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{K.3}$$

where R is the instantaneous radius of the turn. Substituting equation (3.14) and equation (3.15) in the equation above leads to

$$\frac{1}{R} = \frac{\omega V_a (V_a + V_\omega \cos(\psi - \psi_\omega))}{(V_a^2 + V_\omega^2 + 2V_a V_\omega (\cos(\psi - \psi_\omega)))^{\frac{3}{2}}} \tag{K.4}$$

For the case of perfect tailwind, hence $\psi(t) = \psi_\omega$, the denominator which resembles the UAV's velocity reaches its maximum. For a constant curvature, this can only be counteracted by increasing the turn rate appropriately. If perfect tailwind is assumed, the required turn rate for a given path curvature and wind speed can be obtained by

$$\omega = \frac{(V_a + V_\omega)^3}{RV_a(V_a + V_\omega)} = \frac{(V_a + V_\omega)^2}{RV_a} \tag{K.5}$$

Hence the maximum turn rate for a given airspeed V_a and wind speed V_ω can be obtained by

$$\omega_{max}(V_a, V_\omega) = \frac{(V_a + V_\omega)^2}{RV_a} \tag{K.6}$$

which can be compared to the case of no wind, where

$$\omega_{max}(V_a, V_\omega = 0) = \frac{V_a}{R} \tag{K.7}$$

To formulate a scaling factor for the no wind case, equation (K.5) can be reformulated as

$$\omega_{max}(V_a, V_\omega) = (\frac{(V_a + V_\omega)}{V_a})^2 \omega_{max}(V_a, V_\omega = 0) \tag{K.8}$$

Introducing a wind speed component ΔV_ω, the maximum turn rate can be expressed in terms of the maximum turn rate corresponding to the reference wind conditions

$$\omega(V_a, V_\omega, \Delta V_\omega) = \frac{(V_a + V_\omega + \Delta V_\omega)^2}{(V_a + V_\omega)^2} \omega_{max}(V_a, V_\omega) \tag{K.9}$$

[46] suggests to additionally introduce a maximum airspeed error ΔV_a which yields the maximum required turn rate to sustain a given flight path curvature at the given wind condition and airspeed tracking accuracy

$$\omega(V_a, V_\omega, \Delta V_\omega, V_a) = \frac{(V_a + V_\omega + \Delta V_\omega + \Delta V_a)^2}{(V_a + V_\omega)^2} \omega_{max}(V_a, V_\omega) \tag{K.10}$$

For the case considered in this work for coordinated turns with zero sideslip and maximum bank angle $\bar{\phi}$ a nominal turn rate can be calculated

$$\omega_{lim} = \frac{g \tan(\phi_{max})}{V_a} \tag{K.11}$$

To plan the path to be feasible in terms of horizontal curvature, the turn rate limit must be greater or equal the maximum turn rate required for the path at the given wind condition and airspeed error

$$\omega_{lim} \geq \omega_{max}(V_a, V_\omega, \Delta V_\omega, \Delta V_a) \tag{K.12}$$

In figure K.3 robust trochoids are compared to nominal trochoids. For all final poses, the time needed for the robust trochoid is divided by the time for the nominal trochoids. Only the results for the path type RSL are shown. To illustrate the difference, wind conditions of $V_\omega = 15 m/s$ are chosen. The biggest difference is shown when both turning segments are long, as for example shown on the top left. For some close configurations, the robust trochoid even may get infeasible for this path type, as for example in the yellow region.

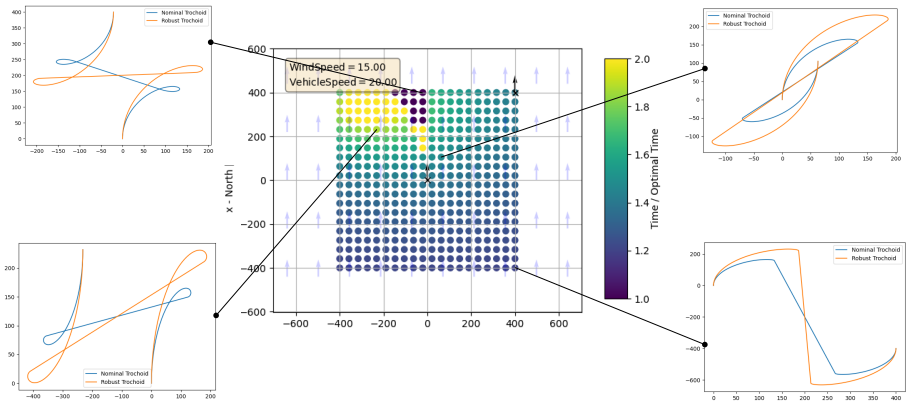


Figure K.3: Optimality indication for robust trochoid paths compared to nominal trochoid for $V_a = 20m/s$, $V_\omega = 15m/s$ and $\psi_\omega = 0°$. Fixed initial pose $x_0 = 0m$, $y_0 = 0m$ and $\psi_0 = 0°$. 20 x 20 final poses equally distributed in a range of $[-400m, 400m]$ in x- and y-direction with final heading constant at$\psi_f = 0°$. The results of the RSL path type are used.