2024-08-07

# Real-Time Dynamically Feasible B-Spline Trajectory Generation for Unmanned Aircraft

David LaVae Christensen
*Brigham Young University*

# Real-Time Dynamically Feasible B-Spline Trajectory Generation for Unmanned Aircraft

David L. Christensen

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Tim McLain, Chair
Marc Killpack
Cammy Peterson

Department of Mechanical Engineering

Brigham Young University

# *Real-Time Dynamically Feasible B-Spline Trajectory Generation for Unmanned Aircraft*

David L. Christensen
Department of Mechanical Engineering
Master of Science

**BYU Engineering**

# *Abstract*

Onboard aircraft systems must frequently plan dynamically feasible paths or trajectories to safely and quickly reach their goal position. Optimized B-splines are promising because they can guarantee kinematic feasibility in a computationally efficient manner and can be applied to a variety of vehicle models. However, current B-spline solutions do not adequately constrain trajectories to the physical limitation of the vehicles they are planning for, or are often too conservative. Most are also limited to path smoothing or require pre-optimized timing parameters. This thesis develops a real-time B-spline optimization-based trajectory generator capable of minimizing time while respecting the velocity, acceleration, and turn limitations of various vehicles. This thesis also develops a minimum-distance B-spline optimization-based path planner with obstacle, curvature, and slope constraints. This work takes advantage of the convex bounding property of B-splines to construct constraints that do not require discrete sampling of the path. This allows the path to scale in size without added computational costs. By converting B-spline control points to Minvo control points, the constraints are also non-conservative in relation to the vehicle limitations. Novel contributions include the development of B-spline optimization constraints that guarantee feasibility over curvature, slope, angular rate, centripetal acceleration, and tangential acceleration. This research also develops a direct obstacle avoidance constraint that does not require pre-planning steps. It additionally implements linear safe flight-corridor constraints into the path optimization. These techniques are validated through successful path and trajectory tracking by a fixed-wing aircraft in simulation.

Keywords: trajectory generation, path planning, B-spline, dynamically feasible, real-time, aircraft, curvature, slope, angular rate, centripetal acceleration, tangential acceleration, collision-free, obstacle, Minvo

# *Acknowledgments*

I would like to first thank my advisor and teacher Dr. Tim McLain for the continued support and guidance he has given me since I started my graduate program at BYU. He has always had high expectations and gave me the freedom to explore and develop my interests and skills in the field of controls and autonomy. He is an inspiring example for me of outstanding character, exceptional work ethic, and engineering brilliance. I would also like to thank my parents, Scott and Amparo Christensen. I thank them for their love and support and for always believing in me no matter what goals I set. To them, I attribute most of my cherished values, principles, and successful habits. Lastly, I would like to thank all the professors and peers in my program who I have worked with and learned from throughout this journey. I am lucky to have brushed shoulders with all of them.

# *Table of Contents*

# List of Figures

# List of Tables

# Nomenclature

**Polynomial Curves**

| | |
|---|---|
| $\alpha$ | time spacing between knot point intervals of a B-spline |
| $\beta_i$ | $i^{\text{th}}$ Bezier control point |
| $\boldsymbol{\beta}$ | set of Bezier control points |
| $\boldsymbol{\nu}$ | set of Minvo control points |
| $\mu$ | number of knot intervals |
| $\nu_i$ | $i^{th}$ Minvo control point |
| $\tau$ | normalized time parameter for a B-spline interval |
| $\mathbf{P}$ | set of B-spline control points |
| $\mathbf{T}$ | set of knot points |
| $A(t)$ | cross term used in the curvature equation of a B-spline |
| $B(t)$ | Bezier curve |
| $b(t)$ | B-spline curve |
| $d$ | space dimension of a polynomial curve |
| $k$ | order or degree of the polynomial |
| $m$ | number of knot points |
| $n$ | number of B-spline control points |
| $n_B$ | number of Bezier control points |
| $n_v$ | number of Minvo control points |
| $N_{i,k}$ | $i^{\text{th}}$ basis function of a k degree B-spline |
| $p(t)$ | polynomial function |
| $P_i$ | $i^{\text{th}}$ B-spline control point |
| $t_i$ | $i^{\text{th}}$ knot point |
| $V(t)$ | Minvo curve |

**Collision Free Paths**

| | |
|---|---|
| $d_{j,i}$ | distance used in constraint for distance from Minvo control point $i$ to an obstacle $j$ |
| $h_{\text{corr},j}$ | height of the $j^{\text{th}}$ safe flight corridor |
| $L_{\text{corr},j}$ | length of the $j^{\text{th}}$ safe flight corridor |

$R_{\text{corr},j}$ rotation that defines the orientation of the $j^{\text{th}}$ safe flight corridor

$R_{j,i}$ rotation for the $j^{\text{th}}$ obstacle and for the set of control points $i$ through $i+k$ that transforms the reference frame so that $V_{j,i}$ aligns with the $x$ axis

$r_j$ radius of the $j^{\text{th}}$ obstacle

$V_{\text{corr},j}$ vector from the inertial frame to the center of the $j^{\text{th}}$ safe flight corridor

$V_{j,i}$ vector from the center of an obstacle to the mean of Minvo control points $i$ through $i+k$

$w_{\text{corr},j}$ width of the $j^{\text{th}}$ safe flight corridor

**Kinematic Terms**

$\omega$ angular rate

**a** acceleration vector

**D** direction vector

**p** position vector

**V** velocity vector

**W** waypoint position vector

$a_R$ centripetal acceleration

$a_T$ tangential acceleration

$C$ curvature

$R$ turn radius

**Vehicle Models**

$\alpha$ angle of attack

$\beta$ bicycle velocity offset angle from the body, or fixed-wing aircraft sideslip angle

$\chi$ heading angle of the vehicle velocity vector

$\Delta_r$ boat rudder steering rate

$\delta_r$ boat rudder steering angle

$\Delta_w$ bicycle wheel steering rate

$\delta_w$ bicycle wheel steering angle

$\gamma$ flight path angle

$\phi$ roll

$\psi$ yaw or heading of the vehicle body

$\theta$ pitch

$c_b$ boat constant

$c_r$ boat rudder constant

$g$ gravity constant

$L$ bicycle length

| | |
|---|---|
| $l_r$ | distance from the center of gravity to the back wheel of the bicycle |
| $p$ | roll rate |
| $q$ | pitch rate |
| $r$ | yaw rate |
| $u$ | body longitudinal velocity |
| $v$ | body lateral velocity |
| $w$ | body vertical velocity |

# 1 Introduction

## 1.1 Motivation and Background

The scope and demand for unmanned aerial vehicles (UAVs) has expanded rapidly over the past few decades. UAVs have been promoted as primary tools for package delivery, infrastructure maintenance, disaster response, intelligence data collection, and airspace security [1]–[4]. Unmanned aircraft are well suited for this type of work because of their speed, payload capacity, and ability to operate in hazardous environments. Fixed-wing aircraft are especially attractive for surveillance and payload transportation due to their long-duration flight capability [5]. Figure 1.1 shows several drones currently used in industry for the mentioned tasks. Image A in Figure 1.1 shows a Zipline drone delivering a medical supply package [6], image B shows the Elios 3 inspecting a potentially hazardous mine [7], image C depicts the Bat UAV carrying out a surveillance mission [8], and image D shows the DroneHunter F700 capturing a drone threat [9].

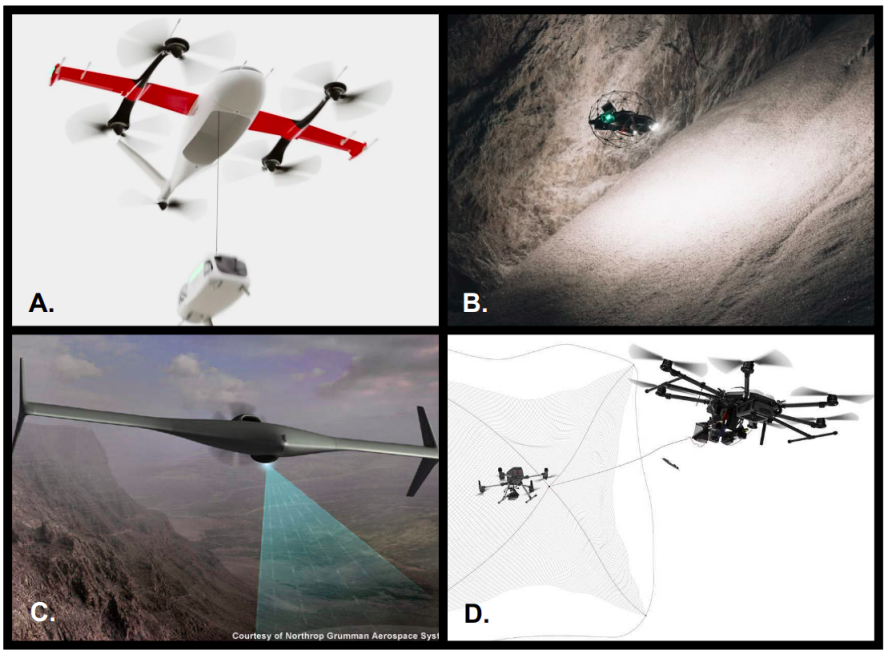



Figure 1.1: UAVs employed for various applications (Images courtesy of [6]–[9])

Most of these use cases involve monotonous tasks, require skilled piloting, or place the UAV in situations where the aircraft can lose communication with ground control. Each of these challenges, respectively, make remote piloting either too expensive, too difficult, or impossible. Autonomous guidance and control is therefore vital for integrating UAVs into these type of flight missions. However, autonomous guidance and control has its own set of hurdles to overcome.

Many autonomous applications require UAVs to travel near large buildings, trees, or other complicated terrain. To navigate this terrain safely and efficiently, UAV systems must frequently generate dynamically feasible paths through their surroundings. This is because vehicles in crowded spaces must respond quickly to unexpected obstacles and sporadic pose updates common to GPS-degraded environments [10]. Frequent path generation is also crucial for scenarios that involve tracking a moving target or flying in conditions with significant wind disturbances. In these situations, either the goal pose changes, the path is obstructed, or the UAV veers off of the original path. Figure 1.2 demonstrates each of these scenarios. In each of the images, the aircraft was following a previously planned path (dashed red line), but was then required to plan a new path (dashed green line). If not for quick re-planning, the aircraft could not complete its mission successfully, or at least not efficiently. This is especially true for time-constrained missions, or for certain aircraft that require trajectories to define their motion.

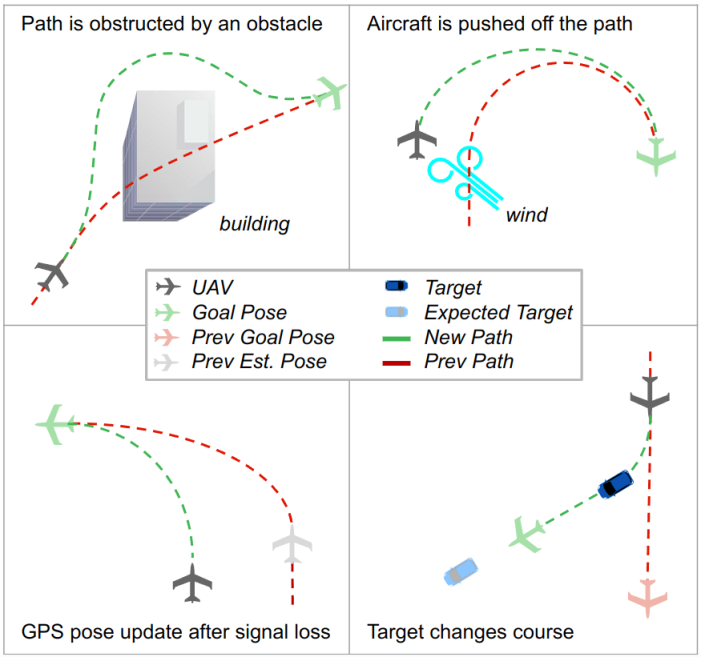



Figure 1.2: Scenarios where real-time path planning is needed

The biggest difficulty in creating optimal paths or trajectories is generating them in real time. Limited communication bandwidth and signal dropout require that planning be completed on the aircraft's onboard computer,

which has limited processing power. The planning algorithms must be computationally efficient while still respecting the dynamics and surroundings of the vehicle. Unfortunately, many of the existing trajectory generation methods are computationally expensive, require complex system models, or work only in two dimensions [11], [12]. These methods also construct trajectories composed of discrete segments with dynamic discontinuities at their connection points. This makes them more computationally expensive and does not guarantee dynamic feasibility at the discontinuities. Other methods are reactive and direct the UAV in a general direction, which often produces less efficient paths, or can lead to situations beyond the vehicle's control authority [13], [14].

Another branch of trajectory generation research focuses on the use of B-splines to efficiently generate paths. This method takes advantage of the convex bounding property of B-splines to produce fast optimal trajectories within the constraint space of the vehicle's kinematics and surroundings [15]. It also leverages the flat output space of the vehicles to accurately define a feasible path or trajectory [16]. B-splines are continuously differentiable up to their polynomial order and can therefore define a smooth differentiable path. However, most of the existing methods are limited in how well they can profile the kinematics and dynamics of a vehicle, thus limiting the true agility of the respective vehicle. For example, the majority of B-spline trajectory planning algorithms neglect turning limitations such as angular rate or centripetal acceleration. Some consider the curvature restrictions of a vehicle, but are limited to second- or third-degree polynomials, or two-dimensional space [17], [18]. Others are limited to discrete sampling, or path smoothing of an existing discretized path [19]. Most of these methods also require a pre-optimized traversal time.

The goal of this research is to develop a quick and efficient B-spline trajectory generator that can produce optimal, dynamically feasible, and collision-free trajectories for a variety of aircraft systems. The guidance algorithm should be computationally efficient enough to work onboard a moving vehicle and should also not be overly conservative with respect to the physical limitations of the vehicle.

## 1.2 Contributions

The research detailed in this thesis makes the following contributions:

- **Constrained Optimization-Based B-spline Path Planner:** The development of a minimum-distance path planner with maximum-curvature constraints, maximum-slope constraints, direct obstacle avoidance constraints, and direction and curvature continuity constraints at the endpoints using third-order three-dimensional B-splines. The obstacle constraints do not require any pre-planning steps and achieve a locally optimal solution.

- **Control-Point Curvature Constraint:** The creation of a second maximum-curvature constraint for B-spline path optimization that relies only on

the control points to limit the curve. This method is more conservative in its bounds but has the potential to be faster. It can also serve B-splines of any polynomial order.

- **Constrained Optimization-Based B-spline Trajectory Generator:** The development of a minimum-distance and time trajectory generator with maximum-angular-rate, maximum-centripetal-acceleration, maximum-tangential-acceleration, and maximum-velocity constraints using third-order three-dimensional B-splines. The trajectories also have constraints on endpoint position, velocity, and acceleration.

- **Fixed-Wing Aircraft Generic Path Follower:** The design of a fixed-wing aircraft path following algorithm that can follow any path representation given the position, direction, and center point of the osculating circle for the point on the path nearest to the aircraft.

- **B-spline Evaluation Library:** The creation of a uniform B-spline evaluation library that can convert uniform B-splines to Minvo and Bezier curves using the matrix evaluation method. The software is implemented in python and comparable in speed to Scipy's BSpline class. It is faster than Scipy for large data sets because it takes advantage of the computational speed of Python's matrix operations. It is capable of evaluating both clamped and open uniform B-splines.

Some other minor contributions include the development of a fixed-wing aircraft trajectory tracker and several trajectory tracking controllers for a variety of 2D vehicle models. This thesis also implemented box-shaped safe flight corridor constraints into the B-spline path optimization. These constraints can be rotated to any orientation and maintain linear complexity.

The contributions are demonstrated in Chapters 3 through 6 by data and simulation results. Each of the path or trajectory generation methods described guarantees constraint feasibility in a non-conservative fashion and does not rely on discrete sampling of the B-spline. Non-discrete sampling gives these algorithms the ability to create paths of any length without any loss of computational efficiency. Also, non-conservative constraint bounds are achieved by converting B-spline control points to Minvo or Bezier curve control points. The GitHub repository for the B-spline evaluation library can be found at *https://github.com/byu-magicc/bspline_evaluator*.

## 1.3 Thesis Outline

This thesis begins with a background and a tutorial on B-splines for trajectory generation in Chapter 2. It draws concepts from other works, but also builds upon the work. Chapter 3 designs two new methods for constraining the curvature of a B-spline path and compares them against traditional methods. Chapter 4 develops a direct obstacle avoidance technique for B-splines, and also details how to construct linear safe flight corridor constraints. The algorithms produced from these chapters are then used to create a fixed-wing aircraft path planner in Chapter 5. Slope constraints are added to the path

planner in this chapter and a path follower is developed for accurate tracking using a fixed-wing. Chapter 6 extends concepts from Chapter 3 to create a B-spline trajectory generator with curvature, angular rate, and centripetal acceleration constraints. Tangential acceleration constraints are also developed in this chapter. Three 2D vehicle models and a fixed-wing aircraft model are used to track and validate the feasibility of the generated trajectories. Also, each of the chapters 3 through 6 contain plots and simulation results that validate the optimization algorithms developed in this thesis. The last chapter emphasizes key results and provides recommendations for further development.

# 2 B-Splines for Path and Trajectory Generation

Various path representations are used to generate and define vehicle paths or trajectories. Some of these include waypoint sequences, Dubins paths [20], vector fields [21], model predictive states [22], and polynomial curves [23]. Today, many of the leading real-time path planning methods use some form of piecewise polynomials to define a path [15], [24], [25]. Among these representations is a B-spline, or basis spline, which is a piecewise polynomial that is continuous at each connection point.

B-spline paths are commonly used because they contain properties that are pertinent and advantageous for trajectory generation [26]. For example, a B-spline is continuously differentiable, which means that it can represent smooth and differentially continuous trajectories. A B-spline is also bounded within the convex hull of its control points, which guarantees boundedness for the path and its derivatives. Basis splines are also invariant to affine transformations in space and time. This enables a path or trajectory to be scaled and transformed as needed for optimization or mapping realignment [27]. In contrast to a discrete-path representation, B-spline paths are nondiscrete and therefore more rapidly optimized and evaluated. This is crucial for online trajectory generation.

## 2.1 B-Splines

A B-spline is a continuous piece-wise polynomial function of degree $k$ [28]. If we were to evaluate a B-spline analytically, it would be expressed as

$$b(t) = \begin{cases} p_0(t) & \text{for} \quad t_k \leq t < t_{k+1} \\ p_1(t) & \text{for} \quad t_{k+1} \leq t < t_{k+2} \\ \quad \vdots & \\ p_{\mu-1}(t) & \text{for} \quad t_{k+\mu-1} \leq t < t_{k+\mu} \end{cases} \tag{2.1}$$

where $\mu$ is the number of intervals in the spline, $t_k$ is the start time or starting knot point for the first interval, and $[p_0, p_1, ..., p_{\mu-1}]$ are the polynomials that define the spline. A defined knot point is the moment in time when a polynomial begins or ends its existence in the spline. The set of defined knot points for a B-spline is $[t_k, t_{k+1}..., t_{k+\mu}]$. Equation 2.1 is depicted in Figure 2.1 for a one-dimensional third-order B-spline. Note that each polynomial is defined exclusively within its respective knot interval.

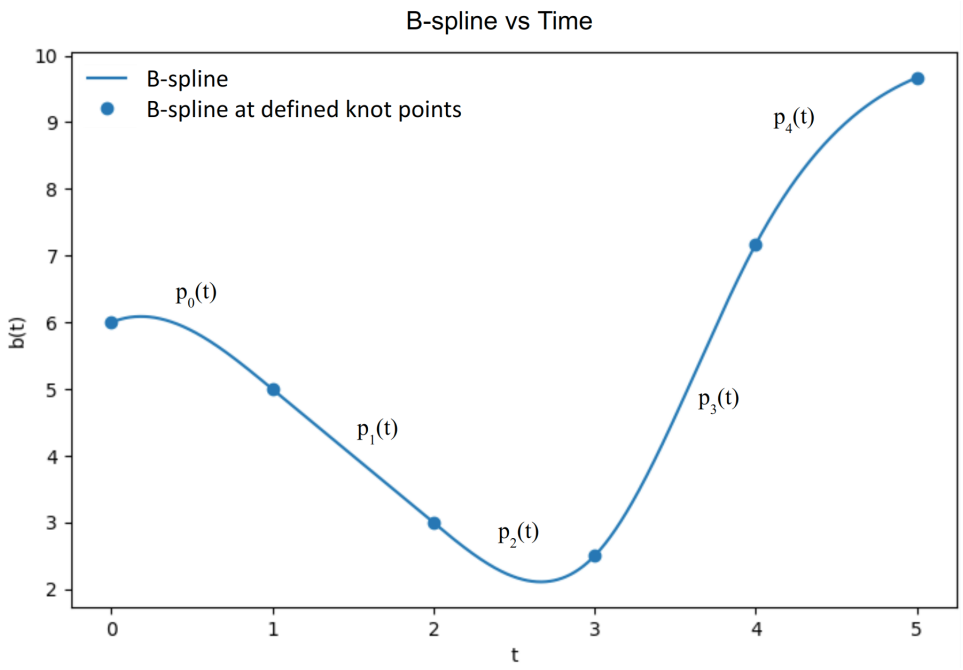



Figure 2.1: Third-order B-spline broken up into five sections or polynomials with $t_k = 0$ and $t_{k+\mu} = 5$

Each polynomial can be written as

$$p_l(t) = c_{l,0} + c_{l,1}\left(\frac{t - t_j}{\alpha}\right) + c_{l,2}\left(\frac{t - t_j}{\alpha}\right)^2 + \cdots + c_{l,k}\left(\frac{t - t_j}{\alpha}\right)^k \tag{2.2}$$

where $l$ is the index of the polynomial, $j$ is the index of the starting knot point, and $k$ is the order or degree of the polynomial. The $c_i$ values are constants, and $\alpha$ is a scale factor. The scale factor is the uniform time spacing between knot points and is equal to $t_{j+1} - t_j$. It is a key variable for scaling and optimizing the timing constraints of a B-spline trajectory.

### 2.1.1 Control Points

One of the main advantages of B-splines is the utility of its control points. Control points shape the physical coordinates of the spline. For a $d$-dimensional B-spline, each of its control points can be expressed as a vector, $P_i \in \mathbb{R}^{d\times 1}$. Figure 2.2 shows a given set of control points shaping a second- and fifth-order uniform B-spline in two dimensions. Notice that higher-order B-splines require more control points to define a single knot interval.

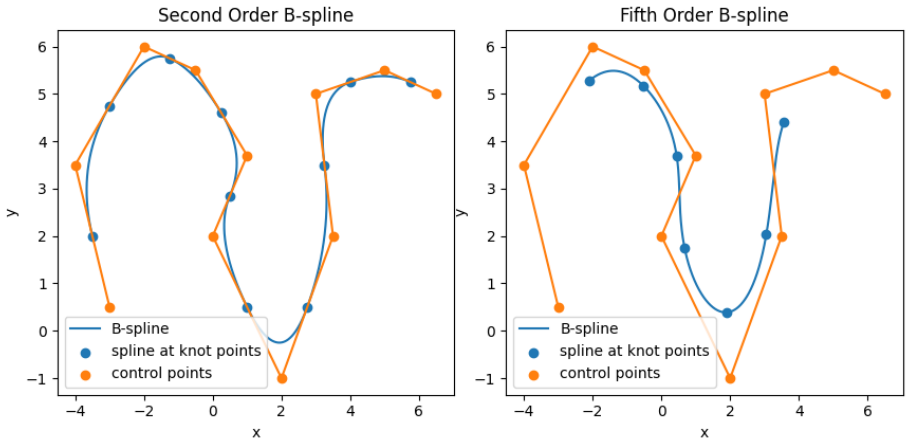



Figure 2.2: Second- and fifth-order B-splines in two dimensions

A B-spline has a fixed set of $n$ control points $\mathbf{P} = [P_0, P_1, ..., P_{n-1}]$. Each polynomial $p_l(t)$ of the B-spline is defined by $k+1$ of these control points. Specifically, the polynomial in the interval $[t_j, t_{j+1}]$ is defined by control points $[P_i, P_{i+1}, ..., P_{i+k}]$, where $i = j - k$. For example, the definition of a polynomial in terms of its control points is written as

$$p_l(t) = N_{i,k}(t)P_i \; + \; N_{i+1,k}(t)P_{i+1} \; + \; \cdots \; + \; N_{i+k,k}(t)P_{i+k} \tag{2.3}$$

where each $N_{\_,\_}(t)$ is a basis function. Using this representation, the entire B-spline can be expressed as

$$b(t) = \begin{cases} N_{0,k}(t)P_0 + N_{1,k}(t)P_1 \cdots + N_{k,k}(t)P_k & \text{for } t_k \le t < t_{k+1} \\ N_{1,k}(t)P_1 + N_{2,k}(t)P_2 \cdots + N_{k+1,k}(t)P_{k+1} & \text{for } t_{k+1} \le t < t_{k+2} \\ \quad \vdots & \\ N_{n-k-1,k}(t)P_{n-k-1} + \; N_{n-k,k}(t)P_{n-k} \cdots & \\ \quad + N_{n-1,k}(t)P_{n-1} & \text{for } t_{k+\mu-1} \le t < t_{k+\mu} \end{cases} \tag{2.4}$$

Equation 2.4 shows that each polynomial is a linear combination of its defining control points, with the basis functions serving as coefficients. This explains why a B-spline curve is shaped by its control points.

### 2.1.2 Basis Functions

A basis function determines the weight that each control point has on the spline at a specific point in time. The basis function $N_{i,\kappa}(t)$ is defined using the Cox-de Boor recursion formula and is defined as

$$N_{i,\kappa}(t) = \frac{t - t_i}{t_{i+\kappa} - t_i} N_{i,\kappa-1}(t) + \frac{t_{i+\kappa+1} - t}{t_{i+\kappa+1} - t_{i+1}} N_{i+1,\kappa-1}(t) \tag{2.5}$$

$$N_{i,0} = \begin{cases} 1, & \text{if } t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

where the first iteration of $\kappa$ is equal to $k$ [28].

A B-spline has a total of $n$ basis functions, and at any given time, the basis functions must sum to one. Each interval of the spline is influenced by $k+1$ basis functions. This is apparent from Figure 2.3 where each basis function is plotted for a second- and fifth-order B-spline, and the knot point intervals are one second long. Notice also that each basis function affects at most $k+1$ intervals of the spline and is equal to zero for the other remaining intervals. This property leads to a more compact expression for B-splines.

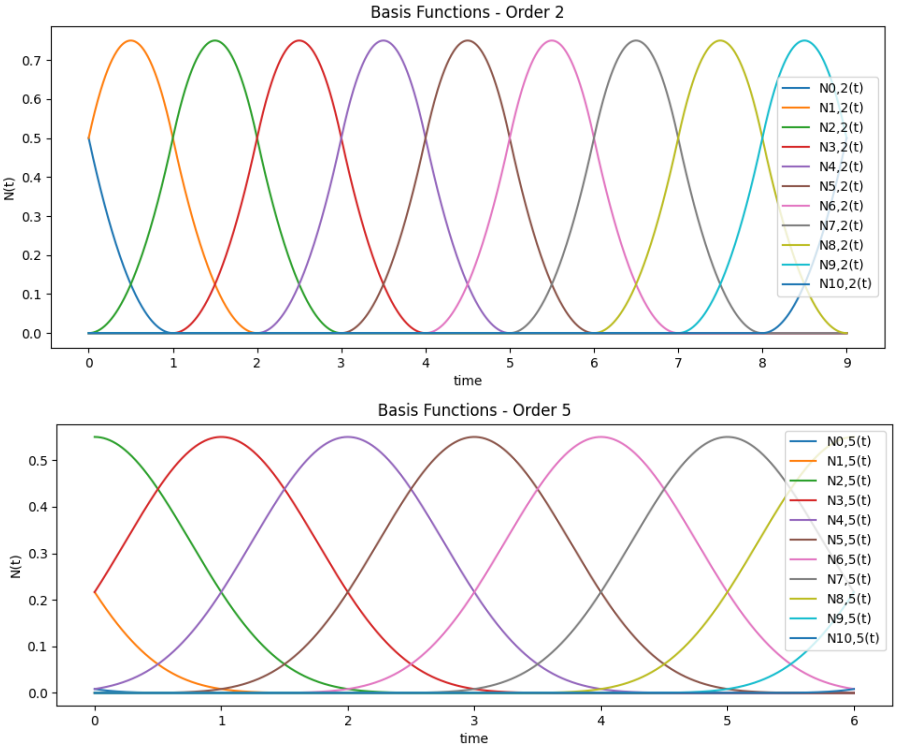



Figure 2.3: Second- and fifth-order basis functions with ten control points

The formal definition of a B-spline is expressed as

$$b(t) = \sum_{i=0}^{n-1} N_{i,k}(t) P_i \tag{2.7}$$

Equation 2.7 defines a B-spline by the summation of the product of all its control points $P_i$ and all its basis functions $N_{i,k}(t)$ [28]. However, because each basis function is active only during certain intervals, this equation is equivalent to Equation 2.4. The use of basis functions also ensures a differentiable continuity between each interval of the spline.

### 2.1.3 Knot Points

The evaluation of basis functions using the Cox-de Boor recursion formula depends on a set of knot points $\mathbf{T} = [t_0, t_1, ..., t_{m-1}]$ that are equally spaced. Each knot point $t_i$ must be less than knot point $t_{i+1}$, and the difference

between each consecutive knot point is equal to $\alpha$. $m$ is the total number of knot points equal to $n + k + 1$. Recall that $n$ is the number of control points and $k$ is the order of the B-spline. We note that a B-spline is defined only between knot points $t_k$ and $t_n$. This results in $\mu$ knot intervals in the curve, where $\mu = n - k$. Undefined knot intervals exist for the sole purpose of evaluating the Cox-de Boor recursion formula. A list of knot points where the defined knot points are highlighted is written as

$$\mathbf{T} = \Big[t_0,\ t_1, \cdots t_{k-1},\ \underbrace{t_k,\ t_{k+1}, \cdots t_n}_{\text{defined}},\ t_{m-k},\ t_{m-k+1}, \cdots t_{m-1}\Big] \tag{2.8}$$

The knot point values can be shifted forward or backward in time without affecting the shape of the spline. This is a useful property for shifting and concatenating B-spline trajectories through time.

### 2.1.4 Derivatives

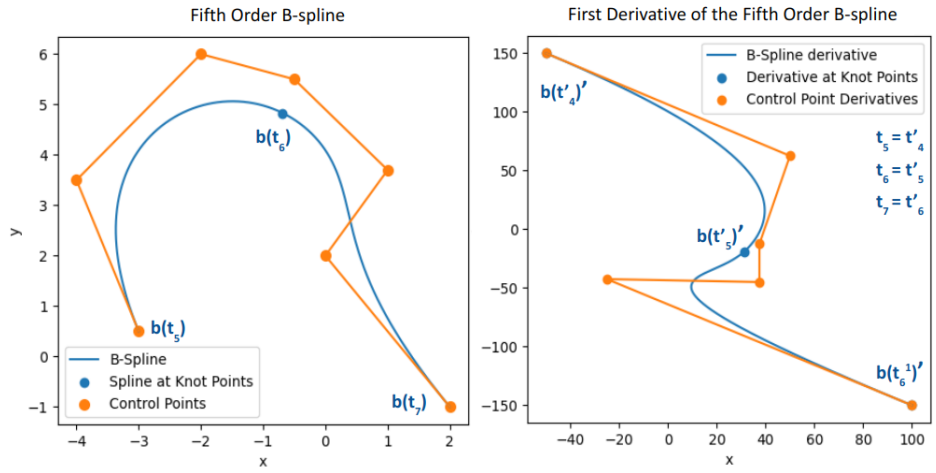



Figure 2.4: Derivative of a fifth-order B-spline and its control-point derivatives

When taking the derivative of a B-spline, a new B-spline of order $k - 1$ is derived with a new set of control points $\mathbf{P'}$ and knot vector $\mathbf{T'}$. The equation for the derivative with respect to time is

$$b(t)^{(r)} = \sum_{i=0}^{n-r-1} N_{i,k-r}(t) P_i^{(r)} \tag{2.9}$$

Figure 2.4 shows a fifth-order B-spline and its derivative with the accompanying control-point derivatives. Note that the number of control points is reduced by one when taking a derivative. Knot point values are also shifted down by one index. However, the number of knot point intervals $\mu$ remains the same. The values of the defined knot points remain the same. The set of

knot points for the $r^{\text{th}}$ derivative is expressed as

$$T^{(r)} = \Big[t_0,\ t_1, \cdots t_{k-r-1},\ \underbrace{t_{k-r},\ t_{k-r+1}, \cdots t_{n-r},}_{\text{defined}}\ t_{m-k-2r},\ t_{m-k-2r+1}, \cdots t_{m-2r-1}\Big] \tag{2.10}$$

### Control-Point Derivatives

Evaluating control-point derivatives is key to the use of B-splines for trajectory generation because the derivative of a B-spline curve is ruled by its respective control-point derivatives. The control points for the derivative of an open uniform B-spline are defined as

$$P_i' = \frac{P_{i+1} - P_i}{\alpha} \tag{2.11}$$

The $r^{\text{th}}$ control-point derivative of an open uniform B-spline is given by

$$P_i^{(r)} = \frac{P_{i+1}^{(r-1)} - P_i^{(r-1)}}{\alpha} \tag{2.12}$$

By bounding and optimizing the control-point derivatives, we can bound and optimize the derivatives of a B-spline trajectory.

## 2.1.5 Matrix Representation

The matrix representation for a B-spline is useful for deriving analytical solutions and constraints of a B-spline path or trajectory [29]. It also enables rapid computation of the spline. Otherwise, one would have to solve the Cox-de Boor recursion formula for each calculation, which has a polynomial-time complexity.

### Matrix Evaluation

The matrix equation for each interval of an open uniform B-Spline is

$$b(t) = \mathbf{P}_i M L \qquad \text{for } t_j \le t < t_{j+1} \tag{2.13}$$

where

$$M \in \mathbb{R}^{k+1 \times k+1} \tag{2.14}$$

$$\mathbf{P}_i = \begin{bmatrix} P_i & P_{i+1} & \cdots & P_{i+k} \end{bmatrix} \tag{2.15}$$

$$L = \begin{bmatrix} \tau^k \\ \vdots \\ \tau^2 \\ \tau \\ 1 \end{bmatrix} \tag{2.16}$$

$$\tau = \frac{t - t_j}{\alpha} \tag{2.17}$$

$$i = j - k \tag{2.18}$$

Note that $M$ is a matrix of constants unique to the order of the B spline, and $L$ is a vector containing time parameters of the polynomial equation. An expanded form of equation (2.13) is

$$c_0 + c_1 t + c_2 t^2 + \cdots + c_k t^k = \begin{bmatrix} P_i & P_{i+1} & \cdots & P_{i+k} \end{bmatrix} M \begin{bmatrix} \left(\frac{t-t_j}{\alpha}\right)^k \\ \vdots \\ \left(\frac{t-t_j}{\alpha}\right)^2 \\ \left(\frac{t-t_j}{\alpha}\right) \\ 1 \end{bmatrix} \tag{2.19}$$

We can derive the values for the matrix $M$ by symbolically solving Equation 2.7. For example, the symbolic solution of Equation 2.7 for a second-order B-spline is

$$\begin{aligned} b(t) &= \sum_{i=0}^{n-1} N_{i,2}(t) P_i \\ &= \left(\frac{P_i}{2} - P_{i+1} + \frac{P_{i+2}}{2}\right)\tau^2 + (P_{i+1} - P_i)\tau + \frac{P_i + P_{i+1}}{2} \\ &= P_{i+2}\frac{\tau^2}{2} + P_{i+1}\left(-\tau^2 + \tau + \frac{1}{2}\right) + P_i\left(\frac{\tau^2}{2} - \tau + \frac{1}{2}\right) \\ &= \begin{bmatrix} P_i & P_{i+1} & P_{i+2} \end{bmatrix} \begin{bmatrix} \left(\frac{\tau^2}{2} - \tau + \frac{1}{2}\right) \\ \left(-\tau^2 + \tau + \frac{1}{2}\right) \\ \frac{\tau^2}{2} \end{bmatrix} \\ &= \begin{bmatrix} P_i & P_{i+1} & P_{i+2} \end{bmatrix} \begin{bmatrix} 1/2 & -1 & 1/2 \\ -1 & 1 & 1/2 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau^2 \\ \tau \\ 1 \end{bmatrix} \end{aligned} \tag{2.20}$$

### The M Matrices

The matrix of constants in the last line of Equation 2.20 is the $M$ matrix for second-order uniform B-splines. The value of $M$ for each polynomial of order one through five is defined as

$$\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{if} \quad k = 1 \tag{2.21}$$

$$\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 2 \tag{2.22}$$

$$\frac{1}{12}\begin{bmatrix} -2 & 6 & -6 & 2 \\ 6 & -12 & 0 & 8 \\ -6 & 6 & 6 & 2 \\ 2 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 3 \tag{2.23}$$

$$\frac{1}{24}\begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -6 & -12 & 11 \\ 6 & -12 & -6 & 12 & 11 \\ -4 & 4 & 6 & 4 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 4 \tag{2.24}$$

$$\frac{1}{120}\begin{bmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 20 & 20 & -50 & 26 \\ -10 & 30 & 0 & -60 & 0 & 66 \\ 10 & -20 & -20 & 20 & 50 & 26 \\ -5 & 5 & 10 & 10 & 5 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 5 \tag{2.25}$$

### Matrix Derivative Evaluation

The matrix equation to evaluate the derivative of a B-spline with respect to time is

$$b(t)^{(r)} = \mathbf{P}_i M L^{(r)} = \begin{bmatrix} P_i & P_{i+1} & \cdots & P_{i+k} \end{bmatrix} M \begin{bmatrix} \frac{(t-t_j)^{k-r}}{\alpha^k} \prod_{j=k-r+1}^{k} j \\ \frac{(t-t_j)^{k-1-r}}{\alpha^{k-1}} \prod_{j=k-r}^{k-1} j \\ \vdots \\ \frac{(t-t_j)}{\alpha^{r+1}} \prod_{j=2}^{r+1} j \\ \frac{1}{\alpha^r} \prod_{j=1}^{r} j \\ \mathbf{0} \in \mathbb{R}^{r \times 1} \end{bmatrix} \tag{2.26}$$

where $r$ is the derivative order and must be a positive integer ($r \in \mathbb{Z}+$).

## 2.2 B-Spline Representation Benefits

This section describes properties of B-splines that are advantageous for path and trajectory generation. These properties include being piecewise and continuous, being bounded by the convex hull of its control points, being subject to affine invariance, and being capable of isolated local modifications.

### 2.2.1 Piece-wise Polynomial

Piecewise polynomials are advantageous because a curve of any length and resolution can be generated with any number of control points, without affecting the polynomial order of the curve. This also means that we can add onto a B-spline without affecting the already existing curves.

### 2.2.2 Continuously Differentiable

Continuously differentiable equations are important for creating smooth trackable paths. A B-spline is continuous from start to end, and the derivatives of a B-spline are also continuous up to the $(k-1)^{\text{th}}$ derivative. Figure 2.5 demonstrates the continuity of a third-order spline. The trajectory has a smooth path and velocity profile and a continuous acceleration profile.

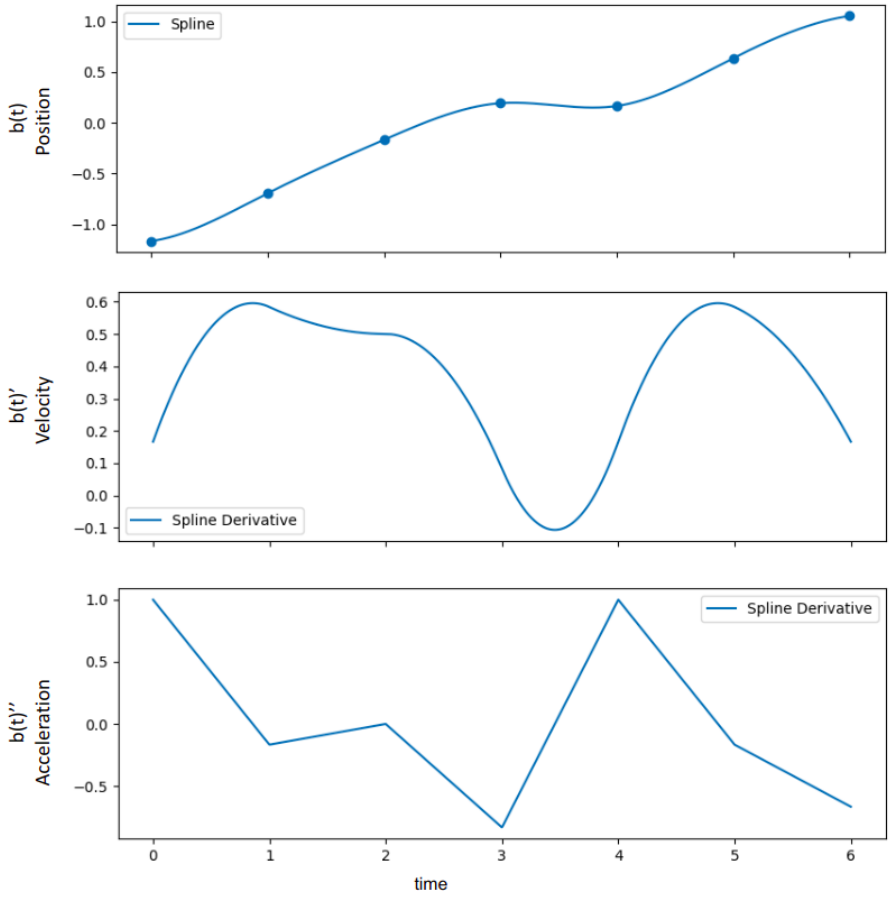



Figure 2.5: The continuity of a third-order B-spline and its derivatives

### 2.2.3 Convex Hull Bounds

A B-spline is also contained within the convex hull of its control points. A convex hull is the smallest convex set that contains a shape or a set of points. If we refer back to Section 2.1.2, we can see that a B-spline is a convex combination of its control points, and is expressed as

$$b(t) = \sum_{i=0}^{n-1} N_{i,k}(t) P_i \quad = \begin{bmatrix} P_0 & P_1 & \cdots & P_{n-1} \end{bmatrix} \begin{bmatrix} N_{0,k} \\ N_{1,k} \\ \vdots \\ N_{n-1,k} \end{bmatrix} = \mathbf{P}\mathbf{N}^T \tag{2.27}$$

A convex combination is a linear combination, or weighted average, where all the coefficients are nonnegative and sum to one. This quality ensures that a B-spline will always reside within the convex set of its control points.

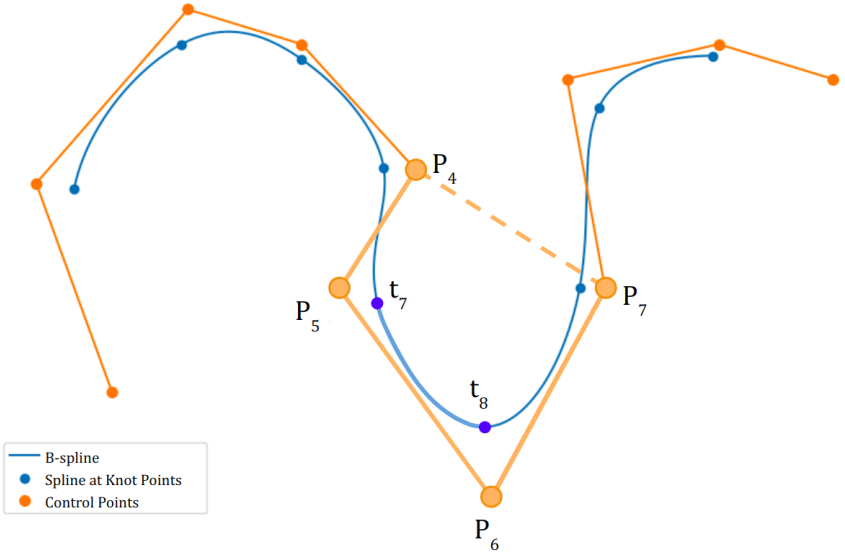



Figure 2.6: Strong convex hull property demonstrated on a third-order B-spline

This is also true for the set of control points that define a specific knot interval. The section of a B-spline that exists within the knot interval $[t_j, t_{j+1}]$, must reside within the convex hull of the control points $[P_i, P_{i+1}, ..., P_{i+k}]$, where $i = j - k$. Figure 2.6 demonstrates this boundedness for a third-order B-spline. Here, the knot interval $[t_7, t_8]$ is contained within the convex hull of control points $[P_4, P_5, P_6, P_7]$. This property is important for trajectory generation to avoid obstacles, as well as to bound the derivatives of the path.

### 2.2.4 Locally Isolated Segments

A modification to one section of the B-spline does not affect other sections of the B-spline. Specifically, moving control point $P_j$ only affects the section of the B-spline that exists between the interval $[t_j, t_{j+k+1}]$. Therefore, a single control point affects the shape of $k+1$ knot intervals. This is true for intervals between indices

$$k \leq j < n - k \tag{2.28}$$

As $j$ approaches 0 from $k$, or approaches $n$ from $n - k$, the number of knot intervals shaped by the $j^{\text{th}}$ control point decreases. In other words, the control points closer to the endpoints of the spline have an influence on fewer knot intervals.

Figure 2.7 demonstrates an isolated local modification on a third-order B-spline. Notice that shifting the sixth control point modifies the spline between the knot intervals $t_6$ through $t_{10}$. This property allows for isolated local modifications of a path without affecting other sections of the path. This is useful when an obstacle obstructs one path section but not the entire path.

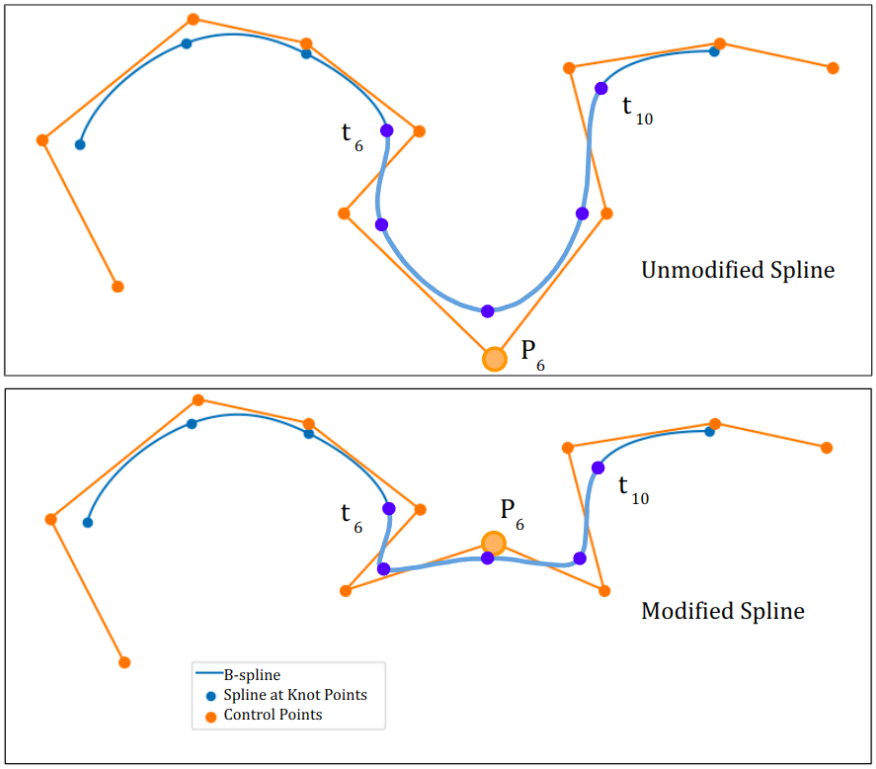



Figure 2.7: Isolated local modification of a control point on a third-order B-spline

### 2.2.5 Affine Invariance

If an affine transformation is applied to the control points of a B-spline, the same transformation is applied to the B-spline curve. Affine transformations include translation, rotation, scale, reflection, shear, and the identity transformation. Figure 2.8 shows an affine transformation performed over a B-spline curve. This property is useful when we need to rotate or translate a path after receiving an estimation update.

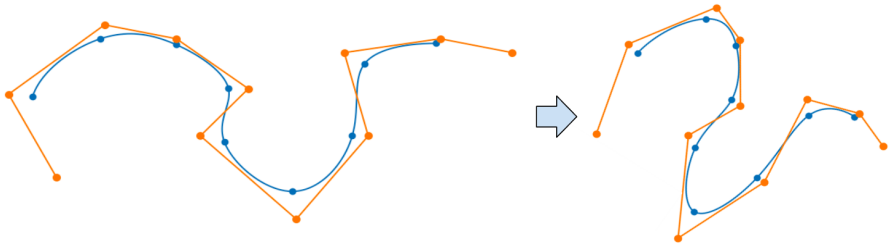


Figure 2.8: Translation, rotation and scaling on the control points

## 2.3 Other Parametric Representations

There are several parametric representations with properties beneficial for path generation. Two of these include Bezier curves and Minvo curves. Both

of these parametric representations use control points with tighter bounds around the curve. Minvo control points create the smallest bounded area around a parametric curve [30]. The bounded area of the control points of the Bezier curve is not as small; however, the end points of the curve are equal to the terminal control points [28]. This is useful for path generation because it allows the creation of tighter control point bounds around a path. This results in less conservative constraints and therefore shorter paths around obstacles. This also enables less conservative maximum bounds for trajectory derivatives, such as velocity and acceleration.

### 2.3.1 Bezier Curves

A Bezier curve is a parametric curve or polynomial of degree $k$ [28]. It is defined between the interval $[t_0, t_0 + \alpha]$. A set of discrete control points $\boldsymbol{\beta}$ shape the curve. Figure 2.9 shows Bezier curves of three different orders. Note that the endpoints of the curve are equal to the first and last control points.

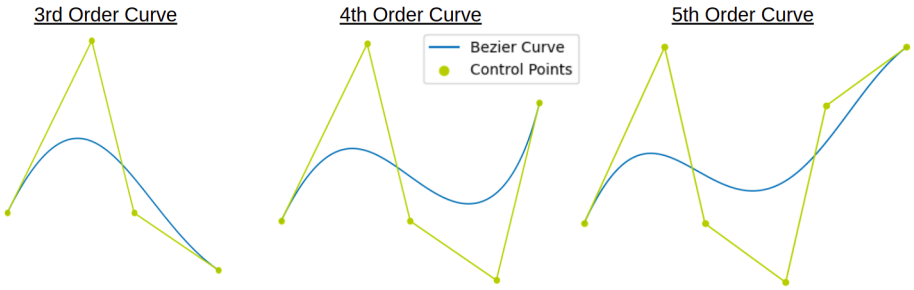



Figure 2.9: Bezier curves of order three through five

We can evaluate a Bezier curve using the matrix method. The matrix equation for a Bezier curve is

$$B(t) = \boldsymbol{\beta} M_B L \tag{2.29}$$

where

$$M_B \in \mathbb{R}^{k+1 \times k+1} \tag{2.30}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 & \beta_1 & \cdots & \beta_k \end{bmatrix} \tag{2.31}$$

$$L = \begin{bmatrix} \tau^k \\ \vdots \\ \tau^2 \\ \tau \\ 1 \end{bmatrix} \tag{2.32}$$

and

$$\tau = \frac{t - t_0}{\alpha}, \quad t_0 \leq t < (t_0 + \alpha) \tag{2.33}$$

The $M_B$ matrices for a first- through fifth-order Bezier curve are given in Section A.1 of the appendix. These values were derived in the same way that the matrices for B-spline curves were evaluated in Section 2.1.5. However, the symbolic and explicit polynomial form of a Bezier curve is evaluated using the Bernstein basis function [31].

### 2.3.2 Minvo Curves

Minvo curve representation is valuable because it gives the smallest convex simplex, or bounding area of the curve [30]. A Minvo curve is a parametric curve or polynomial of degree $k$, defined between the interval $[t_0, t_0 + \alpha]$. A set of discrete control points $\boldsymbol{\nu}$ shape the curve. Figure 2.9 shows Minvo curves of three different orders. Note how tightly the control points bound the curve in comparison to previous images of B-spline and Bezier-curve control points.

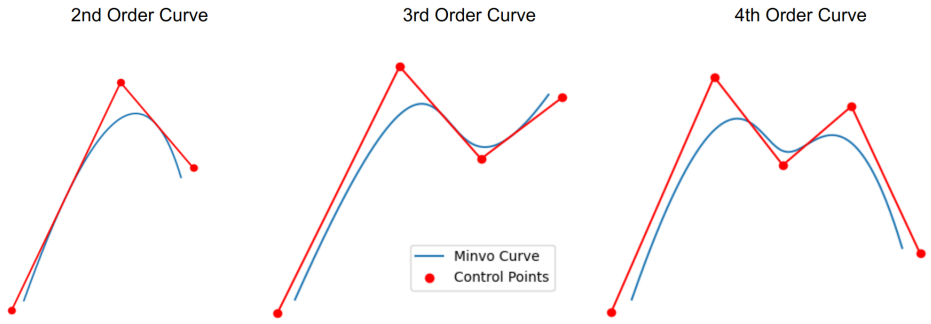



Figure 2.10: Minvo curves of order two through four

The equation for the matrix evaluation of a Minvo curve is

$$V(t) = \boldsymbol{\nu} M_V L \tag{2.34}$$

where

$$M_V \in \mathbb{R}^{k+1 \times k+1} \tag{2.35}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \nu_0 & \nu_1 & \cdots & \nu_k \end{bmatrix} \tag{2.36}$$

$$L = \begin{bmatrix} \tau^k \\ \vdots \\ \tau^2 \\ \tau \\ 1 \end{bmatrix} \tag{2.37}$$

and

$$\tau = \frac{t - t_0}{\alpha}, \quad t_0 \leq t < (t_0 + \alpha) \tag{2.38}$$

The $M_V$ matrices are obtained from Jesus Tordesillas' work in [30]. The values of these matrices are given in Section A.2 of the appendix.

### 2.3.3 Conversion to Other Representations

Converting between parametric representations allows us to take advantage of each representation for a single path generation algorithm. For example, we can leverage the property of a continuous piecewise polynomial of a B-spline, while also taking advantage of the smallest simplex property of a Minvo curve.

A B-spline is composed of $n - k$ polynomial curves where the endpoints connecting each curve are differentiable up to the $(k-1)^{\text{th}}$ derivative. Each of these curves can be transformed into a Bezier or Minvo representation. This means that the control points $\mathbf{P}_i$ that define a knot interval can be transformed into Bezier control points $\boldsymbol{\beta}$ or Minvo control points $\boldsymbol{\nu}$ [32].

#### Converting to Bezier Curves

Figure 2.11 shows a set of B-spline control points converted to Bezier control points. The equation to transform a set of B-spline control points to Bezier control points is

$$\boldsymbol{\beta} = \mathbf{P}_i F_\beta \tag{2.39}$$

or

$$\begin{bmatrix}\beta_0 & \beta_1 & \cdots & \beta_k\end{bmatrix} = \begin{bmatrix}P_i & P_{i+1} & \cdots & P_{i+k}\end{bmatrix} F_\beta \tag{2.40}$$

where $F_\beta$ is a transformation matrix.

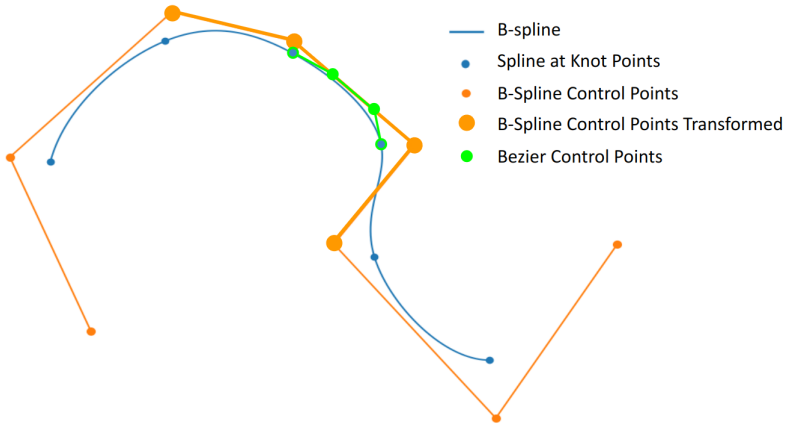



Figure 2.11: Set of B-spline control points converted to Bezier control points

The matrix $F_\beta$ is derived by setting a B-spline curve representation equal to its respective Bezier curve representation at $k+1$ points in time. The symbolic form of this equation for a third-order spline is expressed as

$$\boldsymbol{\beta}\begin{bmatrix}M_B L_0 & M_B L_1 & M_B L_2 & M_B L_3\end{bmatrix} = \boldsymbol{P_i}\begin{bmatrix}ML_0 & ML_1 & ML_2 & ML_3\end{bmatrix} \tag{2.41}$$

Taking the inverse of the matrix to the right of $\boldsymbol{\beta}$ gives

$$\begin{aligned}\boldsymbol{\beta} &= \boldsymbol{P_i}\begin{bmatrix}ML_0 & ML_1 & ML_2 & ML_3\end{bmatrix}\begin{bmatrix}M_B L_0 & M_B L_1 & M_B L_2 & M_B L_3\end{bmatrix}^{-1} \\ \boldsymbol{\beta} &= \boldsymbol{P_i} F_\beta\end{aligned} \tag{2.42}$$

The values of $F_\beta$ for curves of order one through five are given in Section A.3 of the appendix.

### Converting to Minvo Curves

Figure 2.12 shows a set of B-spline control points converted to Minvo control points.

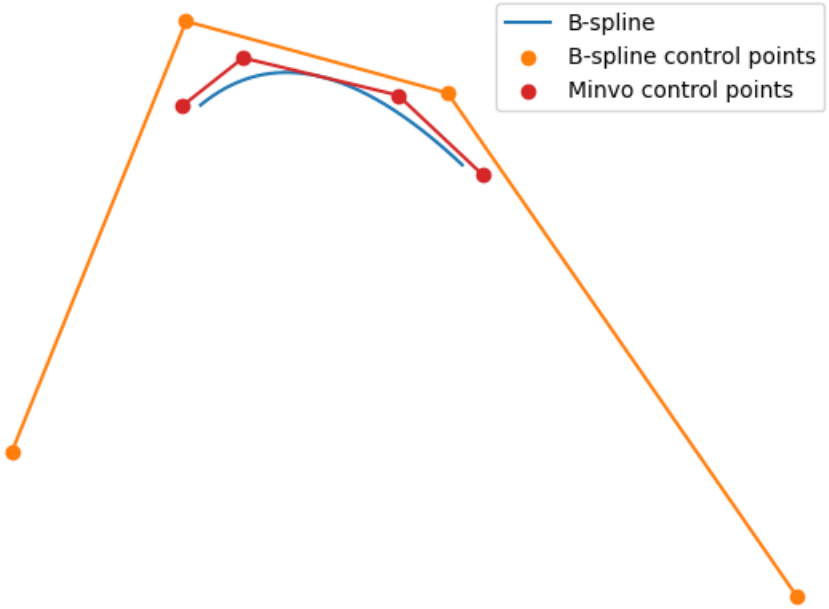



Figure 2.12: Set of B-spline control points converted to Minvo control points.

The equation to transform a set of B-spline control points to Minvo control points is

$$\boldsymbol{\nu} = \mathbf{P}_i F_\nu \tag{2.43}$$

or

$$\begin{bmatrix} \nu_0 & \nu_1 & \cdots & \nu_k \end{bmatrix} = \begin{bmatrix} P_i & P_{i+1} & \cdots & P_{i+k} \end{bmatrix} F_\nu \tag{2.44}$$

where $F_\nu$ is a transformation matrix. The values of $F_\nu$ for curves of order one through five are given in Section A.4 of the appendix. The values for $F_\nu$ can be derived in the same way as the values for $F_\beta$ were derived in Section 2.3.3.

## 2.4 B-Spline Trajectory Objectives and Constraints

Optimization is commonly used to create B-spline paths that accomplish some objective. This objective is formulated mathematically in the form of a cost function or objective function. Restrictions to the path are also formulated mathematically as constraints.

### 2.4.1 B-Spline Objectives Functions

A common cost function for generating B-spline paths is a smoothing spline over lines created from desired point-to-point goal locations [33].

This method smooths out an already calculated path. Another common objective function is the integral of the squared derivative of the spline [34]. This is done to minimize aspects of the trajectory such as jerk or snap. Unfortunately, these objective functions are often complicated and require discretized information of the path. Currently, the most common objective functions for spline trajectory generation involves minimizing the squared magnitude of the control-point derivatives [15], [24], [25]. With this approach we can create a spline shape that minimizes the magnitude of any derivative of the spline such as velocity, acceleration, jerk, or snap. This approach is advantageous because it is quadratic and uses a minimal amount of optimization variables. However, most of these methods determine the knot interval spacing value $\alpha$ prior to optimization. In our approach, we include the scale factor $\alpha$ as an optimization variable, thus simultaneously minimizing time. We also scale the control-point derivatives by the scale factor to achieve a quadratic convex objective function.

The objective function for the $r^{\text{th}}$ control-point derivative will have the form

$$\min_{P_i,\alpha} J = \|\alpha^r P_1^{(r)}\|^2 + \|\alpha^r P_2^{(r)}\|^2 + \cdots \|\alpha^r P_n^{(r)}\|^2 + \alpha^2 \tag{2.45}$$

The last term in this cost function minimizes time, and the other terms minimize the magnitude of the control point derivatives. This is the general form that will be used for each of the cost functions described in the next few subsections.

### Minimizing the Velocity Control Points

Minimizing the velocity control-point magnitudes is useful for creating shortest-path trajectories. When minimizing the velocity control points of a B-spline, the control-point portion of Equation 2.45 becomes

$$\begin{aligned} \min_{P_i} J_{\text{vel}} &= \|\alpha P_1'\|^2 + \|\alpha P_2'\|^2 + \cdots \|\alpha P_n'\|^2 \\ &= \|P_2 - P_1\|^2 + \|P_3 - P_2\|^2 + \cdots \|P_n - P_{n-1}\|^2 \end{aligned} \tag{2.46}$$

Note that this is equivalent to minimizing the distance between consecutive control points.

### Minimizing the Acceleration Control Points

Minimizing the acceleration control points is useful to create a trajectory with minimal change in velocity. This is useful for creating trajectories that approach a constant velocity. The equation to minimize the acceleration control points is

$$\begin{aligned} \min_{P_i} J_{\text{accel}} &= \|\alpha^2 P_1''\|^2 + \|\alpha^2 P_2''\|^2 + \cdots \|\alpha^2 P_n''\|^2 \\ &= \|P_3 - 2P_2 + P_1\|^2 + \|P_4 - 2P_3 + P_2\|^2 + \cdots \|P_n - 2P_{n-1} + P_{n-2}\|^2 \end{aligned} \tag{2.47}$$

### Minimizing the Jerk Control Points

Minimizing jerk or snap is useful for generating trajectories that use minimal energy. In this section, we will show how to minimize the magnitudes of the

jerk control points. The equation is

$$\begin{aligned}\min_{P_i} J_{\text{jerk}} &= \|\alpha^3 P_1'''\|^2 + \|\alpha^3 P_2'''\|^2 + \cdots \|\alpha^3 P_n'''\|^2 \\ &= \|P_4 - 3P_3 + 3P_2 - P_1\|^2 + \cdots \|P_n - 3P_{n-1} + 3P_{n-2} - P_{n-3}\|^2\end{aligned} \tag{2.48}$$

### 2.4.2 B-Spline Constraints

Using the matrix evaluation of B-splines from Section 2.1.5, we can constrain the start and end conditions of a spline, specifically the position, velocity, acceleration, and direction. This is important for trajectory planning, so that the start of the trajectory can be constrained to the current state of the vehicle. This also allows the trajectory to achieve the target conditions at the end of the path.

We can also produce dynamically feasible trajectories by constraining the derivatives of the spline, thereby bounding the maximum velocity and acceleration. The most common and efficient way to do this is to constrain the magnitude of the control-point derivatives [33]. The bounds produced can be made less conservative by converting the B-spline control points to Minvo control points in the constraint setup [15], [24].

#### Initial and Terminal Position Constraints

Using Equation 2.19, and setting $t = t_j = t_0$ and $P_i = P_0$, we can constrain the initial position of the spline to point $\mathbf{W}_0$. The linear equality constraint is

$$\mathbf{W}_0 = \begin{bmatrix} P_0 & P_1 & \cdots & P_k \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{k\times 1} \\ \\ 1 \end{bmatrix} \tag{2.49}$$

If we set $t = t_f$ and $P_{i+k} = P_f$ we can constrain the end position to a waypoint $\mathbf{W}_f$. Note that $t - t_j = \alpha$. The linear equality constraint for the end position is

$$\mathbf{W}_f = \begin{bmatrix} P_{f-k} & P_{f-k+1} & \cdots & P_f \end{bmatrix} M \begin{bmatrix} \mathbf{1} \in \mathbb{R}^{k+1\times 1} \end{bmatrix} \tag{2.50}$$

#### Initial and Terminal Velocity Constraints

Using Equation 2.26, and setting $t = t_j = t_0$, and $P_i = P_0$, we can constrain the initial velocity of the spline to $\mathbf{V}_0$. The linear equality constraint is

$$\mathbf{V}_0 = \begin{bmatrix} P_0 & P_1 & \cdots & P_k \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{k-1\times 1} \\ 1 \\ 0 \end{bmatrix} \frac{1}{\alpha} \tag{2.51}$$

If we set $t = t_f$ and $P_{i+k} = P_f$ we can constrain the end velocity to $\mathbf{V}_f$. Recall again that $t - t_j = \alpha$. The linear equality constraint for the end velocity is

$$\mathbf{V}_f = \begin{bmatrix} P_{f-k} & P_{f-k+1} & \cdots & P_f \end{bmatrix} M \begin{bmatrix} k \\ k-1 \\ \vdots \\ 2 \\ 1 \\ 0 \end{bmatrix} \frac{1}{\alpha} \tag{2.52}$$

### Initial and Terminal Acceleration Constraints

Following the same principles from the previous section we can constrain the initial acceleration of the spline as

$$\mathbf{a}_0 = \begin{bmatrix} P_0 & P_1 & \cdots & P_k \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{k-2\times 1} \\ 2 \\ \mathbf{0} \in \mathbb{R}^{2\times 1} \end{bmatrix} \frac{1}{\alpha^2} \tag{2.53}$$

Also the final acceleration of the spline is constrained as

$$\mathbf{a}_f = \begin{bmatrix} P_{f-k} & P_{f-k+1} & \cdots & P_f \end{bmatrix} M \begin{bmatrix} \prod_{j=k-1}^{k} j \\ \prod_{j=k-2}^{k-1} j \\ \vdots \\ 6 \\ 2 \\ 0 \end{bmatrix} \frac{1}{\alpha^2} \tag{2.54}$$

Unlike the position and velocity constraints, this constraint is non-linear because the scale factor $\alpha$ is squared.

### Initial and Terminal Direction Constraints

Constraining the tangent direction of the spline at a point is similar to constraining the velocity, except that we introduce two new scaling parameters $s_0$ and $s_f$. We include these parameters as optimization variables so that the direction can be constrained without affecting the velocity magnitude.

Following similar methods to the initial velocity constraint, we set the initial direction to $\mathbf{D}_0$. The linear equality constraint is

$$\mathbf{D}_0 = \begin{bmatrix} P_0 & P_1 & \cdots & P_k \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{k-1\times 1} \\ 1 \\ 0 \end{bmatrix} \frac{1}{s_0} \tag{2.55}$$

The linear terminal direction constraint is

$$\mathbf{D}_f = \begin{bmatrix} P_{f-k} & P_{f-k+1} & \cdots & P_f \end{bmatrix} M \begin{bmatrix} k \\ k-1 \\ \vdots \\ 2 \\ 1 \\ 0 \end{bmatrix} \frac{1}{s_f} \tag{2.56}$$

### Initial and Terminal Curvature Constraints

To constrain the curvature at a point on the path, we must constrain the tangent direction and the acceleration direction at that point. This allows us to specify the direction the path is turning in. Constraints for the tangent

direction were described in the previous subsection. Here we describe how to constrain the acceleration direction.

Using the same scaling parameter $s_0$ as the initial direction constraint, the initial acceleration direction is constrained as

$$\mathbf{D}_{a,0} = \begin{bmatrix} P_0 & P_1 & \cdots & P_k \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{k-2\times 1} \\ 2 \\ \mathbf{0} \in \mathbb{R}^{2\times 1} \end{bmatrix} \frac{1}{s_0^2} \tag{2.57}$$

Using the same scaling parameter $s_f$ as the terminal direction constraint, the terminal acceleration constraint is expressed as

$$\mathbf{D}_{a,f} = \begin{bmatrix} P_{f-k} & P_{f-k+1} & \cdots & P_f \end{bmatrix} M \begin{bmatrix} \prod_{j=k-1}^{k} j \\ \prod_{j=k-2}^{k-1} j \\ \vdots \\ 6 \\ 2 \\ 0 \end{bmatrix} \frac{1}{s_f^2} \tag{2.58}$$

Like the acceleration constraints, these constraints are nonlinear because the scaling parameters $s_0$ and $s_f$ are squared.

### Maximum Velocity Constraint

The maximum derivative of a spline is restricted by constraining the magnitude of the control-point derivatives. For the velocity, this results in a quadratic constraint. Given a maximum velocity $V_{\max}$, the constraint for each velocity control point is

$$\begin{aligned} \|P_i'\|^2 &= \left\| \frac{P_{i+1} - P_i}{\alpha} \right\|^2 \leq V_{\max}^2 \\ \|P_{i+1} - P_i\|^2 &\leq \alpha^2 V_{\max}^2 \end{aligned} \tag{2.59}$$

For tighter and less conservative bounds, we constrain the Minvo control-point derivatives. Each interval of the spline has $k+1-r$ Minvo control-point derivatives, where $r$ is the derivative order. The Minvo velocity control points are equal to

$$\begin{bmatrix} v_{0,i}' & v_{1,i}' & \cdots & v_{k-1,i}' \end{bmatrix} = \begin{bmatrix} P_i' & P_{i+1}' & \cdots & P_{i+k-1}' \end{bmatrix} F_{v,k-1} \tag{2.60}$$

and the constraint for each point is

$$\|v_{j,i}'\|^2 \leq V_{\max}^2 \tag{2.61}$$

Note that Bezier curve control points could also be used and substituted for Minvo control points.

### Maximum Acceleration Constraint

Given a maximum acceleration $a_{\max}$, the acceleration magnitude constraint for a B-spline is

$$\begin{aligned} \|P_i''\|^2 &= \left\| \frac{P_{i+2} - 2P_{i+1} - 2P_i}{\alpha^2} \right\|^2 \leq a_{\max}^2 \\ \|P_{i+2} - 2P_{i+1} - 2P_i\|^2 &\leq \alpha^4 a_{\max}^2 \end{aligned} \tag{2.62}$$

We can also constrain the acceleration using the Minvo acceleration control points. The Minvo acceleration control points for interval $i$ are equal to

$$\begin{bmatrix} v_{0,i}'' & v_{1,i}'' & \cdots & v_{k-2,i}' \end{bmatrix} = \begin{bmatrix} P_i'' & P_{i+1}'' & \cdots & P_{i+k-2}'' \end{bmatrix} F_{v,k-2} \tag{2.63}$$

The constraint for each Minvo acceleration control point is

$$\|v_{j,i}''\|^2 \leq a_{\max}^2 \tag{2.64}$$

### Trajectory Optimization Example

Figure 2.13 shows an example of a third-order B-spline trajectory generated by minimizing the velocity control points and scale factor. It is constrained by maximum velocity and maximum acceleration. The start and end points are also constrained in position, velocity, and acceleration. The goal of this optimization was to create a shortest-distance and shortest-time trajectory that is dynamically feasible. Since we include the scale factor as an optimization variable in the acceleration constraint, the optimization is nonlinear and non-convex. If the scale factor $\alpha$ was predetermined, the optimization problem would be convex. The optimization was performed using the Scipy optimization library.

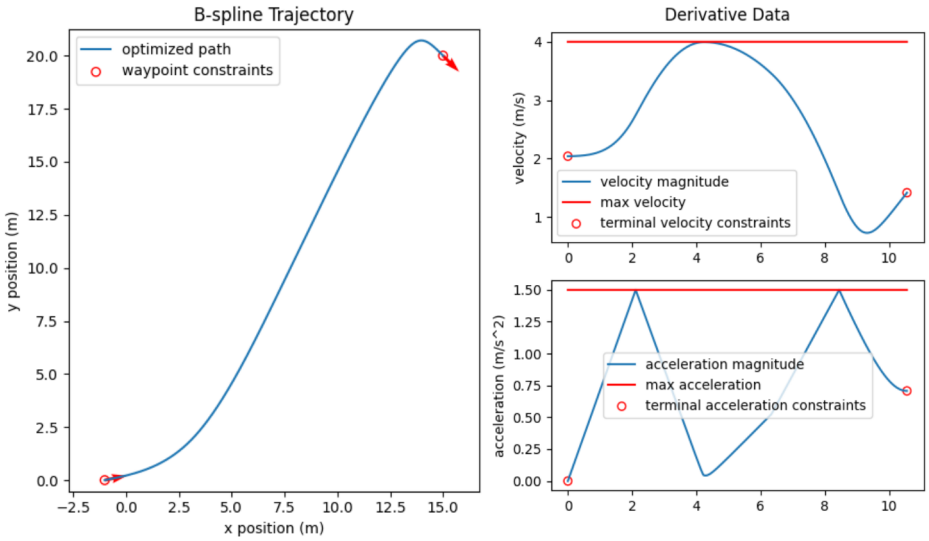



Figure 2.13: Third-order constrained B-spline trajectory

## 2.5 Summary

This chapter introduced B-spline methods for parameterizing, creating, and optimizing trajectories. Methods for evaluating B-splines were presented along with the specific properties of B-splines that make them advantageous for path-planning and trajectory generation. These include properties that enable the generation of continuously differentiable paths that are guaranteed to be feasible within some bounds. These paths are quickly optimized using control points. The chapter also covered the conversion of B-spline control points to control points of other parametric representations, such as Bezier and Minvo control points. These representations are valuable because they introduce a convex set with less conservative bounds around the curves of a B-spline. Lastly, this chapter went over cost functions and constraints specific to generating dynamically feasible B-spline paths. The properties and defining equations from this chapter will be used to define and advance path and trajectory generation algorithms in later chapters.

# 3 Curvature-Constrained B-Spline Path Generation

Curvature-constrained paths are used in various computer-aided applications such as computer graphics, highway design, path generation, and data estimation. This constraint is relevant to path generation for vehicles that are constrained by their turning radius, such as cars and fixed-wing aircraft. Some traditional methods such as the Markov-Dubins approach use straight lines and arcs to generate paths [35]. More recent work combines arcs with monotonic Bezier curves [36]. Spline paths are desirable because they can produce continuous-curvature paths of any shape. However, most spline methods bound the curvature by evaluating the curvature at discrete locations along the curve, and are therefore computationally expensive and do not guarantee feasibility [18], [19]. Other spline methods are limited to two dimensions [37], [38], or work only for second-degree splines [17]. Yang developed a method for generating curvature-constrained and curvature-continuous cubic Bezier curves in real-time; however, this method can only smooth an already existing path and requires special Bezier curves with monotonically increasing curvature. [39]. This chapter describes new methods for generating curvature-constrained paths that are both fast and guarantee curvature boundedness. This chapter also compares these solutions against other legacy methods.

## 3.1 B-Spline Curvature Background

Curvature is defined by the osculating circle, or the circle that is tangent and best approximates the shape of the curve at that point. It is formally defined by the inverse radius of the circle. Figure 3.1 shows the osculating circle at a point along a curve.

The mathematical definition of the curvature of a B-spline $b(t)$ in three dimensions is expressed as

$$C(t) = \frac{\|b(t)' \times b(t)''\|}{\|b(t)'\|^3} \text{ [40]} \tag{3.1}$$

where $b(t)'$ and $b(t)''$ are the first and second derivatives of $b(t)$ with respect to time. For two-dimensional splines, we can add a fictitious zero-valued third dimension to evaluate the cross product. Additionally, using the rules for the derivative of a cross product, and the derivative of a norm, the first

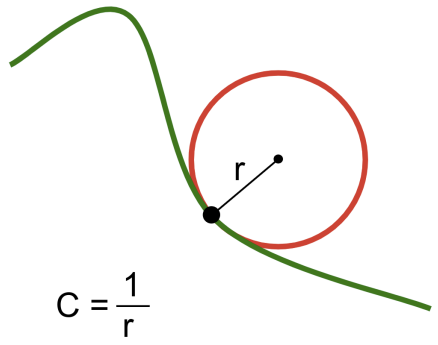



Figure 3.1: Curvature defined by the osculating circle

derivative of the curvature can be expressed as

$$C(t)' = \frac{(b(t)' \times b(t)'') \cdot (b(t)' \times b(t)''')}{\|b(t)' \times b(t)''\| \ \|b(t)'\|^3} - \frac{3(b(t)' \cdot b(t)'')\|b(t)' \times b(t)''\|}{\|b(t)'\|^5} \tag{3.2}$$

Equation (3.2) also uses the third derivative of the spline $b(t)'''$.

Another way to define the curvature is with the portion of the second derivative that is perpendicular to the first derivative. Curvature is equal to the centripetal acceleration divided by the velocity squared. Figure 3.2 shows how centripetal acceleration is related to curvature.

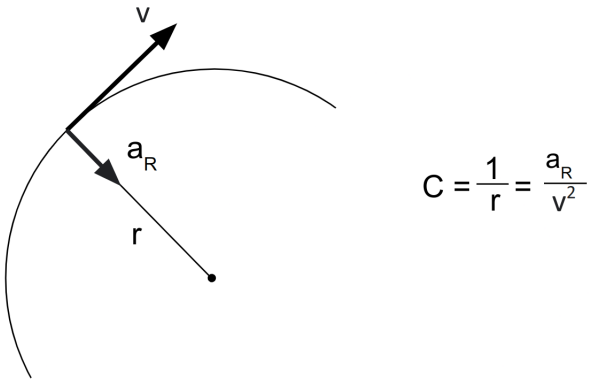



Figure 3.2: Centripetal acceleration equation

The centripetal acceleration of a B-spline is defined as

$$\begin{aligned} b(t)''_R &= b(t)'' - \Big(b(t)' \cdot b(t)''\Big) \frac{b(t)'}{\|b(t)'\|^2} \\ &= \frac{\|b(t)' \times b(t)''\|}{\|b(t)'\|} \end{aligned} \tag{3.3}$$

This is similar to the curvature equation 3.1, but is missing the velocity squared factor in the numerator. In terms of the centripetal acceleration of a

B-spline, the curvature can thus be defined as

$$C(t) = \frac{\|b(t)''_R\|}{\|b(t)'\|^2} \tag{3.4}$$

The equations for curvature and its derivatives will be used to create constraints for curvature in generating paths.

## 3.2 Legacy B-Spline Curvature Evaluation Methods

This section describes various legacy methods for evaluating the curvature bounds of a B-spline. The legacy methods in this subsection will be used as comparative benchmarks for two new methods introduced in the next section. The goal of this research is to produce a new method that is computationally fast and guarantees boundedness, but does not produce overly conservative bounds.

### 3.2.1 Geometric Method for Second-Order Splines

The geometric method is taken from [17]. The idea is that by knowing the position of the control points of a Bezier curve, one can calculate the curvature extrema of the spline segment. This method is computationally efficient, but it only works for second-order splines.

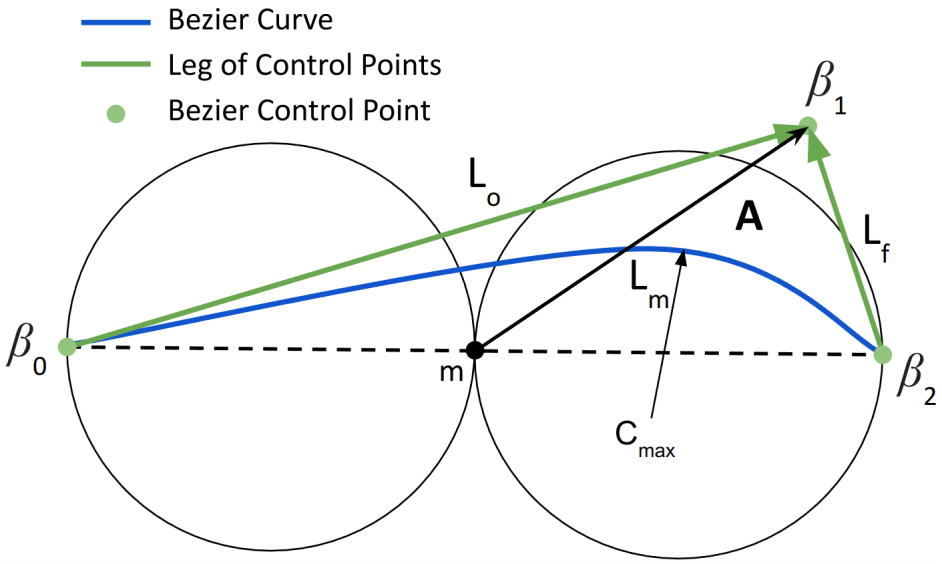



Figure 3.3: Geometric properties of a Bezier curve control points

Given a set of Bezier control points $\boldsymbol{\beta}$ converted from B-spline control

points, the curvature extrema is

$$C_{\max} = \begin{cases} \dfrac{A}{\|\mathbf{L}_0\|^3} & \text{if} \quad \mathbf{L}_0 \cdot \mathbf{L}_m \leq 0 \\ \dfrac{A}{\|\mathbf{L}_f\|^3} & \text{if} \quad \mathbf{L}_m \cdot \mathbf{L}_f \leq 0 \\ \dfrac{\|\mathbf{L}_m\|^3}{A^2} & \text{otherwise} \end{cases} \tag{3.5}$$

where

$$A = \|\mathbf{L}_0 \times \mathbf{L}_f\|/2 \tag{3.6}$$

$$\begin{aligned} \mathbf{L}_0 &= \beta_1 - \beta_0 \\ \mathbf{L}_m &= \beta_1 - m \\ \mathbf{L}_f &= \beta_1 - \beta_2 \end{aligned} \tag{3.7}$$

$$m = (\beta_0 + \beta_2)/2 \tag{3.8}$$

Note that in figure 3.3, $A$ is the area of the triangle formed by the Bezier control points.

### 3.2.2 Root-finding of the Curvature Derivative

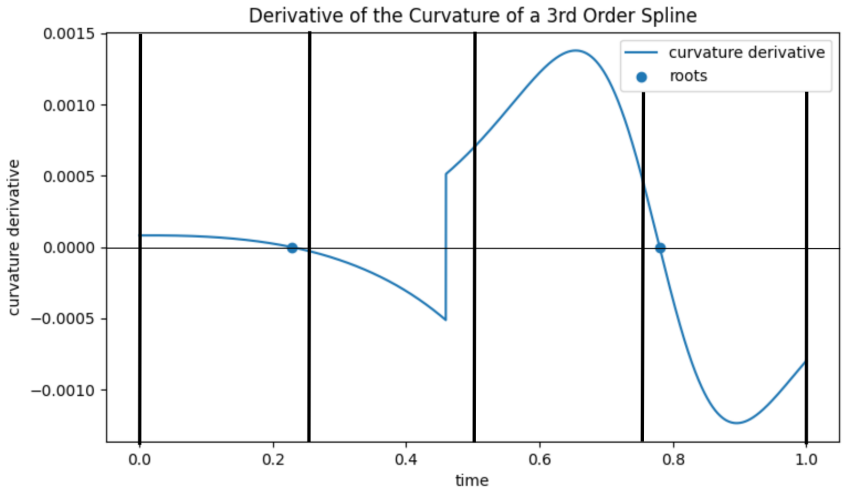



Figure 3.4: Finding roots of the curvature derivative using the Brent-Decker method

This method utilizes the Brent-Dekker optimization to find the roots of the curvature derivative [41], and can be formulated as

$$\begin{aligned} &\text{find roots of} \quad C(t)' = 0 \\ &\text{bounded by} \quad t \in [0, 1] \end{aligned} \tag{3.9}$$

Since we are looking for a global maximum curvature, we split the intervals for which we perform a root search into $k + 1$ intervals or brackets. $k + 1$ is the number of control points that define each interval of the B-spline. By

experimentation, we found that the use of $k+1$ search brackets has shown success in finding the global optimum 98% of time. Figure 3.4 shows the root-finding method performed over one interval of the curvature derivative of a third-order B-spline, using four search brackets. The maximum values are located at the zeros of the curvature derivative.

Searching over multiple intervals makes finding a global solution more likely, but it does not guarantee finding the maximum. Breaking up the optimization into separate intervals also slows the solution process.

### 3.2.3 Maximize the Curvature Equation Using the Newton Method

In this method, the curvature equation is maximized, or the negative of the equation is minimized. This can be expressed as an optimization problem as

$$\begin{aligned} \text{minimize} \quad & -C(t) \\ \text{by varying} \quad & t \in [0,1] \\ \text{with Jacobian} \quad & -C(t)' \end{aligned} \tag{3.10}$$

The Sequential Quadratic Programming approach is used to perform optimization, specifically the Han–Powell quasi–Newton method with a BFGS (Broyden–Fletcher–Goldfarb–Shanno) update of the B–matrix [42]. To find a global maximum, optimization is performed over $k+1$ initial conditions evenly spread over the knot interval. The initial conditions for $t$ are

$$t_{\text{initial}} = \left(0, \frac{1}{k}, \frac{2}{k}, \cdots, \frac{k}{k}\right) \tag{3.11}$$

Figure 3.5 shows the optimization performed over a third-order B-spline. Like the curvature derivative root finding method, it does not guarantee a maxima, and is slowed down by introducing $k$ different start conditions.

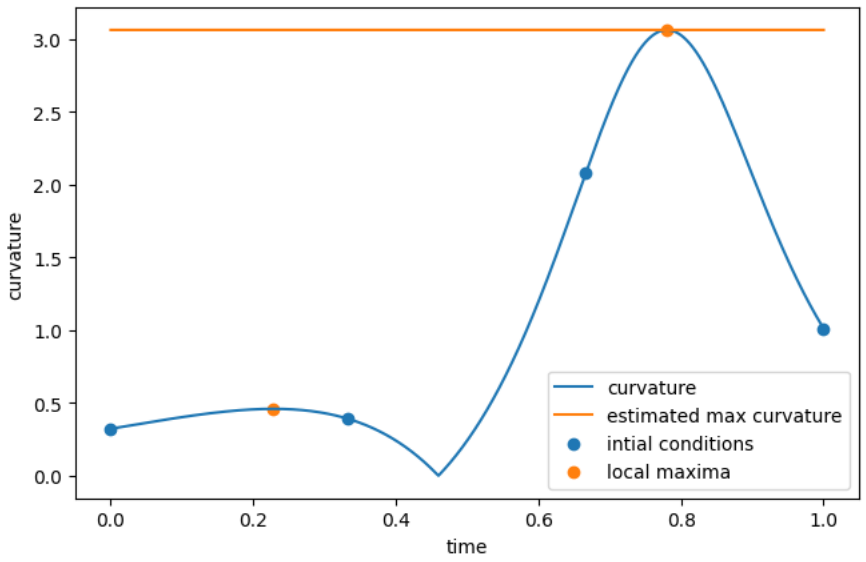



Figure 3.5: Finding the maximum curvature using the SQP approach

### 3.2.4 Evaluating Discrete Points

This method involves finding the curvature at $\eta$ points along the spline interval. Accuracy increases with $\eta$, but so does computation time. Figure 3.6 demonstrates curvature sampling of a third-order spline. This is a brute-force search, or exhaustive search method, and can be computationally expensive.

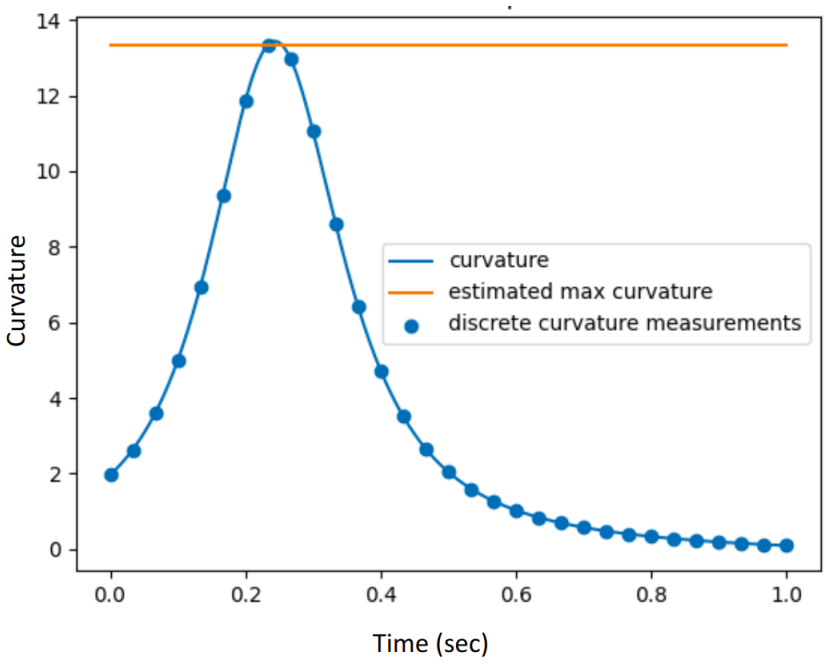



Figure 3.6: Curvature at $\eta$ discrete points

## 3.3 Novel Curvature Evaluation Methods

This section introduces two new methods for calculating an upper bound on the curvature of a B-spline: the analytical roots method and the control-point hull method. These approaches are fast and ensure curvature feasibility. The analytical roots method is fast and feasible because there is a direct solution to finding all the roots of the bounding equations of each interval of the spline. The control-point hull method guarantees boundedness using the convex hull property of B-splines and is efficient because it relies only on the control points of the spline to bound curvature.

### 3.3.1 Upper Bounds from the Curvature Equation

An upper bound on the curvature of a spline can be derived from the maximum value of the numerator and the minimum value of the denominator of the curvature equation, expressed as

$$C(t) = \frac{\|b(t)' \times b(t)''\|}{\|b(t)'\|^3} \leq \frac{\max\{\|b(t)' \times b(t)''\|\}}{\min\{\|b(t)'\|\}^3} \quad \text{for } t_j \leq t \leq t_{j+1} \tag{3.12}$$

Leveraging the centripetal acceleration equation for curvature, we can create a more conservative upper bound for curvature using the maximum

acceleration and minimum velocity of the curve. This upper bound on curvature can be expressed as

$$C(t) = \frac{\|b(t)''_R\|}{\|b(t)'\|^2} \leq \frac{\|b(t)''\|}{\|b(t)'\|^2} \leq \frac{\max\{\|b(t)''\|\}}{\min\{\|b(t)'\|\}^2} \quad \text{for } t_j \leq t \leq t_{j+1} \tag{3.13}$$

When the acceleration vector is nearly parallel to the tangent vector of the spline, equation (3.13) will be overly conservative. However, when the acceleration is perpendicular to the tangent vector, it will create a tight bound. These principles of bounding the curvature equation are used both for the analytical roots method and the control-point hull method.

### 3.3.2 Analytical Roots Method

The maximum numerator and minimum denominator of the curvature equation can be found through their respective roots. Identifying these roots analytically is paramount for creating bounds that can be computed efficiently with low computational complexity. Searching for roots analytically also ensures the discovery of the global maximum for the numerator and the global minimum for the denominator of each interval. This guarantees satisfactory bounds over the curvature. However, since analytical solutions for higher-order polynomials are not possible, this section is limited to finding roots for second- and third-order splines. These are evaluated using the quadratic formula and Cardano's formula, respectively. Cardano's formula is defined in [43].

#### Roots of the Velocity Term

To find the minimum velocity, we must determine the roots of the derivative of the velocity magnitude. We set

$$\frac{d\|b(t)'\|}{dt} = \frac{b(t)' \cdot b(t)''}{\|b(t)'\|} = 0 \tag{3.14}$$

and only need to find the roots of the numerator, so we then set

$$b(t)' \cdot b(t)'' = 0 \tag{3.15}$$

Solving for $t$, we get a set of roots

$$\left[t_1, t_2, ..., t_f\right] \tag{3.16}$$

The minimum velocity for the spline interval is

$$\|b(t)'\|_{\min} = \min\{\|b(t_1)'\|,\ \|b(t_2)'\|, \cdots, \|b(t_f)'\|\} \tag{3.17}$$

We now explain how to solve for the roots of the B-spline. In matrix form, Equation 3.15 is expressed as

$$L'^T M^T \mathbf{P}_i^T \mathbf{P}_i M L'' = 0 \tag{3.18}$$

where

$$\mathbf{P}_i = \begin{bmatrix} P_i & P_{i+1} & P_{i+2} & \cdots & P_{i+k} \end{bmatrix} \tag{3.19}$$

$$L' = \begin{bmatrix} k\tau^{k-1} \\ \vdots \\ 3\tau^2 \\ 2\tau \\ 1 \\ 0 \end{bmatrix} \quad L'' = \begin{bmatrix} k(k-1)\tau^{k-2} \\ \vdots \\ 6\tau \\ 2 \\ 0 \\ 0 \end{bmatrix} \tag{3.20}$$

and

$$\tau = \frac{t - t_j}{\alpha} \tag{3.21}$$

The $M$ matrices are defined in Section 2.1.5. For a third-order B-spline ($k = 3$), the left side of equation 3.18 is a third-order polynomial. For a second-order B-spline ($k = 2$), the left side of equation 3.18 is a first-order polynomial. Each polynomial has root solutions in linear time.

By defining the middle quadratic terms of Equation 3.18 as

$$J = M^T \mathbf{P}_i^T \mathbf{P}_i M \tag{3.22}$$

we can express Equation 3.18 in polynomial form. For a second-order spline, Equation 3.18 is expressed as

$$b(t)' \cdot b(t)'' = 8J_{11}\tau + 4J_{21} \tag{3.23}$$

where

$$J = M^T \mathbf{P}_i^T \mathbf{P}_i M = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \tag{3.24}$$

The roots of Equation 3.23 are determined using the quadratic formula. For a third-order spline, Equation 3.18 is expressed as

$$b(t)' \cdot b(t)'' = 36J_{11}\tau^3 + (12J_{12} + 24J_{21})\tau^2 + (8J_{22} + 12J_{31})\tau + 4J_{32} \tag{3.25}$$

where

$$J = M^T \mathbf{P}_i^T \mathbf{P}_i M = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ J_{31} & J_{32} & J_{33} & J_{34} \\ J_{41} & J_{42} & J_{43} & J_{44} \end{bmatrix} \tag{3.26}$$

The roots of Equation 3.25 are determined using Cardano's formula. These roots can then be used to find the minimum velocity.

### The Acceleration Term

Instead of finding the roots of the acceleration equation, we note specific properties of the acceleration magnitude of a second- and third-order B-spline. The equation for the acceleration magnitude of a B-spline interval in matrix form is

$$\|b(t)''\| = \sqrt{L''^T M^T \mathbf{P}_i^T \mathbf{P}_i M L''} \tag{3.27}$$

For a second-order B-spline, $L''$ is constant and expressed as

$$L'' = \begin{bmatrix} \frac{2}{\alpha^2} \\ 0 \\ 0 \end{bmatrix} \tag{3.28}$$

Therefore, the acceleration is also constant and expressed as

$$\|b(t)''\| = \frac{1}{\alpha^2}\sqrt{\begin{bmatrix}1 & -2 & 1\end{bmatrix}\mathbf{P}_i^T\mathbf{P}_i\begin{bmatrix}1\\-2\\1\end{bmatrix}} \tag{3.29}$$

For a third-order B-spline, $L''$ is expressed as

$$L'' = \begin{bmatrix}\frac{6(t-t_j)}{\alpha^3}\\ \frac{2}{\alpha^2}\\ 0\\ 0\end{bmatrix} \tag{3.30}$$

Therefore, the acceleration magnitude will be either a linear function or a positive quadratic. This means that the maximum acceleration will be at one of the endpoints of the spline segment. The acceleration magnitude at the start point is expressed as

$$\|b(t_j)''\| = \frac{1}{\alpha^2}\sqrt{\begin{bmatrix}1 & -2 & 1 & 0\end{bmatrix}\mathbf{P}_i^T\mathbf{P}_i\begin{bmatrix}1\\-2\\1\\0\end{bmatrix}} \tag{3.31}$$

and the acceleration magnitude at the endpoint is expressed as

$$\|b(t_{j+1})''\| = \frac{1}{\alpha^2}\sqrt{\begin{bmatrix}0 & 1 & -2 & 1\end{bmatrix}\mathbf{P}_i^T\mathbf{P}_i\begin{bmatrix}0\\1\\-2\\1\end{bmatrix}} \tag{3.32}$$

The maximum acceleration magnitude of a third-order spline segment is written as

$$\|b(t)''\|_{\max} = \max\{\|b(t_j)''\|, \|b(t_{j+1})''\|\} \tag{3.33}$$

### Roots of the Cross Term

To find the maximum value of the cross term magnitude, we find the roots of the derivative. We will call the term inside the numerator of the curvature equation the cross term $A(t)$. The derivative of the cross term magnitude can be expressed as

$$\frac{d\|A(t)\|}{dt} = \frac{d\|b(t)' \times b(t)''\|}{dt} = \frac{\left(b(t)' \times b(t)''\right) \cdot \left(b(t)' \times b(t)'''\right)}{\|b(t)' \times b(t)''\|} \tag{3.34}$$

The roots of Equation 3.34 can be found by setting the numerator equal to zero:

$$\left(b(t)' \times b(t)''\right) \cdot \left(b(t)' \times b(t)'''\right) = 0 \tag{3.35}$$

In terms of control points, this is expressed as

$$\left(\mathbf{P}_i ML' \times \mathbf{P}_i ML''\right)^T \left(\mathbf{P}_i ML' \times \mathbf{P}_i ML'''\right) = 0 \tag{3.36}$$

For a third-order B-spline

$$\mathbf{P_i} = \begin{bmatrix} P_i & P_{i+1} & P_{i+2} & P_{i+3} \end{bmatrix} \tag{3.37}$$

$$L' = \begin{bmatrix} 3\tau^2 \\ 2\tau \\ 1 \\ 0 \end{bmatrix}, \quad L'' = \begin{bmatrix} 6\tau \\ 2 \\ 0 \\ 0 \end{bmatrix}, \quad L''' = \begin{bmatrix} 6 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.38}$$

and

$$M = \frac{1}{12} \begin{bmatrix} -2 & 6 & -6 & 2 \\ 6 & -12 & 0 & 8 \\ -6 & 6 & 6 & 2 \\ 2 & 0 & 0 & 0 \end{bmatrix} \tag{3.39}$$

Using a symbolic solver, the left side of Equation 3.36 can be converted into polynomial form as

$$\left(\mathbf{P}_i M L' \times \mathbf{P}_i M L''\right)^T \left(\mathbf{P}_i M L' \times \mathbf{P}_i M L'''\right) = c_3\tau^3 + c_2\tau^2 + c_1\tau + c_0 \tag{3.40}$$

The coefficients of the polynomial in terms of the B-spline control points $\mathbf{P}$ are defined in Section B.1 of the appendix. The roots $[t_1,\ t_2, \ldots,\ t_f]$ of the spline interval are determined using Cardano's formula. The maximum value for the cross term in a spline interval is given by

$$A_{\max} = \max\{\|A(t_1)\|,\ \|A(t_2)\|, \cdots,\ \|A(t_f)\|\} \tag{3.41}$$

For a second-order B-spline the cross term is constant for each interval. $L'''$ evaluates to zero because it is the third derivative of a second-order vector, and therefore Equation 3.36 is also evaluated to zero. This means that $\|b(t)' \times b(t)''\|$ is constant. The cross term magnitude of a second-order spline is expressed as

$$\begin{aligned} \|A(t)\| &= \|\mathbf{P}_i M L' \times \mathbf{P}_i M L''\| \\ &= \left\| \mathbf{P}_i \begin{bmatrix} \frac{1}{2} & -1 & \frac{1}{2} \\ -1 & 1 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} 2\tau \\ 1 \\ 0 \end{bmatrix} \times \mathbf{P}_i \begin{bmatrix} \frac{1}{2} & -1 & \frac{1}{2} \\ -1 & 1 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \right\| \end{aligned} \tag{3.42}$$

where

$$\mathbf{P_i} = \begin{bmatrix} P_i & P_{i+1} & P_{i+2} \end{bmatrix} \tag{3.43}$$

Using a symbolic solver the result is a constant expressed as

$$\|A\| = \|(P_i \times P_{i+1}) + (P_{i+1} \times P_{i+2}) + (P_{i+2} \times P_{i+0})\| \frac{1}{\alpha^3} \tag{3.44}$$

### Curvature Bounds

Putting the results for the minimum velocity, maximum acceleration, and maximum cross term together, we can define an equation for the curvature bound. Using the maximum acceleration, the curvature bound is expressed as

$$C_{\text{bound}} = \frac{\|b(t)''\|_{\max}}{\|b(t)'\|^2_{\min}} \tag{3.45}$$

Using the cross term, the bound is defined as

$$C_{\text{bound}} = \frac{A(t)_{\max}}{\|b(t)'\|^3_{\min}} \tag{3.46}$$

Recall that Equation 3.45 is conservative when used over straight intervals, and Equation 3.46 is conservative over curvy intervals of the spline.

### 3.3.3 Control-Point Hull Method

This section describes how to use control points to determine a curvature bound for a B-spline interval. Using control points to bound curvature is advantageous because the bounds are guaranteed through the use of the convex hull property of a B-spline. It is also fast because we only need to look at the control points of the spline versus analyzing discrete points along the spline. This method can also be performed over any order B-spline. However, it is likely to produce more conservative bounds than the analytical roots method.

#### Convex Hull of the Control Points

Uniform B-splines ensure that the curve will always reside within the convex hull of its control points. Therefore, the upper and lower bounds of the magnitude of a B-spline will be the maximum and minimum magnitude of the convex hull. This is demonstrated in Figure 3.7.

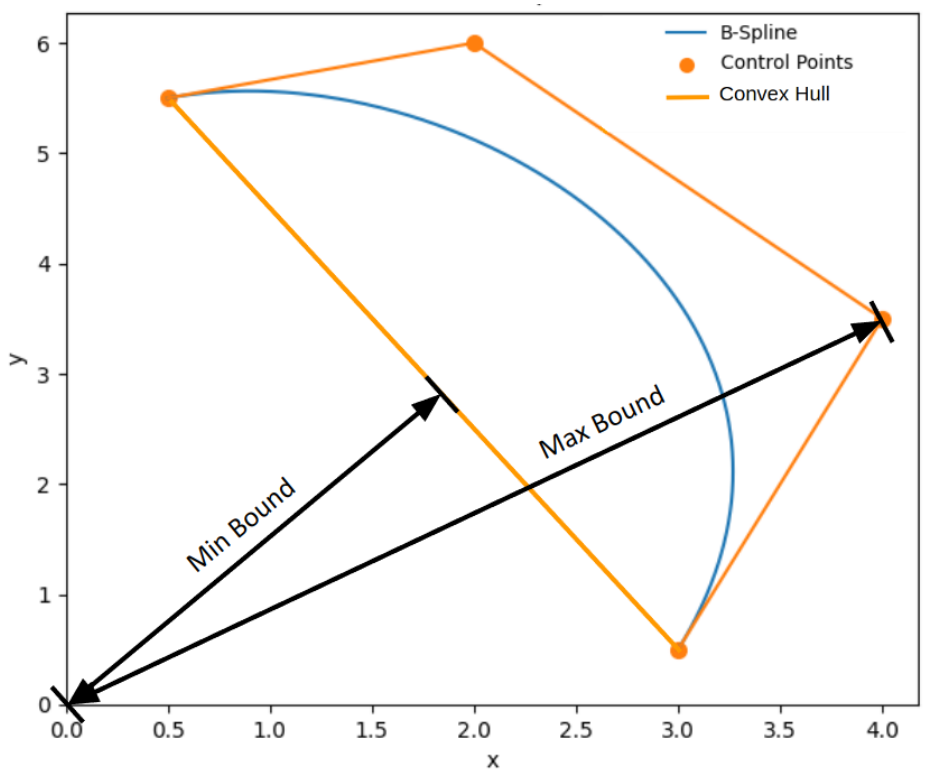



Figure 3.7: Maximum and minimum bounds of a third-order B-spline

**Maximum Bounds of the Convex Hull** The maximum bound will be the norm of the control point furthest from the origin. The maximum is

$$b(t) \leq \max\{\|P_i\|\} \tag{3.47}$$

**Minimum Bounds of the Convex Hull** Finding the minimum bound is a minimum-norm problem [44]. Given a set of control points $\mathbf{P} = \{ P_0,\ P_1, \ldots,\ P_{n-1} \}$, and coefficients $\Lambda = \{ \lambda_0,\ \lambda_1, \ldots,\ \lambda_{n-1} \}$ the convex hull is defined as

$$Z = \{\mathbf{P}\Lambda^T : \text{sum}(\Lambda) = 1,\ \lambda_i \geq 0\} \tag{3.48}$$

and the minimum of the convex hull is

$$\min\{\|z\| \colon z \in Z\} \tag{3.49}$$

Finding the minimum requires the set of coefficients $\Lambda^*$ that minimizes $\|z\|$. Specifically, it requires minimizing $\|\mathbf{P}\Lambda^T\|$) while varying $\Lambda$. One way to do this is to perform a nonlinear optimization.

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{P}\Lambda^T\|^2 \\ \text{by varying} \quad & \Lambda = [\lambda_0,\ \lambda_1, \ldots,\ \lambda_{n-1}] \\ \text{with Jacobian} \quad & \Lambda\mathbf{P}^T\mathbf{P} \\ \text{and constraints} \quad & 0 \leq \lambda_i \leq 1 \\ & \text{sum}(\Lambda) = 1 \end{aligned} \tag{3.50}$$

The fastest approach for this optimization is the Mitchell-Demyanov-Malozemov (MDM) method [44], or the accelerated MDM method. See articles [45] and [46] for a further explanation of the accelerated MDM method. Using this method, we can find the coefficients $\Lambda^*$ that minimize $\|z\|$, and create a minimum bound for the magnitude of the B spline. The minimum bound ensures that

$$\begin{aligned} b(t) &\geq \min(\|z\|) \\ &\geq \|\mathbf{P}\Lambda^{*T}\| \end{aligned} \tag{3.51}$$

### Curvature Bounds Using Control Points

Combining our knowledge of control-point derivatives, and the convex hull properties of a B-spline, we can create bounds for the curvature of a B-spline. The curvature bounds are written as

$$C(t) \leq \frac{\max\{b''(t)\}}{\min\{b'(t)\}^3} \leq \frac{\max\{\|P_i''\|\}}{\|\mathbf{P}'\Lambda^{*T}\|^2} \tag{3.52}$$

Even tighter bounds can be achieved by converting the B-spline control points to Bezier curve control points. A Bezier curve also holds the convex hull property, where the curve is bounded by its control points. If we define $\boldsymbol{\beta}'$ and $\boldsymbol{\beta}''$ as the Bezier control points transformed from $\mathbf{P}'$ and $\mathbf{P}''$ respectively, the curvature bound is given by

$$C(t) \leq \frac{\max\{\|\beta_i''\|\}}{\|\boldsymbol{\beta}'\Lambda^{*T}\|^2} \tag{3.53}$$

where

$$\boldsymbol{\beta}' = \boldsymbol{P}_i' F \tag{3.54}$$

$$\boldsymbol{\beta}'' = \boldsymbol{P}_i'' F \tag{3.55}$$

Minvo control points could also be used to create tighter bounds; however, through experimentation it was found that Bezier control points achieved less conservative bounds on average for B-spline intervals. This is because the terminal control points of a Bezier curve are equal to the end points of the curve, and the spline derivatives often extend radially from the origin. On the other hand, Minvo terminal control points extend beyond the terminal points of the curve.

Using the acceleration control points works well when the acceleration is perpendicular to the tangent velocity of the curve. However, it creates a conservative bound during straight or nearly straight portions of the spline. During these intervals, the cross term $A(t)$ is better suited to find a less conservative bound.

If we evaluate $A$ in terms of $t$ and $\mathbf{P}$, we can rearrange the equation into the B-spline form or the Bezier curve form. This means we can find control points for the cross term and thus maximum bounds for it as well. The set of Bezier control points for $A(t)$ can be defined as

$$\boldsymbol{\beta}_A = \begin{bmatrix} \beta_{A,0} & \beta_{A,1} & \cdots & \beta_{A,k} \end{bmatrix} \tag{3.56}$$

The curvature is bounded by the following expression:

$$C(t) \leq \frac{\max\{\|b(t)' \times b(t)''\|\}}{\min\{b'(t)\}^3} \leq \frac{\max\{\|\beta_{A,i}\|\}}{\|\boldsymbol{\beta}' \Lambda^{*T}\|^3} \tag{3.57}$$

The next subsections describe how to find the control points for the velocity, acceleration, and cross term of a B-spline.

### Evaluating the Velocity Control Points

Individual velocity control points are defined as

$$P_i' = \frac{P_{i+1} - P_i}{\alpha} \tag{3.58}$$

Where for a specific knot interval, the velocity control points are

$$\mathbf{P}_i' = \begin{bmatrix} P_i' & P_{i+1}' & \cdots & P_{i+k}' \end{bmatrix} \tag{3.59}$$

These control points are then converted from B-spline form to Bezier form for less conservative bounds. The Bezier velocity control points are given by

$$\boldsymbol{\beta}' = \boldsymbol{P}_i' F_\beta \tag{3.60}$$

### Evaluating the Acceleration Control Points

Each acceleration control point can be expressed as

$$P_i'' = \frac{P_{i+1}' - P_i'}{\alpha} \tag{3.61}$$

The acceleration control points for each knot interval are

$$\mathbf{P}_i'' = \begin{bmatrix} P_i'' & P_{i+1}'' & \cdots & P_{i+k}'' \end{bmatrix} \tag{3.62}$$

These control points can also be converted to Bezier control points to create tighter bounds. The acceleration Bezier control points are

$$\boldsymbol{\beta}'' = \boldsymbol{P}_i'' F_\beta \tag{3.63}$$

### Evaluating the Cross Term Control Points

The Bezier control points of the cross term $A(t)$ must be represented in terms of the B-spline control points of $b(t)$. For *second-order B-splines*, $A(t)$ is constant for each knot interval and is expressed as

$$A(t) = \Big((P_0 \times P_1) + (P_1 \times P_2) + (P_2 \times P_0)\Big)\frac{1}{\alpha} \tag{3.64}$$

For a *third-order B-spline*, the cross term $A(t)$ is a second-order polynomial expressed as

$$A(t) = c_2\tau^2 + c_1\tau + c_0 \tag{3.65}$$

The coefficients are derived in section B.2.1 of the appendix. In terms of its Bezier control points $\boldsymbol{\beta}$, $A(t)$ is written as

$$\begin{aligned} A(t) &= (1-\tau)^2\beta_0 + 2(1-\tau)\tau\beta_1 + \tau^2\beta_2 \\ &= (\beta_0 - 2\beta_1 + \beta_2)\tau^2 + (2\beta_1 - 2\beta_0)\tau + \beta_0 \end{aligned} \tag{3.66}$$

By setting the coefficients of Equation 3.65 equal to the coefficients of Equation 3.66, we can find the Bezier control points of $A(t)$ in terms of the original B-spline control points. Recall that $\beta_i$ is the $i^{\text{th}}$ Bezier control point of the cross term. The Bezier control points of $A(t)$ are evaluated as

$$\beta_0 = c_0 \tag{3.67}$$

$$\beta_1 = \frac{2\beta_0 + c_1}{2} \tag{3.68}$$

$$\beta_2 = 2\beta_1 - \beta_0 + c_2 \tag{3.69}$$

An advantage of using control points to find the maximum versus using the analytical roots is that we can derive bounds for splines of orders higher than three. Higher-order splines are useful for objective functions that seek to minimize jerk or snap. For a *fourth-order B-spline*, the cross term $A(t)$ is a fourth-order polynomial expressed as

$$A(t) = c_4\tau^4 + c_3\tau^3 + c_2\tau^2 + c_1\tau + c_0 \tag{3.70}$$

where the coefficients are derived in section B.2.2 of the appendix.

In terms of its Bezier control points $\boldsymbol{\beta}$, A(t) is written as

$$\begin{aligned} A(t) =& (1-\tau)^4\beta_0 + 4\tau(1-\tau)^3\beta_1 + 6\tau^2(1-\tau)^2\beta_2 + 4\tau^3(1-\tau)\beta_3 + \tau^4\beta_4 \\ =& (\beta_0 - 4\beta_1 + 6\beta_2 - 4\beta_3 + \beta_4)\tau^4 + (12\beta_1 - 4\beta_0 - 12\beta_2 + 4\beta_3)\tau^3 + \\ & (6\beta_0 - 12\beta_1 + 6\beta_2)\tau^2 + (4\beta_1 - 4\beta_0)\tau + \beta_0 \end{aligned} \tag{3.71}$$

If we set equations (3.71) and (3.70) equal to each other, we can solve for the Bezier control points of $A(t)$. The control points are evaluated as

$$\beta_0 = c_0 \tag{3.72}$$

$$\beta_1 = \frac{c_1 + 4\beta_0}{4} \tag{3.73}$$

$$\beta_2 = \frac{c_2 + 12\beta_1 - 6\beta_0}{6} \tag{3.74}$$

$$\beta_3 = \frac{c_3 + 12\beta_2 + 4\beta_0 - 12\beta_1}{4} \tag{3.75}$$

$$\beta_4 = c_4 + 4\beta_1 + 4\beta_3 - 6\beta_2 - \beta_0 \tag{3.76}$$

The control points for the velocity, acceleration, and cross term can then be used to generate curvature bounds for the B-spline.

## 3.4 Comparison and Results

This section contains test results that compare the performance of the various curvature bounding methods in terms of conservativeness and speed. It also compares the performance of these methods when used during B-spline path generation. Recall that the goal is to create a method that is computationally efficient and guarantees boundedness, but is not overly conservative. The tests were performed on an Intel Core i7-10750H CPU at 2.60 GHz using the Scipy SLSQP library.

### 3.4.1 Evaluating Curvature Maxima

Figures 3.8, 3.9, and 3.10 each contain 1000 curvature evaluation samples for random B-splines of order two, three, and four respectively. A random B-spline was created for each sample by randomly generating control points within some box bounds. Each figure contains a plot showing how well each method bounds the curvature, as well as a time analysis of the speed of the evaluations. The left plot in each figure shows the curvature evaluation error of each method. In this plot, points above the zero line sufficiently bound the curvature, and points below underestimate the maximum curvature. The curvature bounding error is calculated as

$$C_{\text{error}} = C_{\text{est. max}} - C_{\text{true max}} \tag{3.77}$$

The right plot in each figure shows the average computation time and the distribution of the computation time of each method.

For a second-order spline in Figure 3.8, each method performs with nearly the same level of effectiveness in bounding the curvature, except for the discrete method. The geometric method has the shortest computation time on average, and the new methods (5 and 6) have the next shortest computation time. The discrete method has the worst timing performance.

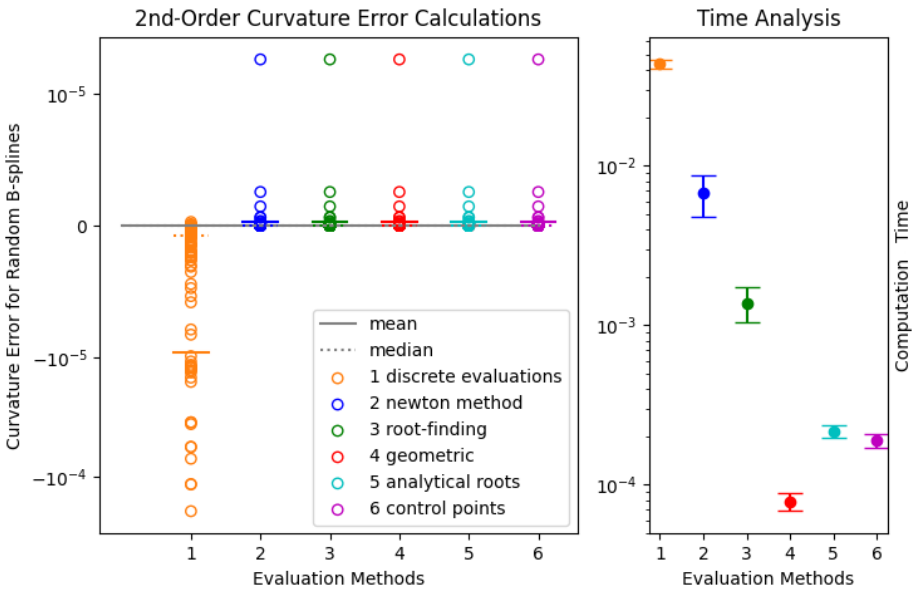



Figure 3.8: Curvature bounding methods for second-order splines

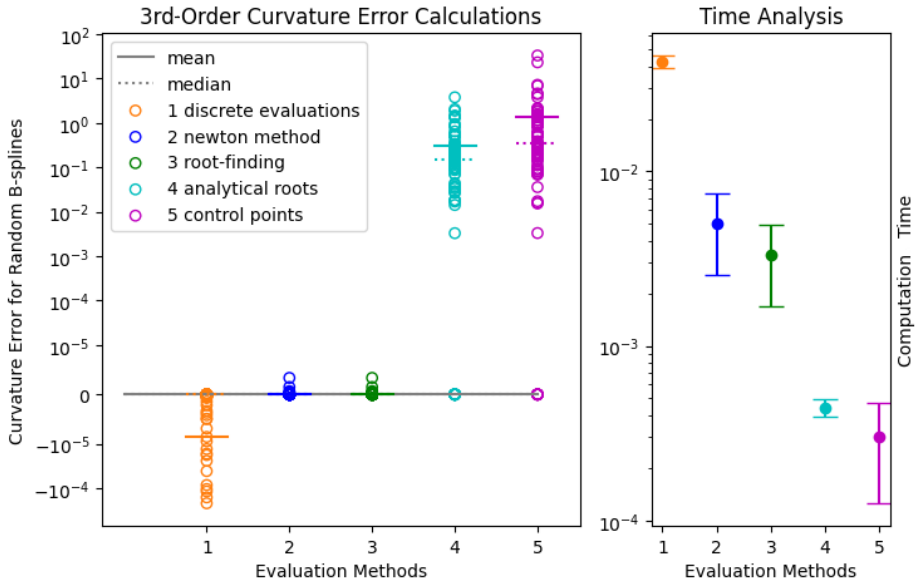



Figure 3.9: Curvature bounding methods for third-order splines

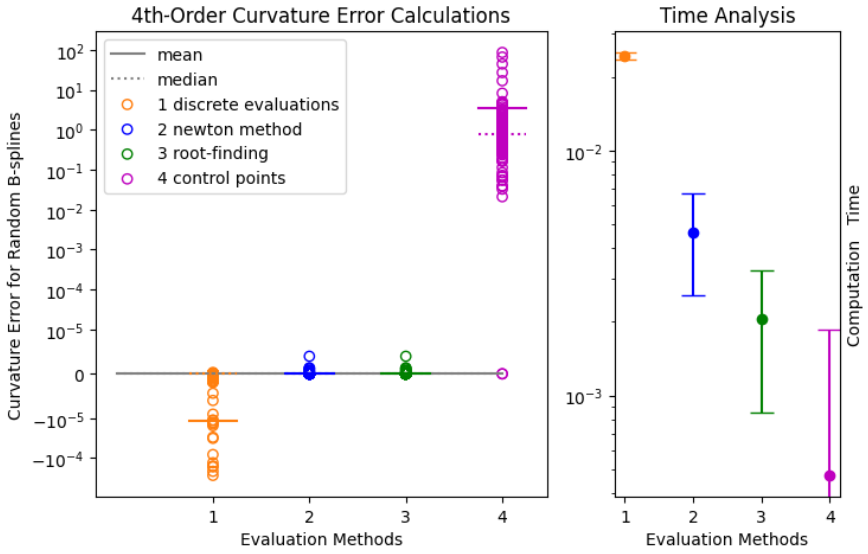



Figure 3.10: Curvature bounding methods for fourth-order splines

In Figure 3.9, third-order spline evaluations exclude the geometric method, as it only works for second-order splines. The most effective methods for creating tight and robust bounds over the curvature are the Newton (2) and numerical roots (3) methods. The new methods (4 and 5) are on average an order of magnitude faster, but they are also more conservative.

The fourth-order spline evaluations exclude the analytical roots method but overall contain similar results to those of the third-order spline. The control-point hull method is the fastest, but more conservative than the Newton and numerical roots method.

### 3.4.2 Generating Curvature-Bounded Paths

This section contains results for path generation using selected curvature bounding methods from the previous section. The level of performance is judged by computational speed, robustness, path length, velocity variance, and how well it bounds the maximum curvature. The geometric method was not considered because it only works for second-order splines and thus does not produce continuous-curvature paths.

The root finding method was chosen as a benchmark because, of all the legacy optimizer methods, it rarely failed to find a feasible solution. The methods evaluated include the analytical roots method and the control-point hull method. These methods were subdivided into two approaches: a direct approach and an indirect approach. The *direct approach* constrains curvature through the inequality

$$\min\left(\frac{\|b''(t)\|_{\max}}{\|b'(t)\|^2_{\min}}, \frac{\|b'(t)\times b''(t)\|_{\max}}{\|b'(t)\|^3_{\min}}\right) \leq C_{\max} \tag{3.78}$$

The *indirect approach* constrains the curvature by constraining the maximum acceleration and the minimum velocity. The minimum velocity constraint is

$$\|b'(t)\| \geq v_{\min} \tag{3.79}$$

and the maximum acceleration constraint is

$$\|b'(t)\| \leq C_{\max} v_{\min}^2 \tag{3.80}$$

Both of these approaches are applied to the analytical roots and control-point hull methods.

### Comparing Bounding Methods for Path Generation

Six different cases are presented in this section to demonstrate the performance of each of the bounding methods. The objective function and endpoint conditions vary for each case. The colors in each graph represent and are unique to a specific bounding method.

Case 1 is a straight path that minimizes the velocity control points of the spline. The results are shown in Figure 3.11. Each method has about the same path-length performance, except for the indirect control-point hull method, which is slightly longer. The most computationally efficient is the indirect analytical roots method. Case 2 is a right-hand turn that minimizes the jerk control-point magnitudes. The results for Case 2 are shown in Figure 3.12. The indirect control-point method has the smallest computation time in this case, but again has the longest path length. All of the methods have comparable standard deviations for velocity.

Figure 3.13 contains results for Case 3, which is a semicircle that minimizes the acceleration control-point magnitudes. All the new methods have about the same level of performance in terms of path length; however, the indirect analytical roots method computes a solution slightly faster than the rest. Case 4 generates a figure eight or circular path and minimizes the velocity control-point magnitudes. See Figure 3.14. The indirect control-point hull method generates an unreasonably long path. The method that generates the shortest path is the analytical root method. The method with the shortest evaluation time is the indirect analytical roots approach.

Figure 3.15 shows the results for Case 5. Case 5 produces an S-shaped path and minimizes the acceleration control-point magnitudes. Both of the control-point hull methods are disqualified because they violate the curvature constraint. The indirect analytical roots approach produces the shortest path and has the smallest evaluation time of the qualified methods. Case 6 produces a hook-shaped path and minimizes the jerk control-point magnitudes. Figure 3.16 shows that the method with the shortest evaluation time for this case is the indirect control-point hull approach. Of the newly developed methods, the analytical roots method takes the longest to compute but produces the shortest path. The indirect analytical roots method produces a long suboptimal path.

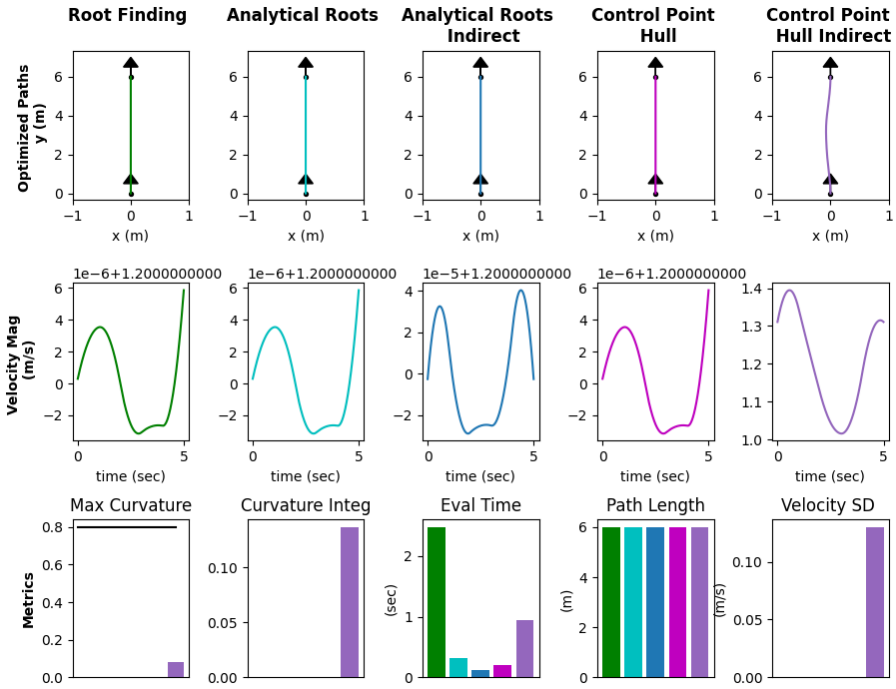



Figure 3.11: Curvature bounding methods comparison - case 1

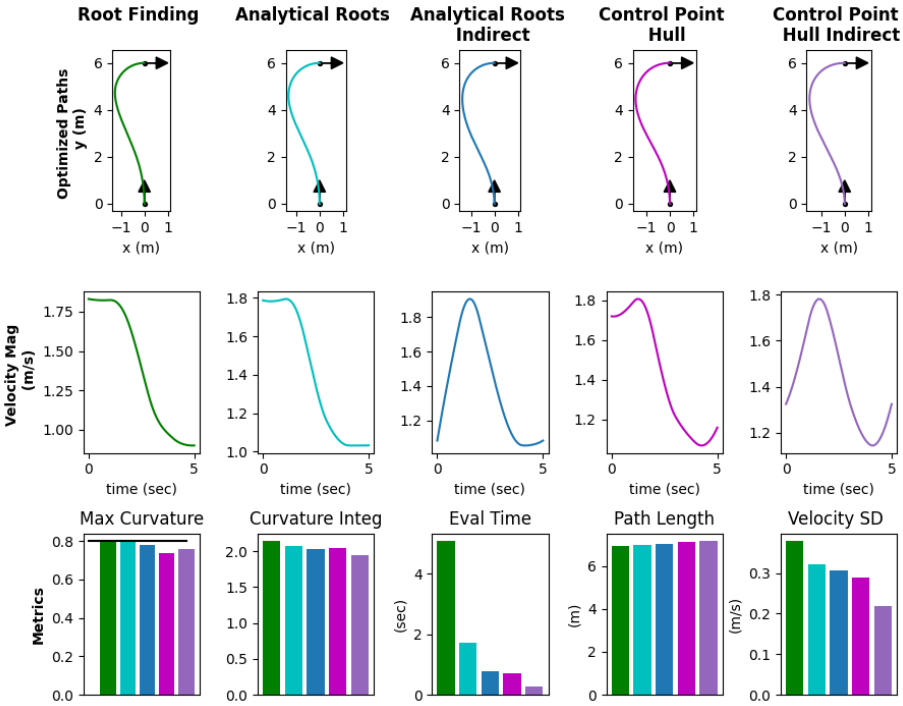



Figure 3.12: Curvature bounding methods comparison - case 2

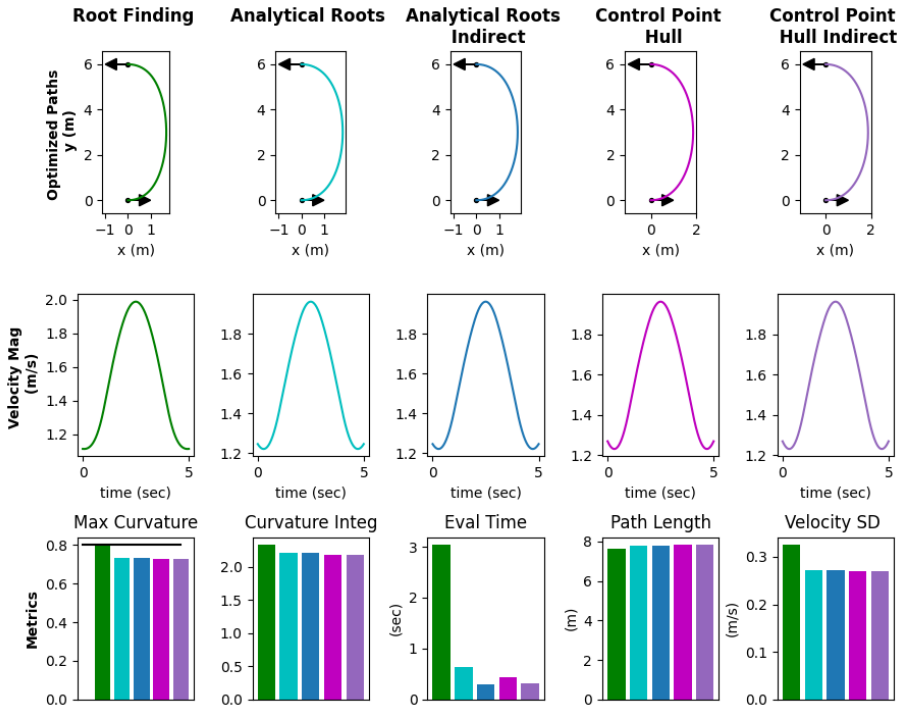



Figure 3.13: Curvature bounding methods comparison - case 3

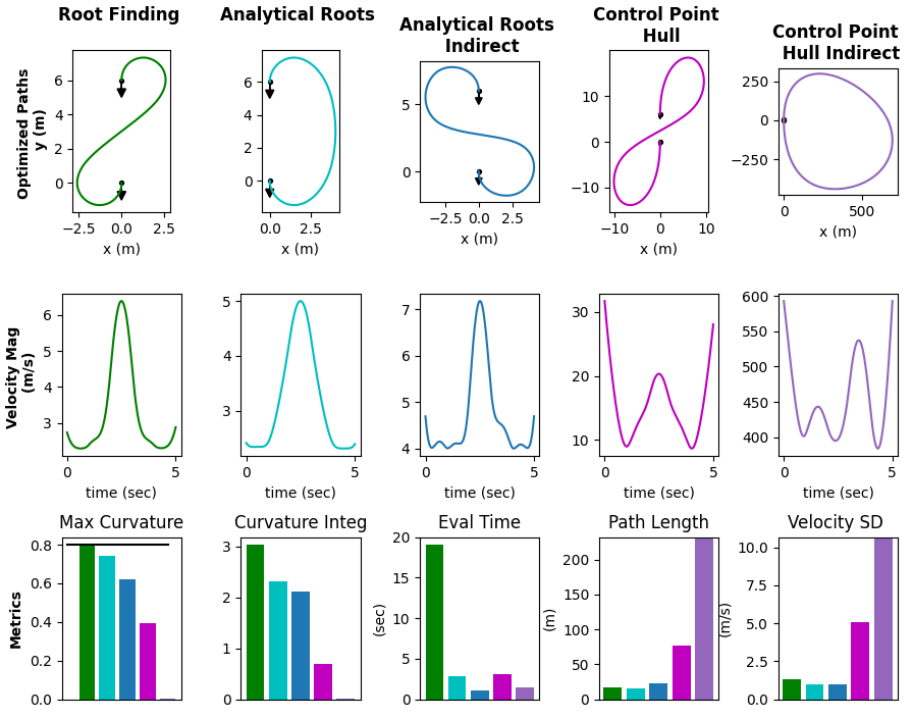



Figure 3.14: Curvature bounding methods comparison - case 4

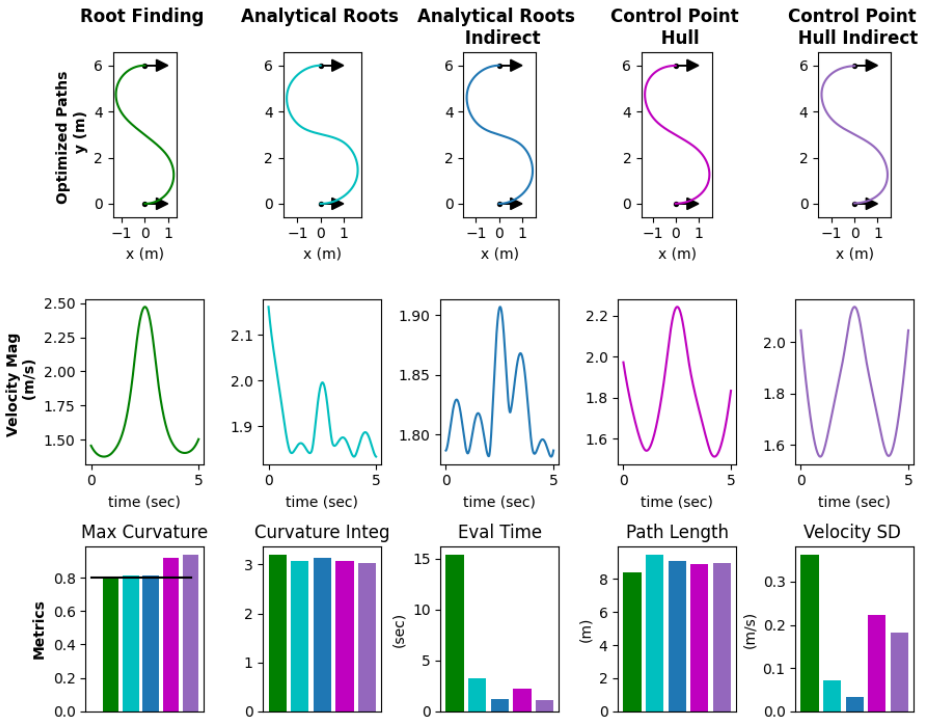



Figure 3.15: Curvature bounding methods comparison - case 5

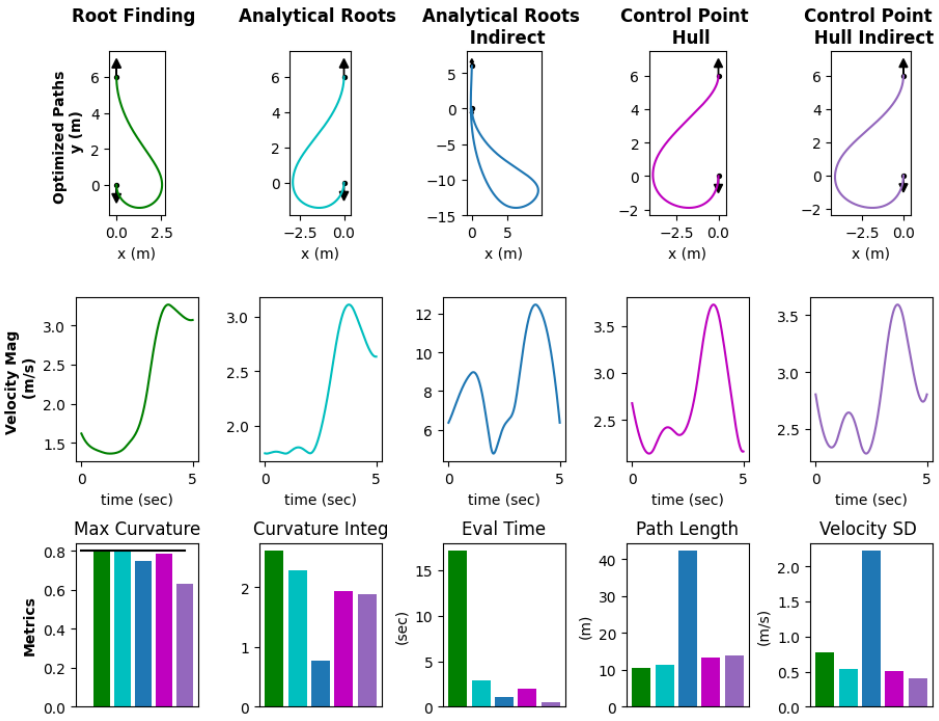



Figure 3.16: Curvature bounding methods comparison - case 6

On average, the most time-efficient method is the indirect control-point hull method. This method, however, occasionally produces nonsensical paths that are either too long or violate the curvature constraint. The direct control-point hull, and indirect analytical roots methods are almost as fast computationally; however, they also occasionally produce undesirable paths. The most robust method is the direct analytical roots method. Although it is on average 1.5 times slower, this method will be used for the rest of the chapters in this thesis because it is the most reliable. It is also still much faster than traditional methods. The direct methods also have the advantage of being viable options for trajectory generation. The indirect methods do not because they have the side effect of independently constraining the minimum velocity and maximum acceleration.

### Comparing Objective Functions

After selecting an optimal curvature bounding method, we also performed tests to select the best optimization function for path generation. Figure 3.17 shows the average results of three different objective functions over 1,000 random cases. Each case has random start and end conditions and a random maximum curvature. The metrics used to compare the objective functions are path length, evaluation time, and standard deviation of the velocity magnitude. The bar graphs are colored cyan to be consistent with the color of the analytical roots method in the previous plots.

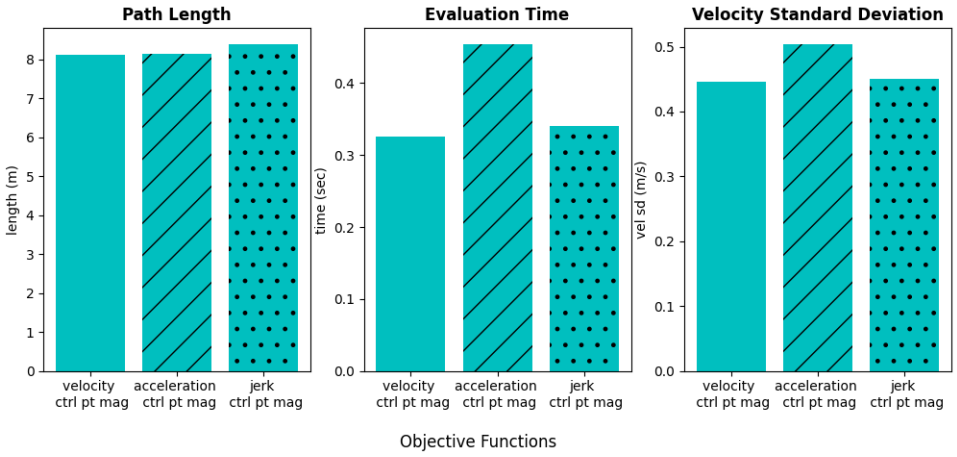



Figure 3.17: Comparing objective functions

The methods that produce the shortest paths on average are the two objectives that minimize the magnitude of the velocity control points and the magnitude of the acceleration control points. However, the most time-efficient objectives are the velocity control-point magnitude and the jerk control-point magnitude objectives. These also produce paths that have the least velocity variation. It is obvious that the optimal objective for shortest path optimization is minimization of the velocity control-point magnitude. For the rest of this thesis, we will minimize the velocity control-point magnitude to optimize B-spline paths.

## 3.5 Summary

This chapter covered the definition of curvature in relation to B-splines. It also covered various methods for evaluating curvature bounds for a B-spline. Two new methods for evaluating curvature bounds were introduced, and tests were performed to benchmark these approaches against traditional methods. These bounding methods were also tested in a path generation framework and optimized for computational efficiency and path length. The method selected for further analysis was the direct analytical roots method, with an objective function that minimizes the magnitude of the velocity control points. This approach was chosen because it was the most robust and, even though it was slightly slower than the other new method, it was still an order of magnitude faster than traditional methods.

# 4 Collision-Free B-Spline Path Generation

## 4.1 Introduction to Collision-Free Path Generation

The ability to generate collision-free paths in real time is crucial for navigation through cluttered environments and confined map spaces. This is especially true when there are moving obstacles, when pose estimates are sporadically corrected, or when tracking a moving target. Collision-free path-generation typically consists of three main steps: detection, mapping, and path-planning. Detection includes sensor and estimation algorithms to identify objects and surroundings in the environment. Mapping entails creating a representation of the surroundings and correlating it with the pose of the vehicle and some inertial reference frame. Path-planning is concerned with course optimality, kinematic feasibility, and the dynamic feasibility of the vehicle.

In this chapter, we focus only on the path-planning portion of collision-free navigation. Specifically, we focus on optimal B-spline path-generation methods that are computationally efficient. This chapter discusses current methods and develops and adds to them.

### 4.1.1 Collision-Free B-Spline Path Methods

Some of the inherent advantages of B-splines for use in collision-free path-generation are their efficiency and safety guarantees. The convex-hull property of B-splines and Minvo curves enable the generation of trajectories that are guaranteed collision-free as long as the control points reside within some convex boundary. This is also more efficient because discrete points along the path do not require inspection.

There are two approaches that prevail to generate collision-free B-spline paths. The first is direct avoidance of obstacles, where there are direct constraints on the curve to maintain a safe distance from the obstacles. For example, Tang [25] ensures a collision-free path by making sure that no control point or intersecting line passes through an obstacle, as well as by making sure that the step length between control points is always shorter than the distance to the nearest obstacle. Tordesillas [15] constrains each B-spline interval to be a safe distance away using segmentation planes placed in front of an obstacle.

The second approach to generate collision-free B-spline paths entails the generation of a sequence of safe flight corridors (SFCs) in which the path should reside. An SFC is any obstacle-free convex shape that the vehicle of interest can safely traverse. In Wang's paper [24], convex polyhedra are used as SFC's to constrain the Minvo basis control points of the B-spline. Park [47]

uses an ellipsoid to constrain the control points of the Bezier curve for each section of the path. Both of their methods are convex, but not necessarily linear.

### 4.1.2 Proposed Solutions to Collision-Free Path Generation

We address both approaches to collision-free B-spline path generation: directly avoiding an obstacle and traversing through safe flight corridors. The approach we take to avoid an obstacle is to generate rotation matrices during the B-spline optimization that define the orientation of bounding planes that separate the obstacles from the control points. The control points of that interval are then constrained a safe distance away along the normal vector of the bounding plane. This is similar to the Tordesillas approach [15], except that the bounding planes are updated at each iteration of the optimization. The approach we take for safe flight corridors is to generate overlapping bounding boxes for each segment of the B-spline path. Contributions to the SFC approach include enabling the rotation of these bounding boxes to any orientation. The next two sections in this chapter describe in detail how to implement these approaches. The last two sections validate them and make recommendations for their use.

## 4.2 Direct Obstacle Avoidance

Direct obstacle avoidance is valuable in situations where there is no general path defined yet. Thus, it has the potential for rapid avoidance of unforeseen obstacles. It can also be less conservative than following an SFC, thus producing more optimal paths. The objective of direct obstacle avoidance is to reach a location or pose as quickly as possible without running into an obstruction. The objective function for this optimization is described in Section 2.4.1 where the optimization variables are the B-spline control points $\mathbf{P}$, and the objective minimizes the distance between the control points to minimize the path distance. For convenience, the objective function is written below as

$$J = \|P_2 - P_1\|^2 + \|P_3 - P_2\|^2 + \cdots \|P_n - P_{n-1}\|^2 \tag{4.1}$$

The biggest computational cost of this problem is estimating the optimal orientation of the bounding plane that separates the obstacle from the spline interval. We estimate the normal vector of the bounding plane by averaging the Minvo control points of the interval of interest, and then evaluating the vector from the center of the obstacle to that average point. Spheres and circles are used to represent obstacles in this section; however, any convex set of points could be substituted as an obstacle.

### 4.2.1 Avoiding Spheres or Circles

The derivation to avoid a sphere or circle is illustrated in Figure 4.1. In the figure, there is an obstacle $j$ and a vector $\mathbf{V}_{j,i}$ that starts from the center of the obstacle $j$ and goes to the mean of Minvo control points $i$ through $i + k$. $\mathbf{V}_{j,i}$ determines the orientation of the normal vector for the bounding plane that

separates the obstacle and Minvo control points. The Minvo control points were calculated from the B-spline control points that define the interval of interest.

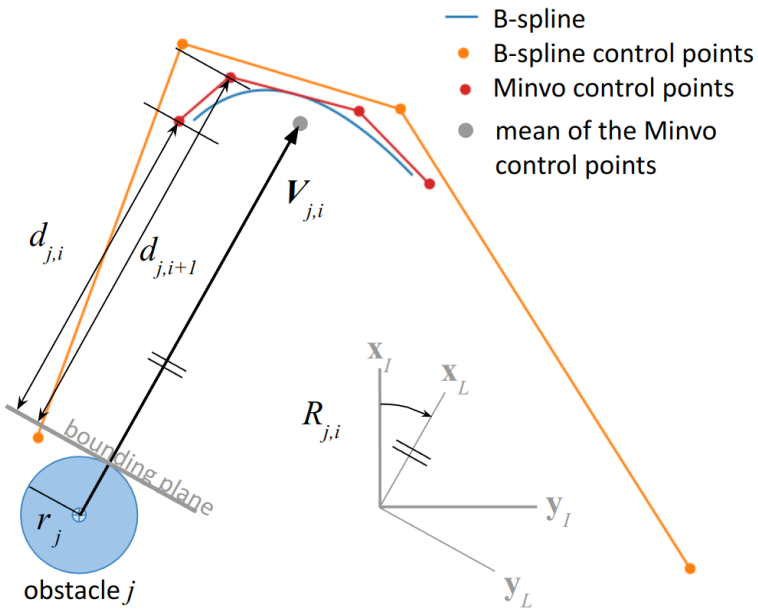



Figure 4.1: Obstacle avoidance constraint derivation for spheres or circles

To create a linear equation for the obstacle bounds, we rotate the reference frame by $R_{j,i}$ so that the $x$ axis is aligned with $\mathbf{V}_{j,i}$. The rotation matrix $R_{j,i}$ is calculated from the equation that determines the rotation that transforms a unit vector onto another unit vector [48]. The rotation is evaluated as

$$R_{j,i} = I + [\hat{\mathbf{x}}_I \times \hat{\mathbf{V}}_{j,i}]_\times + \left([\hat{\mathbf{x}}_I \times \hat{\mathbf{V}}_{j,i}]_\times\right)^2 \frac{1}{1 + \hat{\mathbf{x}}_I \cdot \hat{\mathbf{V}}_{j,i}} \tag{4.2}$$

where $\times$ is a cross product, $[]_\times$ denotes a skew-symmetric cross-product matrix, and the superscript 2 denotes a matrix multiplication square operation.

After rotating the reference frame, each distance $d_{j,*}$ is restricted to be greater than zero. Note that we use the subscript $j$ to denote the obstacle index and the subscript $i$ to denote the set of control point indices $\{i, i+k\}$. The obstacle avoidance constraint for a single obstacle and a set of Minvo control points is expressed as

$$\begin{bmatrix} d_{j,i} \\ d_{j,i+1} \\ \vdots \\ d_{j,i+k} \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.3}$$

where each distance $d_{j,*}$ is evaluated as

$$\begin{bmatrix} d_{j,i} & d_{j,i+1} & \cdots & d_{j,i+k} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,0} \\ c_{2,0} & c_{2,1} & \cdots & c_{2,0} \end{bmatrix} = \left( \begin{bmatrix} P_i & P_{i+1} \cdots & P_{i+k} \end{bmatrix} F_v - \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \right) R_{j,i}^{-1} - \begin{bmatrix} r_j \\ 0 \\ 0 \end{bmatrix} \quad (4.4)$$

On the right side of Equation 4.4 the control points $\mathbf{P}_i$ are transformed by $F_v$ into Minvo control points and then subtracted by the coordinates of the obstacle center. The difference vectors are then rotated by $R_{j,i}^{-1}$ and subtracted by the obstacle radius $r_j$ in the $x$ dimension. Rotating the points by the inverse of $R_{j,i}$ is mathematically equivalent to rotating the reference frame by $R_{j,i}$. Also note that the $c_{*,*}$ values are irrelevant real numbers because the constraint is only concerned with the $x$ dimension. Since $R_{j,i}$ is calculated at each iteration of the B-spline optimization, the implementation of this constraint is nonlinear and non-convex. However, if the rotation matrix $R_{j,i}$ was predefined, this constraint would be linear.

## 4.3 Safe Flight Corridor Paths

Safe flight corridor constraints require prior path knowledge, but are inherently convex and guaranteed safe constraints for B-splines. They are also efficient because they do not increase in complexity with increasing number of obstacles. In the approach we take, the SFC constraints are also linear, and therefore more computationally efficient. This makes them attractive for online operation.

The goal of SFC path generation is to create the shortest path from one pose to another while staying within a series of confined spaces. The objective function we use for this optimization is the same as the objective function used for direct obstacle avoidance, as discussed in Section 4.2. The difference is in the constraints applied to the optimization.

### 4.3.1 Safe Flight Corridor Constraints

The derivation of an SFC constraint is demonstrated in Figure 4.2. Here, an SFC with a predefined position vector $\mathbf{V}_{\text{corr},j}$, length $l_{\text{corr},j}$, width $w_{\text{corr},j}$, height $h_{\text{corr j}}$, and orientation $R_{\text{corr},j}$ is shown. To create less conservative constraints, the control points for each B-spline interval are converted to Minvo control points. The reference frame is then rotated by $R_{\text{corr},j}$ so that the axes $x$, $y$ and $z$ are aligned with the length, width and height of the SFC. The rotated Minvo control points are then restricted to remain within the bounds of the safe flight corridor.

In equation form, the SFC constraint for a single B-spline interval is

$$\mathbf{V}_{\text{corr},j} R_{\text{corr},j}^{-1} - \begin{bmatrix} \frac{l}{2} \\ \frac{w}{2} \\ \frac{h}{2} \end{bmatrix} \leq \begin{bmatrix} P_i & P_{i+1} & \cdots & P_{i+k} \end{bmatrix} F_v R_{\text{corr},j}^{-1} \leq \mathbf{V}_{\text{corr},j} R_{\text{corr},j}^{-1} + \begin{bmatrix} \frac{l}{2} \\ \frac{w}{2} \\ \frac{h}{2} \end{bmatrix} \quad (4.5)$$

The lower bounds are calculated by rotating the center of the SFC $\mathbf{V}_{\text{corr},j}$ by $R_{\text{corr},j}^{-1}$ and subtracting half of the length, width, and height of the SFC. The

upper bounds are found the same way except that the length, width, and height are added instead of subtracted. The center term of the inequality transforms the B-spline control points into Minvo control points and rotates the points into the same frame as the bounds. We rotate all the points by $R^{-1}_{\text{corr},j}$ because it is mathematically equivalent to rotating the reference frame by $R_{\text{corr},j}$.

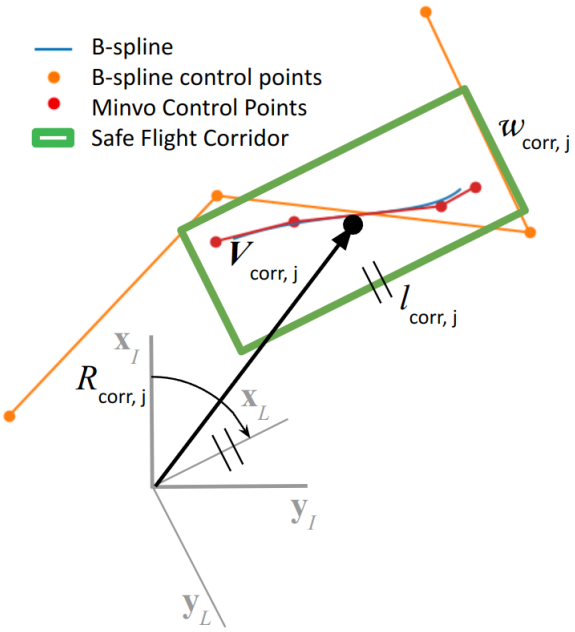



Figure 4.2: Safe flight corridor constraint derivation

For a full B-spline path, consecutive corridors and constraints are placed over each spline interval. The number of intervals per corridor is determined by the length of the corridor. Since each interval of the B-spline is scaled by the same $\alpha$ value, allotting each SFC a number of intervals proportional to the length of the corridor is necessary to avoid nonsensical or suboptimal paths.

## 4.4 Tests and Discussion

To test the efficiency and optimality of paths generated with the constraints described in Sections 4.2 and 4.3, we performed six tests for 2D obstacles, six tests for 3D obstacles, three tests for 2D SFCs, three tests for 3D SFCs, and eight tests for SFCs with obstacles. Each of these test cases are compared against an obstruction-free generated path. The metrics under consideration are evaluation time, path length, and scaling efficiency when increasing the number of obstacles or corridors. Each path was constrained by position and direction at the start and end of the path, as well as by a maximum curvature. All tests were performed on an Intel Core i7-10750H CPU at 2.60 GHz processing speed. The SLSQP Scipy optimizer library was used

to generate the paths. The purpose of these tests was to see if the collision-free path-generation methods described are optimal and viable for online operation. Optimal paths in these cases mean shortest paths.

### 4.4.1 Direct Obstacle Avoidance Tests

Figure 4.3 shows several optimized paths with three different start and end conditions in each row. The first column shows an obstruction-free case for comparison, and columns two and three have paths that avoid one and two obstacles, respectively.

Calculating the evaluation time for cases with one obstacle, we calculated an average of about 0.87 seconds, suggesting that online operation is feasible for one 2D obstacle. However, in column three, we can see that the evaluation time increases on average by 212% with the addition of another obstacle. This means that this solution does not scale well with increasing number of obstacles. The highest increase in path length due to the addition of obstacles was 5.4% for case three with one obstacle. The average increase in path length is 1.3%. In general, the paths are similar in length to the obstruction-free paths, suggesting that they are optimal in terms of path length.

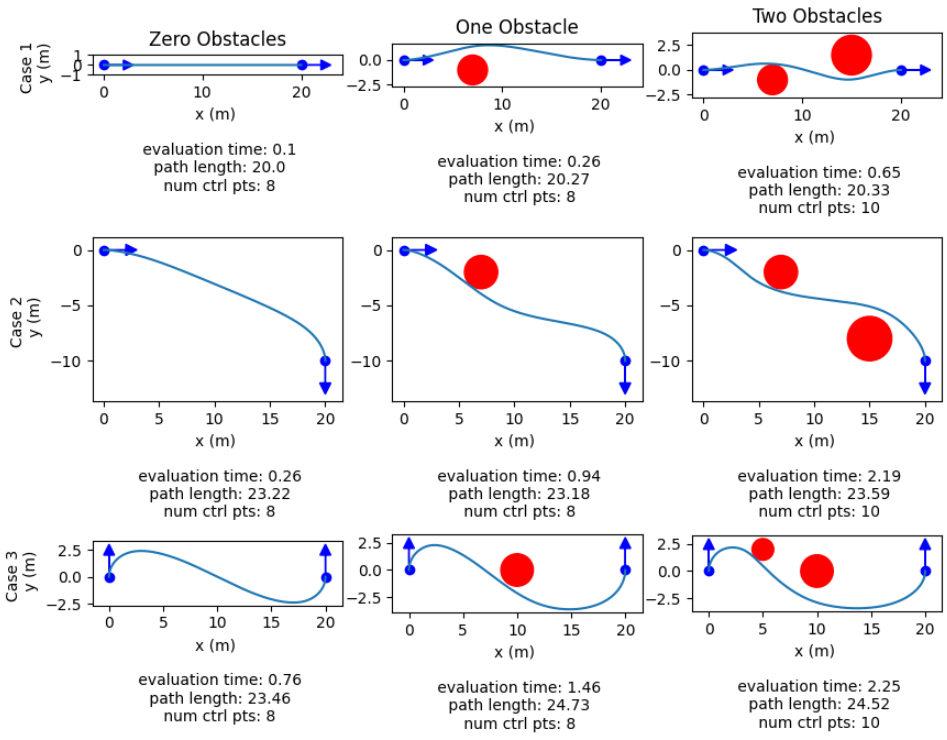



Figure 4.3: Paths avoiding circles

Figure 4.4 shows several optimized 3D paths with three different start and end conditions for each row. The average increase in path length when adding an obstacle in 3D is 0.6% with a maximum increase of 1.8%. This

suggests that the paths created are the optimal shortest paths. However, the increase in the evaluation time to add an obstacle is 256% on average. The average evaluation time to generate a path around one obstacle in 3D was 1.54 seconds. The evaluation time increase to add another obstacle is 193% on average. This suggests that the direct obstacle avoidance method could work for one obstacle, but will not scale well for multiple obstacles.

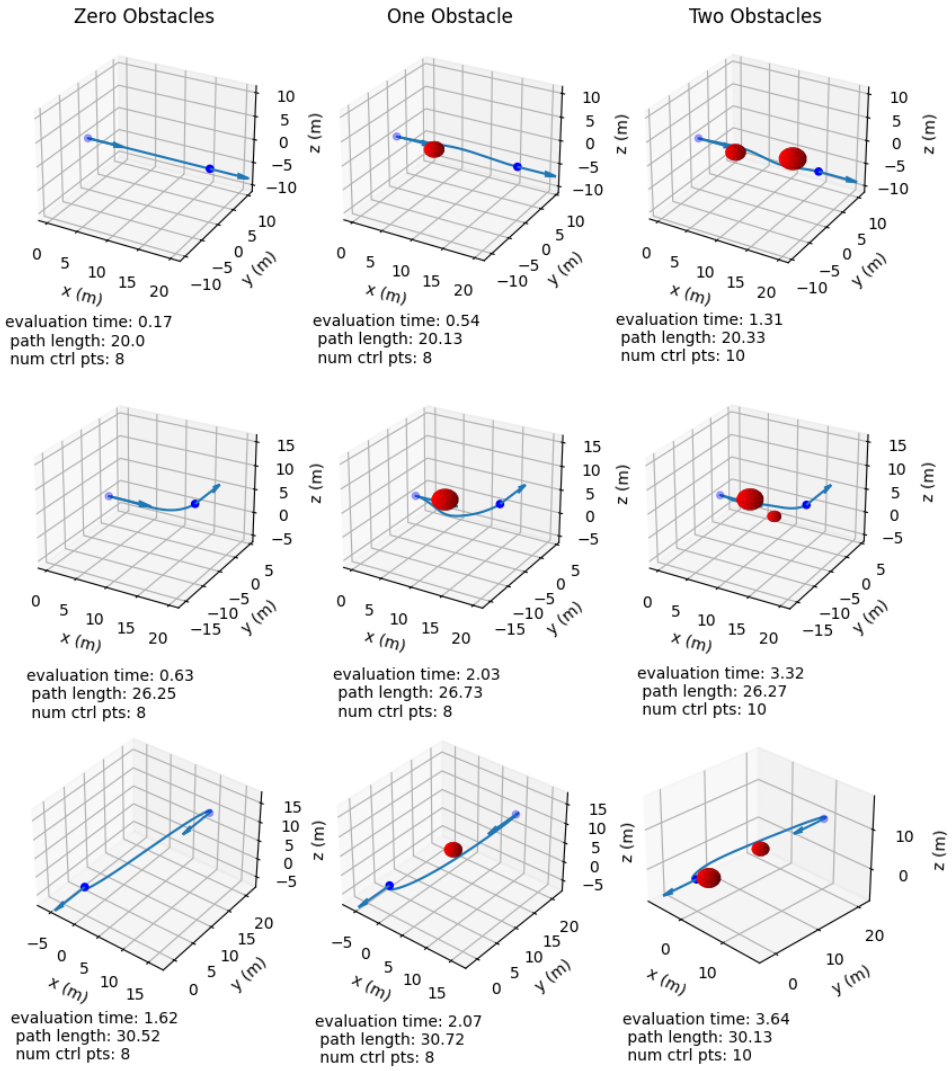



Figure 4.4: Paths avoiding spheres

### 4.4.2 Safe Flight Corridor Path Tests

Figure 4.5 shows three 2D tests, with an obstruction-free path in the left column as a benchmark and paths constrained by SFCs in the right column. Each successive test has an added SFC. Notice that the path always touches an intersecting corner of the SFCs, which shows that the path approaches the optimal shortest path. Since SFCs significantly change the shape of the path compared to the benchmark path, we do not look at empirical data

for path length. We also do not look at empirical data for the path length because it is directly related to the algorithms that generate and select the SFCs, which we do not discuss in this section.

For Cases 1 and 2, the use of SFC constraints in the optimization does not increase the timing costs. For case three, the timing cost increases by 460%. However, this is likely due to the increase in number of control points, because the optimizer does not handle a large number of optimization variables well. This suggests that the evaluation time does not scale directly with the number of SFCs, but with path complexity. In fact, the timing decreases for cases one and two when adding SFCs. The average evaluation time for these three tests is 0.67 seconds, suggesting that the solution is promising for online operation in two dimensions.

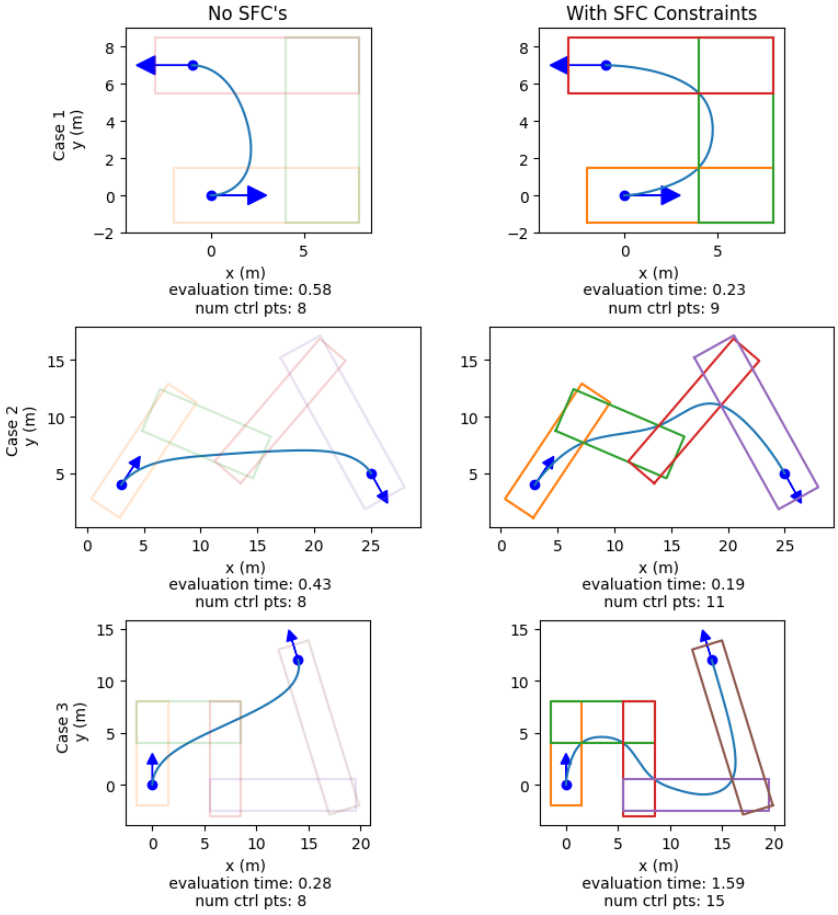



Figure 4.5: 2D safe flight corridor paths

Figure 4.6 shows three SFC tests in 3D. The left column has obstruction-free paths as benchmarks, and the right column has paths with the same

start and end conditions, but with SFC constraints. By observation, one can see that the paths take short and direct routes through the SFCs. The average increase in evaluation time to include SFCs is 7.92%, showing that it scales well with an increasing number of SFCs. Since the addition of SFCs to the optimization proves to be computationally efficient in 3D, we conclude that this approach is viable for online operation.

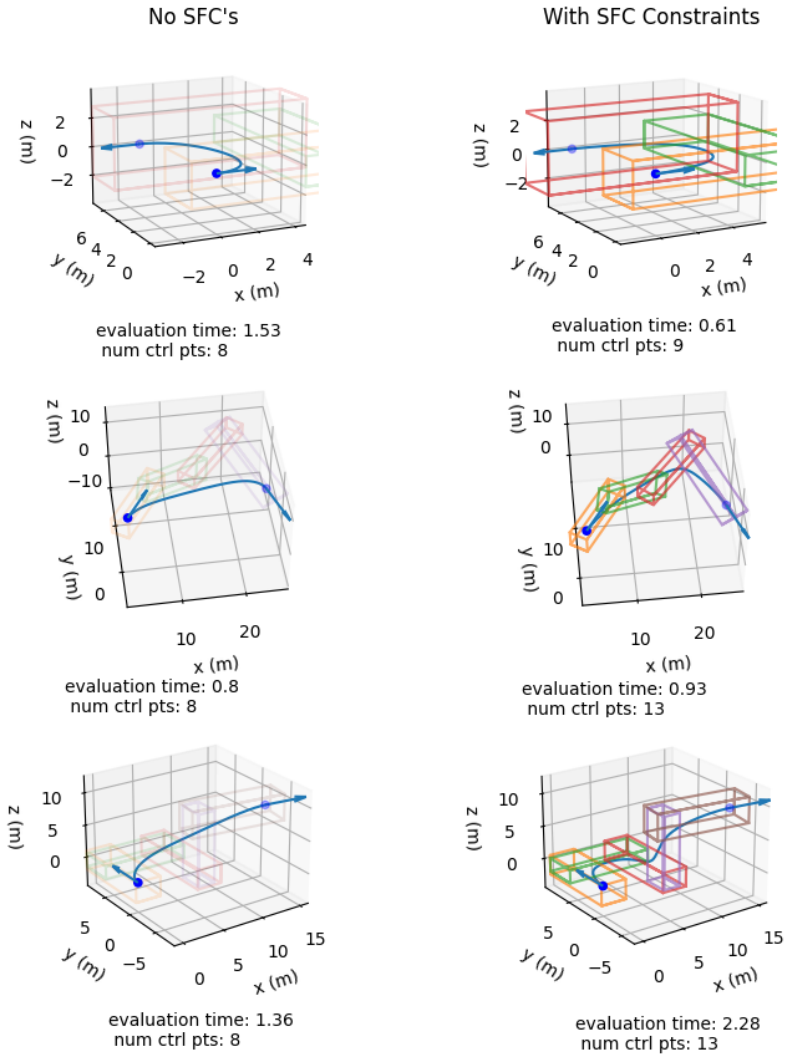



Figure 4.6: 3D safe flight corridor paths

### 4.4.3 SFC and Direct Obstacle Avoidance Tests

Figure 4.7 shows that we can combine direct obstacle avoidance with SFC constraints. Column one contains obstruction-free paths, column two contains the same initial conditions but with SFC constraints, and column three adds an obstacle to the SFCs. Notice that columns two and three use the same number of control points. This shows that including an obstacle in the SFC's does not increase the number of control points required for the

optimization. However, we draw the same conclusions as we draw from the other tests. Using SFCs is a good solution for collision-free paths, and direct obstacle avoidance is viable for at most one obstacle.

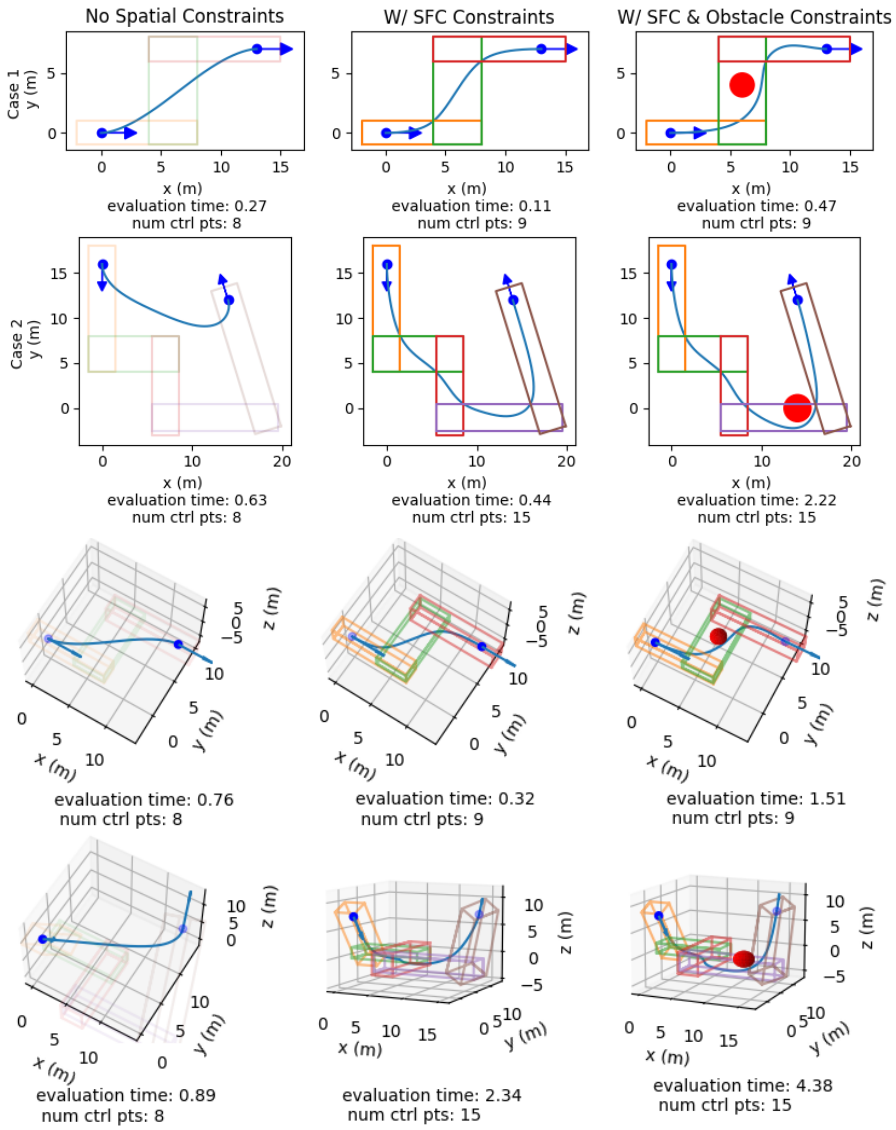



Figure 4.7: SFC and obstacle constrained paths

Some improvements to the direct obstacle avoidance constraint could improve the computation speed. For example, constraining only the intervals inside the SFC with an obstacle would reduce time. Additionally, pre-determining the rotations for each direct obstacle avoidance constraint would make the constraint linear and therefore significantly improve computational efficiency. However, this would also require some pre-planning.

## 4.5 Summary

In this chapter, we covered two approaches for producing collision-free paths. The first method did not require prior information about the shape of the path and implemented direct obstacle avoidance constraints on the path. However, the speed of the direct obstacle avoidance approach does not scale well with the number of obstacles and requires some algorithm improvements to achieve real-time operation with multiple obstacles. The second approach uses a sequence of overlapping safe flight corridor constraints. This is fast and efficient, and scales well with the number of obstacles and corridors. However, prior knowledge about the course is required as input to the optimization, specifically the sequence of SFCs. Both of these approaches produce optimal collision-free paths and will be used in the following chapter.

# 5 Real-Time Path Planning and Following for Fixed-Wing UAVs

## 5.1 Guidance of Fixed-Wing UAVs

One of the biggest challenges for autonomous operation of a fixed-wing aircraft is finding the quickest path from one pose to another, while respecting the dynamics of the aircraft and adjusting to the environment. This includes navigating restricted airspaces, avoiding obstacles, and adapting to wind disturbances. Autonomous navigation in these conditions requires regular planning updates. Furthermore, for a fixed-wing miniature-air vehicle (MAV), it is especially difficult to follow a time-dependent trajectory, because wind speeds commonly reach 20-60 percent of the desired speed of the aircraft [49]. It is easier for a fixed-wing aircraft to follow a path than to follow a trajectory while adjusting to unpredictable wind. For this reason, regularly updated time-independent paths are best suited for fixed-wing MAV guidance and navigation.

Dynamically feasible paths must respect the minimum turning radius and maximum climbing angle of a fixed-wing aircraft. The minimum turning radius is determined by the maximum bank angle and the desired operating speed of the vehicle. The bank angle is limited by the structural load limit of the aircraft and the amount of power available to overcome drag while turning [50]. Therefore, limiting the bank angle respective of the desired airspeed will allow the aircraft to operate within its power and structural constraints.

The maximum climb angle, or flight-path angle, is determined by the maximum climb rate and the desired operating speed. Climb rate is limited by the amount of power available to the aircraft [51]. Furthermore, if the aircraft climbs during a turn, it consumes even more power because the angle of attack must increase and creates more drag. Therefore, the maximum flight-path angle should be adjusted according to the allowable turn radius of the flight path.

### 5.1.1 Contemporary Guidance Approaches

A traditional method for planning curvature-constrained paths for fixed-wing aircraft in open space is to generate paths using Dubins paths [52]. Dubins paths consist of two arcs and a straight line that minimize the path length from a start pose to an end pose. Improvements to Dubins paths allow the aircraft to climb during straight-line portions of the path [53]. However,

more sophisticated algorithms are required to account for obstacles and crowded areas. For example, Song uses a Dubins path based A* search to plan curvature constrained paths through cluttered environments [54]. Beard and McLain implement a Dubins path-based RRT approach [49].

Lee, on the other hand, does away with Dubins paths and uses cubic Bezier curves to implement a spline-based RRT* approach [55]. Parametric curves enable the creation of more optimal paths than Dubins paths because they facilitate gradual changes in heading rate. Putri improves Lee's work by using the Theta* search algorithm to find the shortest path, and then smooths the path with Bezier curves. [56]. The problem with Dubins paths and Bezier curves is that they are not curvature-continuous. They satisfy endpoint interpolation and the endpoint tangent property but do not have a smooth curvature transition between sections. This causes tracking issues for the vehicle.

Another solution is to use potential fields with kinematic constraints based on the motion characteristics of fixed-wing aircrafts [57]. This can be sufficiently smooth, but unfortunately, the computational cost increases with increasing resolution of the field. On the other hand, Bassolillo addresses obstacles and curvature continuity using an integrated Theta* search and clothoid approach. [58]. This method is continuous in curvature, but does not account for three-dimensional paths.

### 5.1.2 Proposed Solution

The solution we propose is the generation of curvature- and inclined-constrained B-spline paths. B-splines are inherently curvature-continuous and facilitate obstacle avoidance algorithms. In this research, we avoid obstacles using the direct obstacle avoidance approach and the Safe Flight Corridor approach described in Chapter 4. Our intention is to show that the proposed approach generates shorter and more trackable paths than traditional path planning methods and is capable of online operation.

The following section describes the fixed-wing aircraft model and its physical limitations. Section 5.3 describes the path-planning and path-following approaches, and Section 5.4 contains simulation results that validate these approaches. In the results, we compare the proposed solution with an RRT Dubins method.

## 5.2 Fixed-Wing Aircraft Model

The fixed-wing aircraft model we use is taken from *Small Unmanned Aircraft: Theory and Practice* [49]. This model is used to simulate the tracking of the optimized paths. It is also useful for describing the physical limitations of the vehicle.

### 5.2.1 Kinematics and Dynamics

Figure 5.1 illustrates the aircraft model with its pose, angular rates, and velocities, labeled in the images. These values fully describe the state of the vehicle at any given time. The state is given with respect to a North, East,

and Down reference frame and with respect to the wind vector. $[\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i]$ are the inertial axes and $[\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b]$ are the body frame axes of the vehicle. $\mathbf{V}_a$, $\mathbf{V}_g$, $\mathbf{V}_w$ are the velocity vectors of airspeed, ground speed, and wind speed, respectively.

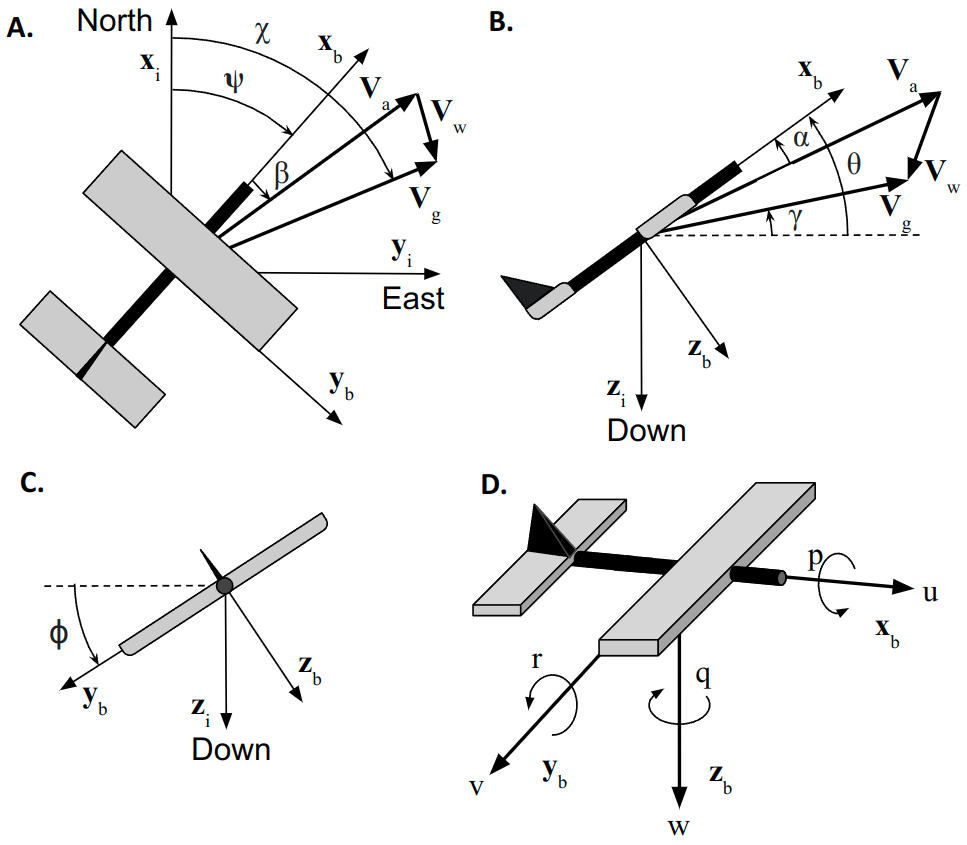



Figure 5.1: Fixed-wing aircraft model

In Figure 5.1, image A shows the course angle $\chi$, the yaw angle $\psi$, and the side slip angle $\beta$. Image B shows the flight-path angle $\gamma$, the pitch angle $\theta$, and the angle of attack $\alpha$. In image C, the roll angle $\phi$ is illustrated, and in image D, the body rates of the vehicle are illustrated. $u$, $v$, and $w$ are the translational velocities in the body frame, and $p$, $q$, and $r$ are the angular velocities in the body frame.

The equations of motion for the fixed-wing aircraft model are

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi c_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{5.2}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & s_\phi t_\theta & c_\phi t_\theta\\ 0 & c_\phi & -s_\phi\\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{5.3}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\Gamma_1 pq - \Gamma_2 qr\\ \Gamma_5 pr - \Gamma_6(p^2 - r^2)\\ \Gamma_7 pq - \Gamma_1 qr\end{bmatrix} + \begin{bmatrix}\Gamma_3 & 0 & \Gamma_4\\ 0 & \frac{1}{J_y} & 0\\ \Gamma_4 & 0 & \Gamma_8\end{bmatrix}\begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} \tag{5.4}$$

$$\mathbf{f} = \begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix},\ \mathbf{M} = \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} \tag{5.5}$$

$$\begin{aligned}\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{J_x J_z - J_{xz}^2},\ \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{J_x J_z - J_{xz}^2},\ \Gamma_3 = \frac{J_z}{J_x J_z - J_{xz}^2},\\ \Gamma_4 = \frac{J_{xz}}{J_x J_z - J_{xz}^2},\ \Gamma_5 = \frac{J_z - Jx}{J_y},\ \Gamma_6 = \frac{J_{xz}}{J_y}\\ \Gamma_7 = \frac{\qquad}{J_x J_z - J_{xz}^2},\ \Gamma_8 = \frac{J_x}{J_x J_z - J_{xz}^2}\end{aligned} \tag{5.6}$$

where $m$ is the mass of the aircraft and the $J_*$ values are the moments of inertia of the aircraft. $\mathbf{f}$ and $\mathbf{M}$ are the forces and moments due to gravity, aerodynamic effects, and propulsion. $\mathbf{f}$ and $\mathbf{M}$ are functions of air speed $V_a$, aileron deflection $\delta_a$, elevator deflection $\delta_e$, rudder deflection $\delta_r$, thrust input $\delta_t$, and wind vector $\mathbf{W}$. For a detailed derivation of the kinematics and dynamics, see Beard and McLain's book on small unmanned aircraft [49].

### 5.2.2 Physical Limitations

For a path to be useful for aircraft flight, the physical limitations of the vehicle must be respected. While traveling at a constant velocity, these limitations can be obeyed by limiting the turn radius and flight-path angle of the trajectory. The turn radius of a fixed-wing aircraft during a coordinated turn is calculated as

$$R = \frac{V_g^2 \cos\gamma}{g \tan\phi\ \cos(\chi - \psi)} \quad [49] \tag{5.7}$$

If we neglect the flight-path angle and assume zero wind conditions, we can simplify Equation 5.7 to be

$$R = \frac{V_g^2}{g \tan\phi} \tag{5.8}$$

The maximum curvature is then limited by the maximum bank angle $\phi$ and is expressed as

$$C_{\max} = \frac{g \tan\phi_{\max}}{V_g^2} \tag{5.9}$$

The bank angle is limited by the structural load the wings can withstand, and the power output of the vehicle. The structural and power limits of a fixed-wing aircraft are determined by design and verified through analysis, along with significant safety factors.

The flight path angle $\gamma$ of the aircraft is calculated as

$$\gamma = \sin^{-1}\left(\frac{\dot{h}}{V_g}\right) \tag{5.10}$$

and is limited by the power output of the aircraft. For the purpose of path optimization, we limit the path angle by limiting the maximum slope of the path. The maximum slope is calculated as

$$\mathrm{m}_{\max} = \tan(\gamma_{\max}) \tag{5.11}$$

We can determine the maximum path angle $\gamma_{\max}$ from the operating ground speed $V_g$ and maximum power output of the aircraft.

## 5.3 Guidance and Control

For a guidance algorithm to be effective, it must be able to respect the dynamic capabilities of the vehicle, avoid obstacles, and update regularly and in a timely manner. Our goal is to develop an online path-planning solution that generates distance-optimal paths that are more easily tracked than traditional or contemporary solutions. This section describes the B-spline path-planning optimization that we propose for online guidance of a fixed-wing aircraft. We also describe a model-based PID path follower that we developed to track B-spline paths, which also works for any generic path.

### 5.3.1 Path Planning

In this subsection, we first derive the slope constraint used for the optimization of B-spline paths. We then explain the B-spline optimization problem as a whole, and finish by describing the method proposed for generating a sequence of safe flight corridors (SFC's).

#### Path Angle Constraint Derivation

The flight-path angle of a B-spline path at time $t$ is expressed as

$$\gamma(t) = \tan^{-1}\left(\frac{b(t)'_z}{\|b(t)'_{xy}\|}\right) \tag{5.12}$$

The flight-path angle can be bounded by restricting the slope of the path m, defined as

$$\mathrm{m}(t) = \frac{b(t)'_z}{\|b(t)'_{xy}\|} \tag{5.13}$$

Therefore the flight-path angle constraint for each spline interval is expressed as

$$\frac{\max(b(t)'_z)}{\min(\|b(t)'_{xy}\|)} \leq \mathrm{m}_{\max} = \tan\gamma_{\max} \quad \forall t \in [t_j, t_{j+1}] \tag{5.14}$$

For a third-order B-spline, $b(t)'_z$ is a second-order polynomial whose extrema points can be found by solving for the roots of $b(t)''_z$, which is a linear equation. The extrema points of $\|b(t)'_{xy}\|$ can also be solved analytically by

finding the roots of $\frac{d\|b(t)'_{xy}\|^2}{dt}$ which is a third-order polynomial. The roots are found using Cardano's formula [43]. The slope constraint is applied to the B-spline optimization using an analytical roots approach similar to the approach described in Section 3.3.2.

### B-Spline Optimization

The intent of this research has been to create a rapid algorithm for generating kinematic and dynamically feasible paths for UAV's. To accomplish this goal, we have focused on generating paths with B-splines using only control points and scale parameters as optimization variables. Optimizing over the control points eliminates the need to optimize discrete points along the path, which leads to a more computationally efficient optimization problem. Using the convex hull property of control points ensures feasibility throughout the entire path.

The B-spline optimization problem can be expressed as

$$\text{minimize:} \quad \|P_2 - P_1\|^2 + \|P_3 - P_2\|^2 + \cdots \|P_n - P_{n-1}\|^2 \tag{5.15}$$

$$\text{by varying:} \quad s_0, s_f \in \mathbb{R}^+, \mathbf{P} \in \mathbb{R}^{3\times n} \tag{5.16}$$

$$\text{such that:} \quad \mathbf{W}_0 = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^3 \\ \\ 1 \end{bmatrix} \tag{5.17}$$

$$\mathbf{W}_f = \begin{bmatrix} P_{n-3} & P_{n-2} & P_{n-1} & P_n \end{bmatrix} M \begin{bmatrix} \mathbf{1} \in \mathbb{R}^{4\times 1} \end{bmatrix}$$

$$\mathbf{D}_0 = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix} M \begin{bmatrix} \mathbf{0} \in \mathbb{R}^{2\times 1} \\ 1 \\ 0 \end{bmatrix} \frac{1}{s_0}$$

$$\mathbf{D}_f = \begin{bmatrix} P_{n-3} & P_{n-2} & P_{n-1} & P_n \end{bmatrix} M \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \end{bmatrix} \frac{1}{s_f}$$

$$\mathbf{D}_{a,0} = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix} M \begin{bmatrix} 0 \\ 2 \\ \mathbf{0} \in \mathbb{R}^{2\times 1} \end{bmatrix} \frac{1}{s_0^2}$$

$$\min\left(\frac{\max(\|b(t)' \times b(t)''\|)}{\min(\|b(t)'\|)^3}, \frac{\max(\|b(t)''\|)}{\min(\|b(t)'\|)^2}\right) \leq C_{\max}$$
$$\forall j \in \{3, 4, \ldots, 3+\mu\}, \text{ where } t \in [t_j, t_{j+1}]$$

$$-\mathrm{m}_{\max} \leq \frac{\max(b(t)'_z)}{\min(\|b(t)'_{xy}\|)} \leq \mathrm{m}_{\max}$$
$$\forall j \in \{3, 4, \ldots, 3+\mu\}, \text{ where } t \in [t_j, t_{j+1}]$$

Recall that $\mu$ is the number of intervals in the B-spline and $n$ is the total number of control points. $\mathbf{W}_0$ and $\mathbf{W}_f$ are the initial and final positions of the path, $\mathbf{D}_0$ and $\mathbf{D}_f$ are the initial and final orientation vectors of the path, $\mathbf{D}_{a,0}$ is the initial direction of the acceleration, $C_{\max}$ is the maximum

curvature, and $m_{max}$ is the maximum slope of the path. Constraining $\mathbf{D}_{a,0}$ allows us to set the curvature at the beginning of the path, which is useful for concatenating continuous-curvature paths. For the derivation of the objective and constraints, see Section 2.4. We solve this problem using the SLSQP solver in the scipy optimization library.

Notice that the only variables that are being optimized are the control points $\mathbf{P}$ and the scaling variables $s_0$ and $s_f$. The equality constraints for the initial and final pose of the vehicle are linear. However, the curvature and slope constraints are nonlinear and non-convex. Nevertheless, the computation can still be relatively fast because it is optimizing a small number of variables.

For paths directly avoiding an obstacle, we add the following constraints.

$$\left(\mathbf{P}_i, xyF_v - \begin{bmatrix} x_{\text{obst}} \\ y_{\text{obst}} \end{bmatrix}\right) R^i_{\text{obst}} - r_{\text{obst}} > \begin{bmatrix} 0 \\ -\infty \end{bmatrix} \quad \forall i \in \{0, 1, ..., \mu - 1\} \tag{5.18}$$

This is applied to every interval of the spline. Also, because $R^i_{\text{obst}}$ is calculated at each iteration of the SLSQP algorithm, these constraints are nonlinear. See Section 4.2 for more details.

For paths traversing a series of safe flight corridors, we add the following constraint for each interval of the B-spline. The constraint is expressed as

$$\mathbf{V}_{\text{corr},l} R_{\text{corr},l} - \begin{bmatrix} \frac{L_l}{2} \\ \frac{w_l}{2} \\ \frac{h_l}{2} \end{bmatrix} \leq \mathbf{P}_i F_v R_{\text{corr},l} \leq \mathbf{V}_{\text{corr},l} R_{\text{corr},l} + \begin{bmatrix} \frac{L_l}{2} \\ \frac{w_l}{2} \\ \frac{h_l}{2} \end{bmatrix} \quad \forall i \in \{0, 1, ..., \mu - 1\} \tag{5.19}$$

where each interval $i$ has a non-unique corresponding corridor $l$. See Section 4.3 for the derivation of this constraint. Note that this constraint is convex and linear. The sequence of SFC's can be created using a Maximally Occupying Convex Space (MOCS) algorithm described in the next subsection.

### SFC Creation

Suh developed a Maximally Occupying Convex Space algorithm for the generation of SFCs for UAV path planning [59]. This algorithm finds every maximum rectangular space in a rectangular world and then creates a graph with each SFC as a node. Every overlapping SFC creates an edge in the graph. Suh then uses Dijkstra's algorithm to find the shortest sequence of SFC's from one location to another. An illustration of the MOCS being generated is shown in Figure 5.2.

Rather than implementing this algorithm in software, we manually select a sequence of SFC's for the B-spline optimization. However, to get an estimate for the time it would take to run Suh's algorithm, we perform a complexity analysis. Suh's results contain the average runtime for a MOCS construction with five to seven obstacles, and the runtime for Dijkstra's algorithm in C++. However, most of the other code used for path planning in this research was

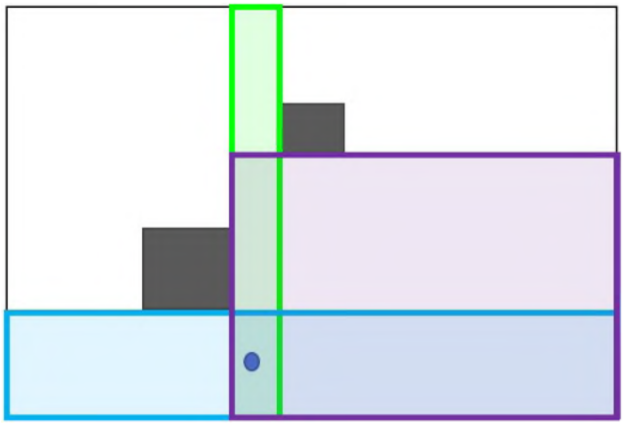


Figure 5.2: Illustration of the MOCS construction taken from Suh's article titled *A Fast and Safe Motion Planning Algorithm in Cluttered Environment using Maximally Occupying Convex Space* [59]

written in Python. For a better timing comparison between methods, we convert the C++ runtime to Python's runtime.

Table 5.1 shows the average timing ratio to convert a C++ runtime to a Python runtime for various Big O complexity algorithms. This information was gathered from Behery and Tamimi's work and from personal experimentation [60], [61].

| Complexity | Ratio | Source |
|---|---|---|
| $O(n)$ | 423.5 | [60] |
| $O(n^2)$ | 32.6 | [60] |
| $O(n^k)$ | 27.04 | [61] |
| $O(n!)$ | 25.8 | [60] |
| $O(n\log n)$ | 37 | heap sort trials |

Table 5.1: Time conversion ratios from C++ to Python

Using the data in Table 5.1, we estimate the time it would take to perform a MOCS construction and Dijkstra's algorithm in Python. Table 5.2 contains the C++ runtime, Big O complexity, and conversion ratio for both Dijkstra's algorithm and MOCS construction. This is used to estimate the Python runtime shown in the last row.

| Complexity | MOCs Creation $O(n^3)$ | Dijkstra's Algorithm $O(n\log n)$ |
|---|---|---|
| C++ runtime (ms) | 8.24 | 0.03 |
| Conversion Ratio | 30.0 | 37 |
| Python Est. runtime (ms) | 247.2 | 1.11 |

Table 5.2: Python time estimate for generating SFC sequences

### 5.3.2 Path Following

To properly validate the dynamic feasibility of a path, the control must be accurate. Otherwise, poor control could be confused with path infeasibility. The control loop diagram for the fixed-wing guidance system is shown in Figure 5.3. The path follower we developed accepts the desired airspeed and B-spline control points of the optimized path and outputs the climb rate, course angle, and airspeed commands. It then sends these commands to the autopilot. The autopilot and fixed-wing model are both detailed in *Small Unmanned Aircraft: Theory and Practice* [49].

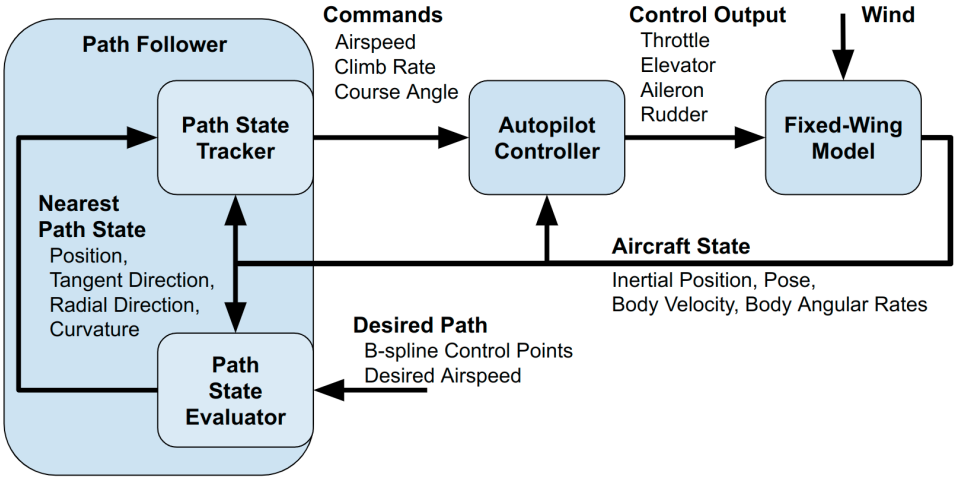



Figure 5.3: Control loop diagram for the fixed-wing aircraft

The path follower is broken down into two steps. The first step finds the desired state of the B-spline path, and the second step computes the input to the autopilot. The next two subsections describe these steps in detail.

#### Path State Evaluator

Figure 5.4 illustrates the derivation of the desired path state evaluator. Given the current position of the vehicle $\mathbf{p}$, we find the closest point on the path $\mathbf{p}_{\text{path}}$ to the vehicle. We simultaneously find the parameter value $t_n$ where the B-spline is equal to $\mathbf{p}_{\text{path}}$. Given $t_n$, we can calculate the tangent direction $\hat{\mathbf{V}}$, the radial direction $\hat{\mathbf{a}}_R$, and the curvature magnitude $C_{\text{path}}$ at $t_n$. The nearest point on the path is found by sampling close to the previous near-point.

In terms of a B-spline $b(t)$, the nearest position is defined as

$$\mathbf{p}_{\text{path}} = b(t_n) \tag{5.20}$$

The tangent direction is expressed as

$$\hat{\mathbf{V}} = \frac{b(t_n)'}{\|b(t_n)'\|} \tag{5.21}$$

and the radial direction is expressed as

$$\hat{\mathbf{a}}_R = \frac{\mathbf{a}_R}{\|\mathbf{a}_R\|} \tag{5.22}$$

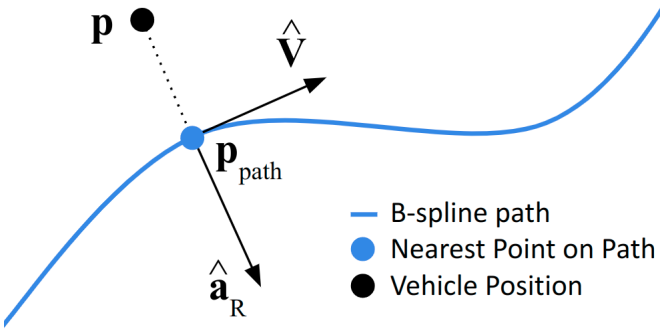



Figure 5.4: Illustration of evaluating the nearest path state

where

$$\mathbf{a}_R = b(t_n)'' - \left(b(t_n)'' \cdot \hat{\mathbf{V}}\right)\hat{\mathbf{V}} \tag{5.23}$$

The radial direction of the path at a position points toward the center of the osculating circle that describes the curvature of the path at that position. The curvature is calculated as

$$C_{\text{path}} = \frac{\|b(t_n)' \times b(t_n)''\|}{\|b(t_n)'\|^3} \tag{5.24}$$

These values fully describe the state of the path at a point. Notice that these values can be drawn from any type of path, not only B-spline paths.

### Path State Tracker

Given the state of the path at the closest point to the vehicle, we can calculate the desired output commands for the autopilot. We first calculate the desired direction of travel $\hat{\mathbf{D}}$ for the vehicle. The desired direction is calculated as

$$\hat{\mathbf{D}} = \frac{\mathbf{D}}{\|\mathbf{D}\|} \tag{5.25}$$

where

$$\mathbf{D} = \begin{cases} \mathbf{p}_e K_p + \int \mathbf{p}_e K_i - \dot{\mathbf{p}}_L K_d + \hat{\mathbf{V}} K_T + \hat{\mathbf{a}}_R K_R C_{\text{path}} & \text{if } \mathbf{p}_e < \mathbf{p}_{\text{tol}} \\ \mathbf{p}_e K_p - \dot{\mathbf{p}}_L K_d + \hat{\mathbf{V}} K_T & \text{else} \end{cases} \tag{5.26}$$

We can see that $\mathbf{D}$ is computed using PID position error control, with a constant gain input in the path direction and a proportional gain input in the radial direction. When the position error $\mathbf{p}_e$ is outside some tolerance $\mathbf{p}_{\text{tol}}$, the radial input and the integral input are not applied.

The integral term of the PID control is calculated as

$$\int \mathbf{p}_e = \int \mathbf{p}_{e,\text{prev}} + (\mathbf{p}_e + \mathbf{p}_{e,\text{prev}})\frac{dt}{2} \tag{5.27}$$

where subscript prev denotes the previous value. When $\mathbf{p}_e$ is outside the given tolerance, $\int \mathbf{p}_e$ is reset to zero. The derivative term is calculated as

$$\dot{\mathbf{p}}_L = \dot{\mathbf{p}} - (\dot{\mathbf{p}} \cdot \hat{\mathbf{V}})\hat{\mathbf{V}} \tag{5.28}$$

where

$$\dot{\mathbf{p}} = \frac{\mathbf{p} - \mathbf{p}_{\text{prev}}}{dt} \tag{5.29}$$

We use the portion of the derivative vector perpendicular to $\hat{\mathbf{V}}$ because we do not want to counteract the path direction term $\hat{\mathbf{V}}K_T$.

Given the desired direction of travel and the desired airspeed, we can calculate the input commands to the autopilot. The course angle command is calculated as

$$\chi_c = \tan^{-1}\left(\frac{\hat{\mathbf{D}}_y}{\hat{\mathbf{D}}_x}\right) \tag{5.30}$$

and the climb rate command is evaluated as

$$\dot{h}_c = -\hat{\mathbf{D}}_z V_{a,d} \tag{5.31}$$

The airspeed command is given and written as

$$V_{a,c} = V_{a,d} \tag{5.32}$$

For details on how these commands are used in the autopilot, see [49].

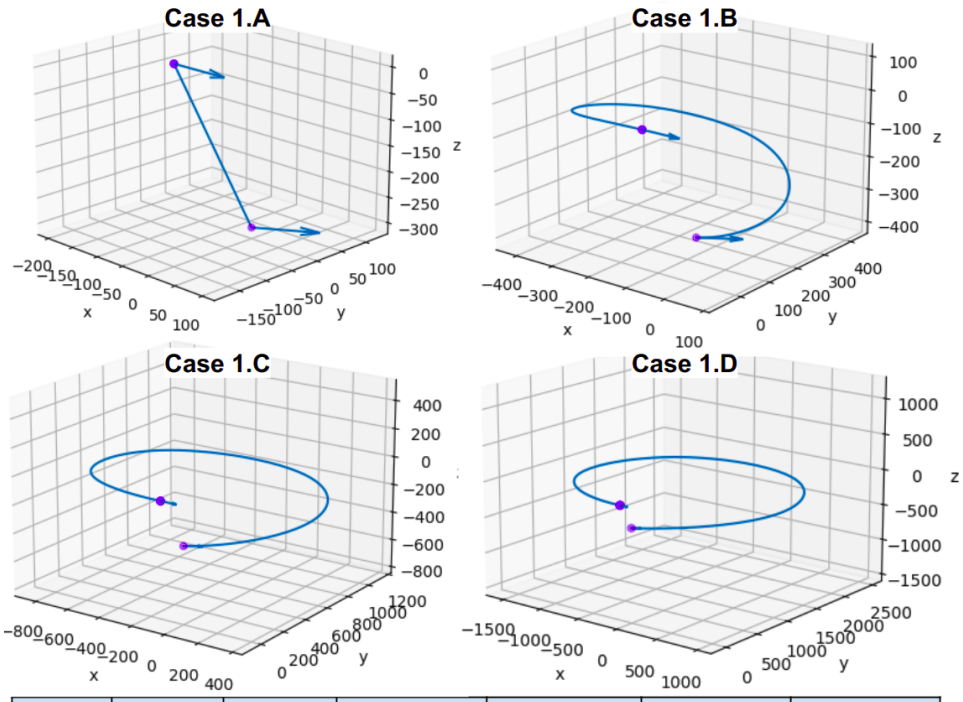
## 5.4 Simulation and Results

In this section we perform four types of tests to validate the performance of the B-spline path generator. The first set of tests verify that the B-spline slope constraint is computationally fast and that it limits the slope sufficiently. The second set of tests compares our direct obstacle avoidance B-spline path planning algorithm with an RRT Dubins path planning algorithm. The RRT Dubins algorithm was taken and implemented from *Small Unmanned Aircraft: Theory and Practice* [49]. Important metrics to consider are the path length, computational speed, and the tracking error of the simulated fixed-wing aircraft. The third set of tests compares the SFC B-spline path optimization with the RRT Dubins path planning optimization. Again, we want to validate that the B-spline path optimization can operate online, that it can produce the shortest path, and that it is easily trackable for a fixed-wing aircraft. The last test verifies that the B-spline path planner can create a sequence of continuous-curvature paths given a sequence of waypoints. All tests were performed on an Intel Core i7-10750H CPU at 2.60 GHz processing speed.

### 5.4.1 Incline Constraint Validation

Each of the test cases in this subsection implements a B-spline optimization using third-order splines with five intervals. That implies eight control points and 24 control-point optimization variables, since the control points are three-dimensional. There are also two scaling parameter variables used in the optimization for the start- and end-orientation constraints.

Figure 5.5 shows a series of B-spline optimization test cases where they each have the same initial and final pose constraints. Case 1.A has no slope or curvature constraint, Case 1.B adds a slope constraint, Case 1.C decreases the slope constraint, and Case 1.D adds a curvature constraint. As the slope limit decreases from infinite to 0.1, the path length increases because the course needs more horizontal space to raise the elevation. This is the expected behavior for a path with slope constraints. Notice also that the maximum slope is near the slope limit for cases 1.A - 1.C. The runtime is also less than one second, suggesting the ability for online operation. When we add a curvature constraint, the path length increases further because it needs more space to turn. This is also expected and desired behavior. However, the computational speed is slower than desired.

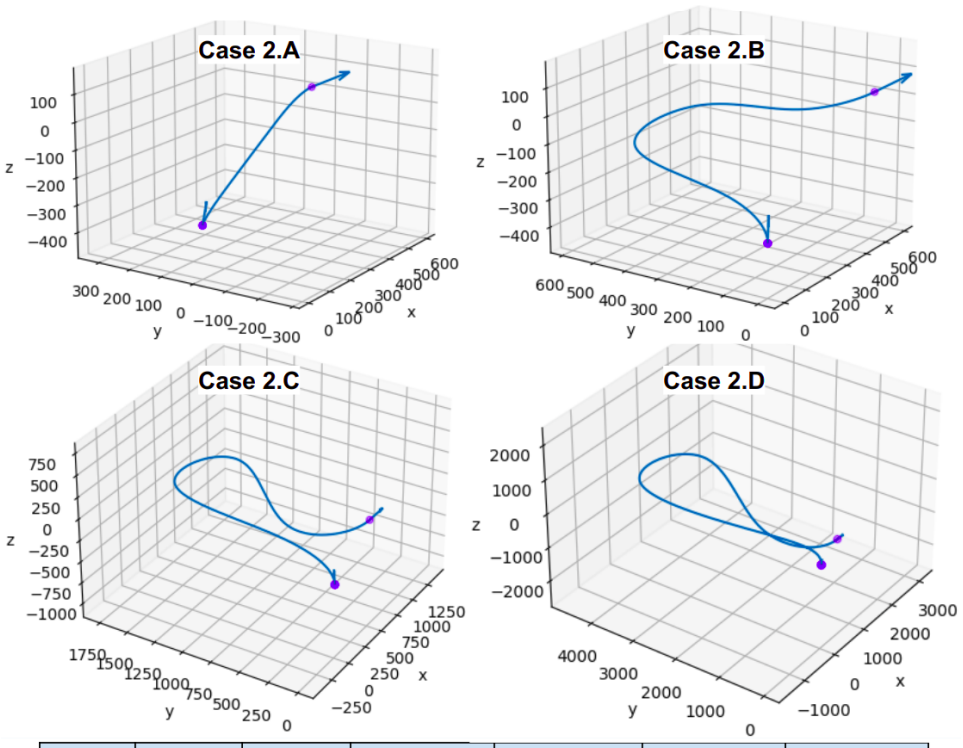



| Case | Slope Bound | Max Slope | Curvature Bound | Max Curvature | Path Length (m) | Gen Time (sec) |
|---|---|---|---|---|---|---|
| **1.A** | None | 2.683 | None | 3.53e14 | 320.2 | 0.171 |
| **1.B** | 0.2679 | 0.267 | None | 0.0071 | 1424.8 | 0.897 |
| **1.C** | 0.1 | 0.0998 | None | 0.0021 | 3631.0 | 0.934 |
| **1.D** | 0.1 | 0.0742 | 0.001 | 0.0009 | 7448.2 | 2.363 |

Figure 5.5: Incline constraint validation tests with start conditions Case 1

Figure 5.6 has another series of B-spline optimization test cases with new

start- and end-point constraints. The incline limit decreases from infinite to 0.1 starting from Cases 2.A to 2.C. In these cases, the path length increases as expected, so that the path can have enough length to rise in elevation. The path length also appropriately increases when the curvature constraint is included in the optimization. However, the runtime for path generation is almost doubled when adding the curvature constraint.




| Case | Slope Bound | Max Slope | Curvature Bound | Max Curvature | Path Length (m) | Gen Time (sec) |
|---|---|---|---|---|---|---|
| **2.A** | None | 0.6257 | None | 0.0202 | 681.9 | 0.033 |
| **2.B** | 0.2679 | 0.2679 | None | 0.0096 | 1506.8 | 0.892 |
| **2.C** | 0.1 | 0.0999 | None | 0.0046 | 4459.5 | 0.9119 |
| **2.D** | 0.1 | 0.0662 | 0.002 | 0.0018 | 11076.9 | 2.2903 |

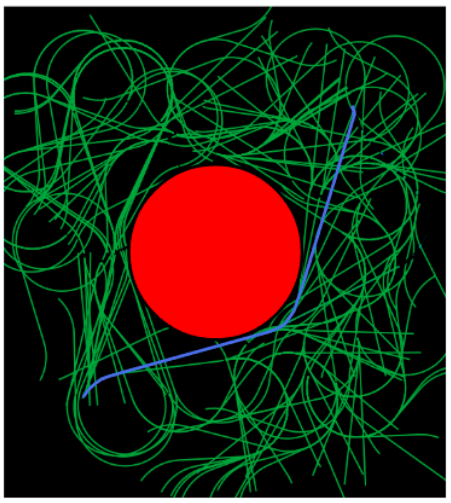
Figure 5.6: Incline constraint validation tests with start conditions Case 2

From these results, we can see that the optimizer sufficiently limits the slope of the path and is also not overly conservative. The runtime is also satisfactory. However, when both slope and curvature are constrained, the computational speed slows significantly. This suggests that the algorithm may need improvements to be viable for online use.

### 5.4.2 Navigation Around a Single Obstacle

In this section, we optimize paths constrained to a start and end pose, by a maximum curvature, by a maximum slope, and to avoid a single cylinder. The paths are generated with the proposed B-spline path generation method and with the RRT Dubins method for comparison. These paths are then tracked by a fixed-wing aircraft. The purpose of these tests is to show that the B-spline path generation method is faster and creates shorter, more trackable paths. Each test figure contains path analytics that detail the path generation runtime, and the path length. The blue highlighted rows indicate that the row contains data for the path shown in the images. The figures also contain a graph showing the tracking error, curvature, and incline of the aircraft's flight path.

RRT Dubins Paths



Path Analytics

| | Path Gen Run Time (sec) | # of Paths Created | Path Length (m) | Distance Obstacle (m) |
|---|---|---|---|---|
| Figure | 1.21 | 18 | 793.7 | 17.8 |
| Mean | 1.31 | 21 | 986.4 | 21.2 |
| High | 1.97 | 30 | 1391.9 | 50.2 |
| Low | 0.42 | 16 | 788.3 | 2.6 |

Fixed Wing Path Following

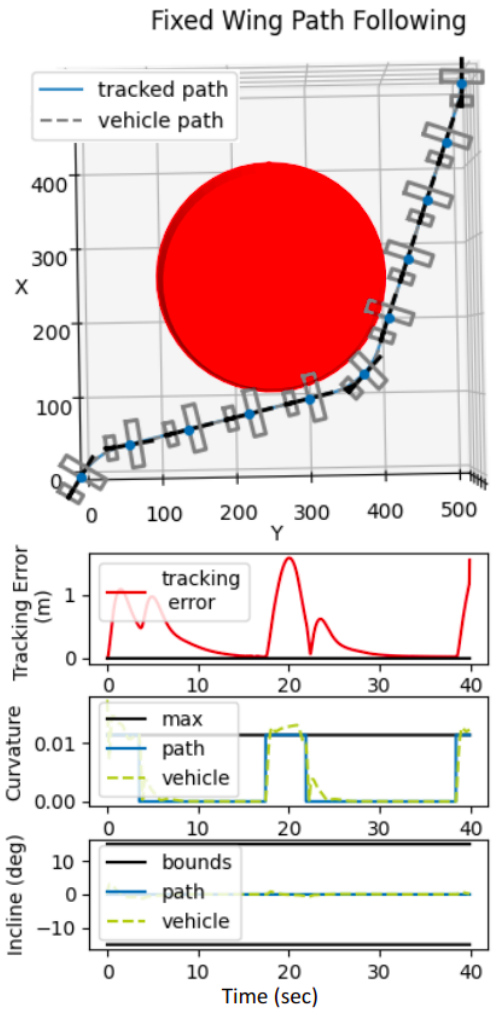
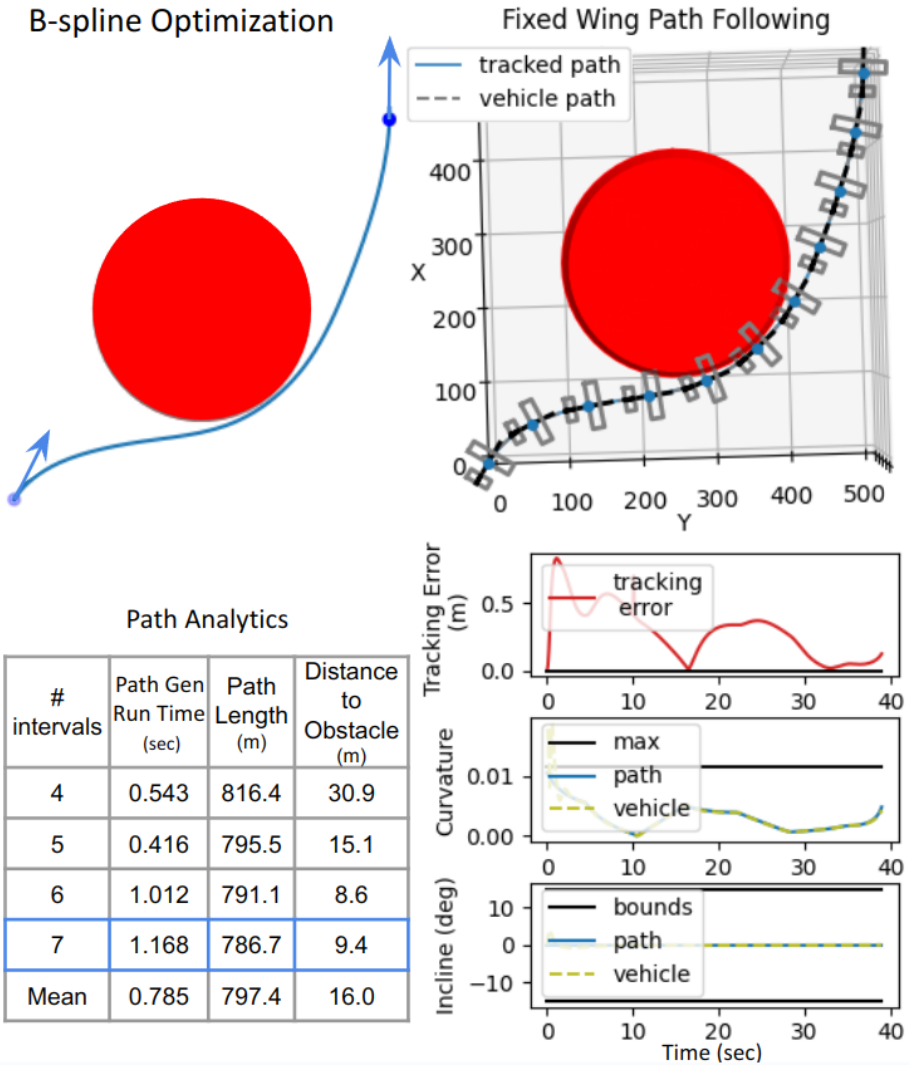



Figure 5.7: Fixed-wing aircraft following a 2D path around an obstacle generated using an RRT Dubins method

Fifteen trial paths were generated for each RRT Dubins flight scenario to obtain a statistical mean, high, and low for each data category of the path

characteristics. This was done because each RRT path varied in shape and generation time for each trial. The optimized B-spline paths, on the other hand, always arrive at the same solution in approximately the same time, given the same constraints and optimization variables. Therefore, we instead gathered a mean for the B-spline paths by varying the number of intervals in the spline per trial.

Figure 5.7 shows the resultant path of the RRT Dubins algorithm and that path being tracked by a fixed-wing aircraft. Figure 5.8 shows a path created using the proposed B-spline path generation algorithm, and that path being tracked by the same aircraft system. Both of these paths were generated on a two-dimensional plane, have the same start and end conditions, and have the same obstacle to overcome. Both paths also have the same curvature bounds equal to 0.0114.

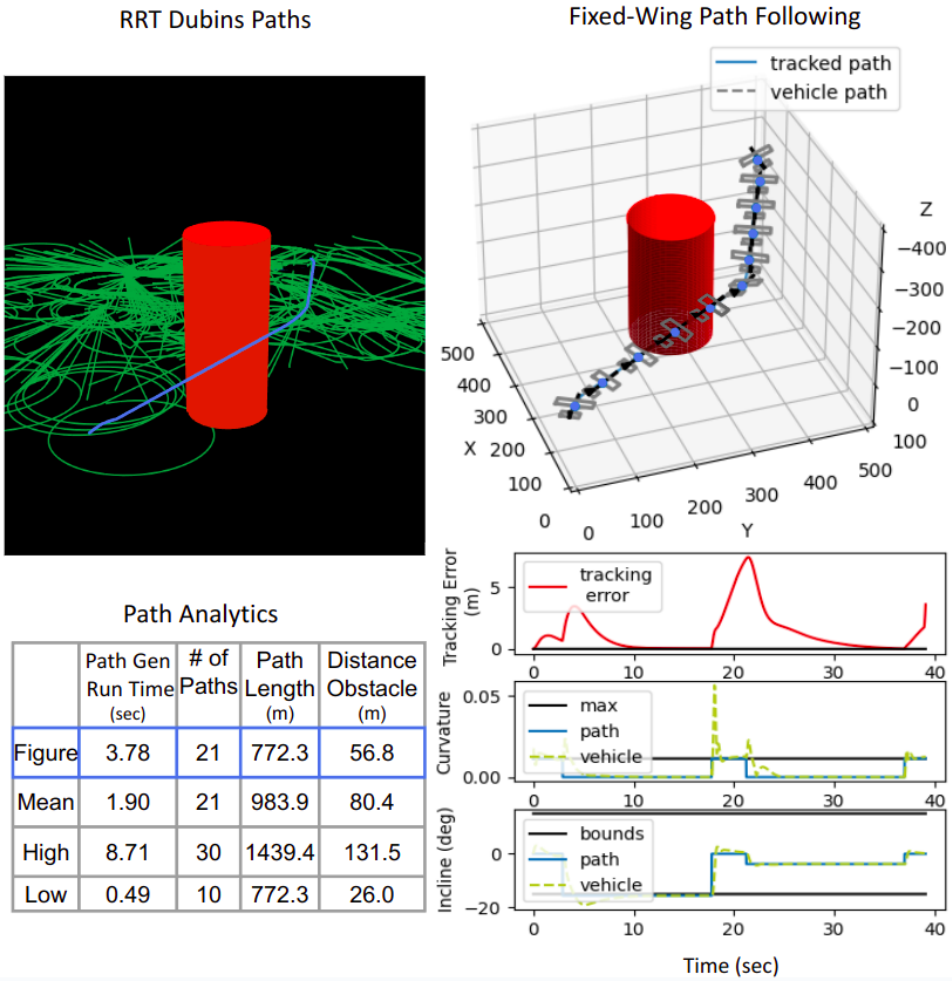



| # intervals | Path Gen Run Time (sec) | Path Length (m) | Distance to Obstacle (m) |
|---|---|---|---|
| 4 | 0.543 | 816.4 | 30.9 |
| 5 | 0.416 | 795.5 | 15.1 |
| 6 | 1.012 | 791.1 | 8.6 |
| 7 | 1.168 | 786.7 | 9.4 |
| Mean | 0.785 | 797.4 | 16.0 |

Figure 5.8: Fixed-wing aircraft following a 2D path around an obstacle generated using a B-spline optimization

The average runtime for paths generated using the B-spline optimization method was 0.785 seconds, and the average runtime for the RRT Dubins method was 1.31 seconds. The B-spline optimization was almost two times faster, and resulted in a path that was on average 20% shorter than the Dubins paths. Additionally, the maximum tracking error while tracking a B-spline path was a third of the tracking error while tracking a Dubins path. This is because the B-spline path had smooth curvature, while the Dubins path did not. Therefore, the greatest tracking errors occurred when the aircraft transitioned between straight-line and curved portions of the Dubins path.

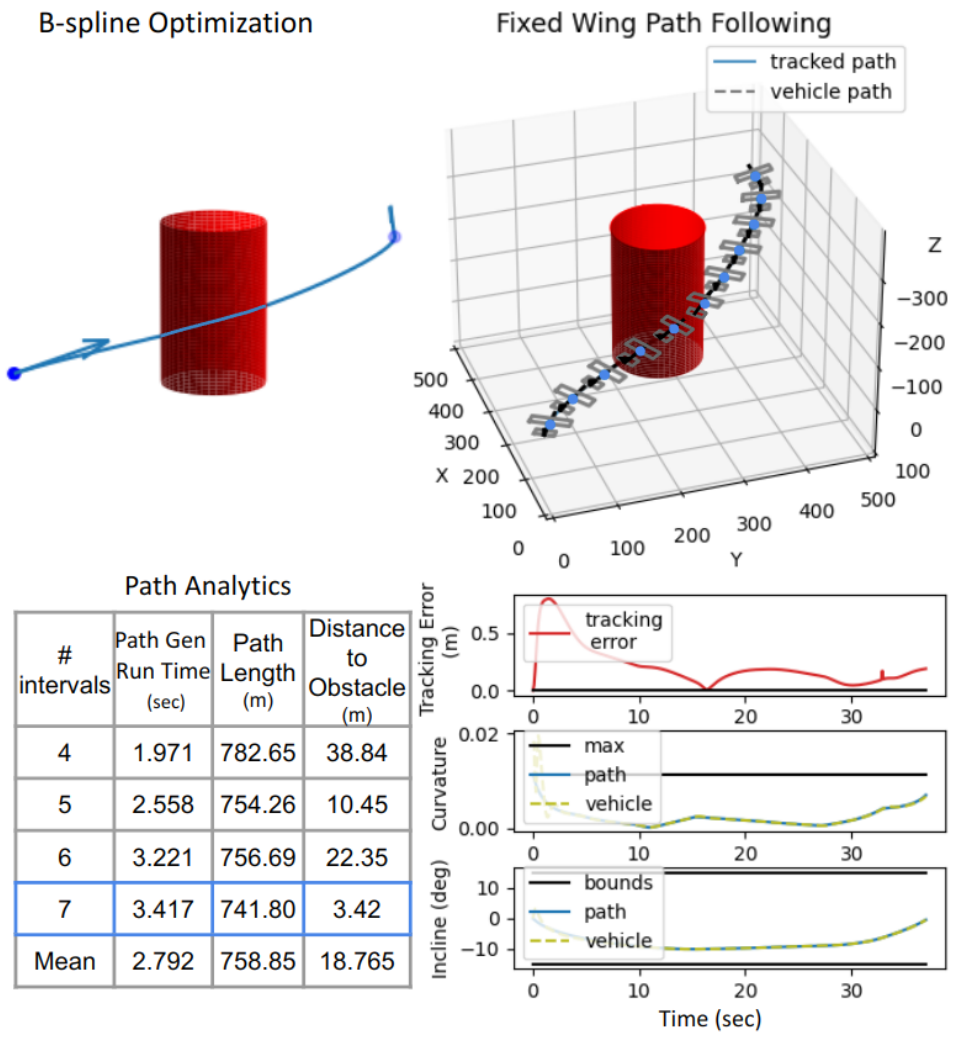
Figures 5.9 and 5.10 contain paths generated in a three-dimensional space. Both paths have the same start and end conditions, the same curvature bounds, and the same obstacle. These paths also have the same slope constraint of 0.173. Figure 5.9 shows a path created using the RRT Dubins method, and Figure 5.10 shows a path generated using the proposed B-spline optimization method.




Path Analytics

| | Path Gen Run Time (sec) | # of Paths | Path Length (m) | Distance Obstacle (m) |
|---|---|---|---|---|
| Figure | 3.78 | 21 | 772.3 | 56.8 |
| Mean | 1.90 | 21 | 983.9 | 80.4 |
| High | 8.71 | 30 | 1439.4 | 131.5 |
| Low | 0.49 | 10 | 772.3 | 26.0 |

Figure 5.9: Fixed-wing aircraft following a 3D path around an obstacle generated using an RRT Dubins method




| # intervals | Path Gen Run Time (sec) | Path Length (m) | Distance to Obstacle (m) |
|---|---|---|---|
| 4 | 1.971 | 782.65 | 38.84 |
| 5 | 2.558 | 754.26 | 10.45 |
| 6 | 3.221 | 756.69 | 22.35 |
| 7 | 3.417 | 741.80 | 3.42 |
| Mean | 2.792 | 758.85 | 18.765 |

Figure 5.10: Fixed-wing aircraft following a 3D path around an obstacle generated using B-spline optimization

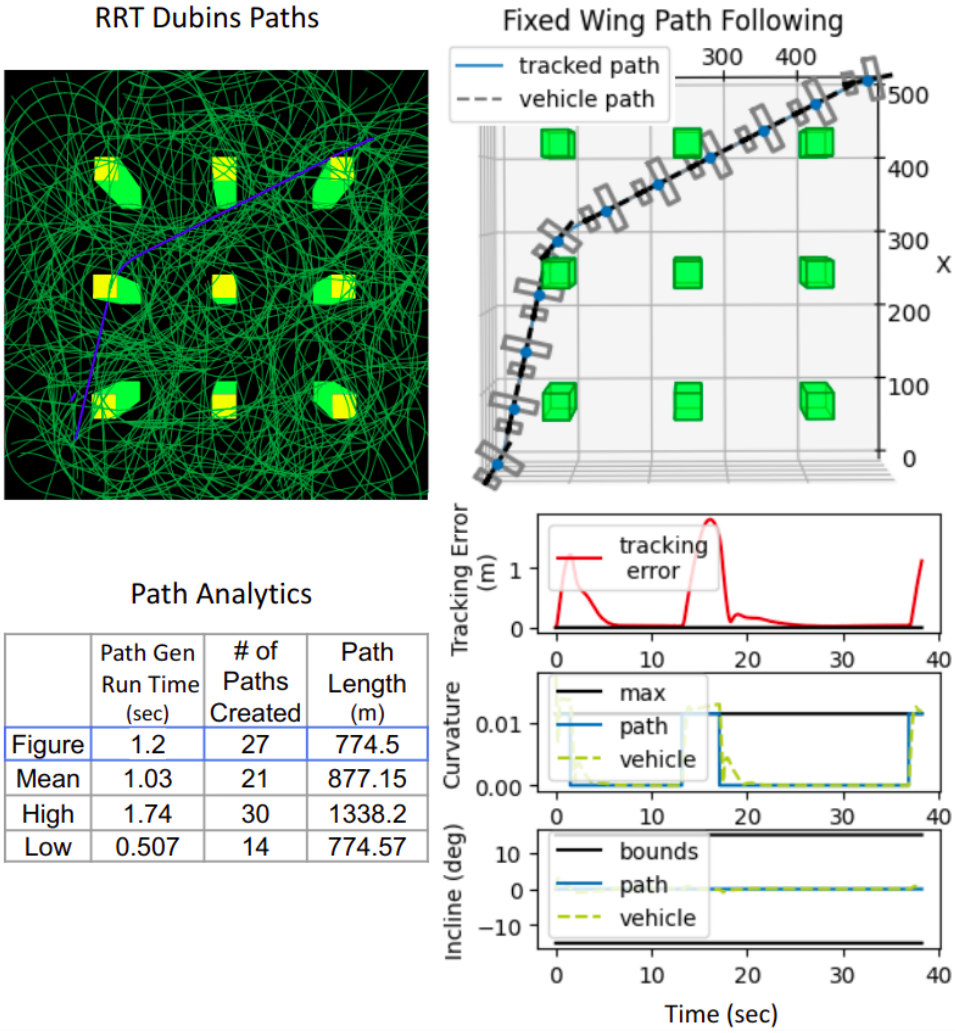
The B-spline optimization has an average runtime of 2.792 seconds and the RRT Dubins method has an average runtime of 1.9 seconds. Also, the fastest B-spline runtime is 1.971 seconds, while the fastest RRT Dubins runtime is 0.49 seconds. This shows that the B-spline optimization method has a worse computational efficiency than the RRT Dubins method in three dimensions. This is because the number of optimization variables increases and because the B-spline optimization now includes slope constraints.

In terms of path-length performance, the B-spline method is superior. The average path length generated by the B-spline optimization is 23% shorter than the average generated with RRT Dubins. The smallest B-spline path is also 4% shorter than the smallest Dubins path. The B-spline path is also easier to track. The maximum tracking error for the B-spline path is 16 times smaller than the maximum tracking error for the Dubins path. This shows that inconsistencies in curvature are even more difficult to handle

when the aircraft is changing elevation.

We can infer that the direct obstacle avoidance B-spline method creates more optimal paths than the RRT Dubins method because its paths are shorter and easier to track. The two-dimensional B-spline method is also faster computationally. However, in three dimensions, the B-spline method has worse computational efficiency than the RRT Dubins method. The main reason is that the constraints for curvature, obstacles, and slope are not convex, and the optimizer is not equipped to handle a large number of optimization variables. Improving the setup and algorithm or improving the computational speed of the hardware is recommended for online operation.

### 5.4.3 Navigation Through Buildings

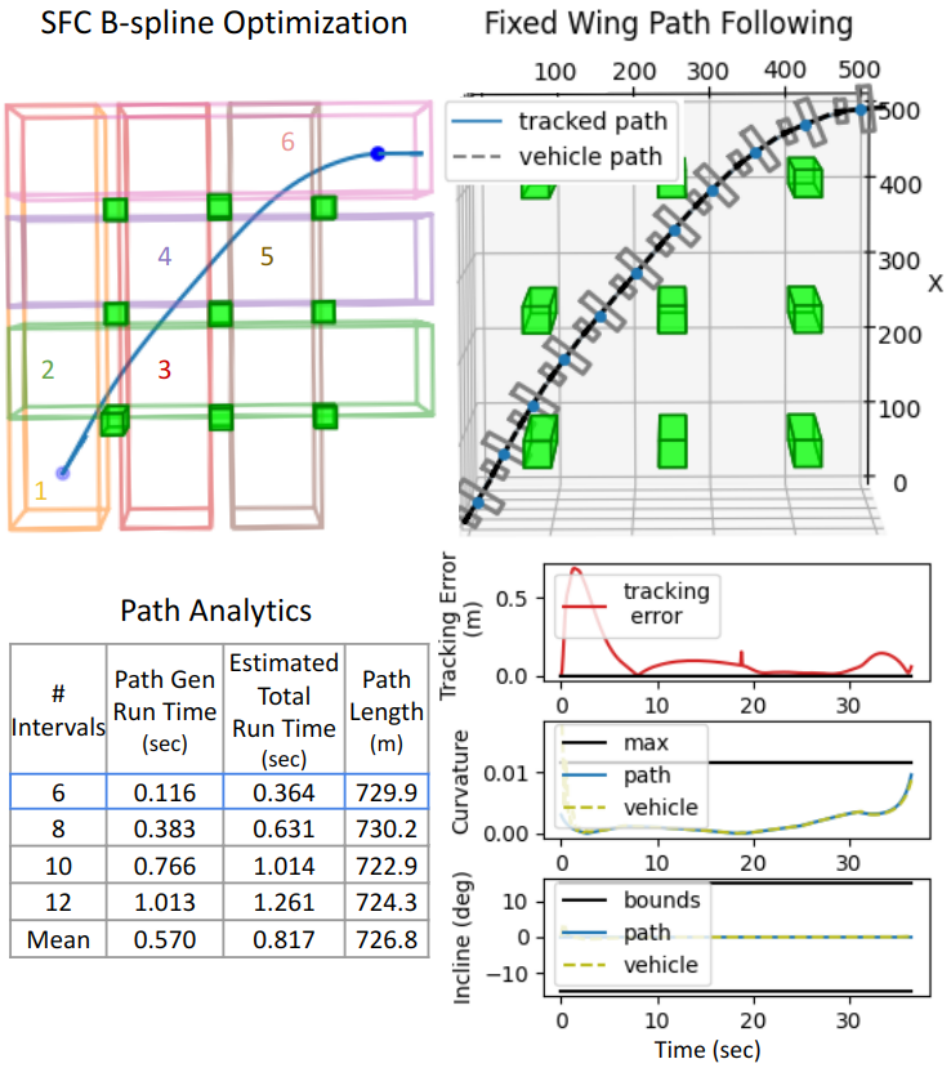



Path Analytics

| | Path Gen Run Time (sec) | # of Paths Created | Path Length (m) |
|---|---|---|---|
| Figure | 1.2 | 27 | 774.5 |
| Mean | 1.03 | 21 | 877.15 |
| High | 1.74 | 30 | 1338.2 |
| Low | 0.507 | 14 | 774.57 |

Figure 5.11: Fixed-wing aircraft following a 2D path through buildings generated using an RRT Dubins method

The set of results in this subsection contains paths that avoid a series of buildings. The B-spline method with SFC constraints and the RRT Dubins method are used to generate these paths. The purpose of these tests is to validate that the SFC B-spline optimization can operate in real time and that it produces shorter and more feasible paths. Fifteen trials were performed for each RRT Dubins scenario to obtain statistics for the path. Statistical averages for the B-spline path data were obtained by varying the number of intervals in the spline for each trial.



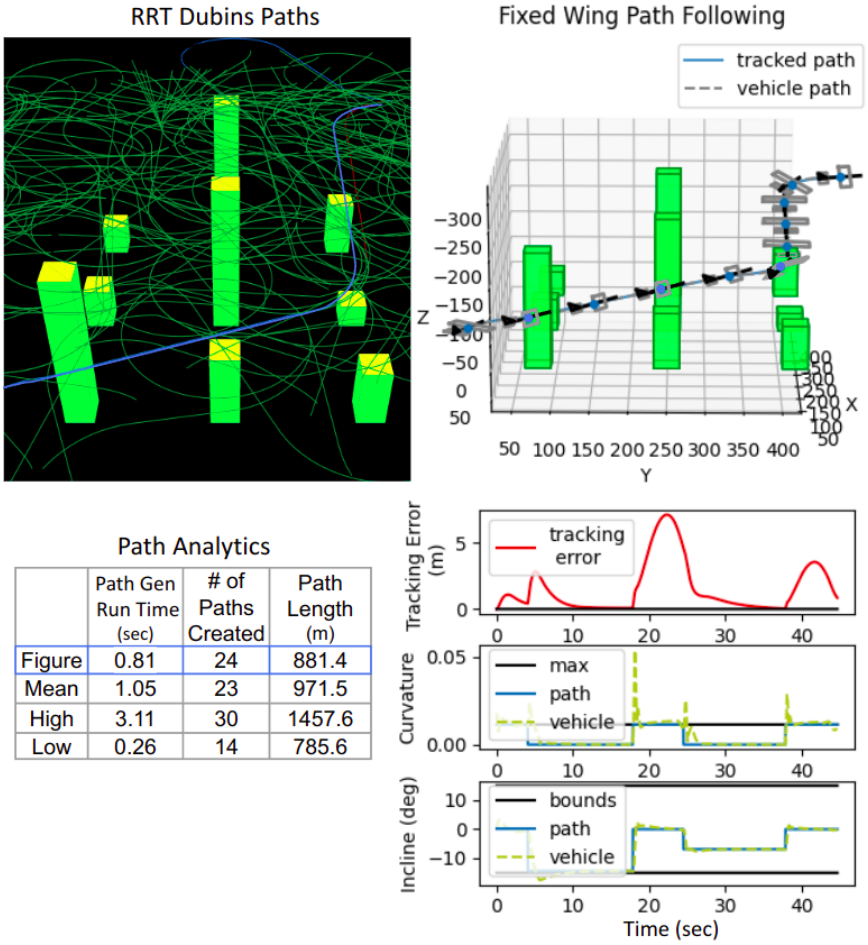

| # Intervals | Path Gen Run Time (sec) | Estimated Total Run Time (sec) | Path Length (m) |
|---|---|---|---|
| 6 | 0.116 | 0.364 | 729.9 |
| 8 | 0.383 | 0.631 | 730.2 |
| 10 | 0.766 | 1.014 | 722.9 |
| 12 | 1.013 | 1.261 | 724.3 |
| Mean | 0.570 | 0.817 | 726.8 |

Figure 5.12: Fixed-wing aircraft following a 2D path through buildings generated using SFC B-spline optimization

Each figure in this subsection contains an image showing an optimized path, an image showing the path being tracked, a table with data about the paths, and a graph showing the tracking data. The highlighted row in each table indicates that the data in that row corresponds to the path shown. The

tables for the B-spline optimization also contain an estimated total runtime when including the MOCS construction and Dijkstra's algorithm. This is only an estimate because we manually selected the SFC's, and do not implement the MOCS construction or Dijkstra's algorithm.

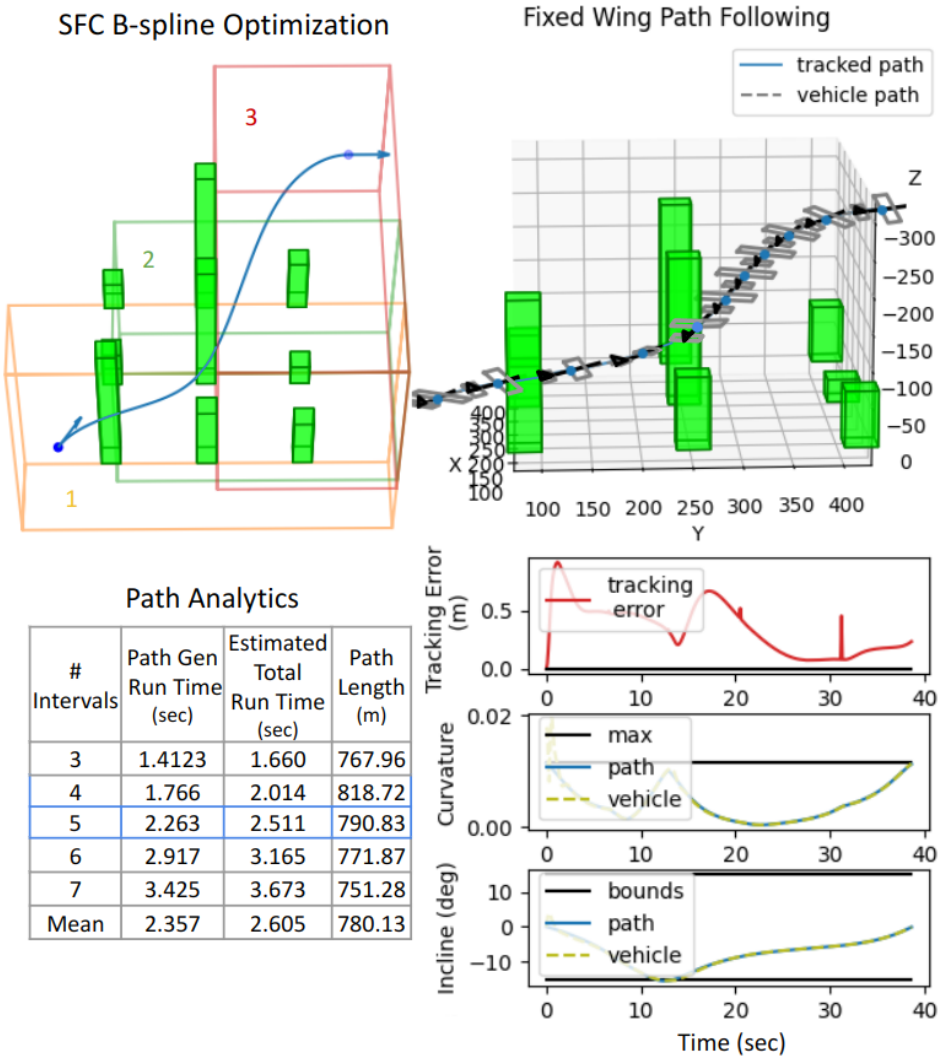
Figures 5.11 and 5.12 contain paths that avoid buildings in a two-dimensional plane. Both cases have the same start and end pose constraints and the same curvature bound of 0.0114. Figure 5.11 shows the results for the RRT Dubins method, and Figure 5.12 shows the results for the B-spline optimization method. The B-spline optimization with the SFC construction has an estimated average runtime of 0.817 seconds, and the RRT Dubins method has an average runtime of 1.03 seconds. The fastest B-spline estimate is also 0.143 seconds faster than the fastest RRT Dubins case. This suggests that the SFC B-spline optimization method is highly capable of operating online in two dimensions. In addition, the length of the average B-spline path is 17% shorter than the average Dubins path. It also has four times less maximum tracking error than the Dubins paths.




| | Path Gen Run Time (sec) | # of Paths Created | Path Length (m) |
|---|---|---|---|
| Figure | 0.81 | 24 | 881.4 |
| Mean | 1.05 | 23 | 971.5 |
| High | 3.11 | 30 | 1457.6 |
| Low | 0.26 | 14 | 785.6 |

Figure 5.13: Fixed-wing aircraft following a 3D path through buildings generated using an RRT Dubins method

Figures 5.13 and 5.14 show optimized paths that avoid a series of buildings of variable height. The building scenario is the same for both paths. The start end end pose conditions are also the same for each path. Each case also has the same curvature constraint of 0.0114 and slope constraint of 0.173. Figure 5.13 shows the results for the RRT Dubins method and Figure 5.14 shows the results for the B-spline optimization method.




| # Intervals | Path Gen Run Time (sec) | Estimated Total Run Time (sec) | Path Length (m) |
|---|---|---|---|
| 3 | 1.4123 | 1.660 | 767.96 |
| 4 | 1.766 | 2.014 | 818.72 |
| 5 | 2.263 | 2.511 | 790.83 |
| 6 | 2.917 | 3.165 | 771.87 |
| 7 | 3.425 | 3.673 | 751.28 |
| Mean | 2.357 | 2.605 | 780.13 |

Figure 5.14: Fixed-wing aircraft following a 3D path through buildings generated using SFC B-spline optimization

The average B-spline optimization runtime is about twice as slow as the average RRT Dubins optimization. Also, the fastest RRT Dubins case is five times faster than the fastest B-spline case. The computational speed of the B-spline optimization is slower because of the extra three-dimensional variables applied to the optimization and because of the slope constraint. However, the resulting paths for the B-spline are superior. The average lengths of the B-spline paths are 20% shorter than the average Dubins paths.

The tracking error is also 10 times smaller than the tracking error for the Dubins path.

We can conclude that the SFC B-spline optimization algorithm produces better paths than the RRT Dubins approach with respect to trackability and path length when avoiding obstacles. In two dimensions, it is a promising solution for online operations. However, when slope constraints are included in the optimization, efficiency improvements to the software, algorithm, or hardware are recommended for online use. For example, the software could be optimized in C++, the algorithm could be convexified, a better optimizer could be employed, or the processor could be upgraded.

### 5.4.4 Navigation to Waypoints

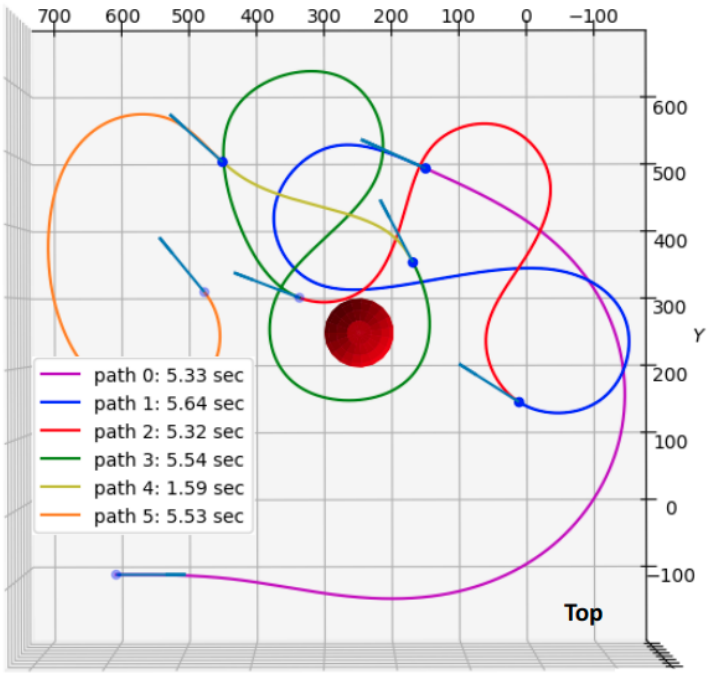



Figure 5.15: Top View of a sequence of B-spline paths through waypoints.

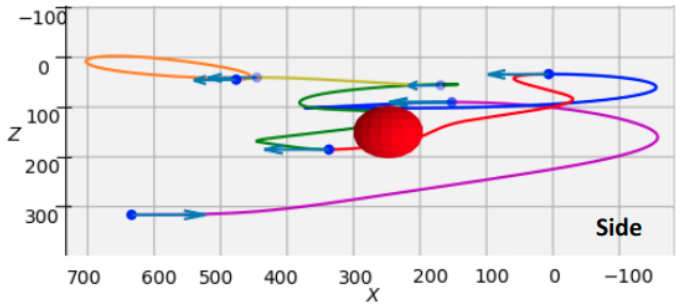



Figure 5.16: Side View of a sequence of B-spline paths through waypoints

The last test in this chapter verifies that the B-spline path planner is capable of generating a sequence of continuous-curvature paths through a given set of waypoints. Each path has a maximum curvature of 0.0114, a maximum slope of 0.2, and must avoid the same obstacle. The waypoints consist of a position and heading. The optimization for each path also includes an acceleration direction constraint for each start point, which is detailed in Section 2.4.2. This allows the path planner to constrain the curvature at the beginning of a path.

Figures 5.15 and 5.16 show two views of the sequence of optimized paths generated using the B-spline path planner. The legend indicates the path number and the runtime to generate the path. Each path is made up of 8 intervals that take about 5.5 seconds to generate. However, Path 4 is created in 1.59 seconds because it is less complex than the other paths. The first path starts as if from the ground level and is significantly longer than the other paths because it must climb more than 200 meters while satisfying a slope constraint. Path 2 also takes wider turns in order to descend to it's goal pose while staying within a slope limitation.

Figure 5.17 shows a simulation of a fixed-wing aircraft tracking the optimized sequence of paths. The waypoints are shown for convenience. Figure 5.18 shows the plotted data for the path and the vehicle trajectory.

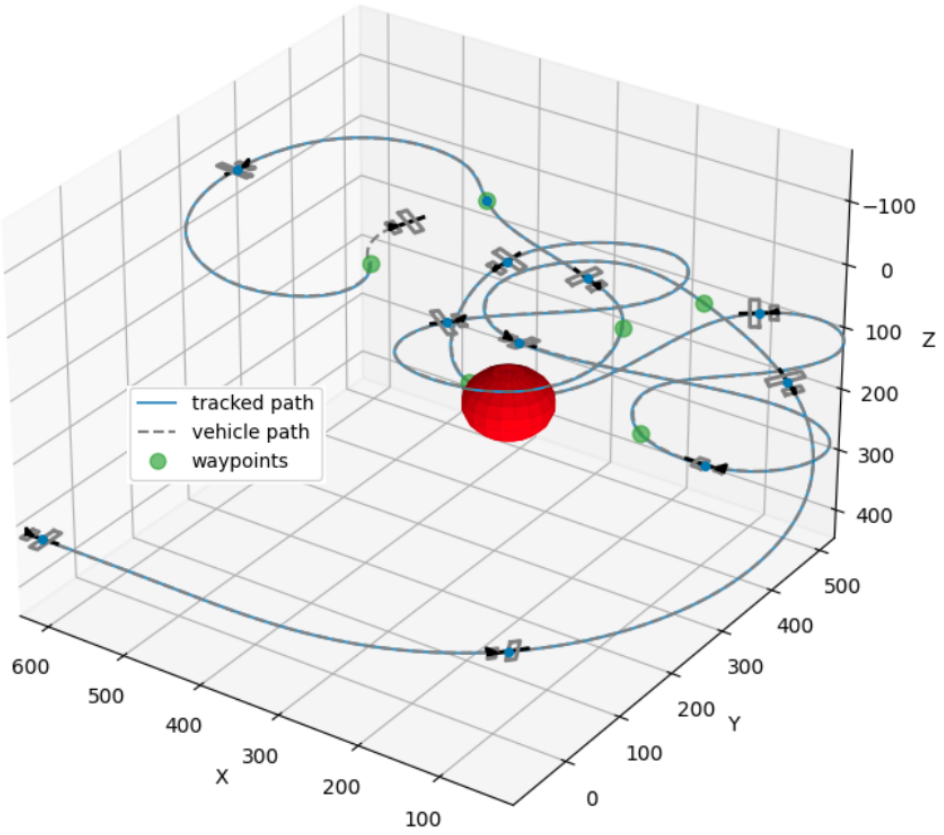



Figure 5.17: Simulation of a fixed-wing aircraft tracking a sequence of paths

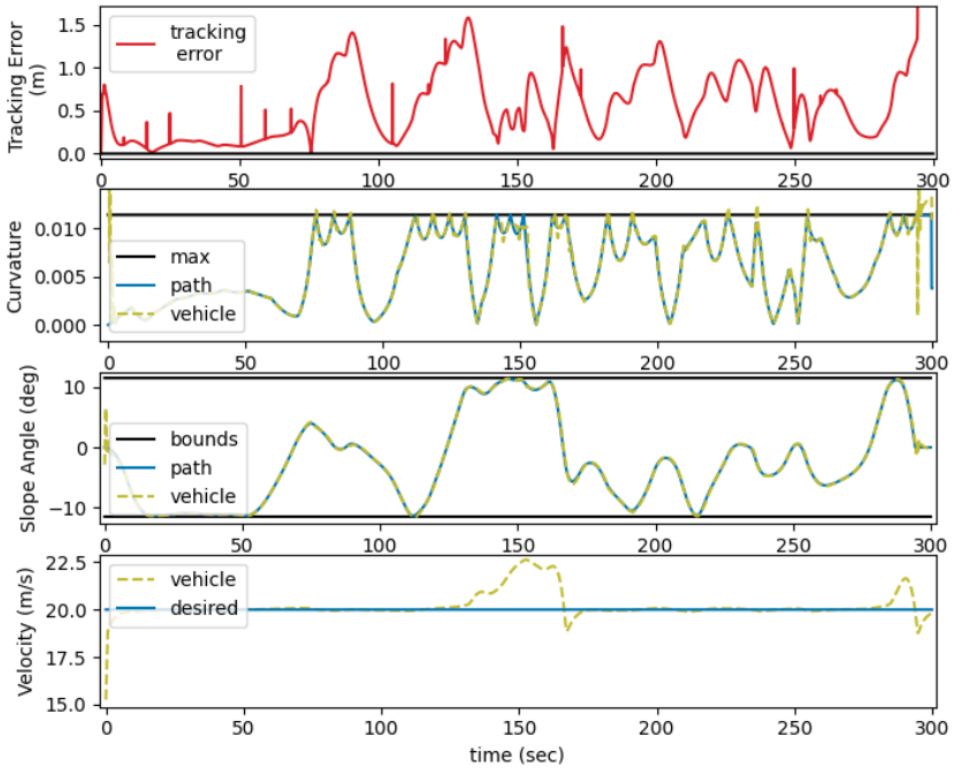



Figure 5.18: Plotted data of a fixed-wing aircraft tracking a sequence of paths

We can observe that the tracking error remains below 1.6 meters and the curvature and slope remain within their designated bounds. The vehicle starts at 15 m/s and approaches 20 m/s for the rest of its trajectory. However, the aircraft occasionally speeds up while descending. Also note that the curvature is continuous throughout the whole path sequence. This makes the entire path trackable for the aircraft. In this scenario, the runtime to generate each path is fast enough to create paths in real time, provided that the next waypoint is given at the start of each path.

## 5.5 Summary

In this chapter, we discussed the challenges and current solutions for autonomous online guidance and control of a fixed-wing aircraft. We proposed a new solution that produces shorter and more easily tracked paths using a B-spline optimization approach with slope constraints. We also described the aircraft model and presented a novel path-following approach to track B-spline paths. The B-spline path planner and the path follower were validated in simulation. From the results, we concluded that the B-spline optimization method produces more optimal paths than other methods in that they are shorter when avoiding obstacles and continuous in curvature. This method is viable for online use in two dimensions; however, improvements in computational efficiency are recommended to operate online in three dimensions.

# 6 Trajectory Generation for Agile Vehicles with Turning Constraints

## 6.1 Introduction to Trajectory Generation for Agile Vehicles

Trajectory generation usually involves the creation of the shortest, fastest, or minimum-energy path to a goal pose or position. To be effective, it must consider the kinematic and dynamic limitations of the vehicle and its environment. The development of real-time trajectory generation algorithms is an ongoing subject of research.

### 6.1.1 Contemporary Trajectory Generation Methods

Traditionally, trajectory-generation methods for agile vehicles have involved setting up the system as an optimal-control problem. This works well for systems that are not overly complicated; however, analytical optimal-control solutions do not permit obstacle constraints to be integrated into the solution. In addition, most non-linear systems need to be solved as discretized optimal control problems, which can significantly degrade computational efficiency. Newer control advances mitigate this issue by convexifying the system through lossless convexification, successive convexification, or a combination of both [62], [63]. These methods are efficient; however, because they are discrete, they do not guarantee feasibility throughout the path. They also scale directly in computational complexity with the distance of the path and require accurate models of the system to be effective.

Another advanced solution is to use spline paths while taking advantage of the flat output space of differentially flat systems to ensure dynamic feasibility [16]. Spline trajectories are smooth and guarantee constraint boundedness throughout the entire path [15], [25]. They are also computationally efficient to compute, and computational costs do not scale with the path length. One shortcoming, however, is that most of these methods do not consider vehicle turning restrictions. Current solutions that consider turning capabilities constrain only the curvature and do so inefficiently by taking discrete samples along the curve [19]. Other methods achieve dynamic feasibility by constraining jerk or by minimizing snap [15], [64]. However, these approaches limit more aggressive and agile maneuvers for the vehicle.

### 6.1.2 B-Spline Trajectories for Vehicles with Turning Limitations

This chapter proposes a new method for generating B-spline trajectories that include the turning restrictions of agile vehicles. It extends the curvature

constraint method from Chapter 3 to also constrain the angular rate and centripetal acceleration of a B-spline path. Additionally, we constrain the tangential acceleration of B-spline trajectories to accommodate the propulsion or deceleration limitations of various vehicles. We then demonstrate how the appropriate turning constraints produce feasible and optimal B-spline trajectories for a specific vehicle. The vehicles used to demonstrate this are 2D models that include the bicycle model, the unicycle model, and a novel boat model. These vehicles are simplified models, and the turning limitations resemble those of a car [65], a two-wheeled robot with differential drive [66], and a tail-controlled missile [67], respectively. Simplified kinematic models are valuable for validating guidance and navigation systems. We finish by demonstrating the effectiveness of turning constraints on trajectories for a fixed-wing aircraft. In the results, we specifically show that constraining curvature produces optimal paths for the bicycle model, constraining angular rate produces optimal paths for the unicycle model, and constraining centripetal acceleration is best suited for the boat model and a fixed-wing aircraft.

## 6.2 Vehicle Constraints

Most agile vehicles used in defense and industry have some turning limitations in the paths they can follow. For example, disregarding slip, a car has a limited turn radius or maximum curvature that it can follow. A two-wheeled robot has course rates bounded by the maximum speed differential of its wheels. A tail-controlled missile can control the acceleration along its lateral axis by adjusting its fins. Similarly, during a coordinated turn, the centripetal acceleration of a fixed-wing aircraft is limited by the maximum load its wings can bear or by the maximum g-force its passengers are allowed to experience.

We propose that the quickest feasible path that a vehicle can traverse from one pose to another has turning constraints that are independent of the speed of the vehicle. In other words, the path design process should respect the vehicle's turning limitation that does not change when the velocity of the vehicle changes. Adding the appropriate turning constraints into the trajectory optimization should produce feasible yet more agile trajectories.

### 6.2.1 Turning Constraints

The equations to constrain the curvature, the angular rate, and the centripetal acceleration of a B-spline below their maximum bounds are respectively

$$\frac{||b(t)' \times b(t)''||}{||b(t)'||^3} \leq \frac{\max\{||b(t)' \times b(t)''||\}}{\min\{||b(t)'||\}^3} \leq C_{\max} \tag{6.1}$$

$$\frac{||b(t)' \times b(t)''||}{||b(t)'||^2} \leq \frac{\max\{||b(t)' \times b(t)''||\}}{\min\{||b(t)'||\}^2} \leq \omega_{\max} \tag{6.2}$$

$$\frac{||b(t)' \times b(t)''||}{||b(t)'||} \leq \frac{\max\{||b(t)' \times b(t)''||\}}{\min\{||b(t)'||\}} \leq a_{R,\max} \tag{6.3}$$

$C_{\max}$ is the curvature bound, $\omega_{\max}$ is the angular rate bound, and $a_{R,\max}$ is the centripetal acceleration bound. Note that the left side of each equation varies only by a factor of the velocity $||b(t)'||$. These constraints are implemented using the analytical roots method described in Section 3.3.2. Although Section 3.3.2 describes only how to constrain curvature, angular rate and centripetal acceleration can be constrained in the same way.

### 6.2.2 Tangential Acceleration Constraint

To create paths that are truly feasible and efficient for the vehicles mentioned, the paths must also consider the vehicle's longitudinal or tangential acceleration limitations. For example, automotive vehicles have limited RPM acceleration, and missiles are limited by the thrust output along their longitudinal axis. Fixed-wing aircraft are also limited by thrust output and even more limited by how fast they can decelerate. Longitudinal acceleration constraints are therefore necessary to generate feasible B-spline paths.

The tangential acceleration $a_T$ of a B-spline is given by

$$a_T(t) = \frac{b(t)'' \cdot b(t)'}{||b(t)'||} \tag{6.4}$$

The tangential acceleration in the direction of travel can be bounded by ensuring that the maximum value of $b(t)'' \cdot b(t)'$ in an interval divided by the minimum value of $||b(t)'||$ in the interval is less than the maximum tangential acceleration. This is expressed as

$$\frac{b(t)'' \cdot b(t)'}{||b(t)'||} \leq \frac{(b(t)'' \cdot b(t)')_{\max}}{||b(t)'||_{\min}} \leq a_{T,\max} \tag{6.5}$$

The tangential deceleration is bounded in the same way, and the constraint is expressed as

$$a_{T,\min} \leq \frac{(b(t)'' \cdot b(t)')_{\min}}{||b(t)'||_{\min}} \leq \frac{b(t)'' \cdot b(t)'}{||b(t)'||} \tag{6.6}$$

where

$$a_{T,\min} < 0 \tag{6.7}$$

For convenience, we will call the numerator of the tangential acceleration G(t). The maximum and minimum values of $G(t)$ are found by finding the roots of its derivative. The derivative and root solution equation is expressed as

$$\frac{dG(t)}{dt} = \frac{d\,(b(t)'' \cdot b(t)')}{dt} = b(t)''' \cdot b(t)' + b(t)'' \cdot b(t)'' = 0 \tag{6.8}$$

In terms of B-spline control points, it is given as

$$(\mathbf{P}_i ML''' \cdot \mathbf{P}_i ML') + (\mathbf{P}_i ML'' \cdot \mathbf{P}_i ML'') = 0 \tag{6.9}$$

Using a symbolic solver, this expression can be converted into polynomial form. For a third-order B-spline it is expressed as

$$(\mathbf{P}_i ML''' \cdot \mathbf{P}_i ML') + (\mathbf{P}_i ML'' \cdot \mathbf{P}_i ML'') = c_2\tau^2 + c_1\tau + c_0 \tag{6.10}$$

The values of the coefficients are given in Section C.1 of the appendix. The maximum numerator value is given by

$$G(t)_{\max} = \max\{G(t_j), G(t_1), G(t_2), G(t_{j+1})\} \tag{6.11}$$

where $t_1$ and $t_2$ are the roots for that interval. Similarly, the minimum is given by

$$G(t)_{\min} = \min\{G(t_j), G(t_1), G(t_2), G(t_{j+1})\} \tag{6.12}$$

To find the minimum values of $\|b(t)'\|$, see Section 3.3.2. These maximum and minimum values are then used to find the tangential acceleration bounds of the B-spline.

## 6.3 Vehicle Models

The 2D vehicle models simulated in this chapter are the bicycle model, the unicycle model, and the boat model. These models have simple kinematics, and the turning limitations of these models resemble those of a car, a two-wheeled robot with differential drive, and a tail-controlled missile, respectively. The bicycle model is limited by curvature, the unicycle model by angular rate, and the boat model by centripetal acceleration. We also simulate a 3D fixed-wing aircraft model. During a coordinated turn, the fixed-wing aircraft is limited by centripetal acceleration.

The following subsections describe the kinematics and equations of motion of the vehicles. The variables $x$ and $y$ are the coordinates of the vehicle in the inertial frame. $\chi$ is the orientation of the vehicle velocity vector $\mathbf{V}$, and $\psi$ is the vehicle heading. $\mathbf{x_I}$ and $\mathbf{y_I}$ describe the inertial frame vectors and $\mathbf{x_b}$ describes the longitudinal body frame vector of the vehicle. These variables are used to describe the motion of each of the 2D vehicle models in this section.

### 6.3.1 Bicycle Model

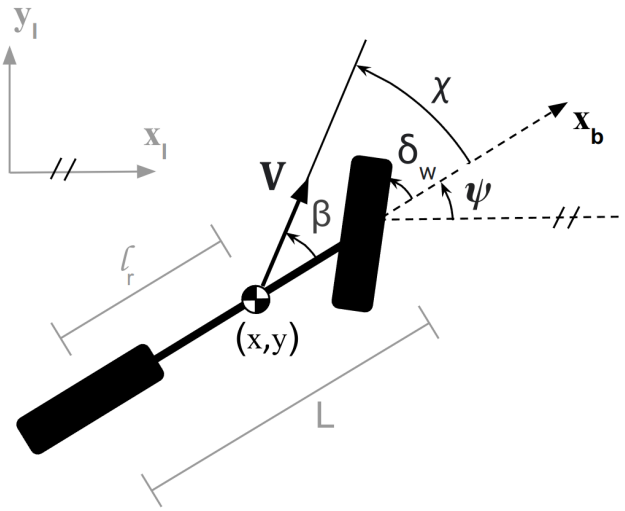



Figure 6.1: Bicycle model

Figure 6.1 shows a graphic representation of the bicycle model. $L$ is the length of the bicycle, and $l_r$ is the distance from the back wheel to the center of gravity or coordinate position of the bicycle. $\delta_w$ is the turn angle of the wheel, and $\beta$ is the offset angle of the velocity vector from the body frame vector $\mathbf{x_b}$. The equations of motion for the bicycle model are

$$\dot{x} = V\cos(\beta + \psi) \tag{6.13}$$

$$\dot{y} = V\sin(\beta + \psi) \tag{6.14}$$

$$\dot{\psi} = \frac{V\ \tan\delta_w\ \cos\beta}{L} \tag{6.15}$$

$$\beta = \tan^{-1}\left(\frac{l_r\tan\delta_w}{L}\right) \tag{6.16}$$

$$\ddot{x} = a_T(t)\cos(\beta + \psi) - (\dot{\psi} + \dot{\beta})V\sin(\beta + \psi) \tag{6.17}$$

$$\ddot{y} = a_T(t)\sin(\beta + \psi) + (\dot{\psi} + \dot{\beta})V\cos(\beta + \psi) \tag{6.18}$$

$$\ddot{\psi} = a_T(t)\cos\beta\tan\delta_w + V\left(\Delta_w(t)\cos\beta\sec^2\delta_w - \dot{\beta}\sin\beta\tan\delta_w\right) \tag{6.19}$$

$$\dot{\beta} = \frac{Ll_r\Delta_w(t)\sec^2\delta_w}{l_r^2\tan^2\delta_w + L^2} \tag{6.20}$$

$$\dot{V} = a_T(t) \tag{6.21}$$

$$\dot{\delta}_w = \Delta_w(t) \tag{6.22}$$

The inputs to the model are the steering angular rate of the wheel $\Delta_w$ and the tangential acceleration $a_T$. Note that the velocity vector of the bicycle model is only collinear with the orientation of the body when $\delta_w$ is equal to zero.

### Bicycle Turning Limitations

The turning constraint best suited for the bicycle model is the maximum curvature constraint because the turn radius of the bicycle, under a no-skid assumption, is independent of its velocity. If we divide both sides of Equation 6.15 by the velocity $V$, and recognize that at a constant steering angle $\dot{\chi}$ is equal to $\dot{\psi}$, we get

$$C_v = \frac{\dot{\chi}}{V} = \frac{\dot{\psi}}{V} = \frac{\tan\delta_w\cos\beta}{L} \tag{6.23}$$

The curvature is dependent only on the steering angle of the wheel.

The maximum curvature occurs when the steering angle of the wheel is saturated and is given by

$$C_{\max} = \frac{\tan(\delta_{w,\max})\cos(\beta_{\max})}{L} \tag{6.24}$$

where

$$\beta_{\max} = \tan^{-1}\left(\frac{l_r\tan\delta_{w,\max}}{L}\right) \tag{6.25}$$

However, because the velocity vector can change in relation to the orientation of the bicycle, the curvature of the vehicle path can exceed this value when

$\delta_w$ is changing. When the steering rate $\Delta_w$ is nonzero, $\dot{\beta}$ is also nonzero, which means that the velocity vector is rotating in reference to the vehicle body. Therefore, the curvature of the vehicle path when $\delta_w$ is changing is evaluated as

$$C_v = \frac{\dot{\beta} + \dot{\psi}}{V} \tag{6.26}$$

Nonetheless, the curvature value when the steering angle is changing cannot be maintained for consistent periods of time. We will therefore limit the path curvature based on the curvature when $\dot{\beta}$ is equal to zero (Equation 6.24).

The maximum angular rate for the bicycle model is expressed as

$$\omega_{\max} = \frac{V_{\max} \tan(\delta_{w,\max}) \cos(\beta_{\max})}{L} \tag{6.27}$$

and the maximum centripetal acceleration is expressed as

$$a_{R,\max} = \frac{V_{\max}^2 \tan(\delta_{w,\max}) \cos(\beta_{\max})}{L} \tag{6.28}$$

However, these values can only be reached when the bicycle is travelling at its maximum velocity.

### 6.3.2 Unicycle Model

Figure 6.2 shows a graphic representation of the unicycle model. This model is simpler because it has direct control over the angular rate $\omega(t)$. In addition, the orientation $\chi$ of the velocity vector is always equal to the orientation of the vehicle heading $\psi$.

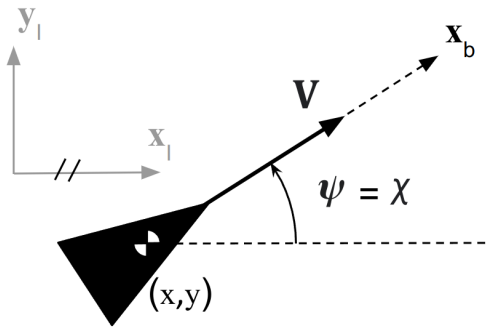



Figure 6.2: Unicycle model

The equations of motion for the unicycle model are

$$\dot{x} = V \cos\psi \tag{6.29}$$

$$\dot{y} = V \sin\psi \tag{6.30}$$

$$\dot{\psi} = \omega(t) \tag{6.31}$$

$$\ddot{x} = a_T(t) \cos\psi - V\omega(t) \sin\psi \tag{6.32}$$

$$\ddot{y} = a_T(t)\sin\psi + V\omega(t)\cos\psi \tag{6.33}$$

$$\dot{V} = a_T(t) \tag{6.34}$$

The inputs to the model are the tangential acceleration $a_T$ and the angular rate $\omega$.

#### Unicycle Turning Limitations

The turning constraint that is best suited for a unicycle model path is a maximum angular rate constraint because the unicycle has direct control over its angular rate, regardless of its velocity. The maximum angular rate is equal to its maximum angular rate input command $\omega_{c,\max}$. Therefore, we express the bound as

$$\omega_{\max} = \omega_{c,\max} \tag{6.35}$$

If we set some lower bound operating velocity $V_{\min}$, we can also find a maximum curvature for the unicycle path. The maximum curvature is

$$C_{\max} = \frac{\omega_{c,\max}}{V_{\min}} \tag{6.36}$$

and can only be reached while travelling at the minimum velocity. This suggests that we can achieve tighter curves at lower velocities. The maximum centripetal acceleration that the unicycle can reach is evaluated as

$$a_{R,\max} = \omega_{c,\max} V_{\max} \tag{6.37}$$

However, this value can only be achieved when the unicycle is moving at its maximum velocity.

### 6.3.3 Boat Model

Figure 6.3 shows a graphic representation of the boat model. $\delta_r$ is the steering angle, $c_r$ is a rudder constant, and $c_b$ is a boat constant that is small in value. The rudder constant is proportional to the size of the rudder, and larger values create faster rotational speeds. In contrast, the boat constant is proportional to the length of the boat, and larger boat constant values create slower rotational speeds. Also, the orientation $\psi$ of the boat is always aligned with the orientation $\chi$ of the velocity vector.

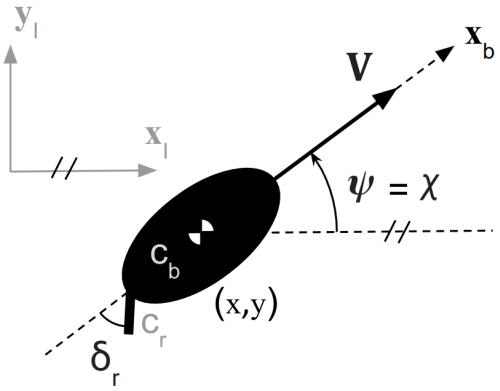



Figure 6.3: Boat model

The equations of motion for the boat model are

$$\dot{x} = V\cos\psi \tag{6.38}$$

$$\dot{y} = V\sin\psi \tag{6.39}$$

$$\dot{\psi} = \frac{c_r \sin(-\delta_r)\ \tan^{-1}(V^2)}{V + c_b} \tag{6.40}$$

$$\ddot{x} = a_T(t)\cos\psi - V\dot{\psi}\sin\psi \tag{6.41}$$

$$\ddot{y} = a_T(t)\sin\psi + V\dot{\psi}\cos\psi \tag{6.42}$$

$$\ddot{\psi} = \frac{c_r a_T(t)\sin(\delta_r)\tan^{-1}(V^2)}{(c_b+V)^2} - \frac{2c_r V a_T(t)\sin(\delta_r)}{(V^4+1)(c_b+V)} - \frac{c_r\Delta_r(t)\cos(\delta_r)\tan^{-1}(V^2)}{c_b+V} \tag{6.43}$$

$$\dot{V} = a_T(t) \tag{6.44}$$

$$\dot{\delta}_r = \Delta_r(t) \tag{6.45}$$

The inputs to the model are the tangential acceleration $a_T$ and the steering rate of the rudder $\Delta_r$. Note that the boat constant $c_b$ prevents infinite angular rate when the velocity is equal to zero.

### Boat Turning Limitations

The turning constraint best suited for a boat model trajectory is a centripetal acceleration constraint. This is because the angular rate of the boat model is almost inversely proportional to the velocity of the boat, and therefore the centripetal acceleration is practically independent of the velocity. The centripetal acceleration of the boat is given by

$$a_{c,v} = \frac{V c_r \sin(-\delta_r)\ \tan^{-1}(V^2)}{V + c_b} \tag{6.46}$$

We can create a simplified equation for the maximum centripetal acceleration by noting that $c_b$ is close to zero, and the maximum value for the inverse tangent of $V^2$ is equal to $\pi/2$. Therefore the maximum centripetal acceleration is approximately

$$a_{R,\max} \approx c_r \sin(\delta_{r,\max})\ \frac{\pi}{2} \tag{6.47}$$

Given a lower limit operating velocity $V_{\min}$, we can also find the maximum curvature and angular rate bounds for the boat model. The maximum curvature is

$$C_{\max} = \frac{c_r \sin(\delta_{r,\max})\ \pi}{2V_{\min}^2} \tag{6.48}$$

and the maximum angular rate is

$$\omega_{\max} = \frac{c_r \sin(\delta_{r,\max})\ \pi}{2V_{\min}} \tag{6.49}$$

Equation 6.48 suggests tighter curves at lower velocities, and Equation 6.49 suggests faster rotations at lower velocities. Note that these values can only be achieved when traveling at the minimum operating velocity.

### 6.3.4 Fixed-wing Aircraft Model

The fixed-wing aircraft model with its kinematics and dynamics is detailed in Section 5.2.

#### Fixed-Wing Aircraft Turning Limitations

Traditionally, curvature is constrained for paths generated for a fixed-wing aircraft. However, the curvature capabilities of a fixed-wing varies with the flight speed of the aircraft. A better suited constraint is bounding the centripetal acceleration of the trajectories created for fixed-wing aircraft. This is because the maximum centripetal acceleration is practically independent of the operating speed of the aircraft.

The maximum centripetal acceleration can be derived from the maximum curvature equation (Equation 5.9) of a fixed-wing aircraft during a coordinated turn with zero wind conditions as

$$\begin{aligned} a_{R,\max} &= V_g^2 C_{\max} \\ &= g \tan \phi_{\max} \end{aligned} \tag{6.50}$$

where $\phi_{\max}$ is the maximum roll angle.

#### Fixed-Wing Aircraft Tangential Acceleration Limitations

A fixed-wing aircraft is also limited in its longitudinal or tangential acceleration. From [49], the longitudinal acceleration is defined as

$$\begin{aligned} \dot{u} &= rv - qw + \frac{f_x}{m} \\ &= rv - qw + \frac{-mg \sin\theta + F_{\mathrm{p}}(V_a, \delta_t) + F_D(V_a, \alpha) + F_{D_e}(V_a, \alpha, \delta_e)}{m} \end{aligned} \tag{6.51}$$

and the drag force is defined as

$$\begin{aligned} F_D(V_a, \alpha) = \frac{1}{2}\rho V_a^2 S \Bigg( &(-C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha) \\ &+ (-C_{D_q}(\alpha)\cos\alpha + C_{L_q}(\alpha)\sin\alpha)\frac{qc}{2V_a} \Bigg) \end{aligned} \tag{6.52}$$

where the $C_*$ values are lift and drag coefficients, $S$ is the planform area of the wing, and the propulsion force $F_{\mathrm{p}}$ is a function of airspeed and throttle. The drag force due to the control surface $F_{D_e}$ is a function of airspeed, angle of attack, and elevator deflection.

In conditions with zero wind and level flight, $r$, $q$, $w$, $\alpha$, $\theta$ and $\delta_e$ are approximately zero. In these conditions, the maximum forward acceleration simplifies to

$$\begin{aligned} a_{T,\max}(V_a) &= \dot{u}_{\max} \\ &= \frac{F_{\mathrm{p}}(V_a, \delta_{t,\max}) + F_D(V_a, \alpha)}{m} \\ &= \frac{F_{\mathrm{p}}(V_a, \delta_{t,\max}) - \frac{1}{2}\rho V_a^2 S C_D(0)}{m} \end{aligned} \tag{6.53}$$

To find the maximum deceleration, or minimum tangential acceleration we remove the propulsion force

$$a_{T,\min}(V_a) = \frac{-\frac{1}{2}\rho V_a^2 S C_D(0)}{m} \tag{6.54}$$

Notice that the tangential acceleration bounds are a function of airspeed. This means that we would have to find the minimum and maximum based on an airspeed range that the aircraft is expected to operate.

## 6.4 Vehicle Control

To control each of the 2D vehicle models we implement a proportional gain trajectory tracker with tangential acceleration control input. Each 2D vehicle model also has its own turning control input. It is important that the vehicle has control capable of accurate tracking so that we can know that any tracking error is due to the vehicle reaching its physical limitations.

Each of the 2D vehicle models accepts a tangential acceleration input $a_{T,c}$ that propels the vehicle in the direction of its velocity vector $\mathbf{V}$. This is important for tracking the speed and position of the trajectory. Each vehicle has the same tangential acceleration control law. A detailed description of this control law can be found in Section C.2.1 of the appendix. The determination of the desired course angle $\chi_d$ is also common for each 2D vehicle. The course angle defines the direction of travel along the velocity vector $\mathbf{V}$. A derivation for $\chi_d$ can be found in Section C.2.2 of the appendix. The trajectory tracking control law for the fixed-wing model is detailed in Section C.3.

### 6.4.1 Bicycle Control

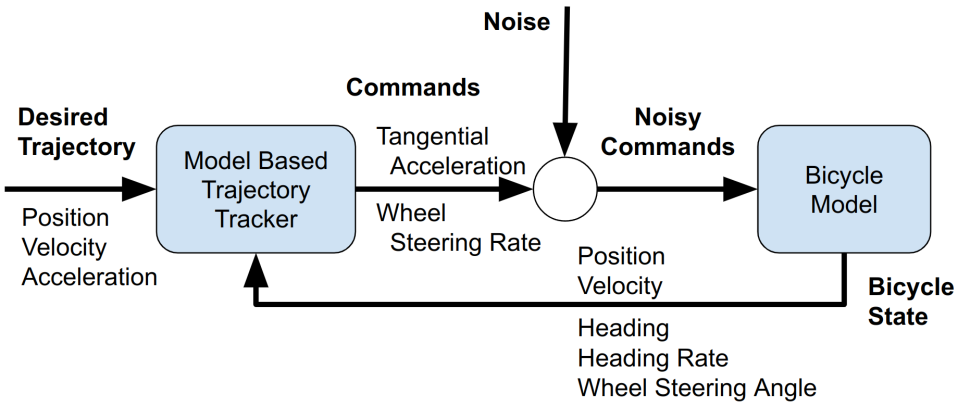



Figure 6.4: Bicycle control loop diagram

The control loop diagram for the bicycle model is shown in Figure 6.4. The input to the controller is the time-dependent trajectory with position, velocity, and acceleration data. The state of the bicycle is returned as feedback. Note

that in the control loop diagram the input to the model also has an added noise term. The two control outputs are the tangential acceleration $a_{T,c}$ and the steering rate $\Delta_{w,c}$.

The course of the bicycle model is controlled by the steering angle $\delta_w$ of the front wheel. This angle determines the direction of the velocity vector and is controlled by $\Delta_{w,c}$. The control law for the steering rate is described in Section C.2.3 of the appendix.

### 6.4.2 Unicycle Control

Figure 6.5 shows the control loop for the unicycle model. The control output is the tangential acceleration $a_{T,c}$ and the angular rate $\omega_R$. This means that the unicycle has direct control over the angular rate of its heading. Since the heading $\psi$ is equal to the course $\chi$, the unicycle also has direct control over its course angle rate. The control law for the angular rate control $\omega_c$ is described in Section C.2.4 of the appendix.

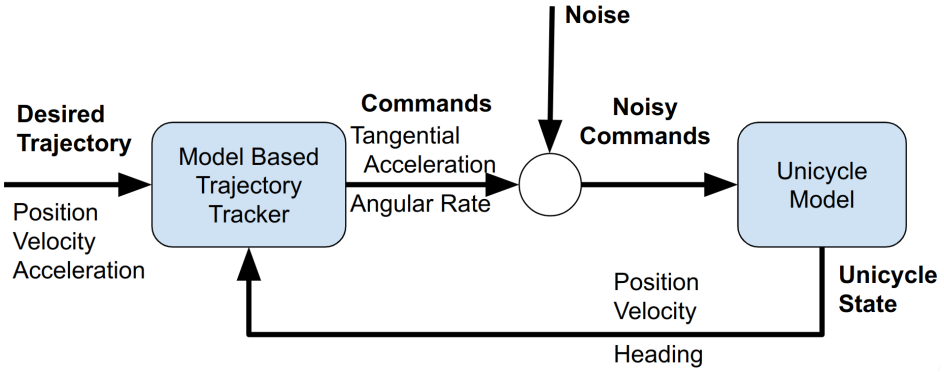



Figure 6.5: Unicycle control loop diagram

The input to this controller contains the position, velocity, and acceleration data of the trajectory. It also receives the unicycle state as feedback, which includes the position, velocity, and heading. Note that this system is differentially flat, meaning that the state and control input of the system can be completely determined from the outputs of the system. This means that trajectories are easier to track with the unicycle model because the full state of the vehicle is described within the trajectory.

### 6.4.3 Boat Control

The control loop for the boat model is shown in Figure 6.6. The input to the controller includes the position, velocity, and acceleration of the trajectory. Feedback includes the position, velocity, orientation, and rudder angle of the boat. The boat uses the angle of the rudder $\delta_r$ to control its course angle $\chi$. The control output is the tangential acceleration $a_{T,c}$ and the rudder steering rate $\Delta_{r,c}$. The control law for the steering rate of the rudder $\Delta_{r,c}$ is detailed in Section C.2.5 of the appendix.

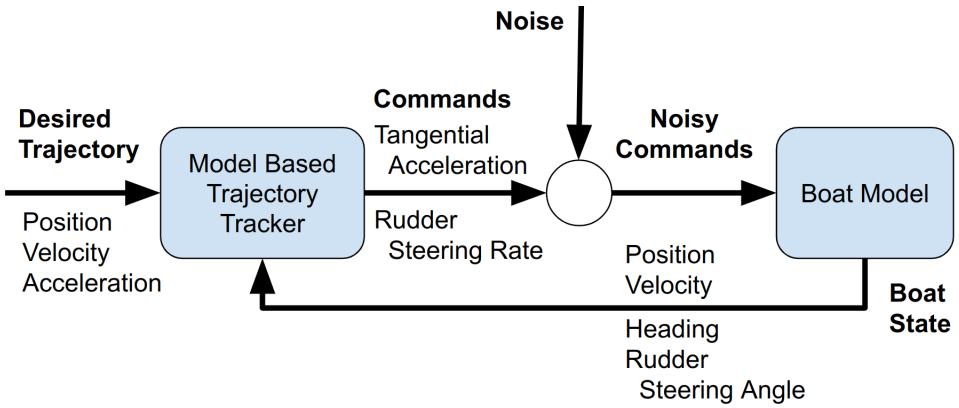



Figure 6.6: Boat control loop diagram

### 6.4.4 Fixed-wing Aircraft Control

The control loop diagram for a fixed-wing aircraft is shown in Figure 6.7. The trajectory tracker accepts the desired trajectory in the form of B-spline control points and a scale factor, to then convert that into time-constrained position, velocity, and acceleration data. The output of the trajectory tracker is a desired airspeed, climb rate, course angle, and a feedforward roll. This is then fed to the autopilot as input commands. The control law for the trajectory tracker is detailed in Section C.3 of the appendix. For details on how these commands are used in the autopilot, see *Small Unmanned Aircraft: Theory and Practice* [49].

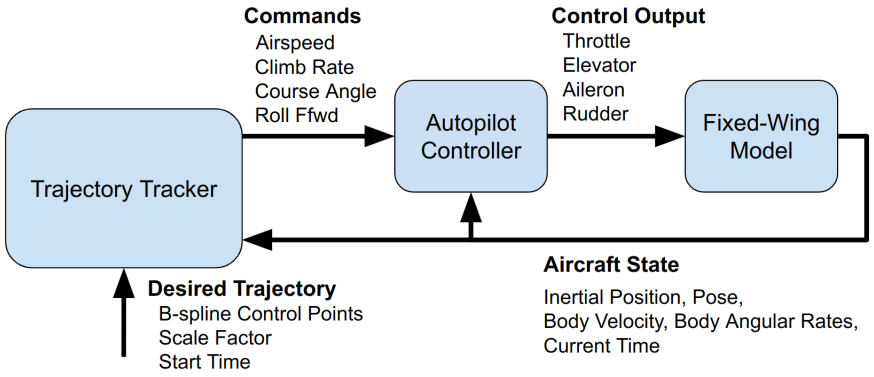



Figure 6.7: Fixed-wing aircraft control loop diagram

## 6.5 Results and Discussion

This section demonstrates the optimality of trajectories for agile vehicles that are limited by turning constraints. Trajectory tracking results are shown for three different 2D vehicle models and one 3D model. The vehicle models tested are the bicycle model, the boat model, the unicycle model, and a fixed-wing aircraft. In all cases, the trajectory is constrained to an initial

and final position and velocity, and by a maximum velocity and maximum tangential acceleration. The goal of these experiments is to show that the optimal trajectory for a vehicle will be constrained by the turning limitation of that vehicle which is independent of velocity. We define optimality as the quickest feasible trajectory for the vehicle. In the figures, the yellow lines are vehicle data, the blue lines are trajectory data, and the red lines are error data.

### 6.5.1 Bicycle Tracking Results

In this subsection, we show three trajectories followed by the bicycle model, each with the same initial and final conditions. Figure 6.8 shows the bicycle tracking a trajectory without any turning constraints. We can see that the bicycle veers off the path when it is attempting to track tight curves, showing that the trajectory is infeasible. The tracking error reaches more than four meters and the curvature of the path exceeds the curvature limitations of the bicycle.

Figure 6.9 shows the bicycle model following a trajectory constrained by the maximum angular rate of the bicycle equal to $\frac{C_{\max}}{V_{\max}}$. However, because the maximum angular rate can only be reached when traveling at full speed, this constraint does not adequately bound the turning capabilities of the bicycle. In other words, the bicycle cannot achieve the desired angular rates of the path at the desired velocities of the path. This makes the trajectory infeasible for the bicycle model.

Figure 6.10 contains the results for the bicycle tracking a trajectory constrained by the maximum curvature limitations of the bicycle. The vehicle remains within 0.0025 meters of the track and traverses the path in 0.53 seconds. This trajectory is feasible and achieves its goal pose quickly.

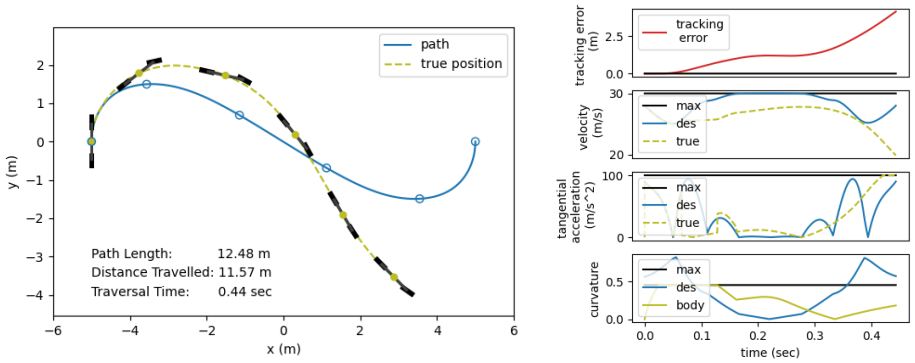



Figure 6.8: Bicycle trajectory with no turning constraints

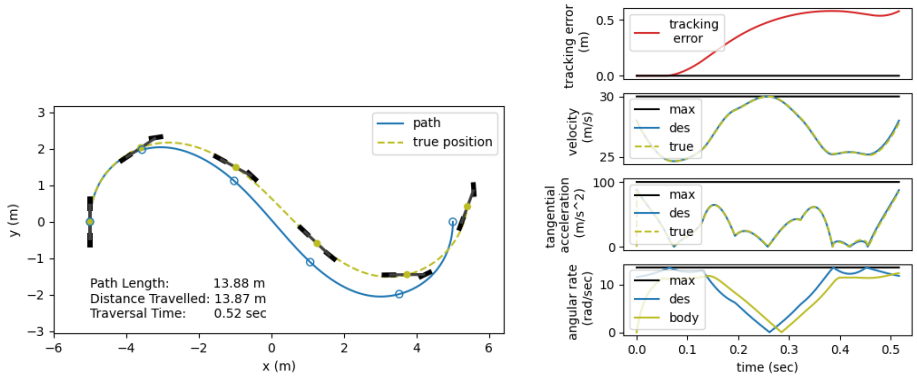



Figure 6.9: Bicycle trajectory with angular rate constraint

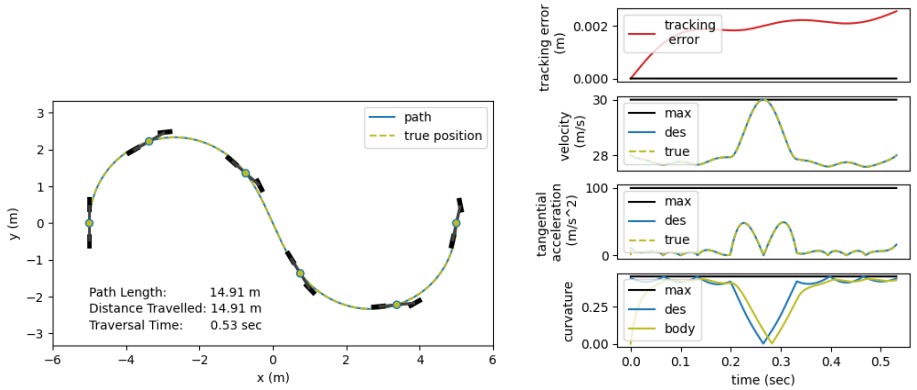



Figure 6.10: Bicycle trajectory with curvature constraint

Although the trajectory created without turning constraints was slightly faster than the trajectory with curvature constraints, it was infeasible. The trajectory created with angular rate constraints was also infeasible because it did not adequately bound the kinematic capabilities of the bicycle. We conclude that the optimal trajectory for the bicycle is the one created with curvature constraints because it maintained feasibility conditions for the bicycle and because it was also able to produce a minimum-time trajectory. This is because the curvature constraint is independent of the operating velocity and the optimizer was free to explore the true limitations of the vehicle.

### 6.5.2 Unicycle Tracking Results

In this subsection, we present results for three different unicycle trajectories. Figure 6.11 shows the unicycle following a trajectory with centripetal acceleration constraints with a maximum centripetal acceleration of $\frac{\omega_{\max}}{V_{\max}}$. The tracking error reaches more than three meters and demonstrates an infeasible trajectory. Note that the centripetal acceleration limit of the path exceeds the limitations of the unicycle when traveling at lower velocities and therefore is infeasible for the unicycle.

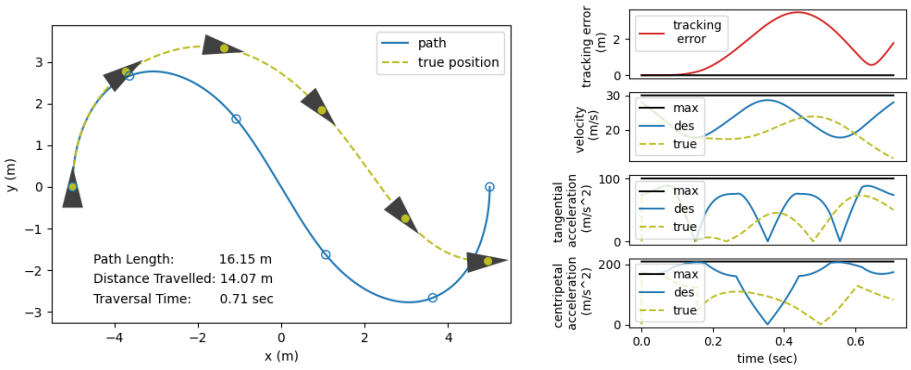



Figure 6.11: Unicycle trajectory with centripetal-acceleration constraint

Figure 6.12 shows data for the unicycle tracking a curvature-constrained path with maximum curvature equal to $(\omega_{\max} V_{\max})$. The scaling of the path is decreased to fit the entire path in the plot. The trajectory is feasible, but long and inefficient. This is because the maximum curvature bound is calculated with respect to the maximum velocity of 28 m/s and does not vary with varying velocities. In other words, the curvature capabilities of the unicycle are greater at lower velocities, but the path does not allow this.

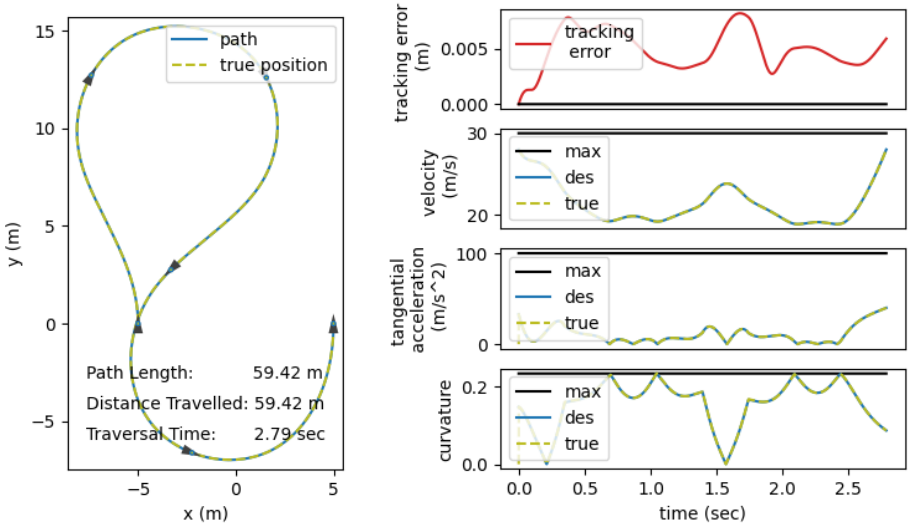



Figure 6.12: Unicycle trajectory with curvature constraint

Figure 6.13 displays the results of the unicycle model that follows a trajectory with angular rate constraints. This trajectory is much faster than the curvature-constrained trajectory and is also feasible. The tracking error remains below 0.0075 meters and reaches its target location in less than 1.5 seconds. None of the kinematic constraints are violated.

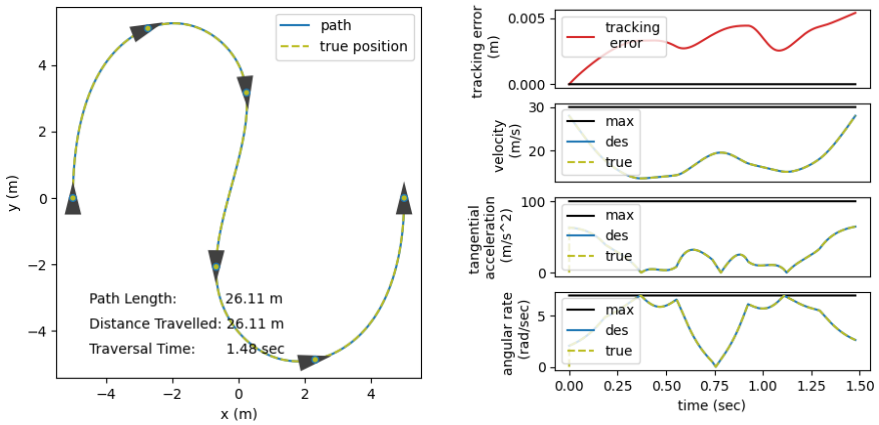



Figure 6.13: Unicycle trajectory with angular rate constraint

The optimal trajectory for the unicycle model includes angular rate constraints because they allow the unicycle to stay on the path with a short traversal time. Angular-rate constraints are best suited for the unicycle model because the angular rate of the unicycle is independent of its velocity.

### 6.5.3 Boat Tracking Results

This subsection contains results for three trajectory tracking cases for the boat model. Figure 6.14 shows the boat tracking a trajectory without a turning constraint. This trajectory is fast, but infeasible to follow. The centripetal acceleration of the boat is saturated when the centripetal acceleration of the trajectory exceeds the boat's limit as dictated by the maximum rudder angle.

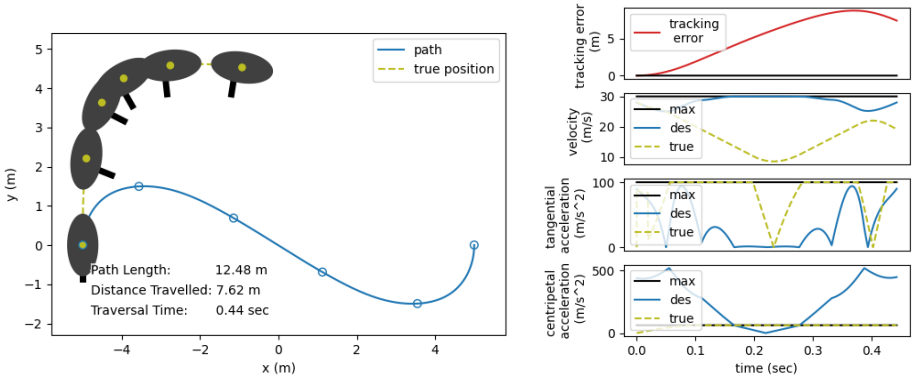



Figure 6.14: Boat trajectory with no turning constraint

Figure 6.15 contains the results of the boat model tracking a trajectory with angular rate constraints. Note that the scaling of the left plot is increased to fit the whole path in the image. The maximum angular rate is equal to $(a_{R,\max} V_{\max})$. From the figure, we can see that the trajectory is feasible but long and inefficient. This trajectory is suboptimal because its angular rate

constraints are too conservative when traveling at velocities lower than the maximum velocity.

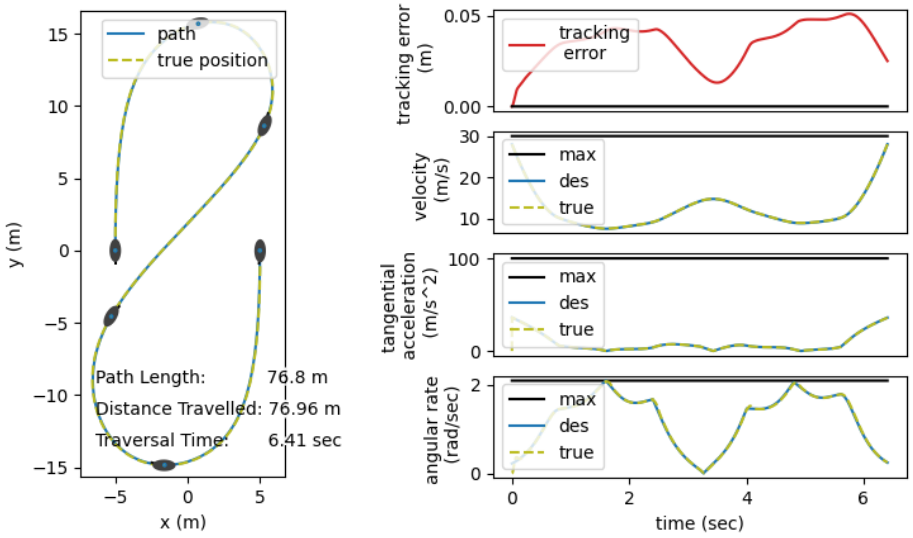



Figure 6.15: Boat trajectory with angular rate constraint

Figure 6.16 demonstrates the results of the boat tracking a trajectory with centripetal acceleration constraints. This path is fast and feasible with a maximum tracking error of 0.09 meters. The optimal trajectory for the boat model is the trajectory with centripetal acceleration constraints. This emphasizes again that an optimal trajectory for a vehicle will have turning constraints that are independent of the vehicle velocity.

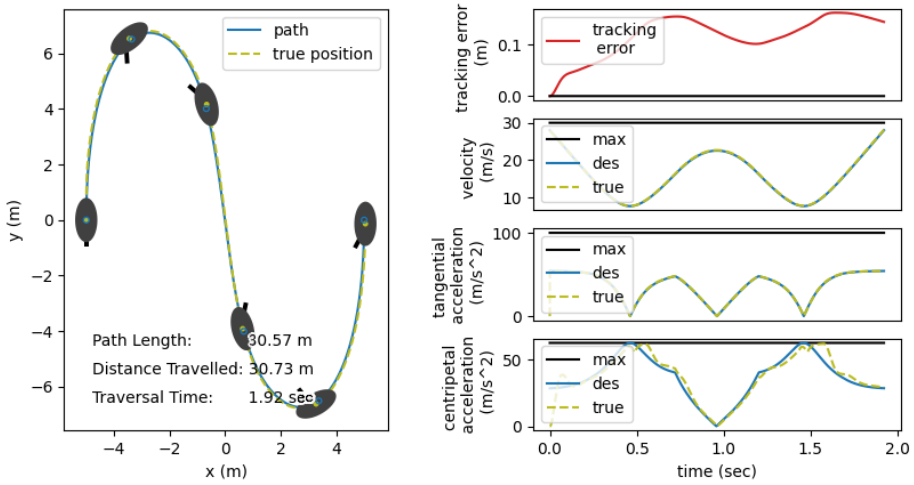



Figure 6.16: Boat trajectory with centripetal acceleration constraint

### 6.5.4 Fixed-Wing Aircraft Tracking Results

In this section, we have the fixed-wing aircraft track three different trajectories; one with only a tangential acceleration constraint, one with a centripetal

acceleration constraint but no tangential acceleration constraint, and one with both centripetal and tangential acceleration constraints. We also have the aircraft track a traditional curvature-constrained path at its maximum velocity. Each trajectory has the same maximum velocity and the same start and end conditions. The goal of these experiments is to show that centripetal acceleration-constrained trajectories produce more optimal paths for a fixed-wing aircraft than curvature-constrained paths. We also show that tangential acceleration constraints are also needed to create feasible trajectories.

Figure 6.17 shows the fixed-wing model tracking a trajectory constrained by tangential acceleration. The trajectory exceeds the centripetal acceleration limitations of the aircraft, which causes it to drift off track by more than 75 meters. This trajectory is infeasible.

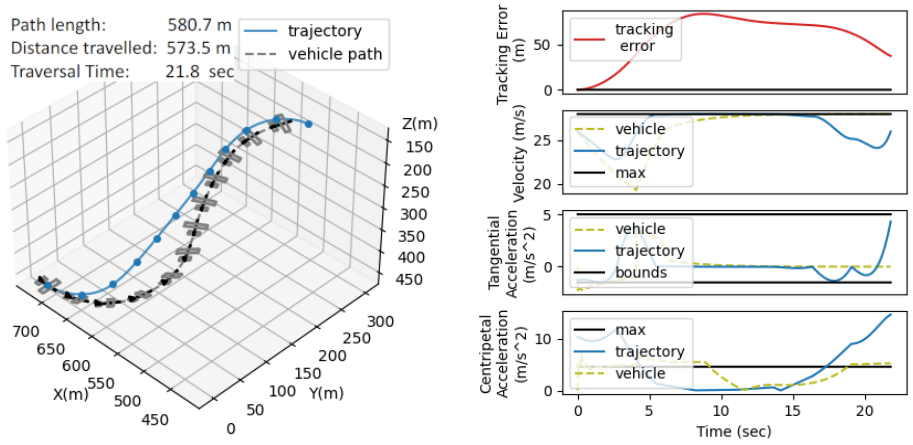



Figure 6.17: Fixed-wing aircraft trajectory with tangential acceleration constraint

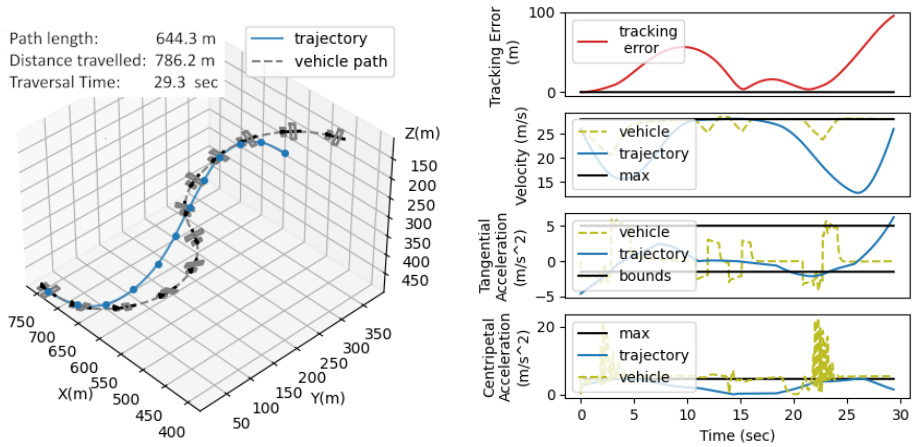



Figure 6.18: Fixed-wing aircraft trajectory with centripetal acceleration constraint

Figure 6.18 shows the fixed-wing model tracking a trajectory constrained by centripetal acceleration. In this scenario, the trajectory exceeds the

tangential acceleration limitations of the vehicle, so much that the aircraft cannot slow down fast enough to make the two turns. This renders the trajectory infeasible.

Figure 6.19 shows a trajectory constrained by both tangential and centripetal acceleration. The fixed-wing aircraft is able to track this trajectory with a maximum tracking error of less than 0.25 meters. It also traverses the path in less than 32 seconds. This trajectory is both feasible and efficient.

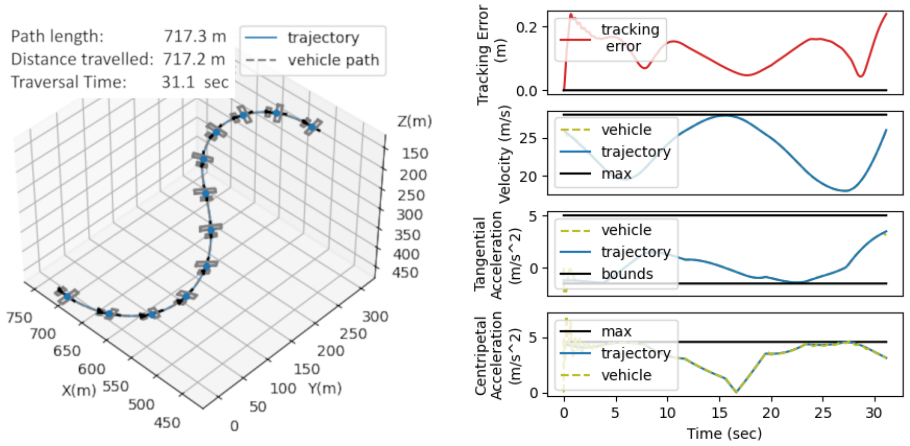



Figure 6.19: Fixed-wing aircraft trajectory with centripetal- and tangential-acceleration constraints

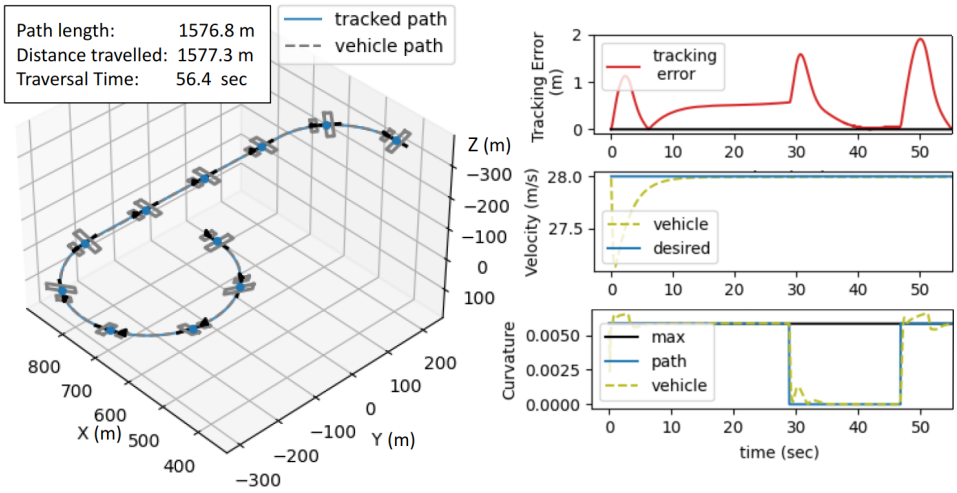



Figure 6.20: Fixed-wing aircraft minimum-radius Dubins path

For comparison, we also show a fixed-wing aircraft that follows a traditional Dubins path with minimum radius, or maximum curvature, in Figure 6.20. The maximum curvature is equal to $\frac{a_{R,\max}}{V_{\max}^2}$. This path is feasible, but even at its maximum speed, the aircraft still takes a total of 56.4 seconds to traverse. This shows that the optimal trajectory for a fixed-wing aircraft should allow the aircraft to decelerate and make tighter turns. We can

conclude that the constraints best suited for the trajectory of a fixed-wing aircraft are tangential and acceleration constraints.

### 6.5.5 Discussion

From these results, we infer that the optimal trajectory for a vehicle is best created by constraining the turn limitation that is independent of velocity for that vehicle. If turning constraints are excluded, the trajectory will likely be infeasible, especially if the trajectory is demanding with respect to the dynamic capabilities of the vehicle. On the other hand, using an inappropriate turning constraint will cause the trajectory to be long and inefficient, or will not adequately bound the limitations of the vehicle for the entire velocity space.

## 6.6 Summary

In this chapter, we introduced curvature, angular-rate, centripetal-acceleration, and tangential-acceleration constraints on a B-spline trajectory. We predicted that including these constraints in the trajectories of vehicles with turning constraints would produce optimal trajectories. The vehicles we used to demonstrate this were the bicycle model, the unicycle model, and the boat model, which have turning limitations similar to a car, a two-wheel differential drive robot, and a tail-controlled missile, respectively. We also demonstrated this using a fixed-wing aircraft model. We described the kinematics and control law for each model and then performed comparative experiments with and without these turning constraints for each vehicle. The conclusion is that the most efficient and feasible trajectories are generated when constructed with a vehicle's velocity-independent turning constraint.

# 7 Conclusions and Recommendations

The motivation for this research was to provide an onboard rapid path planning solution for UAVs used in today's industry. Most of the existing path-planning solutions are either too slow, specific to only certain vehicle systems, or limit the true agility of the vehicle being used. To solve this problem, we developed a B-spline optimization-based path planning solution. B-spline paths are advantageous because they can guarantee kinematic feasibility, are continuous, and can be scaled to any size without increasing computational complexity. They can also be applied to a variety of differentially flat vehicle systems. One shortcoming of current B-spline solutions is that they do not constrain the curvature of a path while minimizing distance or time in a computationally efficient way. There is also no method for constraining the angular rate or centripetal acceleration of a B-spline trajectory. Our solution is a minimum distance and obstacle-avoiding B-spline path planner with curvature and slope constraints that do not rely on discrete sampling of the B-spline path. For time dependent missions, we also developed a minimum-time B-spline trajectory generator with angular rate, centripetal acceleration, and tangential acceleration constraints. Our solution is more computationally efficient because it does not rely on discrete sampling of the path for its constraints.

## 7.1 Key Results

Discoveries from this research include the following key results:

- Bounding the curvature of a B-spline by ensuring that for each interval that the ratio of the maximum acceleration to the minimum velocity, or the ratio of the maximum-acceleration cross term to the minimum velocity, is less than the maximum curvature bound will guarantee curvature feasibility in a non-conservative fashion. It is also an order of magnitude faster than traditional curvature constraint methods. The maximum and minimum are found either through analytical root solvers, or by leveraging the convex hull property of a Minvo curve, each of which can be solved in linear complexity time.
- Optimized B-spline paths with curvature and slope constraints are easier for a fixed-wing aircraft to track than traditional minimum-radius Dubins paths with constant climb rates. This is because B-spline paths have continuous curvature, whereas Dubins paths have discontinuities

when transitioning from straight lines to turns. B-spline paths can also produce shorter paths around obstacles at a faster processing rate than Dubins-based RRT* methods.

- Optimized B-spline trajectories with the appropriate turning constraint produce feasible and shorter time trajectories than B-spline trajectories with derivative constraints alone. The appropriate turning constraint for a vehicle is one that does not depend on the vehicle's velocity. For example, B-spline trajectories with constraints on centripetal acceleration and tangential acceleration are faster and shorter for a fixed-wing aircraft to track than traditional minimum radius Dubins paths at maximum velocity.

- Direct obstacle avoidance constraints can be implemented into a B-spline path optimization without the need for pre-planning steps. This is done by restricting the distance between the closest point on an obstacle to the closest Minvo control point of the interval of interest. However, the paths generated are local optimal solutions, not global optimal solutions.

- A sequence of box-shaped safe flight corridor constraints can be formulated as convex linear constraints for B-spline path optimization. They can also be rotated to any orientation while maintaining their linearity.

## 7.2 Future Work

We recommend several improvements to the B-spline optimization-based path planners developed in this thesis. These improvements could significantly reduce computation time and aid in creating shorter paths or faster trajectories. These recommendations fall into three categories: Convexifying the optimization constraints, relaxing constraints using slack variables, and using non-uniform B-splines for path optimization.

### 7.2.1 Convexification of B-Spline Path Optimization Constraints

Some of the constraints developed in this research were not convex. Non-convex constraints degrade computational performance. To resolve this, we recommend convexifying the B-spline path optimization that was developed in this thesis through sequential convex programming. This will involve convexifying the constraints through linearization and solving convex subproblems [68]. The B-spline optimization can be setup as a convex quadratically constrained quadratic program (QCQP). For trajectory generation, the first step will be to linearize the derivative constraints. This can be done by linearizing the velocity and acceleration constraints with respect to the scale factor. The second step will be to linearize the turning constraints. This can be done by finding the minimum velocity of each interval at the beginning of each convexification step. With the MDM method or the analytical roots method, finding the minimum velocity has a linear convergence rate [69]. The cross term magnitude can then be linearized and constrained in reference to the minimum-velocity magnitude. The slope and tangential

acceleration constraints can be linearized in the same way. However, for the tangential acceleration constraint, the dot product term should be linearized.

The last step will be to linearize the direct obstacle avoidance constraint. This can be done by computing the obstacle vector $\mathbf{V}_{j,i}$ in each convexification step. This will allow the rotation matrices $R_{j,i}$ to be calculated before optimization of the subproblem. In addition, any convex shape could be used as an obstacle in these constraints.

### 7.2.2 Constraint Relaxation

One of the issues that we ran into when using the B-spline path and trajectory generators we developed was that the generator would sometimes output nonsensical paths. An example can be seen in the top right image of Figure 3.14 or the top middle image of Figure 3.16. This occurs either because the combination of constraints is impossible to appease, or the constraint formulation forces the optimization variables to converge on a suboptimal solution. An example of the first scenario is that if the initial velocity of a trajectory is constrained to be 30 m/s, the maximum velocity constraint is 30 m/s, and the initial acceleration is constrained to be positive, then it is impossible to achieve a feasible trajectory under those constraints.

The solution we propose to solve this issue is to relax some of the constraints by using slack variables. This would allow the optimizer more freedom to fully explore the constraint space, or produce a solution that is only slightly infeasible. An example of this technique is detailed in Frison's QCQP programming framework detailed in [70].

### 7.2.3 Non-uniform B-Splines for Path Planning

A shortcoming of uniform B-spline paths is that they use the same scale factor for each interval of the spline. This means that the length of time for each interval is the same. This causes scaling issues for paths with significant turns or dramatic derivative differences along the path. For example, a long trajectory with a small number of B-spline intervals is likely to be conservative in relation to its acceleration or velocity capabilities. This is because there are only so many control variables to work with on such a long path. This also causes issues for safe flight corridors constrained paths if too many intervals are allotted to a corridor in comparison to the number of intervals in another corridor. Each of these scenarios produces suboptimal paths. The solution we propose to resolve this is to use non-uniform B-splines for path optimization.

Nonuniform B-splines can have a unique scale factor for each interval of the spline and could therefore resolve the scaling issue. However, they also have unique and dynamic evaluation matrices. This will create a challenge in designing a rapid convex optimization problem. Nonetheless, this can be resolved through sequential convex programming. At each convexification step, the evaluation matrices can be computed to create a convex subproblem. For this to be computationally efficient, the algorithm for computing the evaluation matrices must have a linear convergence rate.

# References


[1] A. A. Wirabudi, L. Hafiza, and N. R. Fachrurrozi, "Design autonomous drone control for delivery package using prim algorithm and waypoint method," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 1183–1188. DOI: 10.1109/ictc55196.2022.9952874.

[2] P. Wojciechowski and K. Wojtowicz, "Detection of critical infrastructure elements damage with drones," in *2023 IEEE 10th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 2023, pp. 341–345. DOI: 10.1109/metroaerospace57412.2023.10190055.

[3] V. Sowmya, A. A. S. Janani, S. M. Hussain, A. C. Aashica, and S. Arvindh, "Creating a resilient solution: Innovating an emergency response drone for natural disasters," in *2024 10th International Conference on Communication and Signal Processing (ICCSP)*, 2024, pp. 344–348. DOI: 10.1109/iccsp60870.2024.10543657.

[4] S. Richhariya, K. Wanaskar, S. Shrivastava, and J. Gao, "Surveillance drone cloud and intelligence service," in *2023 11th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2023, pp. 1–10. DOI: 10.1109/mobilecloud58788.2023.00007.

[5] Z. Zaludin, C. Chia, and E. Abdullah, "Hover to forward flight transition – longitudinal motion feasibility study of a converted long endurance fixed-wing VTOL UAV," in *2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2021, pp. 50–54. DOI: 10.1109/iccsce52189.2021.9530912.

[6] S. Shankland. "Zipline's new drone brings aerial delivery closer for millions more of us," CNET. (2024), [Online]. Available: https://www.cnet.com/tech/computing/new-zipline-drone-brings-aerial-delivery-closer-for-millions-more-of-us/.

[7] "Elios 3: Streamline your mining operations with visual and metric data," Innovation News Network. (2024), [Online]. Available: https://www.innovationnewsnetwork.com/elios-3-streamline-your-mining-operations-with-visual-and-metric-data/42919/.

[8] "Bat unmanned aerial vehicle (UAV)," Airforce Technology. (2024), [Online]. Available: https://www.airforce-technology.com/projects/northropgrummanbat/?cf-view.

[9] "A better C-UAS option? capture enemy drones in a net," Breaking Defense. (2024), [Online]. Available: https://breakingdefense.com/2023/10/a-better-c-uas-option-capture-enemy-drones-in-a-net/.

[10] T. McLain, D. Wheeler, P. W. Nyholm, *et al.*, "Relative navigation in GPS degraded environments," *Encyclopedia of Aerospace Engineering: UAS*, 2016.

[11] J. L. Sanchez-Lopez, J. Pestana, and P. Campoy, "A robust real-time path planner for the collision-free navigation of multirotor aerial robots in dynamic environments," pp. 316–325, 2017. DOI: 10.1109/icuas.2017.7991354.

[12] M. Neunert, C. de Crousaz, F. Furrer, *et al.*, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," pp. 1398–1404, 2016. DOI: 10.1109/icra.2016.7487274.

[13] M. Dharmadhikari, T. Dang, L. Solanka, *et al.*, "Motion primitives-based path planning for fast and agile exploration using aerial robots," pp. 179–185, 2020. DOI: 10.1109/icra40945.2020.9196964.

[14] F. Belkhouche, "Reactive optimal UAV motion planning in a dynamic world," *Robotics and Autonomous Systems*, vol. 96, pp. 114–123, 2017, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2017.07.006.

[15] J. Tordesillas and J. P. How, "MADER: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2022. DOI: 10.1109/tro.2021.3080235.

[16] Z. He, Y. Fang, X. Liang, H. Lin, Y. Xiao, and X. Zhang, "A time-optimal trajectory generation algorithm for quadrotors with various states constraints," in *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2018, pp. 880–885. DOI: 10.1109/icarm.2018.8610702.

[17] H. Deddi, H. Everett, and S. Lazard, "Interpolation with Curvature Constraints," 2000. [Online]. Available: https://inria.hal.science/inria-00072572.

[18] R. Cimurs, J. Hwang, and I. H. Suh, "Bezier curve-based smoothing for path planner with curvature constraint," *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 241–248, 2017.

[19] H. Kano and H. Fujioka, "B-spline trajectory planning with curvature constraint," *2018 Annual American Control Conference (ACC)*, pp. 1963–1968, 2018.

[20] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli, "A novel formalisation of the Markov-Dubins problem," pp. 1987–1992, 2020. DOI: 10.23919/ecc51009.2020.9143597.

[21] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," pp. 3681–3688, 2017. DOI: 10.1109/iros.2017.8206214.

[22] G. Xu, T. Long, and S. Wang, "Distributed trajectory generation for multi-quadrotors using receding horizon control and sequential convex programming," pp. 837–841, 2019. DOI: 10.1109/icus48101.2019.8996027.

[23] J. A. S. Jayasinghe and A. M. B. G. D. A. Athauda, "Smooth trajectory generation algorithm for an unmanned aerial vehicle (UAV) under dynamic constraints: Using a quadratic Bezier curve for collision avoidance," pp. 1–6, 2016. DOI: 10.1109/mies.2016.7780258.

[24] L. Wang and Y. Guo, "Speed adaptive robot trajectory generation based on derivative property of B-spline curve," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 1905–1911, 2023. DOI: 10.1109/lra.2023.3241812.

[25] L. Tang, H. Wang, P. Li, and Y. Wang, "Real-time trajectory generation for quadrotors using B-spline based non-uniform kinodynamic search," pp. 1133–1138, 2019. DOI: 10.1109/robio49542.2019.8961485.

[26] F. Stoican, D. Popescu, and L. Ichim, "Some comments on the constrained trajectory generation for UAV systems," pp. 470–475, 2019. DOI: 10.1109/med.2019.8798522.

[27] "B-spline curves: Important properties," Michigan Technological University. (2021), [Online]. Available: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-prop.html.

[28] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline Techniques* (Mathematics and Visualization). Springer Berlin, Heidelberg, 2013, ISBN: 978-3-642-07842-2.

[29] K. Qin, "General matrix representations for B-splines," pp. 37–43, 1998. DOI: 10.1109/pccga.1998.731996.

[30] J. Tordesillas and J. P. How, "Minvo basis: Finding simplexes with minimum volume enclosing polynomial curves," *Computer-Aided Design*, vol. 151, p. 103 341, 2022, ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2022.103341.

[31] N. M. Patrikalakis, T. Maekawa, and W. Cho. "Bernstien polynomials." (), [Online]. Available: https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node9.html (visited on 2023).

[32] L. Romani and M. Sabin, "The conversion matrix between uniform B-spline and bézier representations," *Computer Aided Geometric Design*, vol. 21, no. 6, pp. 549–560, 2004, ISSN: 0167-8396. DOI: https://doi.org/10.1016/j.cagd.2004.04.002.

[33] H. Kano and H. Fujioka, "Velocity and acceleration constrained trajectory planning by smoothing splines," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1167–1172. DOI: 10.1109/isie.2017.8001410.

[34] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525. DOI: 10.1109/icra.2011.5980409.

[35] H. Sikha and G. Debasish, "Optimal path planning for an aerial vehicle in 3D space," *49th IEEE Conference on Decision and Control (CDC)*, pp. 4902–4907, 2010.

[36] X. Wang, P. Jiang, D. Li, and T. Sun, "Curvature continuous and bounded path planning for fixed-wing UAVs," *Sensors*, vol. 17, p. 2155, 2017.

[37] D. Walton and D. Meek, "Curvature bounds for planar B-spline curve segments," *Computer-Aided Design*, vol. 20, no. 3, pp. 146–150, 1988.

[38] D. Walton and D. Meek, "Curvature extrema of planar parametric polynomial cubic curves," *Journal of Computational and Applied Mathematics*, vol. 134, no. 1, pp. 69–83, 2001.

[39] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010. DOI: 10.1109/tro.2010.2042990.

[40] P. Dawkins. "Section 12.10 : Curvature." (2022), [Online]. Available: https://tutorial.math.lamar.edu/classes/calciii/curvature.aspx.

[41] E. Jones, T. Oliphant, and P. Peterson. "Scipy optimize root scalar." (2022), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/optimize.root_scalar-brentq.html.

[42] E. Jones, T. Oliphant, and P. Peterson. "SLSQP." (2013), [Online]. Available: https://mdolab-pyoptsparse.readthedocs-hosted.com/en/v2.1.0/optimizers/pyslsqp.html.

[43] E. W. Weisstein. "Cubic formula from Mathworld–a Wolfram web resource." (), [Online]. Available: https://mathworld.wolfram.com/CubicFormula.html (visited on 2022).

[44] D. Kaown and J. Liu, "A fast geometric algorithm for finding the minimum distance between two convex hulls," *Proceedings of the IEEE Conference on Decision and Control*, pp. 1189–1194, Jan. 2010.

[45] L. Vladimirovich, "Analysis of the article on increasing the speed of the MDM method," 2020.

[46] Á. Barbero, J. Lázaro, and J. Dorronsoro, "An accelerated MDM algorithm for SVM training.," pp. 421–426, Jan. 2008.

[47] J. Park and H. J. Kim, "Online trajectory planning for multiple quadrotors in dynamic environments using relative safe flight corridor," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 659–666, 2021. DOI: 10.1109/lra.2020.3047786.

[48] J. van den Berg. "Calculate rotation matrix to align vector A to vector B in 3D?" Mathematics Stack Exchange. (2023), [Online]. Available: https://math.stackexchange.com/q/476311.

[49] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012, ISBN: 9780691149219.

[50] "Maximum rate turns," Aviation Security Service. (2024), [Online]. Available: https://www.aviation.govt.nz/licensing-and-certification/pilots/flight-training/flight-instructor-guide/maximum-rate-turns.

[51] "Climbing and descending," Aviation Security Service. (2024), [Online]. Available: https://www.aviation.govt.nz/licensing-and-certification/pilots/flight-training/flight-instructor-guide/climbing-and-descending/.

[52] I. Lugo-Cárdenas, G. Flores, S. Salazar, and R. Lozano, "Dubins path generation for a fixed wing UAV," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 339–346. DOI: 10.1109/icuas.2014.6842272.

[53] H. Shi, Y. Lu, Z. Hou, and S. Huang, "3D Dubins net-recovery path planning for fixed wing UAV," pp. 604–610, 2018. DOI: 10.1109/ccdc.2018.8407203.

[54] X. Song and S. Hu, "2D path planning with Dubins-path-based A* algorithm for a fixed-wing UAV," in *2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE)*, 2017, pp. 69–73. DOI: 10.1109/ccsse.2017.8087897.

[55] D. Lee, H. Song, and D. H. Shim, "Optimal path planning based on spline-RRT* for fixed-wing UAV's operating in three-dimensional environments," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 2014, pp. 835–839. DOI: 10.1109/iccas.2014.6987895.

[56] D. M. Putri and T. Agustinah, "Path smoothing using Bézier curve with maneuver constraint of fixed-wing UAV," in *2022 IEEE Delhi Section Conference (DELCON)*, 2022, pp. 1–5. DOI: 10.1109/delcon54057.2022.9753042.

[57] Z. Fang Yao, "Piecewise-potential-field-based path planning method for fixed-wing UAV formation," 2023.

[58] S. R. Bassolillo, G. Raspaolo, L. Blasi, E. D'Amato, and I. Notaro, "Path planning for fixed-wing unmanned aerial vehicles: An integrated approach with theta* and clothoids," *Drones*, vol. 8, no. 2, 2024, ISSN: 2504-446X. DOI: 10.3390/drones8020062.

[59] Y. Suh, J. Kang, and D. Lee, "A fast and safe motion planning algorithm in cluttered environment using maximally occupying convex space," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 173–178. DOI: 10.23919/iccas50221.2020.9268421.

[60] A. Behery. "Python vs C++ time complexity analysis." (2024), [Online]. Available: https://www.freecodecamp.org/news/python-vs-c-plus-plus-time-complexity-analysis.

[61] N. Tamimi. "How fast is C++ compared to Python?" (2024), [Online]. Available: https://towardsdatascience.com/how-fast-is-c-compared-to-python-978f18f474c7.

[62] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açikmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4862–4868. DOI: 10.1109/iros.2017.8206363.

[63] Z. Shen, G. Zhou, H. Huang, C. Huang, Y. Wang, and F.-Y. Wang, "Convex optimization-based trajectory planning for quadrotors landing on aerial vehicle carriers," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2023. DOI: 10.1109/tiv.2023.3327263.

[64] A. Iskander, O. Elkassed, and A. El-Badawy, "Minimum snap trajectory tracking for a quadrotor UAV using nonlinear model predictive control," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 344–349. DOI: 10.1109/niles50944.2020.9257897.

[65] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818. DOI: 10.1109/ivs.2017.7995816.

[66] E. Maulana, M. A. Muslim, and A. Zainuri, "Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot," in *2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS)*, 2014, pp. 93–98. DOI: 10.1109/eeccis.2014.7003726.

[67] T. McLain and R. Beard, "Nonlinear optimal control design of a missile autopilot," AIAA, 1998. [Online]. Available: http://hdl.lib.byu.edu/1877/5005.

[68] D. Malyuta, T. P. Reynolds, M. Szmuk, *et al.*, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022. DOI: 10.1109/mcs.2022.3187542.

[69] J. Lopez and J. R. Dorronsoro, "Linear convergence rate for the MDM algorithm for the nearest point problem," *Pattern Recognition*, vol. 48, no. 4, pp. 1510–1522, 2015, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.10.015.

[70] G. Frison, J. Frey, F. Messerer, A. Zanelli, and M. Diehl, "Introducing the quadratically-constrained quadratic programming framework in hpipm," 2021. arXiv: 2112.11872 [math.OC]. [Online]. Available: https://arxiv.org/abs/2112.11872.

# *Appendices*

# A Open Uniform B-splines

## A.1 The Evaluation Matrices of Bezier Curves

The $M_B$ matrices for a first- through fifth-order Bezier curve are

$$\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{if} \quad k = 1 \tag{A.1}$$

$$\frac{1}{2}\begin{bmatrix} 2 & -4 & 2 \\ -4 & 4 & 0 \\ 2 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 2 \tag{A.2}$$

$$\frac{1}{12}\begin{bmatrix} -12 & 36 & -36 & 12 \\ 36 & -72 & 36 & 0 \\ -36 & 36 & 0 & 0 \\ 12 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 3 \tag{A.3}$$

$$\begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 4 \tag{A.4}$$

$$\begin{bmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{if} \quad k = 5 \tag{A.5}$$

## A.2 The Evaluation Matrices of Minvo Curves

The $M_V$ matrices for a first- through fifth-order Minvo curve are

$$\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{if} \quad k = 1 \tag{A.6}$$

$$\begin{bmatrix} 1.49999998 & -2.3660254 & 0.93301271 \\ -2.99999999 & 2.99999999 & 0. \\ 1.49999998 & -0.63397456 & 0.06698729 \end{bmatrix} \quad \text{if} \quad k = 2 \tag{A.7}$$

$$\begin{bmatrix} -3.44163122 & 6.98954850 & -4.46228883 & 0.914371512 \\ 6.67925910 & -11.8459903 & 5.25235983 & 0 \\ -6.67925910 & 8.19178696 & -1.59815645 & 0.0856285552 \\ 3.44163122 & -3.33534516 & 0.808085496 & 0 \end{bmatrix} \quad \text{if} \quad k = 3 \tag{A.8}$$

$$\begin{bmatrix} 8.40833771 & -21.4229882 & 19.1445618 \\ -7.00530881 & 0.899093282 & \\ -17.7354166 & 41.9570300 & -32.4916003 \\ 8.26998651 & 0 & \\ 18.6541578 & -37.3083156e & 21.0544248 \\ -2.40026695 & 0.0772117317 & \\ -17.7354166 & 28.9846365 & -13.0330099 \\ 1.78379042 & 0 & \\ 8.40833771 & -12.2103627 & 5.32562357 \\ -0.648201174 & 0.0236958438 & \end{bmatrix} \quad \text{if} \quad k = 4 \tag{A.9}$$

$$\begin{bmatrix} -23.6539675 & 71.5650800 & -81.3537022 \\ 42.7412549 & -10.1896550 & 0.890989788 \\ 48.0994098 & -141.357620 & 151.542314 \\ -70.0692925 & 11.8183479 & 0 \\ -56.0021106 & 148.684210 & -135.150794 \\ 45.8796401e+01 & -3.48679764 & 0.0758512541 \\ 56.0021106 & -131.326343 & 100.435058 \\ -27.4885841e+01 & 2.45360952 & 0 \\ -48.0994098 & 99.1394293 & -67.1059330 \\ 17.4060284 & -1.37327413 & 0.0331591019 \\ 23.6539675 & -46.7047575 & 31.6330573 \\ -8.46904679 & 0.777769353 & 0 \end{bmatrix} \quad \text{if} \quad k = 5 \tag{A.10}$$

## A.3 Conversion from B-spline to Bezier Curve Control Points

The values of $F_\beta$ for curves of order one through five are

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{if} \quad k = 1 \tag{A.11}$$

$$\frac{1}{2}\begin{bmatrix}1 & 0 & 0\\ 1 & 2 & 1\\ 0 & 0 & 1\end{bmatrix} \quad \text{if} \quad k = 2 \tag{A.12}$$

$$\frac{1}{6}\begin{bmatrix}1 & 0 & 0 & 0\\ 4 & 4 & 2 & 1\\ 1 & 2 & 4 & 4\\ 0 & 0 & 0 & 1\end{bmatrix} \quad \text{if} \quad k = 3 \tag{A.13}$$

$$\frac{1}{24}\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 11 & 8 & 4 & 2 & 1\\ 11 & 14 & 16 & 14 & 11\\ 1 & 2 & 4 & 8 & 11\\ 0 & 0 & 0 & 0 & 1\end{bmatrix} \quad \text{if} \quad k = 4 \tag{A.14}$$

$$\frac{1}{120}\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 26 & 16 & 8 & 4 & 2 & 1\\ 66 & 66 & 60 & 48 & 36 & 26\\ 26 & 36 & 48 & 60 & 66 & 66\\ 1 & 2 & 4 & 8 & 16 & 26\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix} \quad \text{if} \quad k = 5 \tag{A.15}$$

## A.4 Conversion from B-spline to Minvo Curve Control Points

The values of $F_v$ for curves of order one through five are

$$\begin{bmatrix}-1 & 1\\ 1 & 0\end{bmatrix} \quad \text{if} \quad k = 1 \tag{A.16}$$

$$\begin{bmatrix}0.53867513 & 0.08333333 & -0.03867513\\ 0.5 & 0.83333333 & 0.5\\ -0.03867513 & 0.08333333 & 0.53867513\end{bmatrix} \quad \text{if} \quad k = 2 \tag{A.17}$$

$$\begin{bmatrix}0.1837219 & 0.05700954 & -0.01545516 & -0.00533879\\ 0.70176523 & 0.66657382 & 0.2918718 & 0.11985167\\ 0.11985167 & 0.2918718 & 0.66657382 & 0.70176523\\ -0.00533879 & -0.01545516 & 0.05700954 & 0.1837219\end{bmatrix} \quad \text{if} \quad k = 3 \tag{A.18}$$

$$\begin{bmatrix} 0.04654488 & 0.01902005 & -0.0020279 & -0.00193449 \\ -0.00105254 & & & \\ 0.49370783 & 0.40331371 & 0.18162853 & 0.0395948 \\ 0.01773369 & & & \\ 0.44306614 & 0.54000593 & 0.64079874 & 0.54000593 \\ 0.44306614 & & & \\ 0.01773369 & 0.0395948 & 0.18162853 & 0.40331371 \\ 0.49370783 & & & \\ -0.00105254 & -0.00193449 & -0.0020279 & 0.01902005 \\ 0.04654488 & & & \end{bmatrix} \quad \text{if} \quad k = 4 \tag{A.19}$$

$$\begin{bmatrix} 9.36633383^{-3} & 4.59221748^{-3} & -2.41070259^{-5} & -3.64703996^{-4} \\ -3.05969926^{-4} & -1.39856086^{-4} & & \\ 2.35803353^{-1} & 1.88464610^{-1} & 8.68737506^{-2} & 2.46973953^{-2} \\ -6.38854070^{-4} & 2.36477513^{-4} & & \\ 5.64969670^{-1} & 5.74893473^{-1} & 5.12739907^{-1} & 3.76077758^{-1} \\ 2.32994524^{-1} & 1.89764022^{-1} & & \\ 1.89764022^{-1} & 2.32994524^{-1} & 3.76077758^{-1} & 5.12739907^{-1} \\ 5.74893473^{-1} & 5.64969670^{-1} & & \\ 2.36477513^{-4} & -6.38854070^{-4} & 2.46973953^{-2} & 8.68737506^{-2} \\ 1.88464610^{-1} & 2.35803353^{-1} & & \\ -1.39856086^{-4} & -3.05969926^{-4} & -3.64703996^{-4} & -2.41070259^{-5} \\ 4.59221748^{-3} & 9.36633383^{-3} & & \end{bmatrix} \quad \text{if} \quad k = 5 \tag{A.20}$$

# B Curvature Bounds and Extrema

## B.1 Third-Order B-spline Cross-Term Derivative Coefficients

The coefficients for the derivative of the cross-term of a third-order B-spline are determined by solving for $c_0$, $c_1$, $c_2$, and $c_3$ in terms of control points $P_0$, $P_1$, $P_2$, and $P_3$ from the cross-term derivative equation expressed as

$$\begin{aligned} c_3\tau^3 + c_2\tau^2 + c_1\tau + c_0 &= \left(b(t)' \times b(t)''\right) \cdot \left(b(t)' \times b(t)'''\right) \\ &= \left(\mathbf{P}_i ML' \times \mathbf{P}_i ML''\right) \cdot \left(\mathbf{P}_i ML' \times \mathbf{P}_i ML'''\right) \end{aligned} \tag{B.1}$$

where $\mathbf{P}_i$ are the set of control points $[P_0, P_1, P_2, P_3]$. Using a symbolic solver, the coefficients are evaluated to be

$$\begin{aligned} c_3 =& \big((P_{0y} - 2P_{1y} + P_{2y})(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - \\ & (P_{0x} - 2P_{1x} + P_{2x})(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\big)\big((P_{0x}P_{1y})/2 - \\ & (P_{1x}P_{0y})/2 - P_{0x}P_{2y} + P_{2x}P_{0y} + (P_{0x}P_{3y})/2 + (3P_{1x}P_{2y})/2 - \\ & (3P_{2x}P_{1y})/2 - (P_{3x}P_{0y})/2 - P_{1x}P_{3y} + P_{3x}P_{1y} + (P_{2x}P_{3y})/2 - \\ & (P_{3x}P_{2y})/2\big) + \big((P_{0z} - 2P_{1z} + P_{2z})(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - \\ & (P_{0x} - 2P_{1x} + P_{2x})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0x}P_{1z})/2 - \\ & (P_{1x}P_{0z})/2 - P_{0x}P_{2z} + P_{2x}P_{0z} + (P_{0x}P_{3z})/2 + (3P_{1x}P_{2z})/2 - \\ & (3P_{2x}P_{1z})/2 - (P_{3x}P_{0z})/2 - P_{1x}P_{3z} + P_{3x}P_{1z} + (P_{2x}P_{3z})/2 - \\ & (P_{3x}P_{2z})/2\big) + \big((P_{0z} - 2P_{1z} + P_{2z})(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) - \\ & (P_{0y} - 2P_{1y} + P_{2y})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0y}P_{1z})/2 - \\ & (P_{1y}P_{0z})/2 - P_{0y}P_{2z} + P_{2y}P_{0z} + (P_{0y}P_{3z})/2 + (3P_{1y}P_{2z})/2 - \\ & (3P_{2y}P_{1z})/2 - (P_{3y}P_{0z})/2 - P_{1y}P_{3z} + P_{3y}P_{1z} + \\ & (P_{2y}P_{3z})/2 - (P_{3y}P_{2z})/2\big) \end{aligned} \tag{B.2}$$

$$
\begin{aligned}
c_2 = &-\big((P_{0y}/2 - P_{2y}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)\\
&(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\big)\big((P_{0x}P_{1y})/2 - (P_{1x}P_{0y})/2 -\\
&P_{0x}P_{2y} + P_{2x}P_{0y} + (P_{0x}P_{3y})/2 + (3P_{1x}P_{2y})/2 - (3P_{2x}P_{1y})/2 -\\
&(P_{3x}P_{0y})/2 - P_{1x}P_{3y} + P_{3x}P_{1y} + (P_{2x}P_{3y})/2 - (P_{3x}P_{2y})/2\big) -\\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)\\
&(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0x}P_{1z})/2 - (P_{1x}P_{0z})/2 -\\
&P_{0x}P_{2z} + P_{2x}P_{0z} + (P_{0x}P_{3z})/2 + (3P_{1x}P_{2z})/2 - (3P_{2x}P_{1z})/2 -\\
&(P_{3x}P_{0z})/2 - P_{1x}P_{3z} + P_{3x}P_{1z} + (P_{2x}P_{3z})/2 - (P_{3x}P_{2z})/2\big) -\\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) - (P_{0y}/2 - P_{2y}/2)\\
&(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0y}P_{1z})/2 - (P_{1y}P_{0z})/2 -\\
&P_{0y}P_{2z} + P_{2y}P_{0z} + (P_{0y}P_{3z})/2 + (3P_{1y}P_{2z})/2 - (3P_{2y}P_{1z})/2 -\\
&(P_{3y}P_{0z})/2 - P_{1y}P_{3z} + P_{3y}P_{1z} + (P_{2y}P_{3z})/2 - (P_{3y}P_{2z})/2\big) -\\
&\big((P_{0y}/2 - P_{2y}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)\\
&(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\big)\big((P_{0y} - 2P_{1y} + P_{2y})(P_{0x} - 3P_{1x} +\\
&3P_{2x} - P_{3x}) - (P_{0x} - 2P_{1x} + P_{2x})(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\big) -\\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)\\
&(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0z} - 2P_{1z} + P_{2z})(P_{0x} - 3P_{1x} +\\
&3P_{2x} - P_{3x}) - (P_{0x} - 2P_{1x} + P_{2x})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big) -\\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) - (P_{0y}/2 - P_{2y}/2)\\
&(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)\big((P_{0z} - 2P_{1z} + P_{2z})(P_{0y} - 3P_{1y} +\\
&3P_{2y} - P_{3y}) - (P_{0y} - 2P_{1y} + P_{2y})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)
\end{aligned}
\tag{B.3}
$$

$$
\begin{aligned}
c_1 = &\big((P_{0y}/2 - P_{2y}/2)(P_{0x} - 2P_{1x} + P_{2x}) - (P_{0x}/2 - P_{2x}/2)\\
&(P_{0y} - 2P_{1y} + P_{2y})\big)\big((P_{0y} - 2P_{1y} + P_{2y})(P_{0x} - 3P_{1x} +\\
&3P_{2x} - P_{3x}) - (P_{0x} - 2P_{1x} + P_{2x})(P_{0y} - 3P_{1y} + 3P_{2y} -\\
&P_{3y})\big) + \big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 2P_{1x} + P_{2x}) - (P_{0x}/2 -\\
&P_{2x}/2)(P_{0z} - 2P_{1z} + P_{2z})\big)\big((P_{0z} - 2P_{1z} + P_{2z})(P_{0x} - 3P_{1x} +\\
&3P_{2x} - P_{3x}) - (P_{0x} - 2P_{1x} + P_{2x})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big) +\\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0y} - 2P_{1y} + P_{2y}) - (P_{0y}/2 - P_{2y}/2)(P_{0z} -\\
&2P_{1z} + P_{2z})\big)\big((P_{0z} - 2P_{1z} + P_{2z})(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) -\\
&(P_{0y} - 2P_{1y} + P_{2y})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big) + \big((P_{0y}/2 - P_{2y}/2)\\
&(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)(P_{0y} - 3P_{1y} + 3P_{2y} -\\
&P_{3y})\big)^2 + \big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 -\\
&P_{2x}/2)(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)^2 + \big((P_{0z}/2 - P_{2z}/2)(P_{0y} -\\
&3P_{1y} + 3P_{2y} - P_{3y}) - (P_{0y}/2 - P_{2y}/2)(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big)^2
\end{aligned}
\tag{B.4}
$$

$$
\begin{aligned}
c_0 = &- \big((P_{0y}/2 - P_{2y}/2)(P_{0x} - 2P_{1x} + P_{2x}) - (P_{0x}/2 - P_{2x}/2)(P_{0y} - 2P_{1y} + \\
&P_{2y})\big)\big((P_{0y}/2 - P_{2y}/2)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2) \\
&(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\big) - \big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 2P_{1x} + P_{2x}) - \\
&(P_{0x}/2 - P_{2x}/2)(P_{0z} - 2P_{1z} + P_{2z})\big)\big((P_{0z}/2 - P_{2z}/2)(P_{0x} - 3P_{1x} + \\
&3P_{2x} - P_{3x}) - (P_{0x}/2 - P_{2x}/2)(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\big) - \big((P_{0z}/2 - \\
&P_{2z}/2)(P_{0y} - 2P_{1y} + P_{2y}) - (P_{0y}/2 - P_{2y}/2)(P_{0z} - 2P_{1z} + P_{2z})\big) \\
&\big((P_{0z}/2 - P_{2z}/2)(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) - (P_{0y}/2 - P_{2y}/2)(P_{0z} - \\
&3P_{1z} + 3P_{2z} - P_{3z})\big)
\end{aligned}
\tag{B.5}
$$

## B.2 B-spline Cross-Term Coefficients

The cross term $A(t)$ must be represented in terms of the B-spline control points of $b(t)$. The following matrices are used to simplify the math for the derivations of this representation for three-dimensional splines. These matrices are

$$
Y_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad Y_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\tag{B.6}
$$

$$
Y_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} Y_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}
$$

If the spline is two dimensional, the matrices can be written as

$$
Y_1 = Y_4 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad Y_2 = Y_3 \begin{bmatrix} 0 & 1 \end{bmatrix}
\tag{B.7}
$$

In addition, the $\circ$ operator will be used, symbolizing a matrix coefficient-wise multiplication.

### B.2.1 Third-order Cross-Term B-spline Coefficients

For a third-order B-spline, the coefficients of the cross term

$$
A(t) = c_2 t^2 + c_1 t + c_0
\tag{B.8}
$$

are expressed as

$$
\begin{aligned}
c_2 = &Y_1 p_1 \circ Y_2 p_2 - Y_2 p_1 \circ Y_1 p_2 + \\
&Y_3 p_1 \circ 2Y_4 p_2 - Y_4 p_1 \circ 2Y_3 p_2
\end{aligned}
\tag{B.9}
$$

$$
c_1 = Y_1 p_3 \circ 2Y_2 p_2 - Y_2 p_3 \circ 2Y_1 p_2
\tag{B.10}
$$

$$
c_0 = Y_3 p_3 \circ Y_4 p_1 - Y_4 p_3 \circ Y_3 p_1
\tag{B.11}
$$

where

$$
\begin{aligned}
p_1 &= P_0 - 2P_1 + P_2 \\
p_2 &= (P_0 - 3P_1 + 3P_2 - P_3)/2 \\
p_3 &= (P_0 - P_2)/2
\end{aligned}
\tag{B.12}
$$

### B.2.2 Fourth-order Cross-Term B-spline Coefficients

For a fourth-order B-spline, the coefficients of the cross term

$$A(t) = c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0 \tag{B.13}$$

are expressed as

$$\begin{aligned} c_4 =& 2Y_1 p_1 \circ Y_2 p_2 - 2Y_2 p_1 \circ Y_1 p_2 + \\ & Y_3 p_1 \circ 3Y_4 p_2 - Y_4 p_1 \circ 3Y_3 p_2 \end{aligned} \tag{B.14}$$

$$\begin{aligned} c_3 =& Y_2 p_3 \circ Y_1 p_2 - Y_1 p_3 \circ Y_2 p_2 - \\ & Y_3 p_3 \circ 3Y_4 p_2 + Y_4 p_3 \circ 3Y_3 p_2 + \\ & Y_4 p_1 \circ 2Y_3 p_1 - Y_3 p_1 \circ 2Y_4 p_1 \end{aligned} \tag{B.15}$$

$$\begin{aligned} c_2 =& Y_2 p_4 \circ 3Y_1 p_2 - Y_1 p_4 \circ 3Y_2 p_2 + \\ & Y_4 p_3 \circ Y_3 p_1 - Y_3 p_3 \circ Y_4 p_1 - \\ & Y_4 p_3 \circ 2Y_3 p_1 + Y_3 p_3 \circ 2Y_4 p_1 \end{aligned} \tag{B.16}$$

$$c_1 = Y_1 p_4 \circ 2Y_2 p_1 - Y_2 p_4 \circ 2Y_1 p_1 \tag{B.17}$$

$$c_0 = -Y_4 p_4 \circ Y_3 p_3 + Y_4 p_3 \circ Y_3 p_4 \tag{B.18}$$

where

$$\begin{aligned} p_1 =& (P_0 - 3P_1 + 3P_2 - P_3)/2 \\ p_2 =& P_0/6 - 2P_1/3 + P_2 - 2P_3/3 + P_4/6 \\ p_3 =& (P_0 - P_1 - P_2 + P_3)/2 \\ p_4 =& (P_0/3 + P_1 - P_2 - P_3/3)/2 \end{aligned} \tag{B.19}$$

# C Trajectory Generation and Tracking

## C.1 Tangential Acceleration Derivative Coefficients

The polynomial coefficients of the derivative of $b(t)'' \cdot b(t)'$ of a third order B-spline are derived from the equation given by

$$\begin{aligned}\frac{d\,(b(t)'' \cdot b(t)')}{dt} &= (b(t)''' \cdot b(t)') + (b(t)'' \cdot b(t)'') \\ &= (\mathbf{P}_i ML''' \cdot \mathbf{P}_i ML') + (\mathbf{P}_i ML'' \cdot \mathbf{P}_i ML'') \\ &= c_2\tau^2 + c_1\tau + c_0\end{aligned} \tag{C.1}$$

Using a symbolic solver, the coefficients were calculated as

$$\begin{aligned}c_2 =&(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x})\left(\frac{P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}}{2}\right) \\ &+ (P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})\left(\frac{P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}}{2}\right) \\ &+ (P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\left(\frac{P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z}}{2}\right) \\ &+ (P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x})^2 + (P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y})^2 \\ &+ (P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})^2\end{aligned}$$

$$\begin{aligned}c_1 =& -(3P_{0x} - 6P_{1x} + 3P_{2x})(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) \\ &- (3P_{0y} - 6P_{1y} + 3P_{2y})(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) \\ &- (3P_{0z} - 6P_{1z} + 3P_{2z})(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\end{aligned} \tag{C.2}$$

$$\begin{aligned}c_0 =&(P_{0x} - 2P_{1x} + P_{2x})^2 + (P_{0y} - 2P_{1y} + P_{2y})^2 \\ &+ (P_{0z} - 2P_{1z} + P_{2z})^2 \\ &+ \left(\frac{P_{0x} - P_{2x}}{2}\right)(P_{0x} - 3P_{1x} + 3P_{2x} - P_{3x}) \\ &+ \left(\frac{P_{0y} - P_{2y}}{2}\right)(P_{0y} - 3P_{1y} + 3P_{2y} - P_{3y}) \\ &+ \left(\frac{P_{0z} - P_{2z}}{2}\right)(P_{0z} - 3P_{1z} + 3P_{2z} - P_{3z})\end{aligned}$$

## C.2 2D Vehicle Trajectory Tracking

Some common variables used for each vehicle controller are the position, velocity, and acceleration vectors written as

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}, \ \mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, \ \mathbf{a} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{C.3}$$

$V$ is the velocity magnitude, $a_t$ is the tangential acceleration magnitude, and $K$ is a proportional gain sub-scripted with its respective error metric. We also employ the use of subscripts $d$ for a desired value, subscript $c$ for a command value, subscript $e$ as an error term, subscript "fwd" for a feedforward term, and subscript "tol" for a tolerance value.

We also define some commonly used functions for each controller. The saturation function is

$$\text{sat}(x, [a, b]) = \min(\max(a, x) \ , \ b) \tag{C.4}$$

where $a$ is the lower limit and $b$ is the upper limit. To define the magnitude error between angles we use

$$\phi_{\text{error\_value}}(\phi, \phi_d) = \pi - \left| |\phi_d - \phi| - \pi \right| \tag{C.5}$$

and we use the equation

$$\phi_{\text{error\_sign}}(\phi, \phi_d) = \text{sgn}\left( \tan^{-1}\left( \frac{\sin(\phi_d - \phi)}{\cos(\phi_d - \phi)} \right) \right) \tag{C.6}$$

for the sign error between two angles. The complete error term is

$$\phi_{\text{error}}(\phi, \phi_d) = \phi_{\text{error\_sign}}(\phi, \phi_d) \ \phi_{\text{error\_value}}(\phi, \phi_d) \tag{C.7}$$

### C.2.1 Tangential Acceleration Control Law

The saturated commanded tangential acceleration is

$$a_{T,c,\text{sat}} = \text{sat}(a_{T,c}, [-a_{T,c,\max}, a_{T,c,\max}]) \tag{C.8}$$

The unsaturated command has a feedforward term when the vehicle is within a given tolerance of the trajectory and is expressed as

$$a_{T,c} = \begin{cases} a_{T,c} + a_{T,\text{fwd}} & \text{if } \ \mathbf{p}_e < \mathbf{p}_{\text{tol}} \ \text{ and } \ \chi_e < \chi_{\text{tol}} \\ a_{T,d} & \text{else} \end{cases} \tag{C.9}$$

where the error terms are

$$\mathbf{p}_e = \mathbf{p}_{\text{traj}} - \mathbf{p}, \quad \chi_e = \chi_{\text{traj}} - \chi \tag{C.10}$$

We compute the desired tangential acceleration by finding the magnitude of the desired acceleration projected onto the vehicle velocity unit vector.

$$a_{T,d} = \mathbf{a}_d \cdot \hat{\mathbf{V}} \tag{C.11}$$

This means that we accelerate only when a portion of the desired acceleration is aligned with the current velocity vector. The desired acceleration is computed with the proportional gain terms multiplied over the position and velocity error, and is given by

$$\mathbf{a}_d = \mathbf{p}_e K_p + \mathbf{V}_e K_v \tag{C.12}$$

where the velocity error is

$$\mathbf{V}_e = \mathbf{V}_{\text{traj}} - \mathbf{V} \tag{C.13}$$

The feedforward tangential acceleration is calculated by projecting the acceleration vector of the trajectory onto the velocity unit vector of the trajectory, and is expressed as

$$a_{T,\text{fwd}} = \mathbf{a}_{\text{traj}} \cdot \hat{\mathbf{V}}_{\text{traj}} \tag{C.14}$$

In other words the feedforward term is equal to the tangential acceleration of the trajectory at the point of interest. We calculate the velocity unit vector with the course angle of the trajectory as

$$\hat{\mathbf{V}}_{\text{traj}} = \begin{bmatrix} \cos\chi_{\text{traj}} & \sin\chi_{\text{traj}} \end{bmatrix}^\intercal \tag{C.15}$$

where the course of the trajectory at the point of interest is

$$\chi_{\text{traj}} = \tan^{-1}\left(\frac{\dot{y}_{\text{traj}}}{\dot{x}_{\text{traj}}}\right) \tag{C.16}$$

### C.2.2 Desired Course Angle Derivation

The course of the desired velocity vector is calculated as

$$\chi_d = \tan^{-1}\left(\frac{\dot{y}_d}{\dot{x}_d}\right) \tag{C.17}$$

The desired velocity vector is calculated by multiplying some proportional gain with the position error plus the velocity vector of the trajectory at the point of interest. It is expressed as

$$\mathbf{V_d} = \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} = \mathbf{p}_e K_p + \mathbf{V}_{\text{traj}} \tag{C.18}$$

The course rate of the trajectory is also a common term used for course rate control and is derived as

$$\dot{\chi}_{\text{traj}} = \frac{\dot{x}_{\text{traj}}\ddot{y}_{\text{traj}} - \dot{y}_{\text{traj}}\ddot{x}_{\text{traj}}}{\dot{x}_{\text{traj}}^2 + \dot{y}_{\text{traj}}^2} \tag{C.19}$$

This term is used for the course control feedforward terms of each vehicle.

### C.2.3 Bicycle Wheel Steering Rate Control

The wheel steering rate is saturated and evaluated as

$$\Delta_{w,c,\text{sat}} = \text{sat}(\Delta_{w_c}, [-\Delta_{w,\max}, \Delta_{w,\max}]) \tag{C.20}$$

where the unsaturated command is

$$\Delta_{w,c} = \begin{cases} \Delta_{w,d} + \Delta_{w,\text{fwd}} & \text{if } \mathbf{p}_\text{e} < \mathbf{p}_\text{tol} \text{ and } \chi_\text{e} < \chi_\text{tol} \\ \Delta_{w,d} & \text{else} \end{cases} \tag{C.21}$$

Notice that the feedforward term is applied only when the vehicle is within some tolerance of the trajectory position and heading.

The desired wheel turn rate is determined by some proportional gain multiplied by the error in the steering angle. The rate is expressed as

$$\Delta_{w,d} = \delta_e K_\delta \tag{C.22}$$

where the error is

$$\delta_{w,e} = \phi_\text{error}(\delta_w, \delta_{w,c}) \tag{C.23}$$

The steering angle of the bicycle is physically limited, and so the commanded steering angle is saturated and expressed as

$$\delta_{w,c} = \text{sat}(\delta_{w,d}, [-\delta_{w,\max}, \delta_{w,\max}]) \tag{C.24}$$

where the desired steering angle is calculated from Equation 6.16 and is

$$\delta_{w,d} = \tan^{-1}\left(\frac{L\tan(\beta_d)}{l_r}\right) \tag{C.25}$$

The desired offset angle of the velocity vector is equal to the difference between the current heading and the desired course angle. It is written as

$$\beta_d = \text{sat}(\phi_\text{error}(\psi, \chi_d), [-\pi/2, \pi/2]) \tag{C.26}$$

Note that because $\beta_d$ is used as a variable in the tangent function in Equation C.25, it is saturated to stay within positive and negative $\pi/2$.

The feedforward term provides the turning control needed when the vehicle is correctly positioned on the trajectory. The feedforward wheel turn rate is calculated from Equation 6.20 and the trajectory data. It is expressed as

$$\Delta_{w,\text{fwd}} = \frac{\dot{\beta}_\text{traj}\left(l_r^2 \sin^2\delta_\text{traj} + L^2\cos^2\delta_\text{traj}\right)}{L l_r} \tag{C.27}$$

where the velocity vector offset angular rate of the trajectory is

$$\dot{\beta}_\text{traj} = \dot{\chi}_\text{traj} - \dot{\psi} \tag{C.28}$$

and $\dot{\psi}$ is the current angular rate of the bicycle body. The wheel steering angle of the trajectory is

$$\delta_\text{traj} = \tan^{-1}\left(\frac{L\tan(\beta_\text{traj})}{l_r}\right) \tag{C.29}$$

where the velocity offset angle of the trajectory is

$$\beta_\text{traj} = \text{sat}(\phi_\text{error}(\psi, \chi_\text{traj}), [-\pi/2, \pi/2]) \tag{C.30}$$

$\psi$ is the current orientation of the vehicle.

### C.2.4 Unicycle Angular Rate Control

The saturated angular rate command is

$$\omega_{c\,\text{sat}} = \text{sat}(\omega_c, [-\omega_{\max}, \omega_{\max}]) \tag{C.31}$$

and the unsaturated command is

$$\omega_c = \begin{cases} \omega_d + \omega_{\text{fwd}} & \text{if } \mathbf{p}_e < \mathbf{p}_{\text{tol}} \text{ and } \psi_e < \psi_{\text{tol}} \\ \omega_d & \text{else} \end{cases} \tag{C.32}$$

Notice that the angular rate command has a feedforward term when the current position and heading are within some tolerance of the trajectory. The desired angular rate is calculated by multiplying a proportional gain with the angle error and is expressed as

$$\omega_d = \psi_e K_\psi \tag{C.33}$$

where the angle error is

$$\psi_e = \phi_{\text{error}}(\psi, \chi_d) \tag{C.34}$$

where $\chi_d$ is calculated in Section C.2.2. The feedforward angular rate is given by

$$\omega_{\text{fwd}} = \dot{\chi}_{\text{fwd}} \tag{C.35}$$

where $\dot{\chi}_{\text{fwd}}$ is derived from Section C.2.2.

### C.2.5 Boat Rudder Steering Rate Control

The saturated steering rate for the rudder is

$$\Delta_{r,c\,\text{sat}} = \text{sat}(\Delta_{r,c}, [-\Delta_{r,\max}, \Delta_{r,\max}]) \tag{C.36}$$

and the unsaturated rate is evaluated as

$$\Delta_{r,c} = \begin{cases} \Delta_{r,d} + \Delta_{r,\text{fwd}} & \text{if } \mathbf{p}_e < \mathbf{p}_{\text{tol}} \text{ and } \psi_e < \psi_{\text{tol}} \\ \Delta_{r,d} & \text{else} \end{cases} \tag{C.37}$$

Like all the other commands, the forward term is applied only when the boat is within some tolerance of the trajectory position and heading. The desired rudder steering angular rate is evaluated by multiplying the rudder angle error by a proportional gain and is expressed as

$$\Delta_{r,d} = \delta_e K_{\delta_r} \tag{C.38}$$

and the error of the rudder angle is calculated as

$$\delta_{r,e} = \phi_{\text{error}}(\delta_r, \delta_{r,d,\text{sat}}) \tag{C.39}$$

The desired angle of the rudder is physically limited, therefore saturated and written as

$$\delta_{r,d,\text{sat}} = \text{sat}(\delta_{r,d}, [-\delta_{\max}, \delta_{\max}]) \tag{C.40}$$

The unsaturated desired angle is evaluated as

$$\delta_{r_d} = -\sin^{-1}\left(\text{sat}\left(\frac{\omega_d(V + c_b)}{c_r \tan^{-1}(V^2)}, -1, 1\right)\right) \tag{C.41}$$

This is calculated from Equation 6.40. The term inside of the inverse sine function is saturated so that the inverse sine function does not break. The desired angular rate is also calculated by proportional control and is expressed as

$$\omega_d = \psi_e K_\psi \tag{C.42}$$

with an error term written as

$$\psi_e = \phi_{\text{error}}(\psi, \chi_d) \tag{C.43}$$

The feedforward angular rate of the rudder is derived from Equation 6.43 and the desired trajectory. However, when the trajectory rudder angle is saturated, the value of the feedforward term changes. It is expressed as

$$\Delta_{r,\text{fwd}} = \begin{cases} -\ddot{\chi}_{\text{traj}} \quad \text{if} \quad -\delta_{\max} < \delta_{traj} < \delta_{\max} \\ \left(\dfrac{c_r \dot{V}_{\text{traj}} \sin(\delta_{\text{traj}}) \tan^{-1}(V_{\text{traj}}^2)}{(c_b + V_{\text{traj}})^2} - \dfrac{2c_r V_{\text{traj}} \dot{V}_{\text{traj}} \sin(\delta_{\text{traj}})}{(V_{\text{traj}}^4 + 1)(c_b + V_{\text{traj}})} - \ddot{\chi}_{\text{traj}}\right) \\ \left(\dfrac{c_b + V_{\text{traj}}}{c_r \cos(\delta_{\text{traj}}) \tan^{-1}(V_{\text{traj}}^2)}\right) \quad \text{otherwise} \end{cases} \tag{C.44}$$

The angular acceleration of the trajectory is derived as

$$\ddot{\chi}_{\text{traj}} = \frac{(\dot{x}_{\text{traj}}^2 + \dot{y}_{\text{traj}}^2)(\dddot{y}_{\text{traj}} \dot{x}_{\text{traj}} - \ddot{x}_{\text{traj}} \dot{y}_{\text{traj}}) + 2(\ddot{x}_{\text{traj}} \dot{y}_{\text{traj}} - \dot{x}_{\text{traj}} \ddot{y}_{\text{traj}})(\dot{x}_{\text{traj}} \ddot{x}_{\text{traj}} + \dot{y}_{\text{traj}} \ddot{y}_{\text{traj}})}{(\dot{x}_{\text{traj}}^2 + \dot{y}_{\text{traj}}^2)^2} \tag{C.45}$$

The rudder steering angle of the trajectory is

$$\delta_{r_{\text{traj}}} = -\sin^{-1}\left(\text{sat}\left(\frac{\dot{\chi}_{\text{traj}}(V_{\text{traj}} + c_b)}{c_r \tan^{-1}(V_{\text{traj}}^2)}, -1, 1\right)\right) \tag{C.46}$$

The magnitude of the tangential acceleration of the trajectory at the point of interest is

$$\dot{V}_{\text{traj}} = \mathbf{a}_{\text{traj}} \cdot \hat{\mathbf{V}}_{\text{traj}} \tag{C.47}$$

and the magnitude of the velocity of the trajectory at the point of interest is

$$V_{\text{traj}} = ||\mathbf{V}_{\text{traj}}|| \tag{C.48}$$

## C.3 Fixed-Wing Aircraft Trajectory Tracking

This section describes the trajectory tracking control laws used in this thesis for a fixed-wing aircraft.

### C.3.1 Desired Velocity Vector

Given the start time $t_0$, control points $\mathbf{P}$, and scale factor $\alpha$, we can evaluate the position $\mathbf{p}_{\text{traj}}$, velocity $\mathbf{V}_{\text{traj}}$, and acceleration $\mathbf{a}_{\text{traj}}$ of the trajectory at the current point in time $t$. $\mathbf{V}_d$ is computed using PID position error control, with the velocity of the trajectory as a feedforward term. When the position error $\mathbf{p}_e$ is outside some tolerance $\mathbf{p}_{\text{tol}}$, the integral term is set to zero. The desired velocity vector is expressed as

$$\mathbf{V}_d = \begin{cases} \mathbf{p}_e K_p + \int \mathbf{p}_e K_i - \dot{\mathbf{p}}_L K_d + \mathbf{V}_{\text{traj}} & \text{if } \mathbf{p}_e < \mathbf{p}_{\text{tol}} \\ \mathbf{p}_e K_p - \dot{\mathbf{p}}_L K_d + \mathbf{V}_{\text{traj}} & \text{else} \end{cases} \tag{C.49}$$

The error term is defined as

$$\mathbf{p}_e = \mathbf{p}_{\text{traj}} - \mathbf{p} \tag{C.50}$$

and the integral term is calculated as

$$\int \mathbf{p}_e = \int \mathbf{p}_{e,\text{prev}} + (\mathbf{p}_e + \mathbf{p}_{e,\text{prev}}) \frac{dt}{2} \tag{C.51}$$

where subscript prev denotes the previous value. The derivative term is calculated as

$$\dot{\mathbf{p}}_L = \dot{\mathbf{p}} - (\dot{\mathbf{p}} \cdot \hat{\mathbf{V}}_{\text{traj}}) \hat{\mathbf{V}}_{\text{traj}} \tag{C.52}$$

where

$$\dot{\mathbf{p}} = \frac{\mathbf{p} - \mathbf{p}_{\text{prev}}}{dt} \tag{C.53}$$

We use the portion of the derivative vector perpendicular to $\mathbf{V}_{\text{traj}}$ because we do not want to counteract the velocity of the trajectory.

### C.3.2 Autopilot Commands

Given the desired velocity vector, we can calculate the input commands to the autopilot. The course angle command is calculated as

$$\chi_c = \tan^{-1}\left(\frac{\mathbf{V}_{d,y}}{\mathbf{V}_{d,x}}\right) \tag{C.54}$$

The desired airspeed is calculated as

$$V_{a,c} = \text{sat}(\|\mathbf{V}_d\|, [V_{\min}, V_{\max}] \tag{C.55}$$

and the climb rate command as

$$\dot{h}_c = -\hat{\mathbf{V}}_{d,z} V_{a,c} \tag{C.56}$$

Notice that the commanded airspeed command is saturated, and the saturated airspeed is used to calculate a desired climb rate.

The roll feedforward command derivation is taken from [49] and is expressed as

$$\phi_{\text{ffwd}} = \lambda \tan^{-1}\left(\frac{V_a^2}{g r_{\text{flat}}}\right) \tag{C.57}$$

where $g$ is a gravity constant and $r_{\text{flat}}$ is the planar horizontal radius defined as

$$r_{\text{flat}} = \frac{\|\mathbf{V}_{\text{traj},xy}\|^3}{\|A_{xy}\|} \tag{C.58}$$

$\lambda$ is the sign of $A_{xy}$. Since we only consider the horizontal planar curvature of the path, $A_{xy}$ is the cross product of two vectors in the $XY$ plane. It is defined as

$$A_{xy} = \mathbf{V}_{\text{traj},xy} \times \mathbf{a}_{\text{traj},xy} \tag{C.59}$$

and results in a vector magnitude along the $Z$ axis. The roll feedforward is only applied when the vehicle is within a specified tolerance of the current trajectory pose.