Application of the Finite Fourier Series for Smooth Motion Planning of Quadrotors

by

Yevhenii Kovryzhenko

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 5, 2023

Keywords: motion planning, Finite Fourier Series (FFS), minimum snap, trajectory optimization, UAV guidance

Abstract

Motion planning and control system design are crucial in the development of unmanned aerial vehicles (UAVs) for various applications. To maximize the mission performance, trajectory design is often performed to minimize motor output and power consumption along the entire path while avoiding obstacles. Polynomial parameterizations have traditionally been used to express UAV trajectories due to their simplicity and effectiveness of implementation. In this work, an alternative parameterization based on Finite Fourier Series (FFS) is investigated. The results of the FFS-based parameterization are compared with the state-of-the-art polynomial-based trajectory generation algorithms. One unique feature of the FFS parameterization is the theoretical piece-wise infinite differentiability of multi-segment trajectories. For fixed-time minimum-snap motion-planning problems using FFS parameterization, it is 1) shown that motion-planning problems can be formulated as quadratic programming (QP) problems, and 2) derived an analytic solution to an unconstrained QP problem. Leveraging the analytic solution, formulation, and solution of time-allocated minimum-snap multi-segment trajectories is also presented. The practical limitations of the FFS method are also discussed in detail. Finally, formulation and numerical comparison of the minimum-snap, multi-segment trajectories with time-allocation are also presented. Performance (with respect to the computation time, power required, and number of iterations) of the FFS and polynomial parameterizations are compared using five representative trajectories. The results show the comparable performance of the two parameterizations with important differences between the two on high-level derivatives. An in-house quadrotor is used with a six-degree-of-freedom (6DoF) cascaded PID flight control logic to experimentally validate the tracking of the generated smooth trajectories.

Acknowledgments

I am grateful to Dr. Ehsan Taheri for his invaluable guidance and support throughout my thesis journey. His encouragement and technical expertise have been instrumental in the completion of this work. I would also like to thank my committee members, Dr. Nan Li and Dr. Imon Chakraborty, for their insightful comments and feedback that greatly improved the quality of this thesis.

I would like to thank Auburn University, Samuel Ginn College of Engineering, and the Department of Aerospace Engineering for their financial support through the Gavin Fellowship which has partially supported my studies.

I am also thankful to Auburn University faculty and staff for their patience and dedication in providing a conducive research environment. I would like to express my appreciation to Ella Atkins for providing access to the *rc_pilot* firmware, and to Matthew Romano and Behdad Davoudi for their assistance in understanding how to use it properly. Their contributions have been vital in initiating this research and enhancing my understanding of embedded system code and architecture.

Furthermore, I am deeply grateful to Dr. Russell Mailen for sharing his expertise and knowledge and providing constructive feedback on my early thesis draft. His step-by-step guide has been a driving force in the timely completion of this work.

Lastly, I would like to express my heartfelt thanks to my family and friends for their unwavering support and encouragement throughout this journey. Their patience and understanding have been a constant source of motivation and inspiration.

Once again, I express my heartfelt appreciation to all those who contributed to the successful completion of this thesis.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3D LiDAR	3-Dimentional Light Detection and Ranging
F-FFS	Fast Finite Fourier Series
FFS	Finite Fourier Series
MOCAP	Motion Capture System
NLP	Non-Linear Programming
QP	Quadratic Programming
UAS	Unmanned Areal System
UAV	Unmanned Areal Vehicle

Chapter 1

1.1 Introduction

Unmanned aerial vehicles (UAVs) have revolutionized the field of technology and exploration, thanks to rapid advances in microelectronics and manufacturing techniques. These advancements have made small robotic systems more affordable and capable, allowing them to perform tasks that were once only possible with large machinery and manned labor. In response, major commercial companies are investing millions of dollars in autonomous package delivery technologies that use small flying multi-rotors, which produce zero operational emissions.

During the COVID-19 pandemic, the UPS utilized UAVs to transport vibration-sensitive medical packages to local clinics, achieving a 30% boost in logistics efficiency [1]. UAVs offer more efficient solutions to existing problems and have the potential to offer more affordable solutions to previously challenging problems. For instance, practitioners worldwide are exploring the utility of UAVs for search and rescue missions and aerial photography. Recent global events have demonstrated that the introduction of UAVs on the battlefield can be as game-changing as the arrival of tanks and aircraft during the First World War [2]. These and many other fields are driving the development of new aerial platforms, complex on-board algorithms, and novel approaches to control guidance and navigation of UAVs.

By removing the need for a trained pilot on board the aircraft, space, weight, and cost can be saved while greatly increasing the safety of the system and expanding the range of potential missions that can be accomplished. However, designing an unmanned system entails having a capable-enough on-board guidance, navigation, and control system that can replace the on-board pilot and/or allow remote human input. This research effort aims to develop a control and guidance system for a small quadrotor, with a particular focus on a higher-level motion-planning algorithm. Specifically, the focus is on investigating the use of the Finite Fourier Series (FFS) parameterization for trajectory optimization given a set of waypoints. Using an equivalent approach, a comparison of FFS with polynomial parametrization is presented. It is important to emphasize that the minimum snap trajectory optimization that relies on polynomial parametrization is currently a classical technique for rapid trajectory optimization of quadrotors. Due to the differential-flatness properties of the dynamics of quadrotors, this method does not explicitly rely on vehicle dynamics.

1.2 Literature Review and Motivation

Motion- and path-planning algorithms serve as an integral element of any UAV mission design [3, 4, 5]. In the age of space exploration and complex robotic systems, it is no longer sufficient to define the objective and the intended direction of motion. Path planning for aerospace vehicles and robotic systems requires precise motion planning, often involving collision and obstacle avoidance [6, 7, 8, 9, 10, 11]. The tasks of path planning often extend far beyond simple route-planning and generally encompass the functions of the overall system coordination and establish an interface between the machine and a human operator who, generally, defines a global task for the vehicle or its purpose. The necessary commands and interactions are then followed automatically, relaying the necessary information downstream to all the vehicle's subsystems, each executing a well-defined task. For this exact reason, path planning is usually regarded as the highest level of control or the bridge that establishes autonomy [12].

From a mission-designer perspective, it is convenient to think of path planning as an approach to defining a set of tasks or milestones for the vehicle to execute. For example, one can consider a map of local terrain features, such as varying elevation, structures, trees, or else [12] (see Figure 1.1 for a visual explanation of this process). A set of tasks can be defined (e.g., payload delivery to certain places). All the aforementioned tasks can be summarized by a set

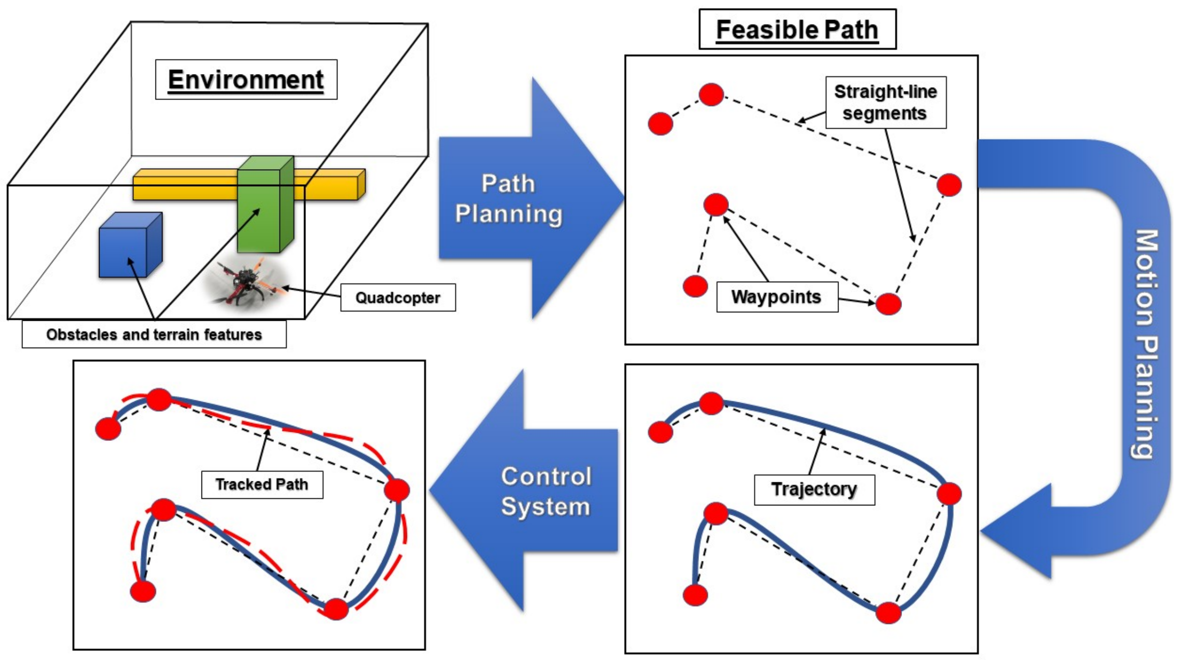


Figure 1.1: General schematics for the path and trajectory planning processes.

of requirements to traverse from a starting waypoint to a target destination. The task of finding a feasible path, irrespective of vehicle dynamics and time, is what can be defined as path planning. It is traditional to separate front-end path planning (i.e., generation of waypoints) and back-end motion planning or trajectory optimization that takes into account dynamic feasibility [13]. In other words, path planning (or front-end) attempts to find a discrete path between the initial and final points or collection of points of interest, while respecting real-world dynamics and constraints. Motion planning (or the back-end), on the other hand, builds upon the set of discrete waypoints along with some notion of dynamic feasibly to perform an end-to-end trajectory optimization in between the waypoints generated by the front-end. A plenitude of path-planning methods exist, ranging from sampling-based [14] to search-based [15, 16, 17] algorithms.

One popular sampling-based algorithm is Randomly-Exploring Random Tree (RRT) [18] search. The foundational principles of this algorithm are based on exploring randomly generated trees of possible junctions and intersections, seeking to connect the point of origin or initialization with the target or destination. The algorithm can be thought of as a tree, with its root planted at the starting point and each subsequent randomly sampled point in the problem space to be a branch or a node of this tree. This tree grows and expands by connecting the closest nodes together if the connection is feasible. The notion of distance and feasibility can be defined based on the specific problem but is usually based on some easy-to-compute heuristics to achieve the computational speeds necessary for real-time requirements. The algorithm iterations terminate when one of the randomly planted nodes is placed close enough to the destination (target, goal) and a feasible (e.g., collision-free segmented path) connection between the start, multiple intermediate waypoints, and end goal exists. Such an algorithm is perfect for exploring the mapped space with obstacles and finding a feasible path without any human intervention [19, 20]. However, sampling-based methods greatly suffer from their random nature and cannot guarantee convergence for real-time applications. Therefore, they are often replaced or complemented by rapid geometry-based re-planing routines [21, 22, 23].

Among some of the common practical applications of path planning blended with trajectory optimization is on-board trajectory generation in a cluttered environment [24]. For example, Hong *et al.* [25] have combined the A^* path-planning algorithm with a polynomial-based, minimum-snap trajectory optimization routine and demonstrated how such a blend of algorithms can be used for real-time collision avoidance when flying through the forest. For path-planning purposes, they have used a 3D LiDAR system for generating point clouds and used an on-board companion computer (*NVIDIA Jetson TX2* ™) to compute the high-level command to the control system. An equivalent combination of 3D LiDAR sensor and on-board companion computer was used by Zhou *et al.* in [26] to compute collision-free trajectories on the fly, but now indoors, flying even a smaller quadrotor. Similar applications are proposed and demonstrated for path-planning tasks of multi-agent systems [27] and exploration of in-door environments [28]. All mentioned applications involve solving a minimum-snap trajectory optimization problem using the polynomial-based parameterization [29].

Modern convex-optimization-based motion-planning techniques have broadened the range of practical applications of UAVs [30, 31]. Trajectory optimization [32] for quadrotors, especially the methods discussed in this work, heavily rely on the concept of *differential flatness* [33, 34]. Mellinger and Kumar [35] established a connection between states of a quadrotor and four *flat outputs*. The concept of differential flatness makes it possible to establish a relation between full (twelve) states of a rigid-body model of a quadrotor to only translational position coordinates X, Y, Z, heading (or yaw) angle, ψ, and their appropriate higher-order derivatives [36]. Mellinger and Kumar have also shown that a solution can be obtained by properly selecting a cost function to be minimized along the trajectory for each of the flat outputs independently to generate smooth trajectories. It is possible to obtain a smooth, continuously differentiable (up to a certain order) path by expressing the position of the vehicle using piecewise polynomials and minimizing its fourth time-derivative squared. They have shown that dynamic constraints can be omitted (for a differentially flat system) to ensure that the path of the vehicle is differentiable and continuous up to the fourth order. More importantly, for obtaining the least-energy path, it is sufficient to minimize the fourth derivative squared (as the cost), without considering complex dynamics in the formulation of the optimization problem. This optimization problem, or minimum-snap optimization, can be solved by formulating the original cost function as a quadratic programming (QP) problem [37], where the solution yields a set of optimal polynomial coefficients that are valid along their respective intervals. The resulting QP problems can be further augmented by incorporating linear equality and inequality constraints to represent path (corridor-like) and boundary constraints on the trajectory, and the resulting QP problems can be solved using QP solvers.

Instead of treating the optimal set of polynomial coefficients as the unknowns, polynomial coefficients can be indirectly incorporated into the solution. In fact, Ritcher *et al.* [38] showed that minimum-snap trajectories can be obtained by reformulating the QP problem as an unconstrained QP problem. However, this time, a direct analytical solution can be obtained without the need to resort to iterative numerical solvers. They have shown that, by leveraging the differential flatness property, a guaranteed feasible solution can be obtained analytically, as long as the involved derivatives are sufficiently bounded or enough time is available for the execution of the maneuver. In other words, by allowing more time for the maneuver, the required velocity, acceleration, and higher-order derivatives can be lowered for the entire trajectory such that they satisfy all the constraints and limitations of the vehicle and onboard control system. Their solution strategy has shown excellent numerical stability and computational speed for long-range trajectories that consist of many segments. Differential flatness property combined with the fact that the flat outputs consist of three important configuration space parameters (i.e., quadrotor position) provides a convenient and powerful tool for rapid design of collision-free, continuous, and time-optimal trajectories even in dense indoor environments [39]. The appealing form of the polynomial-based parameterization provides a simple interface for expressing a path of varied complexity and degrees of freedom, as well as allowing intuitive implementation of the optimization algorithms [40]. Although several optimization methods for quadrotors have been developed, as shown by Kreciglowa *et al.* in [41], minimum-snap optimization leads to the most power-efficient solution for quadrotor flights (when compared with minimum-acceleration and minimum-jerk solutions). The task of path planning and trajectory optimization for quadrotors that consist of many segments and with real-time considerations is an ongoing research topic [42, 43, 44, 45, 46]. The choice of parametrization affects the computational speed and numerical stability of the algorithm. Although polynomials offer an intuitive and simple way of expressing the components of the position vector, an alternative parametrization can potentially lead to faster and more efficient computation while obtaining similar results.

1.3 Earlier work on FFS

This thesis is based upon some previous research, with the original work demonstrating the FFS parameterization being used for generating spacecraft continuous-thrust trajectories by Taheri *et al.* [47, 48]. The original method was then further improved in [49] by reformulating the FFS relations in a compact matrix representation, making the FFS method suitable for programming purposes. In Ref. [50], it is shown that control (maximum thrust magnitude) path constraints can also be handled using the FFS method. Benefiting from its high computational efficiency and flexibility of incorporating various constraints, the FFS method was adopted for shaping spacecraft trajectories with various coordinate representations and propulsion systems [51, 52, 53, 54]. Most recent applications of the FFS method are for solving spacecraft multiple gravity-assist low-thrust trajectory optimization problems [55] and feasibility demonstration for quadrotor motion planning [56]. Parameterized trajectory planning for dynamic soaring using sinusoidal basis functions is also investigated in [57]. Parts of this work have been published in [58]. In all these works, the authors have demonstrated the superior performance of the FFS method in solving a wide range of periodic spacecraft trajectories and have claimed that FFS offers several advantages when compared with other shape-based methods. In summary, the advantages include piece-wise infinite differentiability (theoretical), greater flexibility in handling dynamical and path constraints, and greater computational efficiency. The goal of this work was to directly compare the FFS against polynomial parametrization with an equivalent solution methodology based on simulation and experimental results in the context of trajectory optimization for a quadrotor. To the author's best knowledge, FFS parametrization has not been investigated for solving minimum-snap trajectories for quadrotors. To justify the above claims, detailed theoretical and experimental comparisons between FFS and state-of-the-art technique for quadrotor trajectory optimization is lacking in the literature. Polynomial parametrization is considered the baseline for comparison since it is a very common quadrotor trajectory optimization technique for an input set of waypoints and dynamical constraints. Moreover, the proposed motion-planning methodology is based on obtaining smooth trajectories for a differentially flat system and does not explicitly depend on the system dynamics, which is a very attractive solution approach due to its simplicity of implementation and problem generality.

1.4 Objectives

The use of polynomial parameterizations has been the dominant approach for motion planning of quadrotors, aircraft, and satellites due to their simplicity and effectiveness in generating smooth trajectories. However, this work proposes an alternative parameterization for an existing motion planning algorithm by using FFS. It can be demonstrated that the formulation of minimum-snap trajectories using the FFS method leads to QP problems that can be efficiently solved. To compare the two parameterizations, first, an unconstrained QP problem is formulated, and next, a time-allocation optimization routine for both polynomial and FFS parameterizations is developed. By using an equivalent algorithm and identical waypoints for the two parameterizations, the resulting trajectories are compared against each other based on shape, numerical convergence of the resulting QPs, and computational speed. The comparison involves both theoretical results using simulation and experimental validation with an in-house quadrotor. Some of the peculiar features of the two parameterizations are also discussed in this thesis, all based on five different example trajectories.

This thesis aims to provide additional insight into the advantages and disadvantages of selecting FFS parametrization for minimum-snap motion planning of quadrotors. However, some classes of flight vehicles require the satisfaction of higher-order derivatives beyond snap. For example, the fifth derivative of position is required for expressing all the states and control outputs of a conventional (fixed-wing) aircraft with just position, heading angle, and their higher-order derivatives (see Chapter 14.0.4.1 in [59]). The choice of the FFS parameterization may offer some potential advantages for those classes of problems since FFS trajectories are piece-wise infinitely differentiable. As the results presented in this thesis indicate, the selection of a different base function can also lead to a reduced number of iterations for one of the time-allocation problems. This suggests that FFS can offer computational advantages for problems requiring high-order derivates. Although the FFS parametrization is theoretically piece-wise infinitely differentiable, taking high-order derivatives can lead to numerical instability due to loss of precision, the principal reason of which is outlined in Sec. 2.1.1.2. One of the main goals of this research effort is to compare the formulation of the two parameterization choices for motion planning of quadrotors. For fixed-time minimum-snap motion-planning problems using FFS parameterization, it is 1) shown that motion-planning problems can be formulated as QP problems, and 2) an analytic solution to an unconstrained QP problem is also derived. Leveraging the analytic solution, formulation, and solution of time-allocated minimum-snap multi-segment trajectory optimization solution methodology is also presented. In summary, the primary goal of this thesis is to present a fair one-to-one comparison of the FFS- and polynomial-based formulations for five classes of trajectories and present both computational results and experimental validation using a custom-built quadrotor. Position and heading angle boundary condition information and time of flight data are given for the fixed-time cases in the Appendix for ease of reproducing the test cases.

To validate the proposed method, a quadrotor simulation was developed. Additionally, a heuristic-based framework for “hands-off” gain tuning of multi-rotor vehicles in simulated environments has been developed as one of the byproducts of this research effort. Although there are plenty of gain-tuning techniques already discussed in the literature, they often deal with final refinement and not the entire end-to-end gain-tuning process. In other words, a *stable* set of gains has to be provided to the algorithm at initialization, obtaining which can be a very involved process that has to be done manually. A set of cost functions was considered to incorporate control effort, stability, agility, and overall state tracking into the gain-tuning process, as control gain tuning for a complex non-linear system can be a daunting task. The method allows for intuitively defining the notion of a well-behaved flight and performing control system gain optimization for different classes of trajectories. The proposed method uses MATLAB's `fmincon` optimizer for all gain tuning by starting from a completely unstable set of gains. The method is independent of the exact implementation of the control system and treats it as a set of black boxes with known inputs and outputs. The control system gains are the only variables required to be tuned to obtain optimal flight performance as defined by the compound cost. This method has been developed with the main purpose of saving time spent manually performing gain-tuning. Although it is not fast at all, the method is robust to failure and is entirely "hands-off", meaning that when starting with a random set of controller gains (usually completely unstable), the algorithm can find a feasible set and further optimize it, without any intervention. Just to be clear, this method was developed for in-simulation gain-tuning only and is not the major focus of this work. Only high-level key details are covered here, since the specific control system implementation (and subsequent gain-tuning) is very subjective and is only discussed here for completeness. The main purpose of this work is to compare the performance of the two motion-planning techniques, both theoretically and experimentally. Any control system, as long as it can realize the required trajectory, should suffice.

The remainder of this thesis is organized as follows. In section 2.1, the trajectory generation algorithm is broken down into two parts. First, a solution to an unconstrained QP problem with a fixed-time allocation is derived in Section 2.1.1. Next, the problem solution is augmented with a time-allocation algorithm derived in Section 2.1.2. Section 2.1 is concluded by providing some details about testing the two algorithms, hardware implementation, and details regarding the implementation process in Section 2.1.3. Computational results for fixed-time problems (Sec. 3.1.1) and time-allocation problems (Sec. 3.1.2) that benchmark the two parameterization methods against each other are presented in Section 3. An alternative formulation of the FFS is introduced with computational results shown in Section 3.1.2.5. Finally, experimental validation results are presented in Section 3.2. Section 4.1 summarizes the most important results and key findings.

Chapter 2

2.1 Trajectory Planning

2.1.1 Formulation of fixed-time, minimum-snap trajectory optimization problems

Leveraging the concept of differential flatness for quadrotors [35], it is possible to perform motion planning along each axis (i.e., $x,\ y$ and z) of the motion independently. Additionally, the heading angle can be considered as a separate fourth dimension. Let $P(t,\boldsymbol{p})$ represent an expression for each of the flat outputs, such that it is differentiable with respect to time up to n_{dt} times, where $\boldsymbol{p}$ is a vector of design variables. The general approach is to minimize the square of the n_{dt}-th derivative of $P(t,\boldsymbol{p})$ over the entire interval and for each of the axes of motion. While motion planning of robotic joints, limbs, and manipulators [60, 61] require minimization of the third derivative squared (or minimum-jerk optimization with $n_{\text{dt}} = 3$), motion planning for multirotors must consider minimization of the fourth derivative of the position squared. The Lagrange form cost functional for these classes of problems can be written as:

$$\underset{\boldsymbol{p}}{\text{minimize}} \qquad J = \int_{t_{\text{i}}}^{t_f} \left[\frac{d^{n_{\text{dt}}}}{dt^{n_{\text{dt}}}} P(t,\boldsymbol{p}) \right]^2 dt, \tag{2.1}$$

in which $t \in [t_{\text{i}}, t_f]$. The exact same cost functional is used along each dimension (i.e., $x,\ y,\ z$ and heading angle) and the total cost required to be minimized is the summation of the costs for each axis of motion. The problem defined in Eq. (2.1) is a standard *variational problem*, and the calculus of variations techniques are used to find an extremal solution.

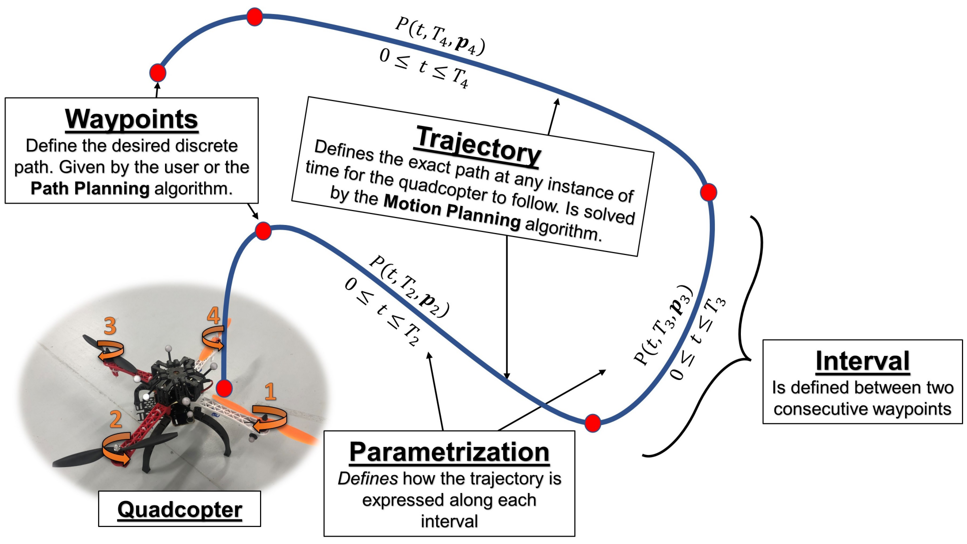

Waypoints
Define the desired discrete path. Given by the user or the Path Planning algorithm.
Trajectory
Defines the exact path at any instance of time for the quadcopter to follow. Is solved by the Motion Planning algorithm.
$P(t, T_4, \boldsymbol{p}_4)$
$0 \leq t \leq T_4$
$P(t, T_2, \boldsymbol{p}_2)$
$0 \leq t \leq T_2$
$P(t, T_3, \boldsymbol{p}_3)$
$0 \leq t \leq T_3$
Interval
Is defined between two consecutive waypoints
Parametrization
Defines how the trajectory is expressed along each interval
Quadcopter
1
2
3
4

Figure 2.1: Definition of a multi-segment/interval trajectory. “interval” and “segment” are used interchangeably.

A more general approach would be to consider linear equality and inequality constraints, which can then be reformulated as a constrained QP problem. Such a solution strategy has been explored by the authors in [56]. While this method offers flexibility such as directly incorporating corridor and path constraints, the main limitation is the computational cost and number of iterations required to achieve a fixed-time solution. Instead, only linear equality constraints can be considered to simplify the process and obtain a fixed-time solution analytically. For this thesis, the quadrotor motion-planning problem is formulated. First introduced in [35], a quadrotor is a differently-flat system. In other words, the quadrotor states (position, velocity, acceleration, attitude, and body frame angular rates) can all be expressed in terms of a set of flat output variables (i.e., spacial position and heading angle) and their derivatives. More importantly, the control output (i.e., required thrust and torque value) has been shown to be a function of the flat output variables and their derivatives. A detailed derivation has been presented in [34], but the key takeaway is that the motor speed required to execute a certain trajectory is a direct function of position, heading angle, and their derivatives (fourth derivative of position, and second derivative of the heading angle). In Ref. [62], it is also shown that the minimum-jerk quadrotor cost functional has an interpretation as an upper bound of the product of the inputs (i.e., thrust and angular rates). Therefore, to minimize power consumption on the i-th segment, $n_{\mathrm{dt}} = 4$

was selected in Eq. (2.1), and written as:

$$\begin{aligned} \underset{\boldsymbol{p}_\mathrm{i}}{\text{minimize}} \qquad & J = \int_{t_0=0}^{t_f=T_\mathrm{i}} \left[\frac{d^4}{dt^4}P(t,\boldsymbol{p}_\mathrm{i})\right]^2 dt = \boldsymbol{p}_\mathrm{i}^\top \mathbb{Q}_\mathrm{i}\boldsymbol{p}_\mathrm{i}, \\ \text{s.t.} \qquad & \mathbb{A}_\mathrm{i}\boldsymbol{p}_\mathrm{i} = \boldsymbol{d}_\mathrm{i}, \qquad \text{for } \mathrm{i} \in \{1,\ldots,n_\text{int}\}, \end{aligned} \tag{2.2}$$

where the matrix $\mathbb{A}_\mathrm{i}$ maps the vector of design variables, $\boldsymbol{p}_\mathrm{i}$, to the constraint vector, $\boldsymbol{d}_\mathrm{i}$. The matrix $\mathbb{Q}_\mathrm{i}$ maps the vector of design variables $\boldsymbol{p}_\mathrm{i}$ to the integral of $P(t, T_\mathrm{i}, \boldsymbol{p}_\mathrm{i})$, and T_i is the time allocated for the i-th segment (in a multi-segment trajectory) such that $0 \leq t \leq T_\mathrm{i}$. Note that T_i is assumed to be fixed and known. If more than one segment is used, time is allocated externally for each interval with a total of n_int (number of intervals) such that a time allocation vector can be formed as $\boldsymbol{T} = [T_1, \ldots, T_{\mathrm{n}_\text{int}}]^\top$. Figure 2.1 shows a schematic summarizing the definitions used throughout the paper. Each segment is represented by its own set of design variables, independent of other segments. When multiple segments are present, the individual cost function and constraints for each segment, $\mathrm{i} \in \{1, \cdots, n_\text{int}\}$, are used to express the relations in a block-diagonal manner as:

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \qquad & J = \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{n_\text{int}} \end{bmatrix}^\top \begin{bmatrix} \mathbb{Q}_1 & [0] & [0] \\ [0] & \ddots & [0] \\ [0] & [0] & \mathbb{Q}_{n_\text{int}} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{n_\text{int}} \end{bmatrix}, \\ \text{s.t.} \qquad & \begin{bmatrix} \mathbb{A}_1 & [0] & [0] \\ [0] & \ddots & [0] \\ [0] & [0] & \mathbb{A}_{n_\text{int}} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{n_\text{int}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_{n_\text{int}} \end{bmatrix}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{p}^\top = [\boldsymbol{p}_1^\top, \cdots, \boldsymbol{p}_{n_\text{int}}^\top]$ and $\mathbb{Q}_i$ and $\mathbb{A}_i$ correspond to the i-th segment and are specific to the parameterization of $P(t, \boldsymbol{p}_i)$. The constraint vector, $\boldsymbol{d}^\top = [\boldsymbol{d}_1^\top, \cdots, \boldsymbol{d}_{n_\text{int}}^\top]$, consists of the fixed and free boundary conditions, some of which are assumed to be specified (e.g., position at each waypoint, initial and final velocity, acceleration, etc.), but the rest are free (e.g., velocity, acceleration, and higher-order derivatives at each intermediate waypoint) and will have to be solved for. The solution steps are outlined in Sec. 2.1.1.3, but it is important to start with defining trajectory parameterization methods. First, the polynomial parameterization is reviewed, and then the FFS parameterization is introduced.

2.1.1.1 Generalization of the polynomial and FFS parameterizations

The minimum number of parameters for either method is defined by the number of boundary constraints. For a single-interval, single-axis, minimum-snap trajectory, there are always ten constraints to be satisfied, including: initial and final position, velocity, acceleration, jerk, and snap values. Therefore, there must be at least ten parameters in the general form of the $P(t, \boldsymbol{p}_\mathrm{i})$ (i.e., $\boldsymbol{p}_\mathrm{i} \in \mathbb{R}^{n \geq 10}$). Equality constraints on derivatives allow enforcing continuity up to snap when multiple segments are considered. However, an additional pair of equality constraints on the fifth derivative can be considered to ensure snap is both continuous and smooth. This sets the minimum number of parameters to twelve per axis of motion per interval. The primary focus of this thesis is on standard minimum-snap trajectory optimization with constraints applied only up to the fourth derivative. This method considers five boundary conditions at the beginning and the end of each interval, therefore requiring at least ten parameters for each interval on a single dimension (or axis of motion).

The procedure to calculate the required mapping matrices is identical for both the polynomial and FFS parameterizations. The constraint matrix $\mathbb{A}_\mathrm{i}$ is simply the Jacobian of the path $P(t, \boldsymbol{p}_\mathrm{i})$ and its derivatives with respect to (w.r.t.) the design vector, $\boldsymbol{p}_\mathrm{i}$, which is then evaluated at the initial time $t_0 = 0$ and final time $t_f = T_\mathrm{i}$. The quadratic mapping matrix, $\mathbb{Q}_\mathrm{i}$, is one-half of the Hessian of the cost function (Eq. (2.2)). It is assumed that $\boldsymbol{p}_\mathrm{i} \in \mathbb{R}^{10\times 1}$, $\mathbb{A}_\mathrm{i} \in \mathbb{R}^{10\times 10}$ and $\mathbb{Q}_\mathrm{i} \in \mathbb{R}^{10\times 10}$ for both parameterizations. These dimensions correspond to the i-th segment of a multi-segment trajectory for single-axis motion.

Polynomial parameterization. The most common parameterization for solving these classes of QP problems is a minimum-order polynomial function, which can be written as:

$$P(t, T_{\mathrm{i}}, \boldsymbol{p}_{\mathrm{i}}) = a_0 + a_1 \frac{t}{T_{\mathrm{i}}} + a_2 \left(\frac{t}{T_{\mathrm{i}}}\right)^2 + a_3 \left(\frac{t}{T_{\mathrm{i}}}\right)^3 + \cdots + a_n \left(\frac{t}{T_{\mathrm{i}}}\right)^n , \qquad \text{where} \quad \boldsymbol{p}_{\mathrm{i}} = [a_0, a_1, a_2, a_3, \ldots, a_n]^\top , \tag{2.4}$$

where time t is scaled by the segment time T_{i} allocated for the i-th segment and polynomial order is $n = 9$ for constraint derivative order of four ($n_{\mathrm{dt}} = 4$). Mapping matrices for this formulation are derived in the following manner. The time-dependent constraint matrix, $\mathbb{A}_{\mathrm{i}}(t, T_{\mathrm{i}})$, can be derived by taking the gradient (w.r.t. $\boldsymbol{p}_{\mathrm{i}}$) of the vector consisting of polynomial representation of the path and its derivatives (9th order polynomial, 4th order derivative) as:

$$\mathbb{A}_{\mathrm{i}}(t, T_{\mathrm{i}}) = \nabla_{\boldsymbol{p}_{\mathrm{i}}} \begin{bmatrix} P(t, T_{\mathrm{i}}, \boldsymbol{p}_{\mathrm{i}}) \\ \frac{d}{dt} P(t, T_{\mathrm{i}}, \boldsymbol{p}_{\mathrm{i}}) \\ \vdots \\ \frac{d^{n_{\mathrm{dt}}}}{dt^{n_{\mathrm{dt}}}} P(t, T_{\mathrm{i}}, \boldsymbol{p}_{\mathrm{i}}) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \frac{t}{T_{\mathrm{i}}} & \frac{t^2}{T_{\mathrm{i}}^2} & \frac{t^3}{T_{\mathrm{i}}^3} & \frac{t^4}{T_{\mathrm{i}}^4} & \frac{t^5}{T_{\mathrm{i}}^5} & \frac{t^6}{T_{\mathrm{i}}^6} & \frac{t^7}{T_{\mathrm{i}}^7} & \frac{t^8}{T_{\mathrm{i}}^8} & \frac{t^9}{T_{\mathrm{i}}^9} \\ 0 & \frac{1}{T_{\mathrm{i}}} & \frac{2\,t}{T_{\mathrm{i}}^2} & \frac{3\,t^2}{T_{\mathrm{i}}^3} & \frac{4\,t^3}{T_{\mathrm{i}}^4} & \frac{5\,t^4}{T_{\mathrm{i}}^5} & \frac{6\,t^5}{T_{\mathrm{i}}^6} & \frac{7\,t^6}{T_{\mathrm{i}}^7} & \frac{8\,t^7}{T_{\mathrm{i}}^8} & \frac{9\,t^8}{T_{\mathrm{i}}^9} \\ 0 & 0 & \frac{2}{T_{\mathrm{i}}^2} & \frac{6\,t}{T_{\mathrm{i}}^3} & \frac{12\,t^2}{T_{\mathrm{i}}^4} & \frac{20\,t^3}{T_{\mathrm{i}}^5} & \frac{30\,t^4}{T_{\mathrm{i}}^6} & \frac{42\,t^5}{T_{\mathrm{i}}^7} & \frac{56\,t^6}{T_{\mathrm{i}}^8} & \frac{72\,t^7}{T_{\mathrm{i}}^9} \\ 0 & 0 & 0 & \frac{6}{T_{\mathrm{i}}^3} & \frac{24\,t}{T_{\mathrm{i}}^4} & \frac{60\,t^2}{T_{\mathrm{i}}^5} & \frac{120\,t^3}{T_{\mathrm{i}}^6} & \frac{210\,t^4}{T_{\mathrm{i}}^7} & \frac{336\,t^5}{T_{\mathrm{i}}^8} & \frac{504\,t^6}{T_{\mathrm{i}}^9} \\ 0 & 0 & 0 & 0 & \frac{24}{T_{\mathrm{i}}^4} & \frac{120\,t}{T_{\mathrm{i}}^5} & \frac{360\,t^2}{T_{\mathrm{i}}^6} & \frac{840\,t^3}{T_{\mathrm{i}}^7} & \frac{1680\,t^4}{T_{\mathrm{i}}^8} & \frac{3024\,t^5}{T_{\mathrm{i}}^9} \end{bmatrix} . \tag{2.5}$$

The unconstrained optimization implies that no path constraints along each interval are considered and only the boundary conditions have to be respected. Therefore, Eq. (2.5) can is evaluated at the boundaries and the *constraint* mapping matrix $\mathbb{A}_i(T_{\mathrm{i}})$ is calculated. This leads to a very appealing form that only depends on the total time (which is fixed for this problem) as:

$$\mathbb{A}_{\mathrm{i}}(T_{\mathrm{i}}) = \begin{bmatrix} \mathbb{A}_{\mathrm{i}}(t=0,T_{\mathrm{i}}) \\ \mathbb{A}_{\mathrm{i}}(t=T_{\mathrm{i}},T_{\mathrm{i}}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{T_{\mathrm{i}}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{T_{\mathrm{i}}^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{6}{T_{\mathrm{i}}^3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{24}{T_{\mathrm{i}}^4} & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & \frac{1}{T_{\mathrm{i}}} & \frac{2}{T_{\mathrm{i}}} & \frac{3}{T_{\mathrm{i}}} & \frac{4}{T_{\mathrm{i}}} & \frac{5}{T_{\mathrm{i}}} & \frac{6}{T_{\mathrm{i}}} & \frac{7}{T_{\mathrm{i}}} & \frac{8}{T_{\mathrm{i}}} & \frac{9}{T_{\mathrm{i}}} \\ 0 & 0 & \frac{2}{T_{\mathrm{i}}^2} & \frac{6}{T_{\mathrm{i}}^2} & \frac{12}{T_{\mathrm{i}}^2} & \frac{20}{T_{\mathrm{i}}^2} & \frac{30}{T_{\mathrm{i}}^2} & \frac{42}{T_{\mathrm{i}}^2} & \frac{56}{T_{\mathrm{i}}^2} & \frac{72}{T_{\mathrm{i}}^2} \\ 0 & 0 & 0 & \frac{6}{T_{\mathrm{i}}^3} & \frac{24}{T_{\mathrm{i}}^3} & \frac{60}{T_{\mathrm{i}}^3} & \frac{120}{T_{\mathrm{i}}^3} & \frac{210}{T_{\mathrm{i}}^3} & \frac{336}{T_{\mathrm{i}}^3} & \frac{504}{T_{\mathrm{i}}^3} \\ 0 & 0 & 0 & 0 & \frac{24}{T_{\mathrm{i}}^4} & \frac{120}{T_{\mathrm{i}}^4} & \frac{360}{T_{\mathrm{i}}^4} & \frac{840}{T_{\mathrm{i}}^4} & \frac{1680}{T_{\mathrm{i}}^4} & \frac{3024}{T_{\mathrm{i}}^4} \end{bmatrix}. \quad (2.6)$$

The quadratic mapping matrix $\mathbb{Q}_{\mathrm{i}}(T_{\mathrm{i}})$ is computed by taking the Hessian of the cost function given by Eq. (2.2) w.r.t. the design vector $\boldsymbol{p}_{\mathrm{i}}$ and by applying a factor of $\frac{1}{2}$, which can be written as:

$$\mathbb{Q}_{\mathrm{i}}(T_{\mathrm{i}}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{576}{T_{\mathrm{i}}^7} & \frac{1440}{T_{\mathrm{i}}^7} & \frac{2880}{T_{\mathrm{i}}^7} & \frac{5040}{T_{\mathrm{i}}^7} & \frac{8064}{T_{\mathrm{i}}^7} & \frac{12096}{T_{\mathrm{i}}^7} \\ 0 & 0 & 0 & 0 & \frac{1440}{T_{\mathrm{i}}^7} & \frac{4800}{T_{\mathrm{i}}^7} & \frac{10800}{T_{\mathrm{i}}^7} & \frac{20160}{T_{\mathrm{i}}^7} & \frac{33600}{T_{\mathrm{i}}^7} & \frac{51840}{T_{\mathrm{i}}^7} \\ 0 & 0 & 0 & 0 & \frac{2880}{T_{\mathrm{i}}^7} & \frac{10800}{T_{\mathrm{i}}^7} & \frac{25920}{T_{\mathrm{i}}^7} & \frac{50400}{T_{\mathrm{i}}^7} & \frac{86400}{T_{\mathrm{i}}^7} & \frac{136080}{T_{\mathrm{i}}^7} \\ 0 & 0 & 0 & 0 & \frac{5040}{T_{\mathrm{i}}^7} & \frac{20160}{T_{\mathrm{i}}^7} & \frac{50400}{T_{\mathrm{i}}^7} & \frac{100800}{T_{\mathrm{i}}^7} & \frac{176400}{T_{\mathrm{i}}^7} & \frac{282240}{T_{\mathrm{i}}^7} \\ 0 & 0 & 0 & 0 & \frac{8064}{T_{\mathrm{i}}^7} & \frac{33600}{T_{\mathrm{i}}^7} & \frac{86400}{T_{\mathrm{i}}^7} & \frac{176400}{T_{\mathrm{i}}^7} & \frac{313600}{T_{\mathrm{i}}^7} & \frac{508032}{T_{\mathrm{i}}^7} \\ 0 & 0 & 0 & 0 & \frac{12096}{T_{\mathrm{i}}^7} & \frac{51840}{T_{\mathrm{i}}^7} & \frac{136080}{T_{\mathrm{i}}^7} & \frac{282240}{T_{\mathrm{i}}^7} & \frac{508032}{T_{\mathrm{i}}^7} & \frac{9144576}{11\,T_{\mathrm{i}}^7} \end{bmatrix}. \quad (2.7)$$

Both the linear constraint mapping matrix, $\mathbb{A}_{\mathrm{i}}(T_{\mathrm{i}})$, and quadratic mapping matrix, $\mathbb{Q}_{\mathrm{i}}(T_{\mathrm{i}})$, are valid for cases where only one interval and one axis of motion are considered. For more complex trajectories, corresponding $\mathbb{A}(\boldsymbol{T})$ and $\mathbb{Q}(\boldsymbol{p})$ matrices can be obtained as given in Eq. 2.3. This standard parameterization method is used as a basis of comparison against the FFS parameterization.

FFS parameterization. Similarly, FFS can be used to parameterize each dimension of motion as:

if n_{dt} is odd:
$$\begin{aligned} P(t, T_{\text{i}}, \boldsymbol{p}_{\text{i}}) = \frac{a_0}{2} + \sum_{k=1}^{n_{\text{r}}} \left\{ a_{\text{k}} \cos\left(\frac{1}{2}k\pi\frac{t}{T_{\text{i}}}\right) + b_{\text{k-1}} \sin\left(\frac{1}{2}k\pi\frac{t}{T_{\text{i}}}\right) \right\} \\ + b_{\text{n}_{\text{r}}} \cos\left(\frac{1}{2}n_{\text{r}}\pi\frac{t}{T_{\text{i}}}\right), \end{aligned} \tag{2.8}$$

if n_{dt} is even:
$$\begin{aligned} P(t, T_{\text{i}}, \boldsymbol{p}_{\text{i}}) = \frac{a_0}{2} + \sum_{k=1}^{n_{\text{r}}} \left\{ a_{\text{k}} \cos\left(\frac{1}{2}k\pi\frac{t}{T_{\text{i}}}\right) + b_{\text{k-1}} \sin\left(\frac{1}{2}k\pi\frac{t}{T_{\text{i}}}\right) \right\} \\ + b_{\text{n}_{\text{r}}} \sin\left(\frac{1}{2}n_{\text{r}}\pi\frac{t}{T_{\text{i}}}\right), \end{aligned} \tag{2.9}$$

where design vector is $\boldsymbol{p}_{\text{i}} = [a_0, \ldots, a_{\text{n}_{\text{r}}}, b_0, \ldots, b_{\text{n}_{\text{r}}}]$ with an abuse of notation. Due to the choise of optimization problem ($n_{\text{dt}} = 4$ and $n_{\text{r}} = 4$), only Eq. (2.9) is needed. However, Eq. (2.8) can be used for the cases when the order of constants and/or cost function is odd (from example, for minimum-jerk or minimum-crackle optimization). The two mapping matrices are derived in the exact same manner as for the polynomial parameterization. The constraint mapping matrix $\mathbb{A}_{\text{i}}(t, T_{\text{i}})$ is found by taking the gradient of the vector of derivatives w.r.t. vector of design variables $\boldsymbol{p}_i$. A constant mapping matrix $\mathbb{A}_{\text{i}}(T_{\text{i}})$ is found by evaluating $\mathbb{A}_{\text{i}}(t, T_{\text{i}})$ at the boundaries and staking the results vertically. The constant constraint mapping matrix corresponding to Eq. (2.9) is

$$
\begin{aligned}
A_\mathrm{i}(T_\mathrm{i}) &= \begin{bmatrix} \mathbb{A}_\mathrm{i}(T_\mathrm{i}=0,T_\mathrm{i}) \\ \mathbb{A}_\mathrm{i}(t_f=T_\mathrm{i},T_\mathrm{i}) \end{bmatrix} \\
&= \begin{bmatrix}
\frac{1}{2} & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{\pi}{2\,T_\mathrm{i}} & \frac{\pi}{T_\mathrm{i}} & \frac{3\,\pi}{2\,T_\mathrm{i}} & \frac{2\,\pi}{T_\mathrm{i}} & \frac{5\,\pi}{2\,T_\mathrm{i}} \\
0 & -\frac{\pi^2}{4\,T_\mathrm{i}^2} & -\frac{\pi^2}{T_\mathrm{i}^2} & -\frac{9\,\pi^2}{4\,T_\mathrm{i}^2} & -\frac{4\,\pi^2}{T_\mathrm{i}^2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{\pi^3}{8\,T_\mathrm{i}^3} & -\frac{\pi^3}{T_\mathrm{i}^3} & -\frac{27\,\pi^3}{8\,T_\mathrm{i}^3} & -\frac{8\,\pi^3}{T_\mathrm{i}^3} & -\frac{125\,\pi^3}{8\,T_\mathrm{i}^3} \\
0 & \frac{\pi^4}{16\,T_\mathrm{i}^4} & \frac{\pi^4}{T_\mathrm{i}^4} & \frac{81\,\pi^4}{16\,T_\mathrm{i}^4} & \frac{16\,\pi^4}{T_\mathrm{i}^4} & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & -1 & 0 & 1 & 1 & 0 & -1 & 0 & 1 \\
0 & -\frac{\pi}{2\,T_\mathrm{i}} & 0 & \frac{3\,\pi}{2\,T_\mathrm{i}} & 0 & 0 & -\frac{\pi}{T_\mathrm{i}} & 0 & \frac{2\,\pi}{T_\mathrm{i}} & 0 \\
0 & 0 & \frac{\pi^2}{T_\mathrm{i}^2} & 0 & -\frac{4\,\pi^2}{T_\mathrm{i}^2} & -\frac{\pi^2}{4\,T_\mathrm{i}^2} & 0 & \frac{9\,\pi^2}{4\,T_\mathrm{i}^2} & 0 & -\frac{25\,\pi^2}{4\,T_\mathrm{i}^2} \\
0 & \frac{\pi^3}{8\,T_\mathrm{i}^3} & 0 & -\frac{27\,\pi^3}{8\,T_\mathrm{i}^3} & 0 & 0 & \frac{\pi^3}{T_\mathrm{i}^3} & 0 & -\frac{8\,\pi^3}{T_\mathrm{i}^3} & 0 \\
0 & 0 & -\frac{\pi^4}{T_\mathrm{i}^4} & 0 & \frac{16\,\pi^4}{T_\mathrm{i}^4} & \frac{\pi^4}{16\,T_\mathrm{i}^4} & 0 & -\frac{81\,\pi^4}{16\,T_\mathrm{i}^4} & 0 & \frac{625\,\pi^4}{16\,T_\mathrm{i}^4}
\end{bmatrix}.
\end{aligned}
\tag{2.10}
$$

The Quadratic mapping matrix $\mathbb{Q}_{\mathrm{i}}$ for FFS parameterization is found using the exact same process as the one for polynomial representation, with the final expression written as:

$$\mathbb{Q}_{\mathrm{i}}(T_{\mathrm{i}}) =$$
$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\pi^8}{512\,T_{\mathrm{i}}^7} & \frac{\pi^7}{24\,T_{\mathrm{i}}^7} & 0 & -\frac{2\,\pi^7}{15\,T_{\mathrm{i}}^7} & \frac{\pi^7}{256\,T_{\mathrm{i}}^7} & \frac{\pi^7}{12\,T_{\mathrm{i}}^7} & \frac{81\,\pi^7}{256\,T_{\mathrm{i}}^7} & \frac{8\,\pi^7}{15\,T_{\mathrm{i}}^7} & \frac{625\,\pi^7}{768\,T_{\mathrm{i}}^7} \\
0 & \frac{\pi^7}{24\,T_{\mathrm{i}}^7} & \frac{\pi^8}{2\,T_{\mathrm{i}}^7} & \frac{243\,\pi^7}{40\,T_{\mathrm{i}}^7} & 0 & -\frac{\pi^7}{24\,T_{\mathrm{i}}^7} & 0 & \frac{243\,\pi^7}{40\,T_{\mathrm{i}}^7} & \frac{64\,\pi^7}{3\,T_{\mathrm{i}}^7} & \frac{3125\,\pi^7}{168\,T_{\mathrm{i}}^7} \\
0 & 0 & \frac{243\,\pi^7}{40\,T_{\mathrm{i}}^7} & \frac{6561\,\pi^8}{512\,T_{\mathrm{i}}^7} & \frac{486\,\pi^7}{7\,T_{\mathrm{i}}^7} & -\frac{81\,\pi^7}{256\,T_{\mathrm{i}}^7} & -\frac{81\,\pi^7}{20\,T_{\mathrm{i}}^7} & \frac{2187\,\pi^7}{256\,T_{\mathrm{i}}^7} & \frac{648\,\pi^7}{7\,T_{\mathrm{i}}^7} & \frac{50625\,\pi^7}{256\,T_{\mathrm{i}}^7} \\
0 & -\frac{2\,\pi^7}{15\,T_{\mathrm{i}}^7} & 0 & \frac{486\,\pi^7}{7\,T_{\mathrm{i}}^7} & \frac{128\,\pi^8}{T_{\mathrm{i}}^7} & -\frac{2\,\pi^7}{15\,T_{\mathrm{i}}^7} & -\frac{32\,\pi^7}{3\,T_{\mathrm{i}}^7} & -\frac{486\,\pi^7}{7\,T_{\mathrm{i}}^7} & 0 & \frac{6250\,\pi^7}{9\,T_{\mathrm{i}}^7} \\
0 & \frac{\pi^7}{256\,T_{\mathrm{i}}^7} & -\frac{\pi^7}{24\,T_{\mathrm{i}}^7} & -\frac{81\,\pi^7}{256\,T_{\mathrm{i}}^7} & -\frac{2\,\pi^7}{15\,T_{\mathrm{i}}^7} & \frac{\pi^8}{512\,T_{\mathrm{i}}^7} & \frac{\pi^7}{12\,T_{\mathrm{i}}^7} & 0 & -\frac{8\,\pi^7}{15\,T_{\mathrm{i}}^7} & 0 \\
0 & \frac{\pi^7}{12\,T_{\mathrm{i}}^7} & 0 & -\frac{81\,\pi^7}{20\,T_{\mathrm{i}}^7} & -\frac{32\,\pi^7}{3\,T_{\mathrm{i}}^7} & \frac{\pi^7}{12\,T_{\mathrm{i}}^7} & \frac{\pi^8}{2\,T_{\mathrm{i}}^7} & \frac{81\,\pi^7}{20\,T_{\mathrm{i}}^7} & 0 & -\frac{625\,\pi^7}{84\,T_{\mathrm{i}}^7} \\
0 & \frac{81\,\pi^7}{256\,T_{\mathrm{i}}^7} & \frac{243\,\pi^7}{40\,T_{\mathrm{i}}^7} & \frac{2187\,\pi^7}{256\,T_{\mathrm{i}}^7} & -\frac{486\,\pi^7}{7\,T_{\mathrm{i}}^7} & 0 & \frac{81\,\pi^7}{20\,T_{\mathrm{i}}^7} & \frac{6561\,\pi^8}{512\,T_{\mathrm{i}}^7} & \frac{648\,\pi^7}{7\,T_{\mathrm{i}}^7} & 0 \\
0 & \frac{8\,\pi^7}{15\,T_{\mathrm{i}}^7} & \frac{64\,\pi^7}{3\,T_{\mathrm{i}}^7} & \frac{648\,\pi^7}{7\,T_{\mathrm{i}}^7} & 0 & -\frac{8\,\pi^7}{15\,T_{\mathrm{i}}^7} & 0 & \frac{648\,\pi^7}{7\,T_{\mathrm{i}}^7} & \frac{128\,\pi^8}{T_{\mathrm{i}}^7} & \frac{5000\,\pi^7}{9\,T_{\mathrm{i}}^7} \\
0 & \frac{625\,\pi^7}{768\,T_{\mathrm{i}}^7} & \frac{3125\,\pi^7}{168\,T_{\mathrm{i}}^7} & \frac{50625\,\pi^7}{256\,T_{\mathrm{i}}^7} & \frac{6250\,\pi^7}{9\,T_{\mathrm{i}}^7} & 0 & -\frac{625\,\pi^7}{84\,T_{\mathrm{i}}^7} & 0 & \frac{5000\,\pi^7}{9\,T_{\mathrm{i}}^7} & \frac{390625\,\pi^8}{512\,T_{\mathrm{i}}^7}
\end{bmatrix}. \tag{2.11}$$

A few key features of the mapping matrices for the two parameterizations are noteworthy and are summarized below.

2.1.1.2 Key differences between polynomial and FFS parametrization

A claim can be made that time allocated for each segment of the trajectory is known and fixed, thus allowing analytic derivation of a minimizer for the quadratic cost function given by Eq. (2.3). The derivation presented in Sec. 2.1.1.3 proves this claim. The most immediate implication is that there should exist a unique solution corresponding to a fixed set of boundary conditions. By keeping the global time for the entire trajectory and fixed boundary conditions, but adjusting how much total time is allocated per segment, a family of solutions can be computed for a set of position waypoints. This feature has been explored in [38], but some of the important features will be reviewed in Section 2.1.2.

Since both mapping matrices (given in Eqs. (2.6) and (2.7) for polynomial and Eqs. (2.10) and (2.11) for FFS parametrizations) are fixed in size, with their elements only being functions of the time, T_i, they can be combined directly without the need to recompute them during the solution using the following matrix decomposition:

$$\tilde{\mathbb{Q}}_\text{i}(T_\text{i}) = \mathbb{A}_\text{i}^{-\top}(T_\text{i})\mathbb{Q}_\text{i}(T_\text{i})\mathbb{A}_\text{i}^{-1}(T_\text{i}). \tag{2.12}$$

The origin of the relation given in Eq. (2.12) becomes clear in Sec. 2.1.1.3, but Eq. (2.12) is computed every time irrespective of the parameterization used. In other words, the two mapping matrices derived earlier for each of the parametrization are used equivalently in the subsequent sections and all the differences between polynomial and FFS parametrization are encoded in Eq. (2.12). Since the solution to this problem requires operations with large matrices, it is useful to consider sparsity patterns. The result after computing the product given by Eq. (2.12) for the two parameterizations is at the root of the differences between the two parametrizations. Even though the mapping matrices $\mathbb{A}_\text{i}(T_\text{i})$ and $\mathbb{Q}_\text{i}(T_\text{i})$ have completely different sparsity patterns, the result of applying Eq. (2.12) is a fully dense matrix. In other words, either parameterization results in a $\mathbb{R}^{10\times 10}$ combined mapping matrix $\tilde{\mathbb{Q}}_\text{i}(T_\text{i})$ with every element being non-zero; thus, evaluation of the cost should take approximately the same time for both parameterizations. As is shown in Sec. 2.1.1.3, computing the optimal set of design variables involves mapping parts of the cost function back using only the inverse of constraint matrix $\mathbb{A}_\text{i}(T_\text{i})$ (Eq. (2.19)). Other than size, the constraint mapping matrices $\mathbb{A}_\text{i}(T_\text{i})$ are quite different in terms of the non-zero elements: $45/100$ for polynomial and $47/100$ for the FFS parameterizations, which obviously means that there are more elements to evaluate in the FFS formulation. However, the inverse of the same mapping matrix is needed, which means that the number of non-zero elements grows: $55/100$ for the polynomial and $100/100$ for the FFS parameterizations. This fact ultimately makes the FFS parameterization almost twice as expensive to compute. As a result, if all other parameters were to remain constant, it should be expected that the fixed-time solution obtained using the FFS parameterization is going to be more demanding to compute than the polynomial parameterization. The results in Tables 3.1

confirm this claim. The requirement of computing the inverse of the mapping matrices limits the theoretical "infinite differentiability" of the FFS parametrization. As higher-order derivatives are added into the formulation ($n_{\text{dt}} \geq 5$) since the coefficient in front of each trigonometric term grows in magnitude (with each derivative of Eq. (2.8) or Eq. (2.9)) the inverse is not numerically stable and can lead to inaccurate solutions.

2.1.1.3 Deriving analytic solution for fixed-boundary problem

The goal is to minimize the quadratic cost function given in Eq. (2.2) subject to a set of linear equality constants enforced by the vector $\boldsymbol{d}$. For the simplest case, when only one interval is preset for a single axis of motion, the vector $\boldsymbol{d}$ represents a set of boundary conditions for an interval, or pairs of initial and final position p, velocity v, acceleration a, jerk j, and snap s. Although the order of the elements can be arbitrary, assume the vector $\boldsymbol{d}$ consists of concatenated vectors $\boldsymbol{d}_k$, where each k-th vector is formed as follows:

$$\boldsymbol{d}_k = \left[p_{\text{k}}, v_{\text{k}}, a_{\text{k}}, j_{\text{k}}, s_{\text{k}}, p_{\text{k+1}}, v_{\text{k+1}}, a_{\text{k+1}}, j_{\text{k+1}}, s_{\text{k+1}}\right]^\top, \tag{2.13}$$

where subscripts k and $k+1$ indicate the current and next point (or first and last for a case when only one interval is present). Note that for a multi-segmented trajectory, intermediate waypoints have to be duplicated in the constraint vector $\boldsymbol{d}$ and this is accomplished through regrouping and duplicating matrices below. However, to obtain an analytic solution to this QP problem, an unconstrained QP problem formulation has to be considered. It was proposed by Richter *et al.* [38] and then further improved by Park *et al.* [63]. As will be shown later in this section, a globally optimal solution can be obtained to an unconstrained QP problem without any iterations, thus, speeding up the computations. The basic approach relies on breaking down the constraints into two types: fixed (specified) $\boldsymbol{d}_{\text{F}}$, and free (unspecified) $\boldsymbol{d}_{\text{P}}$. It is assumed that at the very first waypoint (on the first interval) and last waypoint (on the last interval) all derivatives are always fixed (specified) and are set to zero since it is assumed (but not required) that the vehicle starts and ends the trajectory at rest. The position of each waypoint is assumed to be known and specified. Continuity constraints on derivatives between the intervals, if more than one segment exists (i.e., $n_{\text{int}} > 1$), are also considered to be part of the $\boldsymbol{d}_{\text{F}}$ vector. With this rationale, certain grouping has to be performed to convert the constraint vector $\boldsymbol{d}$ into fixed and free constraints (boundary conditions at each waypoint).

To perform reorganization of the fixed and free constraints, an additional mapping matrix, $\mathbb{M}$, is introduced, which has a variable size depending on the number of intervals and constraints. A matrix $\mathbb{M}$ can be obtained by considering a product of two matrices. First, a re-grouping square matrix $\mathbb{M}_{\text{r}} \in \mathbb{R}^{n_{\text{c}} \times n_{\text{c}}}$ of the same size as the number of unique boundary conditions $n_{\text{c}} = (n_{\text{dt}} + 1)(n_{\text{int}} + 1)$ in $\boldsymbol{d} \in \mathbb{R}^{10 n_{\text{int}} \times 1}$ is introduced, which simply re-orders the original vector $\boldsymbol{d}$ such that specified constraints are followed by free ones. A second matrix, $\mathbb{M}_{\text{c}}$, takes care of continuity constraints by duplicating the intermediate waypoints for cases with more than one interval. For cases when only one interval is present, $\mathbb{M}_{\text{c}}$ is a square identity matrix of the same size as the $\mathbb{M}_{\text{r}}$, but for cases with more than one interval, it has to duplicate the interior points such that the constants at the end and at the beginning of the two subsequent intervals match. The total number of constraints should match the length of the state vector of coefficients, in our case, ten per interval (so $\mathbb{M}_{\text{c}} \in \mathbb{R}^{10 n_{\text{int}} \times n_{\text{c}}}$).

Constraint mapping can be summarized as:

$$\boldsymbol{d} = \mathbb{M} \left[\boldsymbol{d}_{\text{F}}^{\top}, \boldsymbol{d}_{\text{P}}^{\top} \right]^{\top}, \qquad \text{where} \qquad \mathbb{M} = \mathbb{M}_{\text{c}} \mathbb{M}_{\text{r}}, \tag{2.14}$$

in which matrices $\mathbb{M}_{\text{c}}, \mathbb{M}_{\text{r}}$ and $\mathbb{M}$ consist of ones and zeros and do not alter the constants in any way other than reordering and duplicating them. Next, the linear constraints given in Eq. (2.2) can be rearranged and solved for the vector of design variables, $\boldsymbol{p}$, in terms of the fixed and free constraints as:

$$\boldsymbol{p} = \mathbb{A}^{-1} \boldsymbol{d} = \mathbb{A}^{-1} \mathbb{M} \left[\boldsymbol{d}_{\text{F}}^{\top}, \boldsymbol{d}_{\text{P}}^{\top} \right]^{\top}. \tag{2.15}$$

Substituting Eq. (2.15) into Eq. (2.2), the resulting quadratic cost function can be rewritten as:

$$J = \boldsymbol{p}^\top \mathbb{Q} \boldsymbol{p} = \begin{bmatrix} \boldsymbol{d}_\text{F} \\ \boldsymbol{d}_\text{P} \end{bmatrix}^\top \underbrace{\mathbb{M}^\top \mathbb{A}^{-\top} \mathbb{Q} \mathbb{A}^{-1} \mathbb{M}}_{\mathbb{R}} \begin{bmatrix} \boldsymbol{d}_\text{F} \\ \boldsymbol{d}_\text{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_\text{F} \\ \boldsymbol{d}_\text{P} \end{bmatrix}^\top \begin{bmatrix} R_\text{PP} & R_\text{FP} \\ R_\text{PF} & R_\text{PP} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_\text{F} \\ \boldsymbol{d}_\text{P} \end{bmatrix}, \tag{2.16}$$

where the product $\mathbb{M}^\top \mathbb{A}^{-\top} \mathbb{Q} \mathbb{A}^{-1} \mathbb{M}$ is grouped together into matrix $\mathbb{R}$ and then partitioned into four blocks based on the fixed and free constraints. The product can further be expanded to obtain the following expression:

$$J = \boldsymbol{d}_\text{F}^\top R_\text{PP} \boldsymbol{d}_\text{F} + \boldsymbol{d}_\text{F}^\top R_\text{PF} \boldsymbol{d}_\text{P} + \boldsymbol{d}_\text{P}^\top R_\text{PF} \boldsymbol{d}_\text{F} + \boldsymbol{d}_\text{P}^\top R_\text{PP} \boldsymbol{d}_\text{P}. \tag{2.17}$$

Since fixed derivatives are already known, the main goal is then to use the free derivatives, $\boldsymbol{d}_\text{P}$, to find the minimum of the cost function as $\partial \boldsymbol{J} / \partial \boldsymbol{d}_\text{P} = \boldsymbol{d}_\text{F}^\top R_\text{FP} + R_\text{PF} \boldsymbol{d}_\text{F} + 2 R_\text{PP} \boldsymbol{d}_\text{P} = 2 R_\text{PF} \boldsymbol{d}_\text{F} + 2 R_\text{PP} \boldsymbol{d}_\text{P}$ where both R_PF and R_FP are assumed to be transposes of each other ($R_\text{PF} = R_\text{FP}^\top$). The vector of free derivatives can be written as:

$$\boldsymbol{d}_\text{P}^* = -R_\text{PP}^{-1} R_\text{PF} \boldsymbol{d}_\text{F} = -R_\text{PP}^{-1} R_\text{FP}^\top \boldsymbol{d}_\text{F}. \tag{2.18}$$

The optimal set of design parameters can be obtained by substituting the result back into Eq. (2.15):

$$\boldsymbol{p}^* = \mathbb{A}^{-1} \mathbb{M} \begin{bmatrix} \boldsymbol{d}_\text{F}^\top, \boldsymbol{d}_\text{P}^{*\top} \end{bmatrix}^\top. \tag{2.19}$$

At this point, an optimal solution to a fixed-time unconstrained problem without any iterations has been derived. The exact same procedure is applied to both polynomial and FFS parameterizations. In fact, the only differences between the two parameterizations are reflected in the values of the mapping matrix, $\mathbb{A}$. Recall that both the quadratic mapping matrix, $\mathbb{Q}$, and linear mapping matrix, $\mathbb{A}$, are only functions of the total time allocated for each segment. In fact, there exists a simple decoupling for both. Consider a single-interval, single-axis case for simplicity. Matrices $\mathbb{Q}$ and $\mathbb{A}$ can be re-written as the following products $\mathbb{Q} = \mathbb{Q}_\mathrm{s}\bar{\mathbb{Q}}\mathbb{Q}_\mathrm{s}$ and $\mathbb{A} = \mathbb{A}_\mathrm{s}\bar{\mathbb{A}}$ where $\mathbb{Q}_\mathrm{s}$ is a square positive diagonal time-scaling matrix and $\bar{\mathbb{Q}}$ is a constant square matrix. Similarly, $\mathbb{A}_\mathrm{s}$ is a square symmetric time-scaling matrix (not diagonal) for constant square matrix $\bar{\mathbb{A}}$. In fact, $\bar{\mathbb{Q}}$ and $\bar{\mathbb{A}}$ are only defined by the cost function and the parameterization method and are invariant to constraints or time allocation and are never changed unless the order of derivatives considered is altered. The cost function given in Eq. (2.16) can be rewritten as:

$$\begin{aligned} J = \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix}^\top \mathbb{M}^\top \mathbb{A}^{-\top}\mathbb{Q}\mathbb{A}^{-1}\mathbb{M} \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix}^\top \mathbb{M}^\top \overbrace{\mathbb{A}_\mathrm{s}\mathbb{Q}_\mathrm{s}}^{\mathbb{S}} \underbrace{\bar{\mathbb{A}}^{-1}\bar{\mathbb{Q}}\bar{\mathbb{A}}^{-\top}}_{\overline{\underline{\mathbb{Q}}}} \overbrace{\mathbb{A}_\mathrm{s}\mathbb{Q}_\mathrm{s}}^{\mathbb{S}} \mathbb{M} \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix}^\top \mathbb{M}^\top \mathbb{S}\overline{\underline{\mathbb{Q}}}\mathbb{S}\mathbb{M} \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix} . \end{aligned} \tag{2.20}$$

What is intriguing is that the scaling pattern given by $\mathbb{S}(T) = \mathbb{A}_\mathrm{s}(T)\mathbb{Q}_\mathrm{s}(T)$ is independent of the parameterization method used and is only defined by the cost function, or the order of the derivative considered (minimum jerk, minimum snap, etc.). This indicates that a major part of the problem, $\overline{\underline{\mathbb{Q}}}$, remains completely invariant to time and, with additional manipulations, can lead to the following time scaled form:

$$\text{minimize } \bar{J} = \bar{\boldsymbol{p}}^\top \mathbb{Q}\bar{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix}^\top \mathbb{M}^\top \overline{\underline{\mathbb{Q}}}\mathbb{M} \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix}, \qquad \text{where,} \tag{2.21}$$

$$\bar{\boldsymbol{p}} = \mathbb{S}^{-1}\boldsymbol{p} = \mathbb{Q}_\mathrm{s}^{-1}\bar{\mathbb{A}}^{-1}\mathbb{M} \begin{bmatrix} \boldsymbol{d}_\mathrm{F} \\ \boldsymbol{d}_\mathrm{P} \end{bmatrix} . \tag{2.22}$$

The problem given in Eq. (2.21) can be solved without time factored into the solution for $\bar{\boldsymbol{p}}$ and the time-dependent coefficients can be recovered using $\mathbb{S}(T)$. However, this approach does not appear to offer any advantages over directly solving for the coefficients with time factored in. Although it may appear simpler, the original problem must still be solved to account for the new free constraints, so it presents no computational advantage. Since both $\mathbb{A}_\mathrm{s}$ and $\mathbb{Q}_\mathrm{s}$ contain elements with time raised to large and low powers simultaneously, the combined contribution of those elements (after solving for the free constraints) amplify the numerical error and final trajectory (in the time domain) may have visible discontinuities at some of the points. This effect is demonstrated with the fixed-time solutions and their time-optimal counterparts in the subsequent sections..

2.1.2 Time-allocation problem

In the previous section, it is assumed that the total time allocated for each interval is constant and, if allocation was done correctly, the optimal trajectory for that sequence of time segments can be obtained analytically. In practice, the trajectory designer might not know the exact timing for each and every waypoint of interest, and more importantly, global time for the entire trajectory can be treated as a far more important design variable rather than how that time is allocated for each individual interval. Let us assume the exact same setup as for the fixed-time problem discussed earlier, where a sequence of waypoints is given as well as intermediate boundary conditions have been specified (if any). However, there is no longer explicit time allocated for each interval, and it is up to the algorithm to find the optimal solution.

The first step is to augment the quadratic cost function in Eq. (2.2) by a linear term with a constant weight k_{T} for the sum of time allocation vector $\boldsymbol{T} = [T_1, \ldots, T_{\mathrm{n_{int}}}]^\top$:

$$J_T = \boldsymbol{p}^\top \mathbb{Q} \boldsymbol{p} + k_{\mathrm{T}} \sum_{\mathrm{i}=1}^{n_{\mathrm{int}}} T_{\mathrm{i}}, \tag{2.23}$$

where time $T = \sum_{\mathrm{i}=1}^{n_{\mathrm{int}}} T_{\mathrm{i}}$ is the global time for the entire trajectory and T_{i} is time of flight between each waypoint i to $i+1$. The time penalty k_{T} gain can be adjusted by the user to increase or decrease the global time, but is arbitrary and depends on the specific mission and requirements. To be more specific, the determination of an appropriate k_{T} value depends on several factors. If any of the dynamic requirements (maximum load factor, speed, etc.) are exceeded for a particular choice of vehicle and a set of waypoints, the k_{T} value can be adjusted accordingly to lower (or raise, if too slow) the total flight time and find a dynamically feasible trajectory. The same value of k_{T} produces a different effect for the two parameterizations, even though the solution scheme and inputs are identical. For the same boundary constraints, the FFS usually leads to a higher cost than polynomial parameterization. Therefore, k_{T} gain has to be larger for the FFS method if the user intends to match the global time of the two solutions. The problem defined in Eq. (2.23) can be solved using quasi-Newton gradient-based NLP solvers such as *MATLAB*'s `fmincon`. A set of linear inequality constraints on time allocated for each segment was considered, and the gradient numerically was computed numerically by perturbing the time allocation vector and solving the fixed-time problem. The performance of the algorithm has been tested with hand-picked waypoints, as well as with a higher-lever randomized path-planning algorithm RRT* of *MATLAB* providing a collision-free path defined by a set of waypoints. A virtual 3-dimensional grid was constructed to re-create the flying arena in ACELAB. Both simulation and experimental results are presented in Sec. 3.

2.1.3 Simulation and experimental setup

A 6DoF quadrotor simulation was developed for theoretical validation of the two methods discussed in previous chapters. The constants and parameters have been defined as in [56]. System dynamics, control system model and implementation details were also identical to those in [56]. The dynamics are standard for a 6DoF rigid-body quadrotor, with all four motors producing thrust perpendicular to the $x - y$ in a north-east-down body frame. To give some physical meaning to the numerical results presented later in the paper, the power consumption was computed as part of the post-processing analysis of the simulated quadrotor flight and included it with the rest of the numerical results. Although the minimum-snap cost given in Eq. (2.3) is directly representative of the power consumed by an ideal quadrotor, for the sake of completeness the total power consumed by the quadrotor was defined (with abuse of notation) and computed as:

$$P = \int_{t_0}^{t_f} \sum_{\mathrm{i}=1}^{4} k\omega_{\mathrm{i}}^{3} dt, \qquad k = 3.0 \times 10^{-9} N\ m/RPM^2, \tag{2.24}$$

where ω_{i} is the motor speed in revolutions per minute (RPM). The quadrotor system dynamics with the control system described in [56] was simulated in order to obtain this theoretical power consumption.

All the motion-planing algorithms and simulations have been developed in *MATLAB* and *Simulink*. For validating the motion-planning algorithms, a full-sized quadrotor was assembled and flown in the lab (Fig. 2.2). All trajectories presented in Section 3 have been computed off-board using *MATLAB* environment. The laptop used for generating all the trajectories runs an *Intel*® i7-10750H 2.60GHz CPU. Once generated, all trajectories were uploaded to the quadrotor as text files and executed using remote keyboard input from the ground computer.

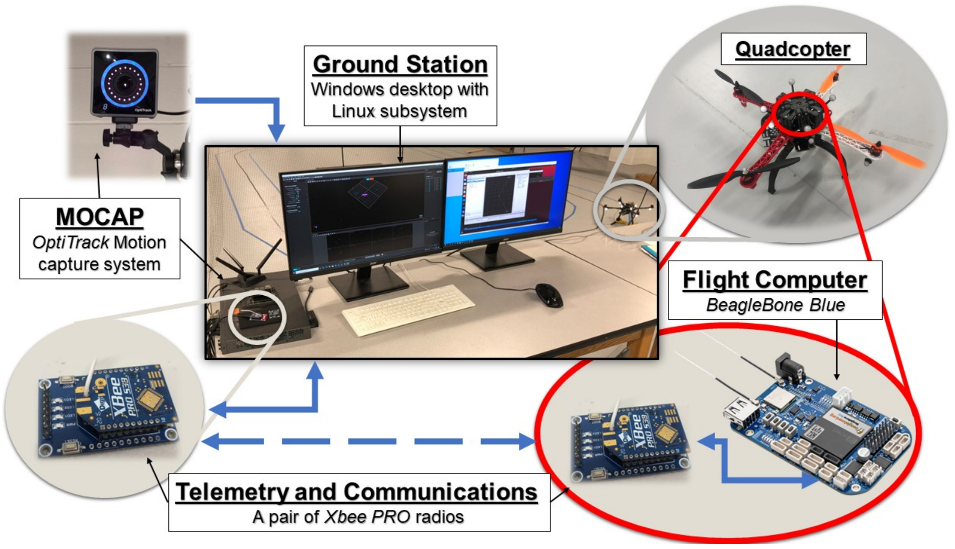

Ground Station
Windows desktop with Linux subsystem
Quadcopter
MOCAP
OptiTrack Motion capture system
Flight Computer
BeagleBone Blue
Telemetry and Communications
A pair of Xbee PRO radios

Figure 2.2: Hardware data flow schematics.

For experimental validation of the method presented in this thesis, a custom-built quadrotor with custom-written in C++ firmware (ACE-pilot) has been used. The onboard Linux computer, *BeagleBone*® *Blue*, was only used to run the flight control system and its peripherals. The communications with the ground station were handled by a pair of *Xbee Pro* radio modules, with a custom data-packet format. The position and heading of the quadrotor were not estimated on-board but sent from the ground station using an OptiTrack motion capture system. The trajectories were loaded in a point-by-point format and used to assign internal position setpoints. No feedforward control was used for this work. To simplify the trajectory generation process, all controls were assumed to be performed in a local, initial position corrected, coordinate frame. The quadrotor was to log the initial state in the inertial frame when a new trajectory was to be executed and use the incoming trajectory references as offsets to the initial position. The same strategy was applied for continuous heading reference setpoints. A simple path-following strategy was adopted. The reference position to the flight control system, $P_{\text{ref}}(t)$ was interpolated from a pre-computed optimal trajectory based on time-since-epoch, t, or the start of the trajectory. The deviation, Δ, from the intended path was defined as:

$$\Delta P_k(t) = P_{\text{ref}}(t) - P_k(t), \qquad \text{for,} \quad k \in \{x,\ y,\ z,\ \psi\}, \tag{2.25}$$

where the state of the vehicle is estimated using *Blue*'s built-in inertial-measurement unit and the motion capture system. Although there can be several enhancements made to the path-following strategy (e.g., considering the perpendicular distance from the intended path to compensate for the time lag), the development of an advanced path-following strategy is considered to be out of the scope of this work, since it will only improve the overall tracking performance. The exact same inputs for trajectory generation, path-following strategy, control system, hardware, and overall setup have been maintained for all experimental flight tests. The only difference is due to *the parametrization method* used to generate an optimal trajectory, which is converted into a set of the reference position and heading reference points to the control system.

2.2 Vehicle Dynamics

Under rigid-body dynamics assumptions, the complete state of the vehicle, $\bar{\boldsymbol{X}}$, can be expressed in terms of the position and velocity vectors of the center of mass in the inertial frame (subscript 'I'), $\boldsymbol{R}_I = [x, y, z]^\top$, $\boldsymbol{V}_I = [\dot{x}, \dot{y}, \dot{z}]^\top$, respectively. The angular velocity vector of the body frame relative to the inertial frame (when expressed in the body frame) is defined as $\boldsymbol{\Omega} = [p, q, r]^\top$. The attitude of the vehicle is expressed as a sequence of three rotations about the three orthogonal axes of the body frame $\boldsymbol{\Theta} = [\phi, \theta, \psi]^\top$. Euler angles denote rotation about the body x_B axis (with roll angle ϕ), y_B axis (with pitch angle θ) and z_B axis (with yaw angle ψ) [64]. For a definition of the coordinate frames, refer to Figure 2.3. Let the state vector be $\bar{\boldsymbol{X}}^\top = [\boldsymbol{R}_I^\top, \boldsymbol{V}_I^\top, \boldsymbol{\Theta}^\top, \boldsymbol{\Omega}^\top]$, the 6DoF dynamics is [64, 65]:

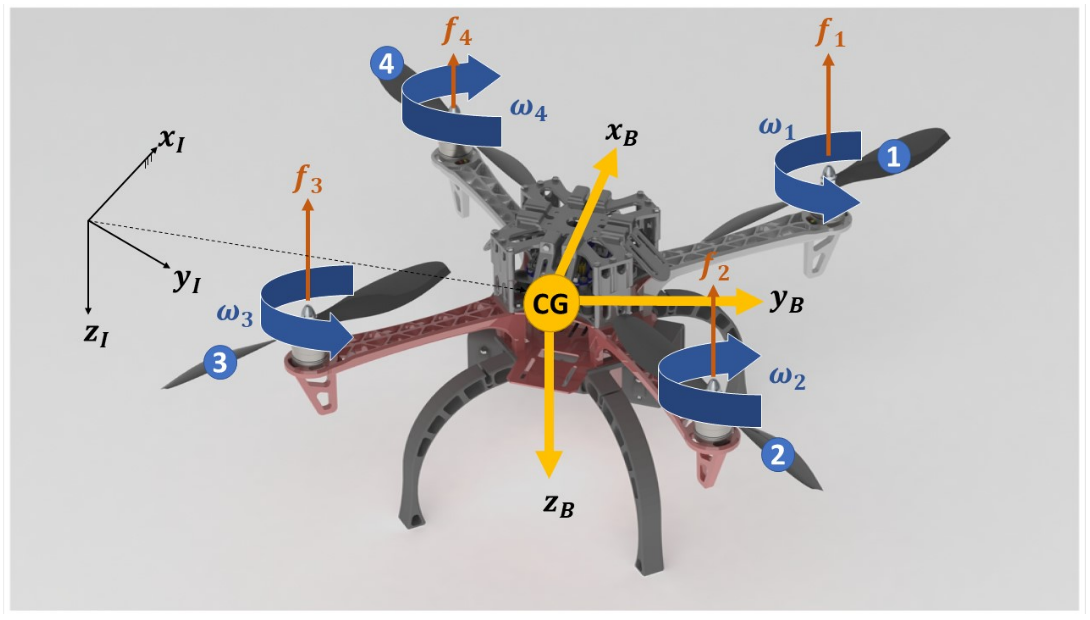


Figure 2.3: Definition of inertial and quadcopter body-fixed frames of reference. Sense of rotation and numbering convention of propellers are shown.

$$\dot{\boldsymbol{R}}_I = \boldsymbol{V}_I,\ \dot{\boldsymbol{V}}_I = \frac{1}{m}\left(\boldsymbol{F}_g + R_{\mathrm{B2I}}\boldsymbol{T}\right), \tag{2.26}$$

$$\dot{\boldsymbol{\Omega}} = I_B^{-1}\left(-\boldsymbol{\Omega}\times I_B\boldsymbol{\Omega} - \boldsymbol{M}_{\mathrm{gyro}} + \boldsymbol{M}\right), \tag{2.27}$$

$$\dot{\boldsymbol{\Theta}} = R_{\mathrm{sq}}\,\boldsymbol{\Omega}, \tag{2.28}$$

$$R_{\mathrm{sq}} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}, \tag{2.29}$$

$$R_{\mathrm{B2I}} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta c_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}, \tag{2.30}$$

where m is the constant total mass of the vehicle, g is gravitational acceleration, $\boldsymbol{F}_g = [0, 0, mg]^\top$ denotes the gravitational force, $R_{\mathrm{B2I}}(\phi, \theta, \psi)$ is the transformation matrix from the body frame to the inertial frame, $s_{(\cdot)} \equiv \sin(\cdot)$ and $c_{(\cdot)} \equiv \cos(\cdot)$. The constant moment of inertia matrix of the vehicle is denoted as I_B. In Eq. (2.27), $\boldsymbol{M} \in \mathbb{R}^3$ denotes the control torque vector produced by the propellers given in Eq. (2.33). Total thrust, $\boldsymbol{T}$, is expressed as:

$$\boldsymbol{T} = \left(\sum_{i=1}^{n} T_{\text{rotor},i}\right)\begin{bmatrix}0\\0\\-1\end{bmatrix} = b\left(\sum_{i=1}^{n}\omega_i^2\right)\begin{bmatrix}0\\0\\-1\end{bmatrix}, \tag{2.31}$$

where thrust output of each propeller, $T_{\text{rotor,i}}$, is related to the square of the motor speed ω_i by a constant coefficient b, i.e., $T_{\text{rotor,i}} = b\omega_i^2$ and that there are $n = 4$ motors (with the same performance). The total moment of inertia of the rotary part of the electric motor with the propeller attached to it is $I_{r,i}$, and the gyroscopic torque, $\boldsymbol{M}_{\text{gyro}}$, due to all n rotors is

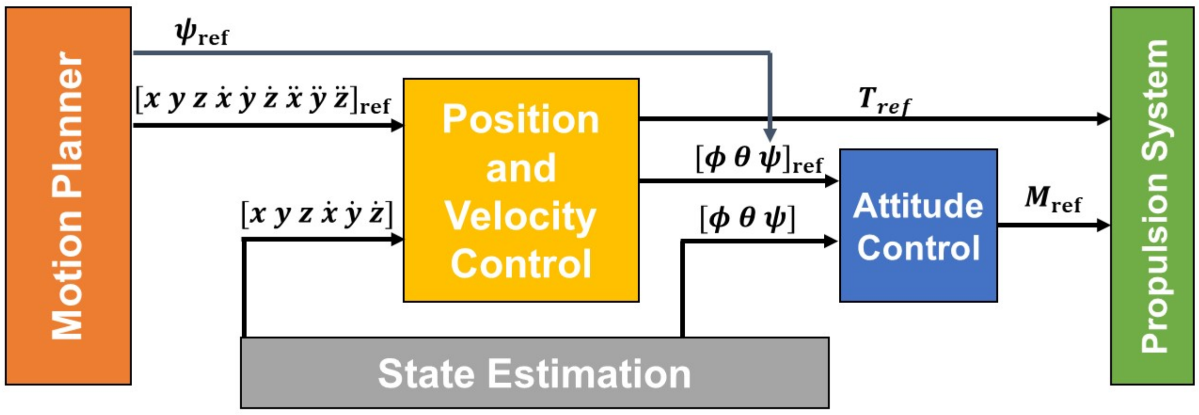
$$\boldsymbol{M}_{\text{gyro}} = \sum_{i=1}^{n}\left(\boldsymbol{\Omega}\times\begin{bmatrix}0\\0\\1\end{bmatrix}\right) I_r\omega_i(-1)^i. \tag{2.32}$$

Reactive torque due to each electric motor and its propeller is related to the square of the speed of rotation with a constant coefficient k and is expressed as $Q_i = k\omega_i^2$ ($i \in \{1,2,3,4\}$). Let h and l denote the roll and pitch moment arms, respectively. The total torque due to propellers is

$$\boldsymbol{M} = \begin{bmatrix}M_x\\M_y\\M_z\end{bmatrix} = \begin{bmatrix}bh(-\omega_1^2-\omega_2^2+\omega_3^2+\omega_4^2)\\ bl(\omega_1^2-\omega_2^2-\omega_3^2+\omega_4^2)\\ k(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2)\end{bmatrix}. \tag{2.33}$$

Combining Eqs. (2.31) and (2.33) yields a relation between the thrust and torque produced in the body frame as:

$$\begin{bmatrix}T_z\\M_x\\M_y\\M_z\end{bmatrix} = \begin{bmatrix}-b & -b & -b & -b\\ -bh & -bh & bh & bh\\ bl & -bl & -bl & bl\\ k & -k & k & -k\end{bmatrix}\begin{bmatrix}\omega_1^2\\\omega_2^2\\\omega_3^2\\\omega_4^2\end{bmatrix}. \tag{2.34}$$



Motion Planner
ψ_{ref}
$[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \ddot{x}\ \ddot{y}\ \ddot{z}]_{\text{ref}}$
$[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]$
Position and Velocity Control
T_{ref}
$[\phi\ \theta\ \psi]_{\text{ref}}$
$[\phi\ \theta\ \psi]$
Attitude Control
M_{ref}
Propulsion System
State Estimation

Figure 2.4: Cascaded position-attitude control hierarchy. The outputs of the control system are reference/requested torque and thrust commands.

2.3 Non-linear Control structure

The multirotor control systems typically consist of two primary control loops, i.e., a position-level control and an attitude-level control. Optionally, the rate of change of both translational and rotational (or velocity and Euler angle rate controllers respectively) motion can have their intermediate control layers [66]. Each control loop aims to minimize the error of its respective state $\bar{\boldsymbol{X}}_i(t)$ with the input reference $(\cdot)_{\text{ref}} \equiv \bar{\boldsymbol{X}}_i^r(t)$ provided by the human pilot, control layer above, or a higher-level motion planner. Similarly, the output of each control layer is the reference to the control layer below it in a cascaded manner all the way to the actuator speed controllers. The control system discussed in this paper is depicted on Figure 2.4. In this work, a non-linear backstepping control is used to highlight the benefits of the proposed tuning method. Originally presented by Zuo [65], the position and velocity controller outputs such as roll reference, ϕ_{ref}, pitch, θ_{ref}, and total thrust T_{ref} values are expressed with the following algebraic expressions:

$$\begin{cases} \theta_{\text{ref}} = \arctan\left(\frac{U_1 \cos\psi_{\text{ref}} + U_2 \sin\psi_{\text{ref}})}{U_3+g}\right), \\ \phi_{\text{ref}} = \arcsin\left(\frac{U_1 \sin\psi_{\text{ref}} - U_2 \cos\psi_{\text{ref}}}{\sqrt{U_1^2+U_2^2+(U_3+g)^2}}\right), \\ T_{\text{ref}} = m[U_1(s_\theta c_\psi c_\phi + s_\psi s_\phi) \\ \qquad +U_2(s_\theta s_\psi c_\phi - c_\psi s_\phi) + (U_3+g)c_\theta c_\phi], \end{cases} \tag{2.35}$$

where a virtual acceleration, $\bar{\boldsymbol{U}} = [U_1, U_2, U_3]^\top$, is obtained

$$\bar{\boldsymbol{U}} = \ddot{\boldsymbol{R}}_{I_{\text{ref}}} + \mathbb{K}_p\,\Delta\boldsymbol{R} + \mathbb{K}_d\,\Delta\boldsymbol{V} + \mathbb{K}_i \int_{t_0}^{t_f} \Delta\boldsymbol{R}dt, \tag{2.36}$$

where $\Delta\boldsymbol{R} = \boldsymbol{R}_{I_{\text{ref}}} - \boldsymbol{R}_I$, $\Delta\boldsymbol{V} = \boldsymbol{V}_{I_{\text{ref}}} - \boldsymbol{V}_I$. Three positive definite proportional, derivative, and integral diagonal matrices are $\mathbb{K}_p = \text{diag}\{K_{p_1}, K_{p_2}, K_{p_3}\}$, $\mathbb{K}_d = \text{diag}\{K_{d_1}, K_{d_2}, K_{d_3}\}$, and $\mathbb{K}_i = \text{diag}\{K_{i_1}, K_{i_2}, K_{i_3}\}$, respectively. The $\text{diag}\{(\cdot),(\cdot),(\cdot)\}$ command constructs a diagonal matrix with the listed arguments. Attitude controller outputs, i.e., the requested torque, $\boldsymbol{\tau}_{\text{ref}} = \boldsymbol{M}$, can be derived as [65]:

$$\begin{aligned} \boldsymbol{\tau}_{\text{ref}} = (\boldsymbol{\Omega} \times I_B\boldsymbol{\Omega}) + \boldsymbol{M}_{\text{gyro}} - I_B\bar{T}\boldsymbol{\Omega}_{\text{ref}} + I_B\bar{T}\boldsymbol{R}_{\text{sq}}^{-1}X_2 \\ -I_B\bar{T}\boldsymbol{R}_{\text{sq}}^{-1}\Gamma_1(\boldsymbol{\Theta} - \boldsymbol{\Theta}_{\text{ref}}) - I_B\boldsymbol{R}_{\text{sq}}^{-1}(\boldsymbol{\Theta} - \boldsymbol{\Theta}_{\text{ref}} - \boldsymbol{\epsilon}) \\ -I_B\Gamma_2(\boldsymbol{\Omega} - \boldsymbol{\Omega}_{\text{ref}}), \end{aligned}$$

which requires propagation of additional states such as:

$$\begin{cases} \dot{\boldsymbol{X}}_1 &= \boldsymbol{X}_2, \\ \dot{\boldsymbol{X}}_2 &= \Lambda^2(\boldsymbol{\Theta}_{\text{ref}} - \boldsymbol{X}_1) - 2\Lambda\boldsymbol{X}_1, \\ \dot{\boldsymbol{\Omega}}_{\text{ref}} &= -\bar{O}(\boldsymbol{\Omega}_{\text{ref}} - \boldsymbol{\Omega}_d), \\ \dot{\boldsymbol{\epsilon}} &= -\Gamma_1\boldsymbol{\epsilon} + \boldsymbol{R}_{\text{sq}}(\boldsymbol{\Omega}_{\text{ref}} - \boldsymbol{\Omega}_d), \end{cases} \tag{2.37}$$

where $\boldsymbol{\Theta}_{\text{ref}} = [\phi_{\text{ref}},\ \theta_{\text{ref}},\ \psi_{\text{ref}}]^\top$, $\boldsymbol{\epsilon}$ is the tracking error compensation filter, Λ, $\bar{O}$, Γ_1, and Γ_2 are positive definite diagonal matrices (each with three gains) to be tuned and $\boldsymbol{\Omega}_d$ is the desired angular velocity:

$$\boldsymbol{\Omega}_d = [p, q, r]^\top_{\text{ref}} = \boldsymbol{R}_{\text{sq}}^{-1}\left(\dot{\boldsymbol{\Theta}}_{\text{ref}} + \Gamma_1(\boldsymbol{\Theta}_{\text{ref}} - \boldsymbol{\Theta})\right). \tag{2.38}$$

2.4 Formulation of the Control System Optimization Problem

The primary goal of this gain-tuning method is to mathematically quantify an appropriate vehicle performance in terms of flight data. The control system would rely on the states of the vehicle and a certain reference provided by the higher-level path planner or human pilot. Intuitively, the human observer evaluates the performance based on visual cues such as overall attitude stability, smoothness of motion, and deviation from the intended path/reference. Visual measures such as vehicle "twitching" and "shaking" are not suitable for an automated tuning algorithm. But once the notion of "good performance" exists as a certain cost to be minimized by adjusting the control system gains, optimization routines such as `fmincon` [67] can be used to obtain a combination of control system gains resulting in an enhanced performance over the entire flight path or certain time intervals. The performance of a controller can be characterized in terms of state tracking, control effort, stability, etc. Recall that $\bar{\boldsymbol{X}} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, p, q, r, \phi, \theta, \psi]^\top$. A state tracking cost function, J_{track}, can be defined as:

$$J_{\text{track}} = \sum_{i=1}^{12} \mathbb{C}_{tr_{i,1}} J_i,\ J_i = \int_{t_0}^{t_f} \left(e^{\mathbb{C}_{tr_{i,2}}|\bar{\boldsymbol{X}}_i^r - \bar{\boldsymbol{X}}_i|} - 1\right) dt, \tag{2.39}$$

where J_i denotes the cumulative error cost associated with the i-th state (and subscript is used to denote the i-th element) and $|(\cdot)|$ denotes the absolute value. Superscript 'r' is used to denote the reference state. A weighting matrix, $\mathbb{C}_{tr}$, with a dimension of, 12×2 contains a set of gains used to assign priority to tracking certain states.

An ideal control system should be able to achieve perfect tracking of the commanded reference or can achieve near-zero error tracking cost given in (2.39) for any trajectory. In practice, however, such a perfect control is generally infeasible. Since reference tracking is the only piece of information given to the optimization routine, the resulting combination of control system gains is not only extremely hard to find, if possible at all, but also may be physically unrealistic due to an extreme control effort, sensitivity and system requirements for realizing such performance. Moreover, there is no guarantee that such a control system would be stable at all since any minor off-nominal disturbance can result in highly unpredictable and chaotic behavior.

One approach to quantify control effort is to consider the resulting speeds of all the electric motors and/or thrust produced by each motor as:

$$J_{\text{RPM}} = c_{\text{RPM}} \sum_{i=1}^{n} \int_{t_0}^{t_f} \left[\omega_i(t) \cdot \frac{30}{\pi} \right]^2 dt, \tag{2.40}$$

where J_{RPM} is the total cost of all n motors on the vehicle, c_{RPM} is a constant gain used to assign priority to this cost. However, Eq. (2.40) does not distinguish between individual control layers and control degrees. Moreover, motor speed might not be as easily obtainable for a physical vehicle, therefore an additional control effort cost was considered as:

$$J_{\text{CTRL}} = \sum_{i=1}^{n_d} \int_{t_0}^{t_f} C_{\text{CTRL}_i} |\{T_{\text{ref}}(t), \boldsymbol{\tau}_{\text{ref}}(t)\}| dt, \tag{2.41}$$

where thrust and torque reference outputs from the control system are directly minimized. It is assumed that the references from the control system can be fully realized by the propulsion system, such that $\{T_{\text{ref}}(t), \boldsymbol{\tau}_{\text{ref}}(t)\} = \{T(t), \boldsymbol{\tau}(t)\}$.

Motor speed cost (2.40) provides a general way to minimize the combined effect of the entire control system, path planner, intermediate motor mixing, and control signal-to-motor signal conversions. By directly minimizing the control system outputs given by (2.41), a way of prioritizing individual degrees of freedom of the vehicle provides more control over the optimization process and can be treated as an overall energy or total power consumption minimization problem. However, such a high-level approach may not be sufficient, since none of the above has considered the actual control parameters and intermediate input and output relations. Even for the most agile vehicles, the multi-copter's attitude is expected to remain close to level for the majority of its flight, unless the trajectory consists of long segments with strong wind or trajectories involving cruise flight at a relatively constant tilt angle. Therefore, any reference command generated either by pure position or position-velocity control loop is expected to be small for the majority of flights. Similarly, an average reference for an Euler angle rate control loop should be as small as possible. Therefore, attitude and rate deviations from hover should carry a certain penalty on their own. All of these requirements can be enforced using an agility cost as

$$\begin{gathered} J_{\text{agility}} = \sum_{i=1}^{12} \left[\mathbb{C}_{\text{agility}_{i,1}} J_{i,1} + \mathbb{C}_{\text{agility}_{i,2}} J_{i,2} \right], \text{ with} \\ J_{i,1} = \int_{t_0}^{t_f} \left| \bar{\boldsymbol{X}}_i^r(t) \right| dt, \text{ and } J_{i,2} = \int_{t_0}^{t_f} \left| \bar{\boldsymbol{X}}_i(t) \right| dt. \end{gathered} \tag{2.42}$$

Here, the entries associated with the yaw angle and translational states (both the states and references) cost should be excluded by zeroing out their respective entries in the cost allocation matrix, $\mathbb{C}_{\text{agility}}$. Observe that the underlying assumptions do not require a certain yaw angle to hold true for maintaining stable flight, in fact, the reference yaw angle is entirely defined by the mission designer and may remain at a fixed non-zero value for almost any trajectory. Unbounded inputs and outputs that may remain far from zero by design (such as position states and references), should be excluded from the cost (by setting their respective $\mathbb{C}_{\text{agility}}$ gains to zero). For certain applications, a cost penalty on translational velocity and the rate of change of attitude can be considered to further shape the desired cost function and enable greater control within the optimization process.

Stability is an important characteristic of any control system. Assuming close-to-ideal state information is available, such that sensor noise and other sources of non-physical behaviors are not present or have been filtered out from the flight profile, any high-frequency oscillatory behavior can be treated as undesirable. In reality, the majority of the vehicles would not be required to perform maneuvers that require rapid changes of the flight states faster than 0.5 to 4 Hz (depending on the vehicle's agility, the specific state considered, and maneuver required to perform). An attempt to rapidly change any of the physical flight states faster than 7 to 10 Hz can be visually described as "shaking", which is comparable to low-frequency vibrations of the vehicle's frame itself. Therefore, a certain cutoff frequency can be considered to filter normal (or expected) vehicle behavior out from undesirable oscillatory behavior induced by the control system. As such, a high-pass filter can be used for each of the states of the vehicle to discard all the desirable behavior and leave the oscillatory profile along the trajectory, $\bar{\boldsymbol{X}}_i^{hp}(t)$. The associated stability cost J_{stab} can then be obtained by integrating the profile similarly to how it was done before:

$$J_{\text{stab}} = \sum_{i=1}^{12} \mathbb{C}_{\text{stab}_{i,1}} J_i, \;\; J_i = \int_{t_0}^{t_f} \left(e^{\mathbb{C}_{\text{stab}_{i,2}} |\bar{\boldsymbol{X}}_i^{hp}(t)|} - 1 \right) dt,$$

Since the equation above can be treated as a measure of instability, the associated cost gains within cost allocation matrix, $\mathbb{C}_{\text{stab}}$, should be large enough to aggressively penalize any undesirable behavior.

At this point, all the necessary building blocks of the total cost function have been introduced. However, it is not sufficient to add all the individual costs together because of their scale relative to each other. For example, the control effort cost obtained from the motor speed is disproportionately high and has to be weighted substantially lower than all other components discussed above such that it only becomes significant if there is severe and prolonged overactuation. The total cost function for the optimization, which is a function of flight states $\bar{\boldsymbol{X}}(t)$ and state references, $\bar{\boldsymbol{X}}^r(t)$ can be expressed as a sum of all the above costs:

$$J_{\text{total}} = J_{\text{track}} + J_{\text{stab}} + J_{\text{agility}} + J_{\text{RPM}} + J_{\text{CTRL}}. \tag{2.43}$$

This cost is decreased by optimizing over a set of adjustable parameters specific to each of the control structures. For the specific controller structure described in section 2.3, the only variable inputs to the optimization routine are the controller gains $\mathbb{K}_p$, $\mathbb{K}_i$, $\mathbb{K}_d$, Λ, $\bar{O}$, Γ_1 and Γ_2, with a total of 21 parameters (three for each), while all other parameters remain constant (including the reference trajectory, vehicle parameters, wind magnitude, and relative direction, etc.) The constant gains of each and every channel, $\mathbb{C}_{(\cdot)}$, are used to highlight the importance of proper scaling: 1) stability is the primary concern. If frequency analysis was done properly, any cost associated with the oscillatory behavior, J_{stab}, has to be the driving factor of the optimization and be prioritized the most. 2) error tracking in xy plane is highly important, but its relative value is significantly lower than all the other terms, so it has to be increased accordingly. Since altitude tracking can be done directly by increasing thrust, and control effort is evaluated separately, changes to altitude should have a relatively low cost. 3) as mentioned earlier, cost associated with motor speed should not dominate and remain very small such that it is only needed for "fine-tuning" because the majority of the control effort is already distributed to individual cost functions. To simplify the initial test runs, the last two terms in equation (2.43) can be dropped, that is, the total control effort, J_{CTRL}, and power consumption, J_{RPM}. 4) certain desired effects have an overlap and can be achieved by adjusting various parts of the control system. 5) since all the analysis is purely numerical, all the flight data should be scaled down by a large factor (1.0×10^{-6} or even smaller for SI units) such that the exponential terms do not explode.

2.5 Practical Considerations For Gain-Optimization

Compiling the simulation using built-in MATLAB tools (i.e., *rapid accelerator* mode) can decrease each simulation run time from several minutes down to just a few seconds. By taking advantage of the parallel computing capabilities, the overall optimization run time can reduce to few hours.

The pseudo-algorithm for automated gain tuning can be described as such: 1) Formulate the cost function and adjust the priority given to each of the cost parameters discussed in section 2.4. 2) Define a desired trajectory, run the simulation, and determine the critical time after each control system fails. 3) Truncate the simulation time such that only a few seconds or less of unstable flight are present (in other words, if starting with a set of unstable gains, the simulation has to be terminated before the simulation crashes, even if it means simulation for less than a second). 4) Run the optimization routine, and inspect the optimized performance. Major issues and instabilities should be drastically decreased, and the system should appear to be stable for the short time considered during optimization. If still unstable with no sign of improvement, revisit previous steps until the short-term performance has improved substantially. 5) Increase the simulation time until the simulated flight becomes unstable again or till the end of the trajectory. Note, for short flight (less than 10 seconds) minor control system instabilities may not have enough time to aggregate, and the resulting system might still be unstable long term. 6) Repeat the optimization process (all the previous steps) until satisfactory performance. The cost function defines the notion of what is considered a high-performance system and should not require major modification (if at all) after all the steps have been completed for a single trajectory. After the satisfactory performance has been defined, trajectory and flight conditions can be changed, and the optimization routine has to be repeated without modification to the cost function (unless the requirement for the flight performance has changed). The class of desired maneuvers required from the control system is important: if some of the control axes (e.g., yaw) are unused and no disturbances require significant yaw stabilization, the optimization routine will not tune the respective control axis. Similarly, the optimization will find and exploit any idealization and simplification such as ideal actuation (with no time delay), simplified propulsion and aerodynamics model (no complex aerodynamic interactions between the propellers and vehicle drag) or no disturbances such as wind, which require the control system to sacrifice tracking precision for a certain degree of robustness. Due to this, it is often required to rerun the gain-tuning routine if the reference trajectory has changed.

Chapter 3

3.1 Numerical Results

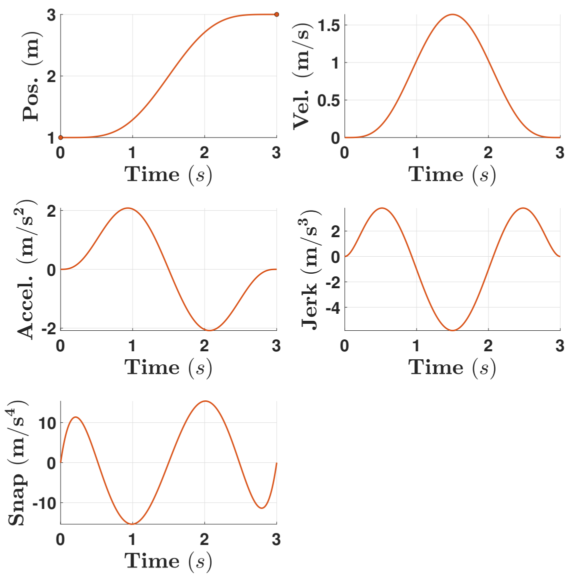
3.1.1 Fixed-time solutions

Let's start by examining fixed-time solutions with an even time allocation. For the sake of simplicity, all trajectories discussed in this section are summarized in Table 3.1, and they are characterized in terms of the following parameters: a) n_{dt}, which denotes the order of derivative considered for boundary conditions. Since the goal is to compute minimum-snap trajectories, $n_{\text{dt}} = 4$ for both the cost function and the boundary conditions. This also sets the number of coefficients per interval to be ten; b) n_{dim}, which denotes the number of dimensions for which motion planning is performed. This number defines how many active degrees of freedom are used for optimization. Even though all trajectories are considered in 3D space, some of them consider constant altitude or only planar XY motion. Any set of waypoints that is zero or constant for a degree of freedom can be considered to always lead to a trivial solution and is removed from the optimization to reduce the computational time; c) n_{int}, which denotes the number of intervals/segments between two waypoints for a single dimension. It is assumed that all dimensions have the same waypoint discretization and no optimization is performed to determine the optimal number of waypoints or intervals; d) T, which denotes the total time allocated for the trajectory. Not to be mistaken for with the vector of individual time segments $\boldsymbol{T}$ allocated for each interval. This defines the total time it physically takes to execute or fly the trajectory. For fixed-time problems, this quantity is decided by the trajectory designer and is fixed; e) T_{solve}, which denotes the time it takes to compute a solution. This quantity defines how long it took to solve a complete problem. This number does not account for initial user-input setup, solution evaluation or plotting; f) J, which denotes the final value of the minimum-snap cost function. Trajectory design requirements are generally widely different based on the application, even for vehicles of the same class.

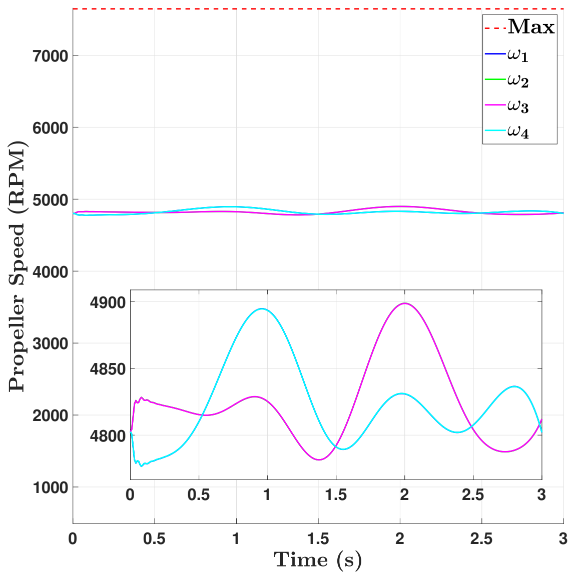
3.1.1.1 "Simple" trajectory.

This trajectory considers motion along a single axis and only between two waypoints. It serves as a baseline for computational speed comparison and control system performance for the different parameterizations. This is considered the simplest case because it can be analytically derived and solved as explained in Sec. 2.1.1.3 with mapping matrices being $\mathbb{A}(\boldsymbol{T}) = \mathbb{A}_1(T)$ and $\mathbb{Q}(\boldsymbol{T}) = \mathbb{Q}_1(T)$, where there is only one segment, so the time allocation vector is a scalar with only one element $\boldsymbol{T} = T = T_1$. The mapping matrix $\mathbb{M}$ is simply the diagonal matrix, since no reordering or duplication of intermediate constraints is needed. The input waypoints are the start and end points at one and three meters, respectively. All other axes are held constant at their initialization values, which, in the case of the simulation, are all zeros. Trajectory optimization is performed on the X axis. First, the trajectory is solved outside the simulator and the required position, velocity, acceleration, jerk, and snap are evaluated using the optimal coefficients.

As can be seen in Figures 3.1a and 3.2a, the path and its derivatives are identical. The boundary-value problem is actually almost trivial in nature due to the assumptions made earlier such that the derivatives at the start and end points of the trajectory are all zeros and only the initial and final position, in this case, are non-zero. The solution process is simple enough to be checked by hand and such a one-dimensional, single-interval example is quite common to be applied in practice, where a simple connecting arc between just two points is required, for example, for takeoff or landing. Although it is more common to use a cubic polynomial, the 9th-degree polynomial and similar complexity FFS solutions lead to almost identical solutions. The difference is, of course, in that the cubic polynomial is not continuous up to the snap level, and a sharp change in motor speed is expected at the beginning and the end of the cubic-polynomial solution. The resulting motor RPM profiles serve as the final arbiter for the performance of

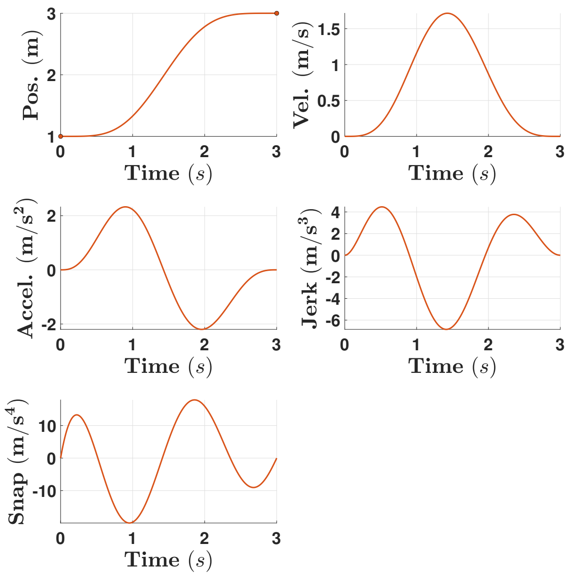


(a) Generated position and higher-order derivative references.

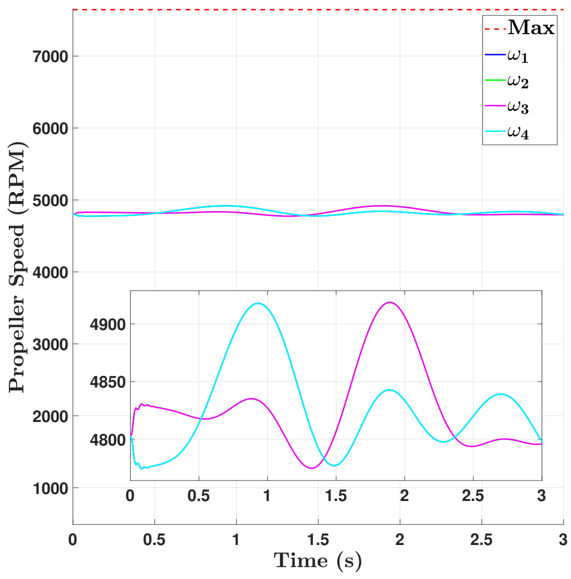


(b) RPM of propellers vs. time.

Figure 3.1: Simulated results for the “Simple” trajectory with polynomial parameterization.



(a) Generated position and higher-order derivative references.



(b) RPM of propellers vs. time.

Figure 3.2: Simulated results for the “Simple” trajectory with FFS parameterization.

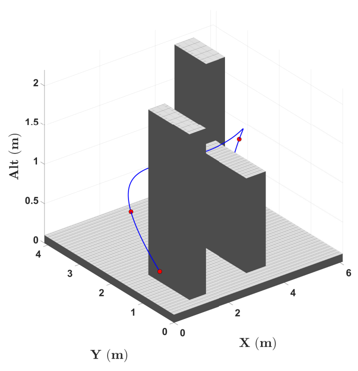
the control system and the trajectory generation algorithm. For this reason, control system plots have been omitted, but all four motor profiles, labeled with different colors, are shown in Figures 3.1b and 3.2b. The sense of rotation of the propellers is given in Fig. 2.1. As expected, motor profiles are identical for the two parameterizations and lead to the almost equivalent power consumed (over the entire trajectory) of $425.6W$ for polynomial and $426.36W$

for FFS. The values of the minimum-snap cost functions are 301.68 and 400.04 for polynomial and FFS parameterizations. Figures 3.1b and 3.2b contain the global plot in the background, where motor profiles are plotted with colored lines on the same scale as their maximum values (dashed red line). Since almost all trajectories do not show significant deviation of the motor profile along the trajectory, a zoomed-in view of the same motor profiles is included on each of the plots. The vertical and horizontal axes are identical for both the background graph and a zoomed-in view. Due to the simple nature of this one-dimensional (1D) trajectory, the four motor profiles are not fully distinguishable, and only overlaid pairs are visible.

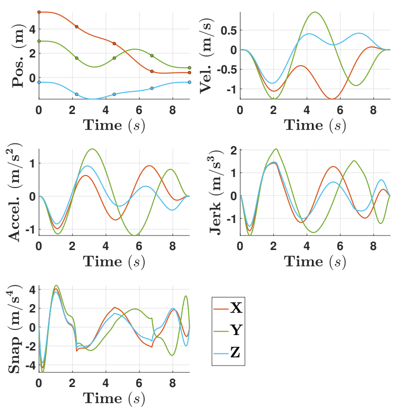
3.1.1.2 "3 Blocks" trajectory.

This trajectory was designed as an extension of the previous simple 1D case, where an open-contour set of points between the start and end goal is required to be joined by a feasible trajectory. First, a 3D grid with a resolution of one decimeter has been constructed to reproduce the flyable lab space in a virtual environment. The objective was to guide the quadrotor from one corner of the lab to the other. To complicate the path-planning task, three static rectangular obstacles have been placed in between the starting point and the destination. Next, the random-search algorithm RRT^* of *MATLAB* has been used to generate a feasible path connecting the start and end points in straight lines. Since there was no collision-free straight line path between the start and goal pose, the algorithm had to introduce intermediate waypoints in between to avoid any collision. The resulting set of waypoints was then used to compute the optimal trajectory using the algorithm derived earlier. This time, the motion was in 3D space with variable waypoints in X, Y, and Z axes. The heading angle was not considered for this problem, so it was assumed to be held constant. The input waypoints are marked with red dots on 3D plots shown in Figures 3.3a and 3.4a . Note that for display purposes, the trajectory generated in NED coordinate frame with its Z axis pointing down, was flipped to represent altitude or $Alt = -Z$.

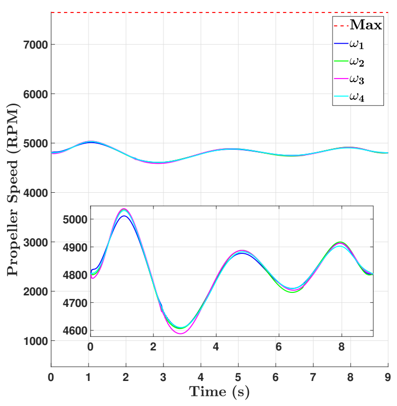
The resulting path for polynomial (Fig. 3.3a) and FFS (Fig. 3.4a) parameterizations is identical to within the numerical precision. The shape does not look distorted for FFS parameterization and fully follows the polynomial solution. A very small, almost unnoticeable



(a) 3D view of the path. Color was removed for clarity.

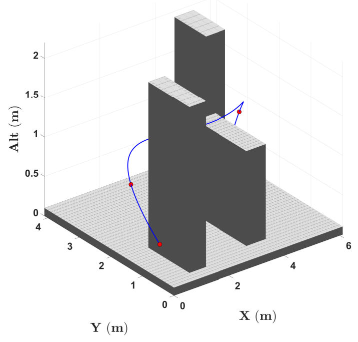


(b) Generated position and higher-order derivative references.

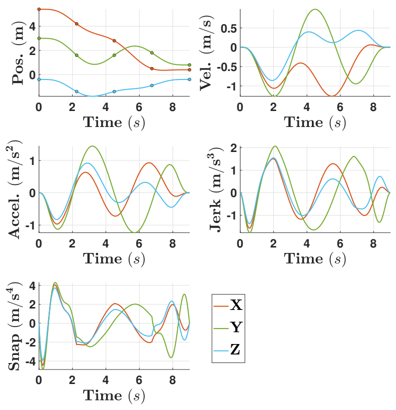


(c) RPM of propellers vs. time for simulated flight.

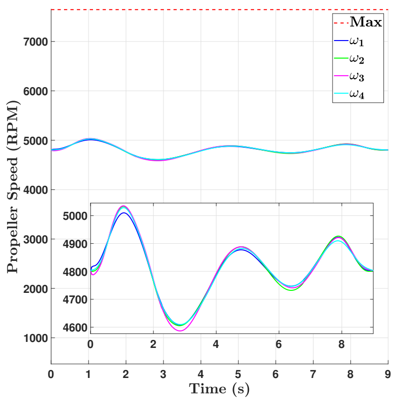
Figure 3.3: Simulated results for the “3 Blocks” trajectory with polynomial parameterization.



(a) 3D view of the path. Color was removed for clarity.



(b) Generated position and higher-order derivative references.



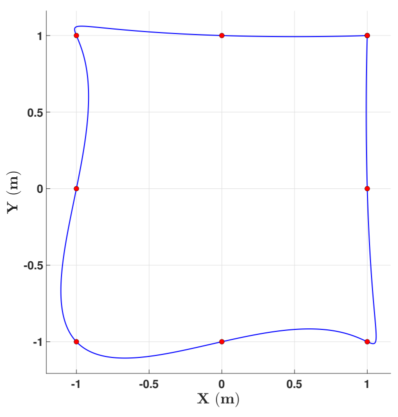
(c) RPM of propellers vs. time for simulated flight.

Figure 3.4: Simulated results for the “3 Blocks” trajectory with FFS parameterization.

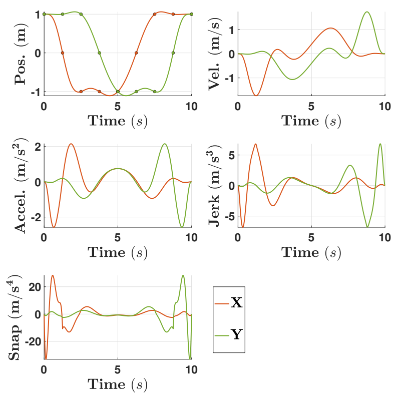
difference is present on the snap level between the two solutions (Fig. 3.3b and Fig. 3.4b). The total power consumption is $1269.97W$ and $1270.27W$ for polynomial and FFS parameterizations, respectively. The minimum-snap cost values are 90.07 for polynomials and 93.85 for FFS methods, which reemphasizes that the cost function values do not directly translate to power consumption. The only general trend is that the cost value and total power consumption, both, are higher for the FFS parameterization, which is expected since according to the principles of calculus of variations, the family of extremal solutions is represented by polynomials. But, the objective of this thesis is to present an alternative smooth trajectory generation method. Even though this trajectory requires the quadrotor to take off, fly over the obstacle and dodge the other two, plenty of time has been given for the maneuver. This results in a very conservative change in motor profiles, as can be seen in Figures 3.3c and 3.4c. A zoomed-in view of the same figures shows a slight variation in all four motor profiles. Although almost identical, motor profiles corresponding to the polynomial solution show a very minor twitch at approximately 2.2 seconds, which is not present in the FFS solution. The two solutions are smooth and FFS again closely matches with the solution of the polynomial parameterization.

3.1.1.3 “Square” trajectory.

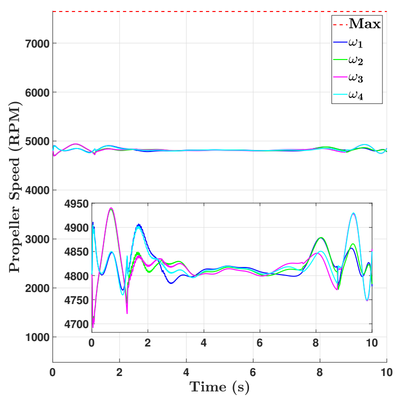
Next, consider a somewhat more standard trajectory. The waypoints for this trajectory define the four corners of a square. Additional waypoints are placed midway between each of the corners to constrain the shape and enforce a motion close to straight lines. This square path is completely in the horizontal $X - Y$ plane, with time being evenly discretized between all the waypoints. Altitude and heading are assumed to remain constant. This trajectory aims to decouple the motion in the X and Y directions and forces the optimal solution to approximate straight lines and corners. For this closed trajectory, the start and end waypoints are at the same location, $X = 1.0$ m and $Y = 1.0$ m, which also defines the first corner of a 2×2 meter square. The motion is counterclockwise.



(a) 2D view of the path.



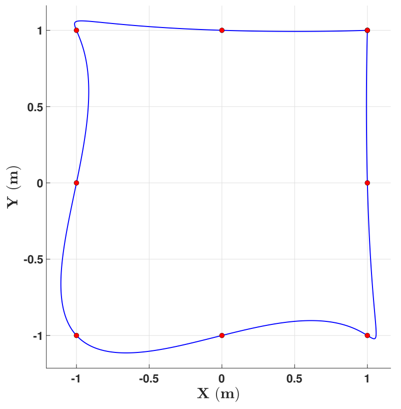
(b) Generated position and higher-order derivative references.



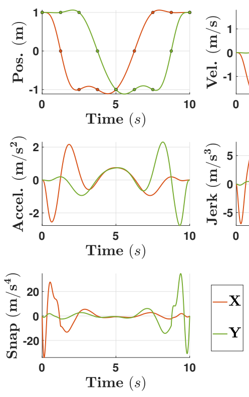
(c) RPM of propellers vs. time for simulated flight.

Figure 3.5: Simulated results for the “Square” trajectory with polynomial parameterization.

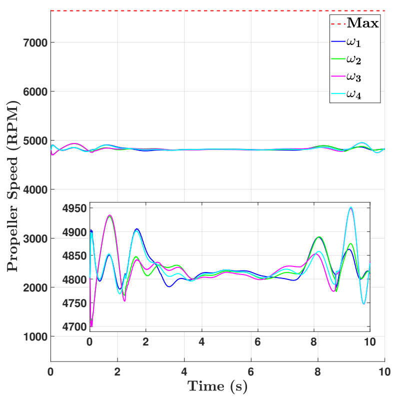
Similarly to the test cases shown earlier, the resulting trajectories are identical, within some numerical error, for the two parameterizations. As can be seen on 2D plots in Figures 3.5a



(a) 2D view of the path.



(b) Generated position and higher-order derivative references.



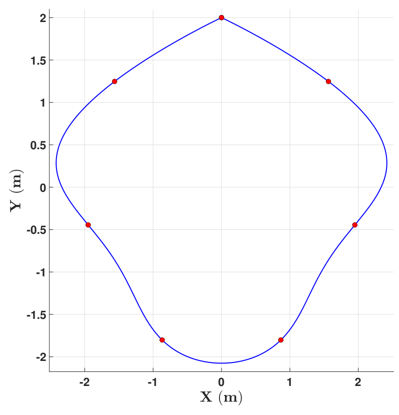
(c) RPM of propellers vs. time for simulated flight.

Figure 3.6: Simulated results for the "Square" trajectory with FFS parameterization.

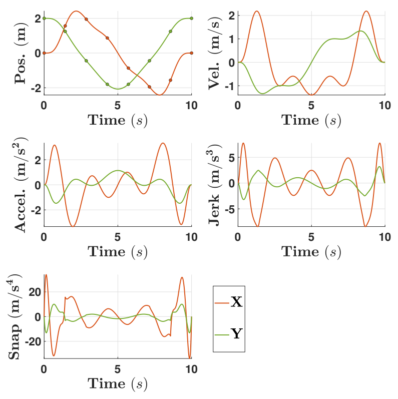
and 3.6a, the differences in the two shapes are indistinguishable. However, some very minor differences are seen between the snap profiles shown in Figures 3.5b and 3.6b at about 1.2 and 8.5 seconds. Although very similar, the FFS solution might appear to be somewhat smoother because it lacks the same small spikes on the snap level of the polynomial solution. This very minor fluctuation (at 1.2 seconds) is also reflected in the profiles of the motors (Fig. 3.5c and 3.6c). Of course, the effect is minimal and is most likely purely attributed to the numerical convergence of the two methods. As is shown later, similar numerical spikes in the snap are also present for some of the FFS solutions. The power consumption (although negligible) is still in favor of the polynomial solution that obtained a value of $1417.68W$ while FFS solution lead to $1418.52W$ of power consumed.

3.1.1.4 "Circle" trajectory.

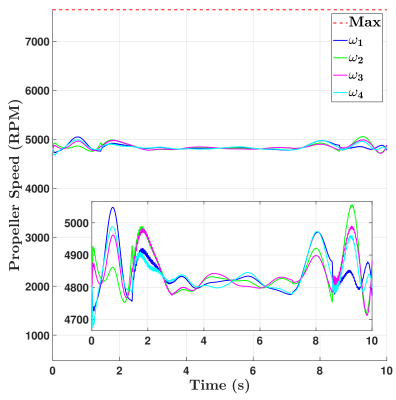
Even though it may not look circular at all from the fixed-time paths shown in Figures 3.7a and 3.8a, the waypoints are tracing a perfect circular shape. This 2D trajectory is very similar to the "Square" one, with time allocated evenly for $t_f = 10$ seconds. The intent is again to obtain a simple circular path which, intuitively, should be the optimal shape for a set of waypoints that already trace a circle of a constant radius. The results in Figures 3.7 and 3.8 show that both solutions are closely matching each other except for small differences on the snap level. These differences are then further reflected in the motor profiles shown in Figures 3.7c and



(a) 2D view of the path.



(b) Generated position and higher-order derivative references.



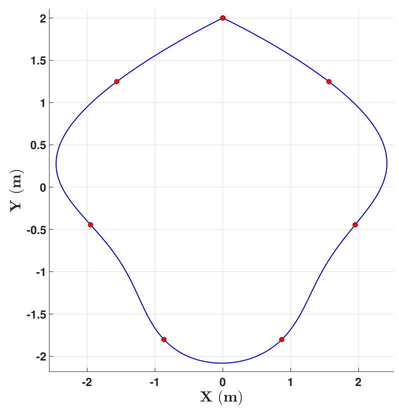
(c) RPM of propellers vs. time for simulated flight.

Figure 3.7: Simulated results for the "Circle" trajectory with polynomial parameterization.

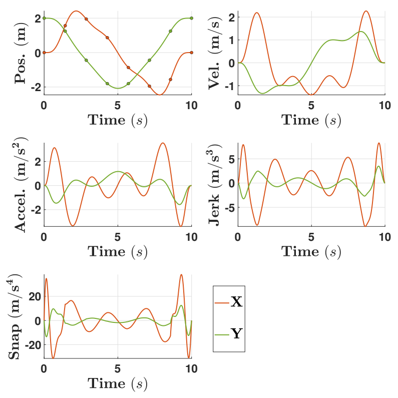
3.8c. Minor numerical noise is visible across the polynomial trajectory, but is missing for the FFS. The irregular shape of the two solutions in Figures 3.7a and 3.8a is due to the evenly discretized time, and can be fixed when the time-allocation problem is solved.

3.1.1.5 "Figure-8" trajectory.

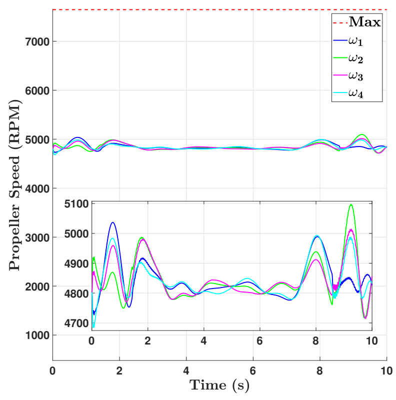
This trajectory traces an "8"-like figure in an inclined plane. The plane itself is tilted such that the altitude is not constant. The heading angle is also set to approximately trace the center of the figure. This way, all four dimensions are active and require trajectory optimization. This high dimensionality and complex 3D shape is intended to stress both the trajectory generation



(a) 2D view of the path.



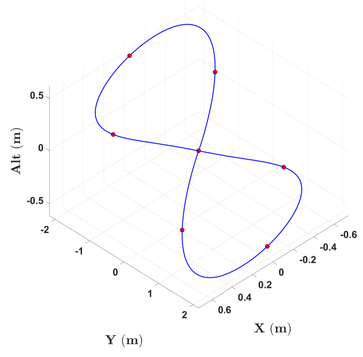
(b) Generated position and higher-order derivative references.

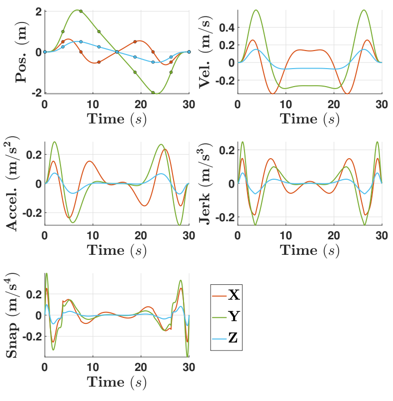


(c) RPM of propellers vs. time for simulated flight.

Figure 3.8: Simulated results for the "Circle" trajectory with FFS parameterization.

methods and the control system. Due to the inherent symmetry of the trajectory, even discretization (for $t_f = 30$ seconds) is close to optimal, and this feature was exploited to verify time allocation.


(a) 3D view of the path.


(b) Generated position and higher-order derivative references.

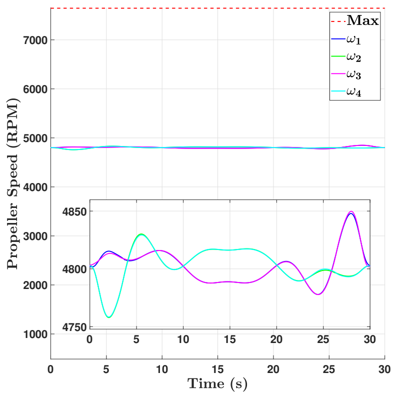
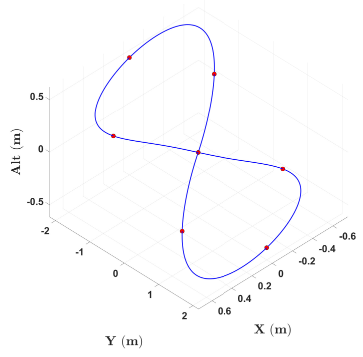
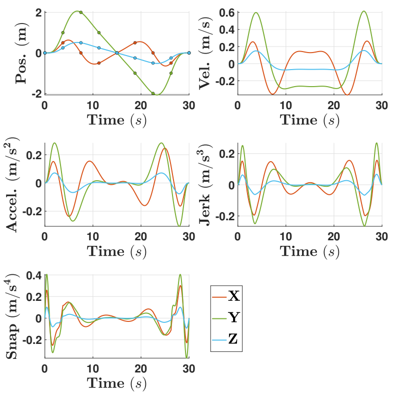

(c) RPM of propellers vs. time for simulated flight.

Figure 3.9: Simulated results for the “Figure-8” trajectory with polynomial parameterization.


(a) 3D view of the path.


(b) Generated position and higher-order derivative references.

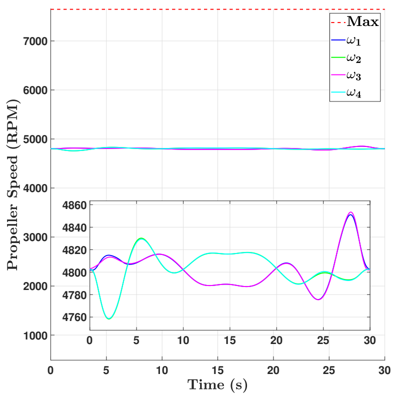

(c) RPM of propellers vs. time for simulated flight.

Figure 3.10: Simulated results for the “Figure-8” trajectory with FFS parameterization.

Even for this relatively complicated motion, the two parameterizations are closely matching (Fig. 3.9 and Fig. 3.10). The total time allocated for this fixed-time solution was 30 seconds, which results in almost flat motor profiles for the two solutions. No noticeable differences are present either in shape, path components, derivatives or motor profiles.

3.1.1.6 Summary of the fixed-time trajectories.

All fixed-time solution metrics are summarized in Table 3.1. The solutions for the two different parameterizations are closely following each other and are mostly identical. As was expected, computational times are on the same order of magnitude but are slower for the FFS parameterization. The power consumption is primarily dominated by the total time of flight since most trajectories require only minor changes in motor speeds around the hover thrust. Although the difference is very small, FFS leads to a slightly higher power consumption for all the trajectories considered. The minimum-snap cost values are following the same trends as power consummations, but the relative scale is unrelated.

Table 3.1: Summary of the fixed-time solutions.

Name	Method	n_{dt}	n_{dim}	n_{int}	n_{coefs}	T (s)	T_{solve} (ms)	J	P (W)
Simple	Polys	4	1	1	10	3	0.37	301.68	425.60
Simple	FFS	4	1	1	10	3	0.62	400.04	426.36
3 Blocks	Polys	4	3	4	120	9	0.85	90.07	1269.97
3 Blocks	FFS	4	3	4	120	9	1.25	93.85	1270.27
Square	Polys	4	2	8	160	10	0.91	1174.49	1417.68
Square	FFS	4	2	8	160	10	1.26	1287.14	1418.52
Circle	Polys	4	2	7	140	10	0.85	1840.02	1435.62
Circle	FFS	4	2	7	140	10	1.22	2004.76	1437.65
Figure-8	Polys	4	4	9	360	30	1.38	1.05	4200.38
Figure-8	FFS	4	4	9	360	30	2.48	1.15	4200.42

3.1.2 Time-allocated solutions

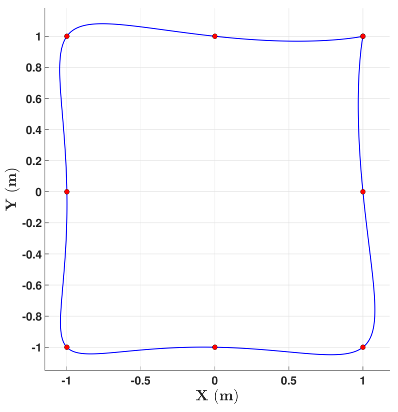
This section continues the discussion of the results, now for the time-allocation formulation presented in Section 2.1.2. In addition to the parameters introduced earlier, the solutions are compared based on the time allocation weight coefficient, k_{T}, and the number of fixed-time iterations, n_{iter}, it took to converge to the optimal solution. The results are summarized in Table 3.2. All the inputs (trajectory waypoints and constraints) are the same as for the fixed-time solutions, the time allocation vector with evenly discretized time is used as an initial guess for the gradient-descent algorithm that attempts to find the optimal time allocation given trajectory-specific weight k_{T}. This weight is arbitrary and is determined by trial and error to match the desired global time $T(s)$ for each trajectory and parametrization combination.

Additional rounds of trajectory generation have been executed to study the effect of the time allocation weight, k_{T}. The choice of this gain determines the global time for executing the entire trajectory, but the values of the minimum snap cost vary between the two parameterizations. For this reason, the same time weight k_{T}, with all other inputs being identical, leads to two different solutions. All graphical results included in this section are using different k_{T} values such that the global time for the two parameterizations match for each trajectory. Since it may also be interesting to compare the two methods when all inputs, including k_{T}, are identical, these results have been also included in Table 3.2.

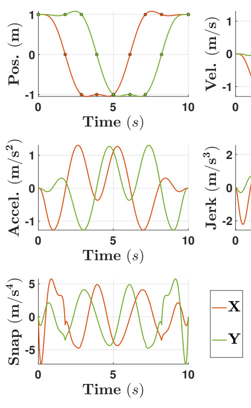
As it was shown in [38], and explained in Section 2.1.2, all the time-optimal solutions for the same set of inputs, other than k_{T} lead to the same path. By varying time allocation gain k_{T}, the global time can be adjusted, but the shape of the solution remains identical. The position components, when plotted w.r.t. time, pass through the exact same points for any reasonable k_{T} values. These points only shift in time, occurring either earlier or later in time. Their derivatives, on the identical plot, not only shift in time but also in their respective magnitudes. This adjustment, if an optimal solution was found, maintains the same position along the trajectory, but increases or decreases the respective velocity, acceleration, jerk, and snap along the path. For this reason, it is expected to obtain the same shape of the time-allocation solution while adjusting time allocation gain k_{T} and keeping all other inputs the same. As will be shown later, this property holds between the two different parameterizations. Although, the errors due to numerical rounding, discussed in Sec. 2.1.1.2, can be more severe for one of the two parameterizations, which can lead to some differences between the two solutions. These differences often increase as the order of the derivatives grows. In fact, first symptoms usually occur on the level of snap, then, by constraining the trajectory even more (by increasing the time allocation weight k_{T}), undesirable large-frequency oscillations propagate to lower derivatives, up until the solution completely breaks down. The invariance of the trajectories to time allocation is quite an interesting characteristic and can be exploited. This means that the feasibility of the trajectories w.r.t. the collision-avoidance constraint takes precedence over the time-allocation problem and the first step (in any trajectory optimization) can be focused on the collision-avoidance step. More specifically, the time-allocation gain, k_{T}, directly affects the total time allocated for the trajectory and, therefore, is directly affecting how demanding the trajectory is to execute. Ref. [40] provides an insightful discussion of the connection between the calculus of variations of kinematics with the corresponding dynamics of the physical vehicle.

3.1.2.1 "Simple" and "3 Blocks" trajectories.

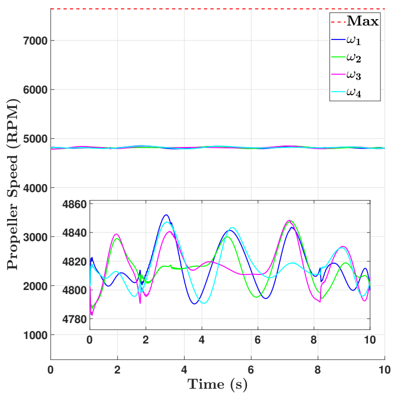
The time-allocation problem for the "Simple" trajectory is trivial, since there exists only one interval and the only optimization that occurs is matching the minimum-snap fixed-time cost with the time weight k_{T}. Since the time factor k_{T} was chosen such that it closely matches the original global time of three seconds, this solution is identical to the one presented earlier (fixed-time solution). The time-optimized solution for the "3 Blocks" trajectory, similar to the "Simple" trajectory, is visibly identical since the number of intervals is still low and even discretization is a good approximation of the optimal time allocation for this trajectory. Due to trajectory similarities, only results are summarized in Table 3.2.



(a) 2D view of the path.

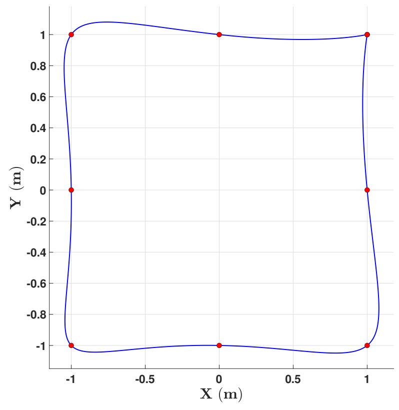


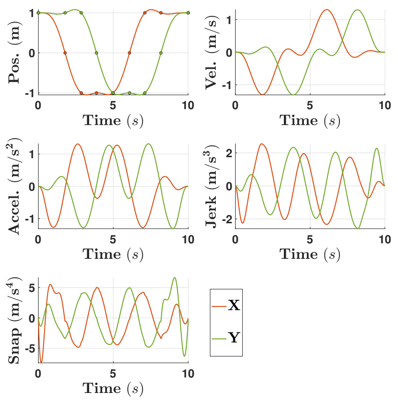
(b) Generated position and higher-order derivative references.



(c) RPM of propellers vs. time for simulated flight.

Figure 3.11: Simulated results for the time-allocated "Square" trajectory with polynomial parameterization.


(a) 2D view of the path.


(b) Generated position and higher-order derivative references.

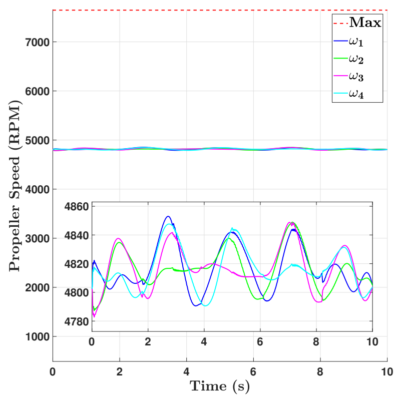

(c) RPM of propellers vs. time for simulated flight.

Figure 3.12: Simulated results for the time-allocated "Square" trajectory with FFS parameterization.

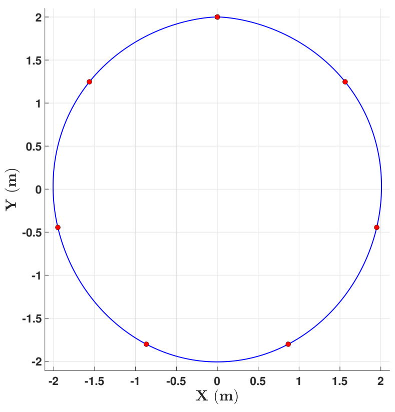
3.1.2.2 "Square" trajectory.

For this problem, with fixed and evenly-distributed time allocation, only two straight edges and some irregularly shaped connections that vaguely resembled the other two edges of a square have been obtained. Intuitively, it should be fairly obvious that an even distribution of time is very often not optimal for a quadcopter that starts from rest, performs some maneuvers, and returns back to a full halt at the end, especially for a fully symmetric path. A better time allocation should allow more time for the initial and final phases of a flight, where the vehicle must overcome its own inertia and add some velocity. If the global time is maintained constant, some time can be removed from the portions of the trajectory that do not require large changes in the flight path and shifted to more demanding phases. Fortunately, this task can be completely automated, and the time allocation algorithm does just what is expected intuitively.

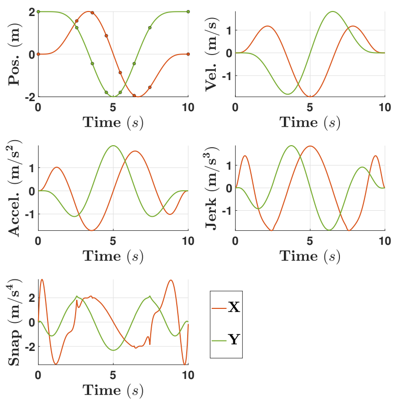
As can be seen in Figures 3.11a and 3.12a, the shape of a square became more clear. Even though the two edges that used to be straight, which corresponded to the initial and final segments of the trajectory, became more curved, the other two have straightened out. The two parameterizations have been able to achieve a close-to-symmetric square shape, which is also visibly identical for the two. The effects of the time allocation are clearly visible when comparing the acceleration plots of the fixed-time solutions (Fig. 3.5b or Fig. 3.6b) with the new time-optimal solutions shown in Figures 3.11b and 3.12b. Large initial and final acceleration spikes that were required for the fixed-time solutions are distributed throughout the entire trajectory. This has lowered the requirements for the control system and allowed a smoother RPM output (Fig. 3.11c and Fig. 3.12c). Even though the global time was maintained the same, proper time allocation has not only restored a more appealing shape but also reduced physical requirements for executing the maneuver. As a general trend, the differences between solutions obtained using different parameterizations are very similar. Very minor distinctions exist on the order of snap, and FFS seems to produce smoother motor output.

3.1.2.3 "Circle" trajectory.

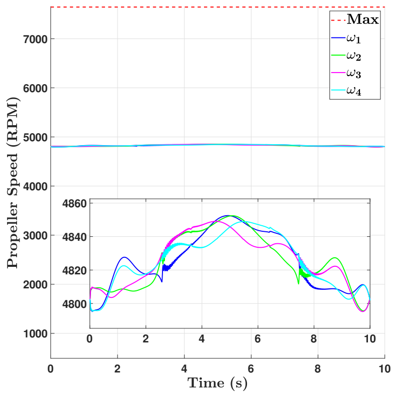
Similar to the "Square" trajectory, proper time allocation was able to fully recover the circular path (Fig. 3.13a and Fig. 3.14a). The two shapes are, again, identical and requirements along the trajectories have been lowered when compared to the fixed-time solutions (see Fig. 3.7 and Fig. 3.8).



(a) 2D view of the path.



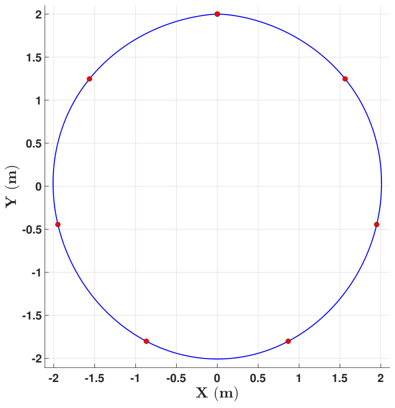
(b) Generated position and higher-order derivative references.

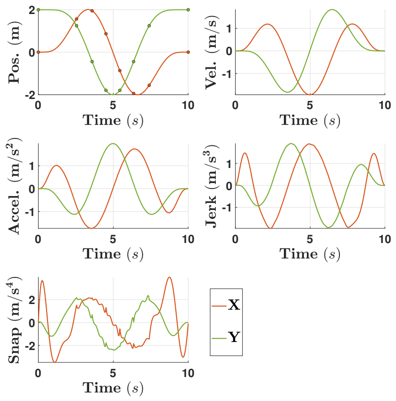


(c) RPM of propellers vs. time for simulated flight.

Figure 3.13: Simulated results for the time-allocated "Circle" trajectory with polynomial parameterization.

Although very similar, the distinctions between the two parameterizations begin to be more clear. When comparing the snap of the two solutions in Fig. 3.13b and Fig. 3.14b, the pattern of the issues changes. The polynomial solution still contains two visible spikes at the two joints between the first and second as well as the last and one before the last intervals,


(a) 2D view of the path.


(b) Generated position and higher-order derivative references.

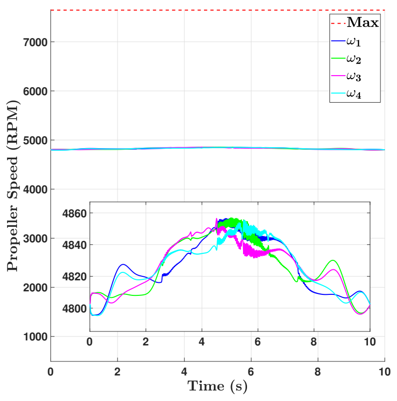

(c) RPM of propellers vs. time for simulated flight.

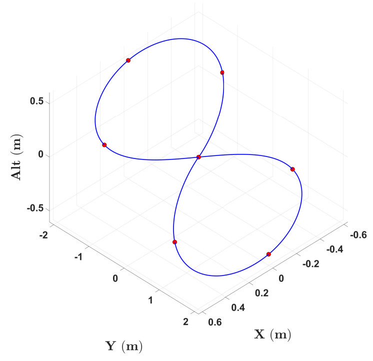
Figure 3.14: Simulated results for the time-allocated "Circle" trajectory with FFS parameterization.

while the FFS solution does not have them (or they are significantly smaller). However, the FFS starts to encounter some difficulties along the intermediate intervals. This, almost like numerical noise, is clearly amplified by the control system and is reflected by the motor profiles plotted in Fig. 3.14c. Although still very minor, if compared with the global scale of the motor profiles, the large-frequency oscillations are only present in the FFS solution and become worse for some of the trajectories or for stricter time-allocation gain values k_T.

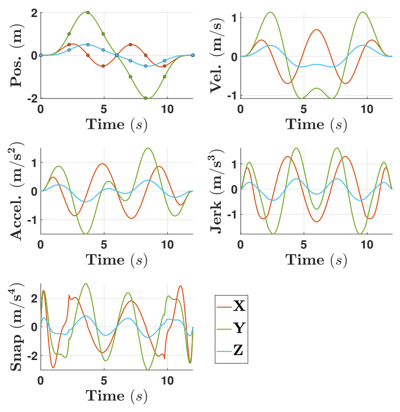
3.1.2.4 "Figure-8" trajectory.

In this case, the shape became more circularized, perfectly tracing the figure "8" (Fig. 3.15a and Fig. 3.16a). Time factors were also chosen such that the global time for the flight is decreased by more than half of the original guess. For this reason, all the ranges for derivatives and motor speeds became somewhat higher than for the fixed-time solution. However, it should be obvious that without a proper time allocation, performing the exact same maneuver, but more than twice as fast, would significantly increase all the requirements. No notable or new differences between the two solutions are seen neither on the level of derivatives (Fig. 3.15b and Fig. 3.16b) nor at the level of control system outputs (Fig. 3.15c and Fig. 3.16c).

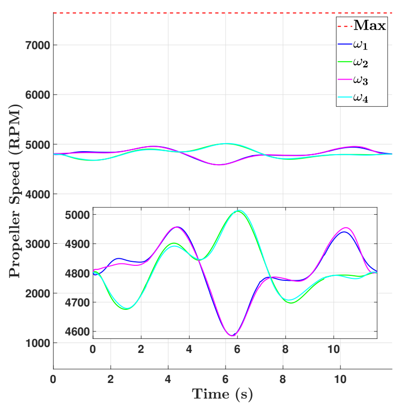
One important point worth mentioning is that even though there are some minor issues on the snap level for both parametrizations, they do not appear to show a distinctive pattern



(a) 3D view of the path.

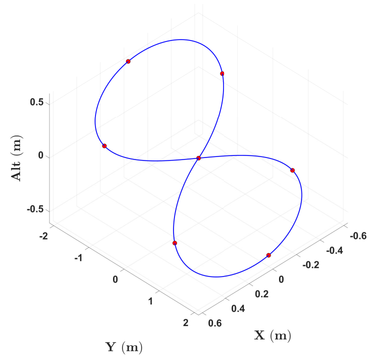


(b) Generated position and higher-order derivative references.

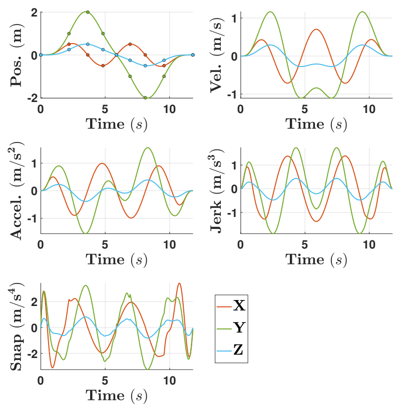


(c) RPM of propellers vs. time for simulated flight.

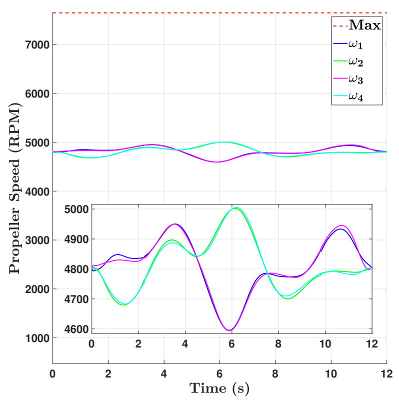
Figure 3.15: Simulated results for the time-allocated "Figure-8" trajectory with polynomial parameterization.



(a) 3D view of the path.



(b) Generated position and higher-order derivative references.



(c) RPM of propellers vs. time for simulated flight.

Figure 3.16: Simulated results for the time-allocated "Figure-8" trajectory with FFS parameterization.

or correlation. In summary, it should be clear that the motion planning of quadrotors can be performed using either of the parameterizations and the FFS shows close convergence to the polynomial solution even when time allocation is performed. In addition to the graphical examples discussed earlier in this section, the solution for the equivalent time factor k_T has been summarized in Table 3.2. Although the shapes of all the solutions for any k_T values remained the same, these results have been included to provide a one-to-one comparison of the two parameterizations when every single input is identical. As it was shown by the fixed-time solution, the minimum-snap cost of the FFS parameterization is higher than the polynomial cost. For this reason, the FFS method requires a higher gain k_T to match the same global time. Thus, the total flight time should always be higher for the FFS solution than for an equivalent polynomial solution.

Table 3.2: Summary of the time-allocation solutions.

Name	Method	n_dt	n_dim	n_int	n_coefs	T (s)	T_solve (ms)	J	P (W)	k_T	n_iter
Simple	Polys	4	1	1	10	3.00	5.05	300.21	425.59	700	5
Simple	FFS	4	1	1	10	3.10	9.09	310.99	439.68	700	6
Simple	FFS	4	1	1	10	3.00	13.59	398.75	426.35	930	5
3 Blocks	Polys	4	3	4	120	10.00	305.00	8.79	1403.42	6.15	34
3 Blocks	FFS	4	3	4	120	10.10	252.69	8.88	1417.40	6.15	22
3 Blocks	FFS	4	3	4	120	10.00	285.33	9.53	1403.52	6.67	23
Square	Polys	4	2	8	160	10.00	423.99	200.02	1411.97	140	35
Square	FFS	4	2	8	160	10.04	609.74	200.87	1417.61	140	32
Square	FFS	4	2	8	160	10.00	563.42	205.89	1412.14	144	29
Circle	Polys	4	2	7	140	10.00	306.61	54.03	1420.25	37.8	30
Circle	FFS	4	2	7	140	10.07	528.96	54.38	1430.07	37.8	32
Circle	FFS	4	2	7	140	10.00	469.94	56.89	1420.67	39.8	27
Figure-8	Polys	4	4	9	360	12.00	515.74	85.72	1692.22	50	22
Figure-8	FFS	4	4	9	360	12.05	895.02	86.11	1699.25	50	24
Figure-8	FFS	4	4	9	360	12.00	892.07	88.81	1665.06	51.8	23

Although the theoretical assumptions and fixed-time solutions show that FFS parameterization is slower than polynomial parameterization, the time-allocation problem may not follow the same trends. In fact, FFS seems to often converge in fewer iterations than an equivalent polynomial method. This is related to the time factor, k_T, but for the combination of inputs used in this work, some solutions have low enough number of iterations such that the total computational time for the time-allocation problem offsets the longer evaluation times of each of the fixed-time solutions. This is clearly shown for the "3 Blocks" trajectory in Table 3.2, where FFS was able to achieve an optimal solution faster than the equivalent formulation using the polynomial ("3 Blocks" problem). The fact that the time-allocation problem shows that the FFS parameterization may often converge to the optimal (with a near-equivalent minimum-snap solution in fewer iterations than an identical algorithm with a polynomial formulation) is intriguing. Some features, specific to the trigonometric functions, allow controlling the rate of convergence and can be further explored.

3.1.2.5 Fast Finite Fourier Series

As was shown in Sec. 2.1.1.3, the FFS method is inherently slower due to the sparsity pattern of the constraint mapping matrix $\mathbb{A}_i(T_\mathrm{i})$. The numerical results for the fixed-time solutions have further verified this in Sec. 3.1.1. However, this is not quite the case if the $\frac{1}{2}$ factor is removed from the argument of $\sin$ and $\cos$ terms and the order of derivative of constraints is increased to be one degree higher (5th derivative or crackle). The cost function can remain the same (e.g. $n_\mathrm{dt} = 4$ for cost function), but since $n_\mathrm{dt} = 5$ for the constraints, the minimum number of design variables per interval is twelve. This also sets the $\mathbb{A}_i(T_\mathrm{i})$ and $\mathbb{Q}_i(T_\mathrm{i})$ to be $\mathbb{R}^{12\times 12}$. The new FFS parametrization is:

$$\text{if } n_\mathrm{dt} \text{ is even:} \quad \begin{aligned} P(t, T_\mathrm{i}, \boldsymbol{p}_\mathrm{i}) = \frac{a_0}{2} &+ \sum_{k=1}^{n_r} \left\{ a_k \cos\left(k\pi \frac{t}{T_\mathrm{i}}\right) + b_{k-1} \sin\left(k\pi \frac{t}{T_\mathrm{i}}\right) \right\} \\ &+ b_{n_r} \cos\left(n_r \pi \frac{t}{T_\mathrm{i}}\right), \end{aligned} \tag{3.1}$$

$$\text{if } n_\mathrm{dt} \text{ is odd:} \quad \begin{aligned} P(t, T_\mathrm{i}, \boldsymbol{p}_\mathrm{i}) = \frac{a_0}{2} &+ \sum_{k=1}^{n_r} \left\{ a_k \cos\left(k\pi \frac{t}{T_\mathrm{i}}\right) + b_{k-1} \sin\left(k\pi \frac{t}{T_\mathrm{i}}\right) \right\} \\ &+ b_{n_r} \sin\left(n_r \pi \frac{t}{T_\mathrm{i}}\right), \end{aligned} \tag{3.2}$$

where the argument of $\sin$ and $\cos$ terms no longer contains a factor of $\frac{1}{2}$. After deriving the new mapping matrices for the FFS (using Eq. (3.2)) and polynomial parametrizations, the same sparse metrics can be computed. Equation (2.12) still leads to a fully dense matrix for both, and the number of non-zero elements in the constraint mapping matrix has grown as expected: $63/144$ for polynomials and $68/144$ for FFS parametrizations. Interestingly enough, the number of non-zero elements in the inverse of the $\mathbb{A}_i(T_\mathrm{i})$ does not follow the same trend: $78/144$ for polynomial and the same $68/144$ for FFS parametrizations. Not only does the number of no-zero elements remain the same for $\mathbb{A}_i(T_\mathrm{i})$ matrix of FFS after inversion, but it becomes lower than one for polynomial parametrization. This clearly suggests that the evaluation time for minimum-snap optimization with boundary constraints up to crackle can be faster for FFS than for polynomial parametrization. The computational results in Tables 3.3 and 3.4 confirm this observation.

However, removing the factor of $\frac{1}{2}$ from the FFS formulation introduces some penalty for complex trajectories, or those having at least 3 intervals. The resulting shape of the trajectory is highly likely to be noticeably different from the polynomial solution. The higher-order derivatives, especially on the snap level may be highly oscillatory, which is very undesirable for quadrotor flight. The solution to these issues is to consider adding back $\frac{1}{2}$ factor in all $\sin$ or all $\cos$ terms. This fixed all known issues with distortions in the shape of the solution and all the undesirable oscillations, but such formulation loses the sparsity benefit. Several combinations of different factors have been considered for arguments of $\sin$ and $\cos$ terms to exploit the benefits from the sparsity of the constraint matrix without affecting the quality of the solution, but these were not successful. In this work, the graphical or experiential results have not been included for any other FFS parametrization other than those given by Eq. 2.9, since it gave the closest math to the polynomial parametrization. However, computational results for Fast Finite Fourier Series or F-FFS version of the algorithm are summarized below.

Table 3.3: Fixed time solutions summary for F-FFS.

Name	Method	n_{dt}	n_{dim}	n_{int}	n_{coefs}	T (s)	T_{solve} (ms)	J
Simple	Polys	5	1	1	12	3	0.40	561.60
Simple	F-FFS	5	1	1	12	3	0.46	1011.40
3 Blocks	Polys	5	3	4	144	9	1.11	112.39
3 Blocks	F-FFS	5	3	4	144	9	1.15	355.54
Square	Polys	5	2	8	192	10	1.28	1526.95
Square	F-FFS	5	2	8	192	10	1.13	10738.88
Circle	Polys	5	2	7	168	10	1.04	2347.83
Circle	F-FFS	5	2	7	168	10	1.09	12850.88
Figure-8	Polys	5	4	9	432	30	1.89	1.36
Figure-8	F-FFS	5	4	9	432	30	1.91	12.87

The fixed time solution for the same set of trajectories is shown in Table 3.3. As was expected, the F-FFS parametrization is solved, if not faster than equivalent order polynomial parametrization, but clearly as fast. The F-FFS is actually faster than the FFF parametrization of the lower order, even though there are more design variables for F-FFS. The primary difference between F-FFS and FFS is the cost value, which affects the relative values of time allocation gain k_{T} for the results shown in Table 3.4. The difference between the cost value for F-FFS and polynomial parametrizations is even more noticeable as the order of constraints has been increased, but follows the same trend.

The most intriguing result is the computational time for the F-FFS and the number of iterations that led to it. By paying attention to the last column of Table 3.4, the number of iterations is the same or lower for F-FFS than for polynomial parametrization, and it results in lower computational time T_{solve} for almost all the test cases. The computational results for F-FFS suggest that Fourier Series may converge faster than polynomials under certain conditions. Although it is currently not quite clear what combination of trigonometric functions and their arguments can lead to faster computational speed while retaining the desired characteristics of the polynomial solutions.

Table 3.4: Time allocation solutions summary for F-FFS.

Name	Method	n_{dt}	n_{dim}	n_{int}	n_{coefs}	T (s)	T_{solve} (ms)	J	k_{T}	n_{iter}
Simple	Polys	5	1	1	12	3.01	11.90	538.88	1250	5
Simple	F-FFS	5	1	1	12	3.24	12.06	580.01	1250	8
Simple	F-FFS	5	1	1	12	2.99	11.65	1026.41	2400	7
3 Blocks	Polys	5	3	4	144	10.09	221.79	10.10	7	23
3 Blocks	F-FFS	5	3	4	144	14.12	211.86	14.13	7	22
3 Blocks	F-FFS	5	3	4	144	10.09	194.98	148.54	130	17
Square	Polys	5	2	8	192	10.03	754.32	215.13	150	39
Square	F-FFS	5	2	8	192	16.11	530.23	345.26	150	31
Square	F-FFS	5	2	8	192	10.03	395.37	9465.88	6600	17
Circle	Polys	5	2	7	168	10.31	405.67	50.12	34	26
Circle	F-FFS	5	2	7	168	19.91	392.93	96.72	34	26
Circle	F-FFS	5	2	7	168	10.30	308.73	9718.09	6600	15
Figure-8	Polys	5	4	9	432	12.75	998.74	61.97	26	34
Figure-8	F-FFS	5	4	9	432	21.99	775.60	106.83	34	20
Figure-8	F-FFS	5	4	9	432	12.75	1098.92	4846.52	2660	25

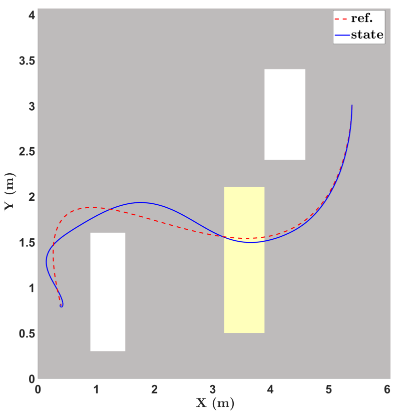
It is important to note that the F-FFS method is not as computationally stable as FFS and is not practical for quadrotor trajectory optimization, since it requires higher-order derivatives beyond snap. The computational results here have been shown to further highlight the potential benefits that can be offered by the FFS parametrization. For the purposes of this work, only FFS and polynomial parametrizations have been considered in the subsequent comparison.

3.2 Experimental Results

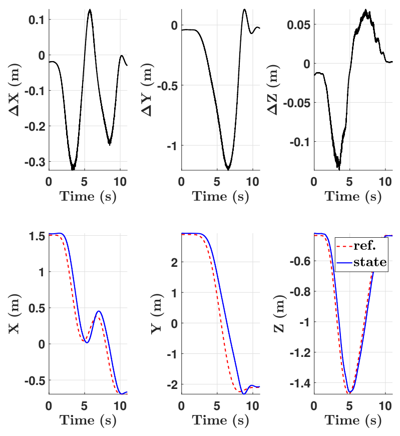
Consider a relatively small flying arena, limited to trajectories confined to a medium-size room (about $4 \times 6 \times 3$ meters). Five different trajectories are considered to compare the usefulness of the trajectories in practice. All time-optimal trajectories discussed in the previous sections have been successfully flown. Both polynomial and FFS parameterizations have shown identical performance. Theoretical and computational results have shown that the differences between the two methods are negligible, even in an ideal environment. Since there was little to no difference between the two methods, the results are reported only for the “3 Blocks”, “Square” and “Figure-8” trajectories.

3.2.1 “3 Blocks” trajectory

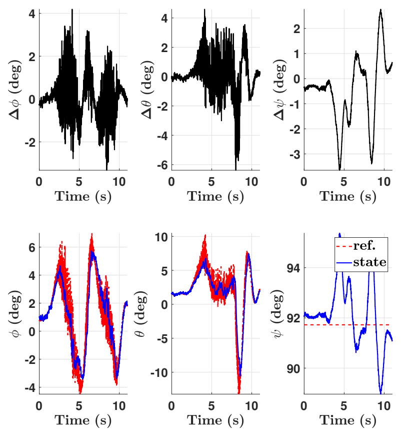
Using the optimal solution for the “3 Blocks” trajectory with time allocation, the control system on the quadrotor had to follow the required position references (red) generated using the two parameterizations. The resulting 2D path (blue) is shown in Figures 3.17a and 3.18a. Although not as perfect as in the simulator, the two trajectories have been closely tracked and no significant deviation occurred for any of the two methods.



(a) 2D view of the flight path.



(b) Tracking of position components.



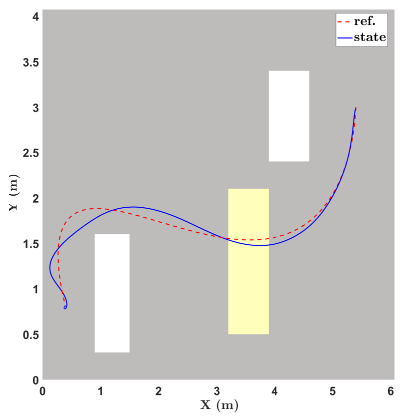
(c) Tracking of attitude components.

Figure 3.17: Experimental results for the time-allocated “3 Blocks” trajectory with polynomial parameterization.

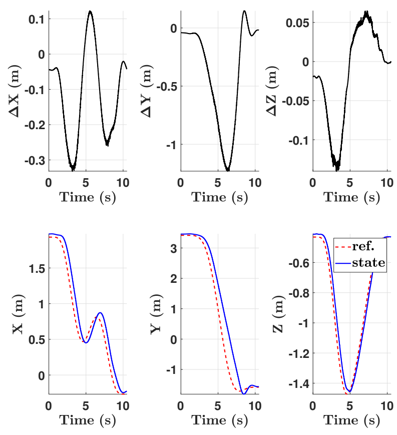
The performance of the position control tracking is shown in Fig. 3.17b and Fig. 3.18b. Fig. 3.17c and Fig. 3.18c show attitude control system tracking for the references generated by the higher-level velocity and position control (details of the flight control system are given in Ref. [56]). It is worth noting that the quadrotor is relatively heavy and a more conservative control system gains have been chosen. For these reasons, in addition to the reference tracking method that was chosen, the quadrotor's state noticeably lags behind the commanded reference, with position-level errors growing in time as the trajectory references run away faster than the quadrotor is capable of catching up. However, the commanded trajectory is still followed with acceptable accuracy.

3.2.2 "Square" trajectory

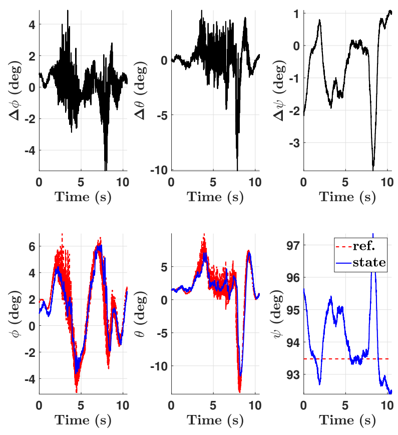
Similarly, taking the optimal solution for the square-shaped trajectory with time allocation, the quadrotor had to follow position-level setpoints. The resulting path is shown in Fig. 3.19a and Fig. 3.20a. Again, the 2D view is shown instead of the 3D view because the motion is completely planar. In fact, the vertical position errors in Fig. 3.19b and Fig. 3.20b clearly show that the altitude is kept within a few millimeters within the initialization vertical setpoint. The performance of the attitude control tracking is shown in Figures 3.19c and 3.20c. Since the implemented control system has a cascaded PID control structure, the references for each control layer are generated by the control layer above, in a cascaded manner. For this reason



(a) 2D view of the flight path.



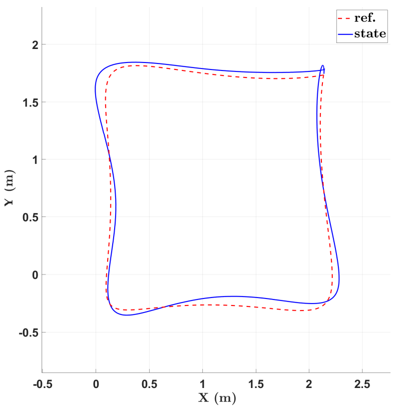
(b) Tracking of position components.



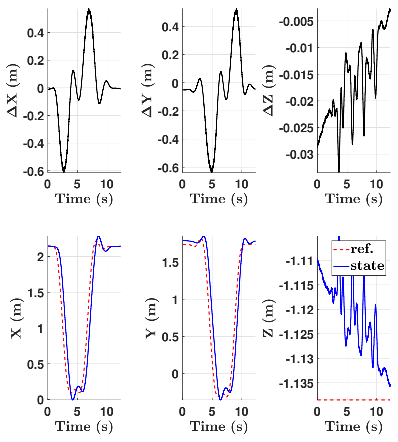
(c) Tracking of attitude components.

Figure 3.18: Experimental flight results for the time-allocated "3 Blocks" trajectory with FFS parameterization.

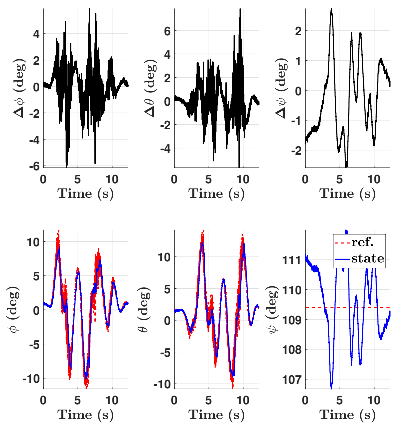
and due to the significant numerical noise of estimated states, the control system references for the attitude layer show noticeable bands of noise.



(a) 2D view of the flight path.

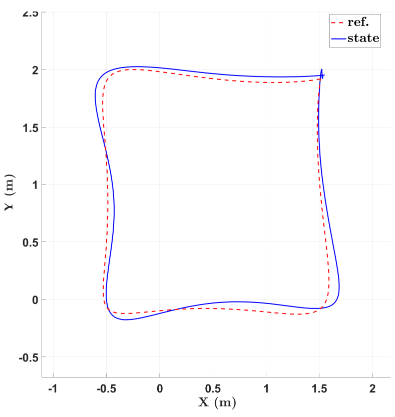


(b) Tracking of position components.

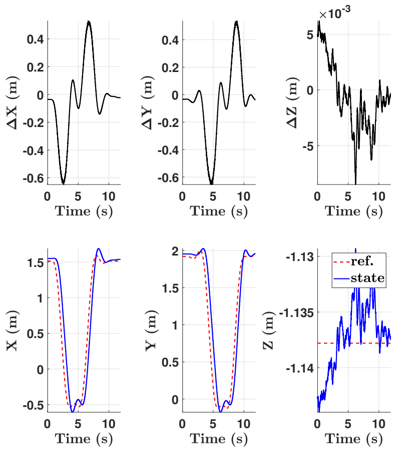


(c) Tracking of attitude components.

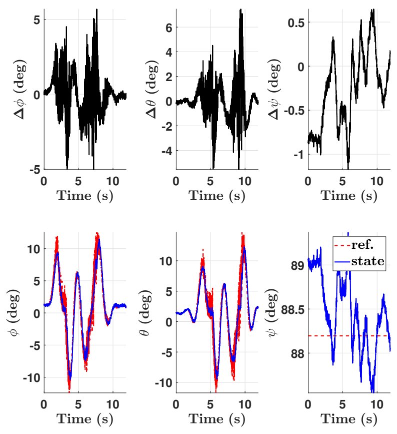
Figure 3.19: Experimental results for the time-allocated "Square" trajectory with polynomial parameterization.



(a) 2D view of the flight path.



(b) Tracking of position components.

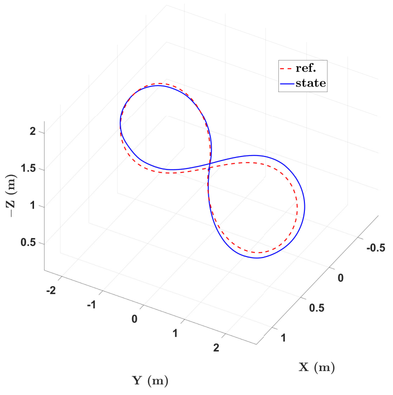


(c) Tracking of attitude components.

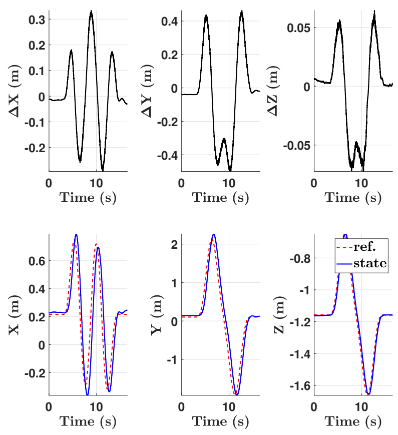
Figure 3.20: Experimental flight results for the time-allocated "Square" trajectory with FFS parameterization.

3.2.3 "Figure-8" trajectory

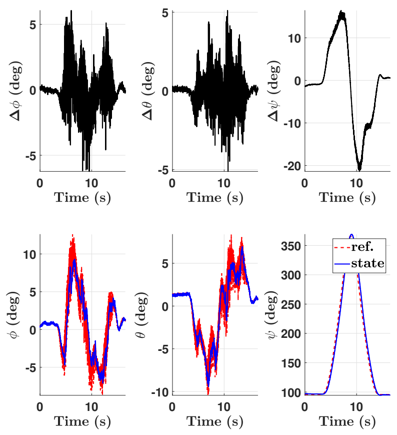
The experimental results for the fast and complex "Figure-8" trajectory are also as expected (Fig. 3.21 and Fig. 3.22). Although the uncertainties due to the environment allow using experimental results as validation, they limit the acquisition of any additional insight into the solutions. No unexpected behavior has been observed, and it is clear that the two methods generated trajectories that can be followed by a quadrotor in the ACELAB.



(a) 3D view of the flight path.

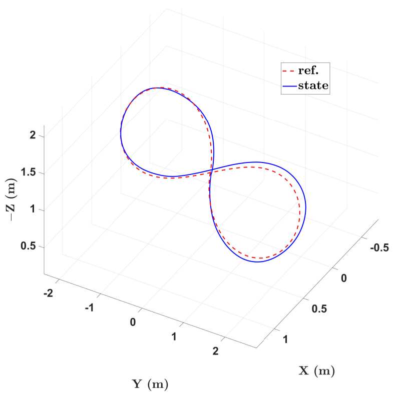


(b) Tracking of position components.

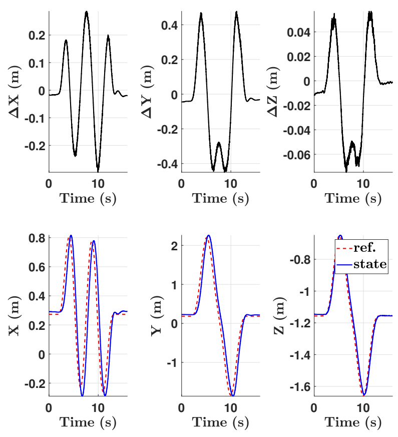


(c) Tracking of attitude components.

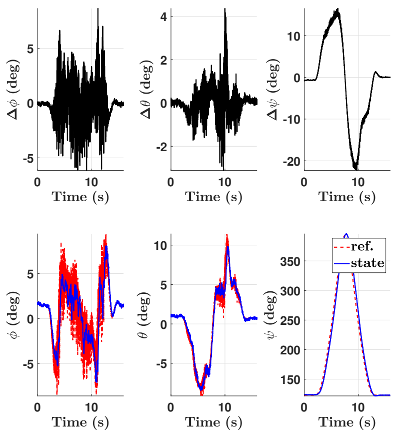
Figure 3.21: Experimental results for the time-allocated “Figure-8” trajectory with polynomial parameterization.



(a) 3D view of the flight path.



(b) Tracking of position components.



(c) Tracking of attitude components.

Figure 3.22: Experimental flight results for the time-allocated “Figure-8” trajectory with FFS parameterization.

Chapter 4

4.1 Conclusion

As the result of this research effort, a quadrotor motion-planning algorithm based on the idea of the Finite Fourier Series (FFS) method for minimum-snap trajectory optimization of quadrotors was successfully developed. It was demonstrated how the minimum-snap trajectory optimization problem can be formulated with the FFS parameterization as an unconstrained quadratic programming problem.

The results show a comparable performance between the FFS and polynomial parametrization methods on five types of trajectories labeled as "simple", "circular", "square", "3 Blocks" and "Figure-8". The results show that for one of the problems, the time-allocation problem that uses FFS parameterization converges in fewer iterations than polynomial parametrization, suggesting that it can potentially offer a reduced overall computational time, given an appropriate input. However, for the majority of the considered trajectory generation problems, the polynomial method offers slightly better computational performance. The FFS method has generated trajectories similar to the polynomial parameterization methods when solving minimum-snap trajectory optimization problems for quadrotors. The one-to-one comparison of the FFS method with polynomial parametrization has also shown an obvious computational advantage of the well-established polynomial minimum-snap optimization. The power consumption along the smooth trajectories generated using the FFS and polynomial parameterizations show less than 0.1% of relative difference. The FFS parameterization has been proven experimentally to be suitable for quadrotor minimum-snap trajectory optimization. It is important to mention that the performance of the control system and reference tracking will most likely degrade when precise motion capture data is no longer available and the quadcopter has to rely on the onboard sensors.

Bibliography

[1] *Drone COVID Vaccine Deliveries*. en. URL: https://about.ups.com/ca/en/our-stories/innovation-driven/drone-covid-vaccine-deliveries (visited on 03/18/2023).

[2] *How the Ukraine drone war is changing the game on the battlefield*. en-US. Section: Military. Sept. 2022. URL: https://newatlas.com/military/ukraine-russia-drone-war/ (visited on 03/18/2023).

[3] C. Goerzen, Z. Kong, and B. Mettler. "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance". en. In: *Journal of Intelligent and Robotic Systems* 57.1-4 (Jan. 2010), pp. 65–100. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-009-9383-1. URL: http://link.springer.com/10.1007/s10846-009-9383-1 (visited on 09/14/2022).

[4] Evan Kawamura and Dilmurat Azimov. "Integrated Extremal Control and Explicit Guidance for Quadcopters". en. In: *Journal of Intelligent & Robotic Systems* 100.3-4 (Dec. 2020), pp. 1583–1613. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-020-01211-2. URL: https://link.springer.com/10.1007/s10846-020-01211-2 (visited on 09/13/2022).

[5] Juan Paredes et al. "Development, implementation, and experimental outdoor evaluation of quadcopter controllers for computationally limited embedded systems". en. In: *Annual Reviews in Control* 52 (2021), pp. 372–389. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2021.06.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1367578821000420 (visited on 09/14/2022).

[6] M.G. Mohanan and Ambuja Salgoankar. “A survey of robotic motion planning in dynamic environments”. en. In: *Robotics and Autonomous Systems* 100 (Feb. 2018), pp. 171–185. ISSN: 09218890. DOI: 10.1016/j.robot.2017.10.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0921889017300313 (visited on 09/14/2022).

[7] E. Trélat. “Optimal Control and Applications to Aerospace: Some Results and Challenges”. en. In: *Journal of Optimization Theory and Applications* 154.3 (Sept. 2012), pp. 713–758. ISSN: 0022-3239, 1573-2878. DOI: 10.1007/s10957-012-0050-5. URL: http://link.springer.com/10.1007/s10957-012-0050-5 (visited on 09/14/2022).

[8] Reza Kamyar and Ehsan Taheri. “Aircraft Optimal Terrain/Threat-Based Trajectory Planning and Control”. en. In: *Journal of Guidance, Control, and Dynamics* 37.2 (Mar. 2014), pp. 466–483. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.61339. URL: https://arc.aiaa.org/doi/10.2514/1.61339 (visited on 09/14/2022).

[9] Ehsan Taheri et al. “A novel approach for optimal trajectory design with multiple operation modes of propulsion system, part 1”. en. In: *Acta Astronautica* 172 (July 2020), pp. 151–165. ISSN: 00945765. DOI: 10.1016/j.actaastro.2020.02.042. URL: https://linkinghub.elsevier.com/retrieve/pii/S0094576520301156 (visited on 09/14/2022).

[10] Paul Williams. “Three-Dimensional Aircraft Terrain-Following via Real-Time Optimal Control”. en. In: *Journal of Guidance, Control, and Dynamics* 30.4 (July 2007), pp. 1201–1206. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.29145. URL: https://arc.aiaa.org/doi/10.2514/1.29145 (visited on 10/28/2022).

[11] Yang Wei, Chun-Lin Shen, and Peter Dorato. “U-parameter design for terrain-following flight control”. en. In: *Journal of Guidance, Control, and Dynamics* 16.2 (Mar. 1993), pp. 387–389. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.21015. URL: https://arc.aiaa.org/doi/10.2514/3.21015 (visited on 10/28/2022).

[12] Lun Quan et al. “Survey of UAV motion planning”. en. In: *IET Cyber-Systems and Robotics* 2.1 (2020). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-csr.2020.0004, pp. 14–21. ISSN: 2631-6315. DOI: 10.1049/iet-csr.2020.0004. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-csr.2020.0004 (visited on 09/14/2022).

[13] Janne Karelahti, Kai Virtanen, and John Öström. “Automated Generation of Realistic Near-Optimal Aircraft Trajectories”. en. In: *Journal of Guidance, Control, and Dynamics* 31.3 (May 2008), pp. 674–688. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.31159. URL: https://arc.aiaa.org/doi/10.2514/1.31159 (visited on 10/28/2022).

[14] Steven M LaValle. “Rapidly-exploring random trees: A new tool for path planning”. In: *Ames, IA, USA* (1998).

[15] Steven M. LaValle. *Planning Algorithms*. en. 1st ed. Cambridge University Press, May 2006. ISBN: 978-0-521-86205-9 978-0-511-54687-7. DOI: 10.1017/CBO9780511546877. URL: https://www.cambridge.org/core/product/identifier/9780511546877/type/book (visited on 10/27/2022).

[16] Alison Eele and Arthur Richards. “Path-Planning with Avoidance Using Nonlinear Branch-and-Bound Optimization”. en. In: *Journal of Guidance, Control, and Dynamics* 32.2 (Mar. 2009), pp. 384–394. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.40034. URL: https://arc.aiaa.org/doi/10.2514/1.40034 (visited on 10/28/2022).

[17] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. “ARA*: Anytime A* with provable bounds on sub-optimality”. In: *Advances in neural information processing systems* 16 (2003).

[18] T. Siméon, J.-P. Laumond, and C. Nissoux. “Visibility-based probabilistic roadmaps for motion planning”. en. In: *Advanced Robotics* 14.6 (Jan. 2000), pp. 477–493. ISSN: 0169-1864, 1568-5535. DOI: 10.1163/156855300741960. URL: https://www.

tandfonline.com/doi/full/10.1163/156855300741960 (visited on 09/14/2022).

[19] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30.7 (June 2011). Publisher: SAGE Publications Ltd STM, pp. 846–894. ISSN: 0278-3649. DOI: 10.1177/0278364911406761. URL: https://doi.org/10.1177/0278364911406761 (visited on 09/14/2022).

[20] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. “Real-Time Motion Planning for Agile Autonomous Vehicles”. en. In: *Journal of Guidance, Control, and Dynamics* 25.1 (Jan. 2002), pp. 116–129. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4856. URL: https://arc.aiaa.org/doi/10.2514/2.4856 (visited on 10/28/2022).

[21] Jay P. Wilhelm and Garrett Clem. “Vector Field UAV Guidance for Path Following and Obstacle Avoidance with Minimal Deviation”. en. In: *Journal of Guidance, Control, and Dynamics* 42.8 (Aug. 2019), pp. 1848–1856. ISSN: 1533-3884. DOI: 10.2514/1.G004053. URL: https://arc.aiaa.org/doi/10.2514/1.G004053 (visited on 10/28/2022).

[22] Vrushabh Vijaykumar Zinage and Satadal Ghosh. “Generalized Shape Expansion-Based Motion Planning in Three-Dimensional Obstacle-Cluttered Environment”. en. In: *Journal of Guidance, Control, and Dynamics* 43.9 (Sept. 2020), pp. 1781–1791. ISSN: 1533-3884. DOI: 10.2514/1.G004756. URL: https://arc.aiaa.org/doi/10.2514/1.G004756 (visited on 10/28/2022).

[23] Anusha Mujumdar and Radhakant Padhi. “Reactive Collision Avoidance of Using Nonlinear Geometric and Differential Geometric Guidance”. en. In: *Journal of Guidance, Control, and Dynamics* 34.1 (Jan. 2011), pp. 303–311. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.50923. URL: https://arc.aiaa.org/doi/10.2514/1.50923 (visited on 10/28/2022).

[24] Ruixiang Du and Raghvendra V. Cowlagi. “Interactive Sensing and Planning for a Quadrotor Vehicle in Partially Known Environments”. en. In: *Journal of Guidance, Control,*

and Dynamics 42.7 (July 2019), pp. 1601–1611. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G003773. URL: https://arc.aiaa.org/doi/10.2514/1.G003773 (visited on 10/28/2022).

[25] Youkyung Hong et al. "Quadrotor path planning using A* search algorithm and minimum snap trajectory generation". en. In: *ETRI Journal* 43.6 (2021). _eprint: https://onlinelibrary.wiley.co 0085, pp. 1013–1023. ISSN: 2233-7326. DOI: 10.4218/etrij.2020-0085. URL: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.2020-0085 (visited on 10/28/2022).

[26] Boyu Zhou et al. "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 3529–3536. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2927938.

[27] D. Reed Robinson et al. "An Efficient Algorithm for Optimal Trajectory Generation for Heterogeneous Multi-Agent Systems in Non-Convex Environments". en. In: *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), pp. 1215–1222. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2794582. URL: https://ieeexplore.ieee.org/document/8260912/ (visited on 09/14/2022).

[28] Eungchang Mason Lee et al. "REAL: Rapid Exploration with Active Loop-Closing toward Large-Scale 3D Mapping using UAVs". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Sept. 2021, pp. 4194–4198. DOI: 10.1109/IROS51168.2021.9636611.

[29] David G. Hull. "Conversion of Optimal Control Problems into Parameter Optimization Problems". en. In: *Journal of Guidance, Control, and Dynamics* 20.1 (Jan. 1997), pp. 57–60. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4033. URL: https://arc.aiaa.org/doi/10.2514/2.4033 (visited on 10/28/2022).

[30] Michael Szmuk et al. "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance". en. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, Sept. 2017,

pp. 4862–4868. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206363. URL: http://ieeexplore.ieee.org/document/8206363/ (visited on 09/14/2022).

[31] Liyang Wang and Xiaoli Bai. "Quadrotor Autonomous Approaching and Landing on a Vessel Deck". en. In: *Journal of Intelligent & Robotic Systems* 92.1 (Sept. 2018), pp. 125–143. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-017-0757-5. URL: http://link.springer.com/10.1007/s10846-017-0757-5 (visited on 09/14/2022).

[32] John T. Betts. "Survey of Numerical Methods for Trajectory Optimization". en. In: *Journal of Guidance, Control, and Dynamics* 21.2 (Mar. 1998), pp. 193–207. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4231. URL: https://arc.aiaa.org/doi/10.2514/2.4231 (visited on 10/28/2022).

[33] Nadeem Faiz, Sunil K. Agrawal, and Richard M. Murray. "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities". en. In: *Journal of Guidance, Control, and Dynamics* 24.2 (Mar. 2001), pp. 219–227. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4732. URL: https://arc.aiaa.org/doi/10.2514/2.4732 (visited on 10/28/2022).

[34] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". en. In: *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), pp. 620–626. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2017.2776353. URL: http://ieeexplore.ieee.org/document/8118153/ (visited on 09/14/2022).

[35] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". en. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 2520–2525. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980409. URL: http://ieeexplore.ieee.org/document/5980409/ (visited on 09/14/2022).

[36] Davide Invernizzi, Simone Panza, and Marco Lovera. “Robust Tuning of Geometric Attitude Controllers for Multirotor Unmanned Aerial Vehicles”. en. In: *Journal of Guidance, Control, and Dynamics* 43.7 (July 2020), pp. 1332–1343. ISSN: 1533-3884. DOI: 10.2514/1.G004457. URL: https://arc.aiaa.org/doi/10.2514/1.G004457 (visited on 10/28/2022).

[37] John T. Betts and William P. Huffman. “Path-constrained trajectory optimization using sparse sequential quadratic programming”. en. In: *Journal of Guidance, Control, and Dynamics* 16.1 (Jan. 1993), pp. 59–68. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.11428. URL: https://arc.aiaa.org/doi/10.2514/3.11428 (visited on 10/28/2022).

[38] Charles Richter, Adam Bry, and Nicholas Roy. “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments”. en. In: *Robotics Research*. Ed. by Masayuki Inaba and Peter Corke. Vol. 114. Series Title: Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2016, pp. 649–666. ISBN: 978-3-319-28870-3 978-3-319-28872-7. DOI: 10.1007/978-3-319-28872-7_37. URL: http://link.springer.com/10.1007/978-3-319-28872-7_37 (visited on 09/14/2022).

[39] Adam Bry et al. “Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments”. In: *The International Journal of Robotics Research* 34.7 (June 2015). Publisher: SAGE Publications Ltd STM, pp. 969–1002. ISSN: 0278-3649. DOI: 10.1177/0278364914558129. URL: https://doi.org/10.1177/0278364914558129 (visited on 09/13/2022).

[40] Oleg A. Yakimenko. “Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories”. en. In: *Journal of Guidance, Control, and Dynamics* 23.5 (Sept. 2000), pp. 865–875. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4616. URL: https://arc.aiaa.org/doi/10.2514/2.4616 (visited on 09/13/2022).

[41] Nadia Kreciglowa, Konstantinos Karydis, and Vijay Kumar. “Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation”. In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. June 2017, pp. 656–662. DOI: 10.1109/ICUAS.2017.7991496.

[42] Markus Hehn and Raffaello D’Andrea. “Real-Time Trajectory Generation for Quadrocopters”. In: *IEEE Transactions on Robotics* 31.4 (Aug. 2015), pp. 877–892. ISSN: 1941-0468. DOI: 10.1109/TRO.2015.2432611.

[43] Marcelino M. de Almeida, Rahul Moghe, and Maruthi Akella. “Real-Time Minimum Snap Trajectory Generation for Quadcopters: Algorithm Speed-up Through Machine Learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2019, pp. 683–689. DOI: 10.1109/ICRA.2019.8793569.

[44] Declan Burke, Airlie Chapman, and Iman Shames. “Generating Minimum-Snap Quadrotor Trajectories Really Fast”. en. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 1487–1492. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341794. URL: https://ieeexplore.ieee.org/document/9341794/ (visited on 09/14/2022).

[45] Zhepei Wang et al. “Generating Large-Scale Trajectories Efficiently using Double Descriptions of Polynomials”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 7436–7442. DOI: 10.1109/ICRA48506.2021.9561585.

[46] Aleix Paris, Brett T. Lopez, and Jonathan P. How. “Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2020, pp. 9577–9583. DOI: 10.1109/ICRA40945.2020.9197081.

[47] Ehsan Taheri and Ossama Abdelkhalik. “Shape Based Approximation of Constrained Low-Thrust Space Trajectories using Fourier Series”. en. In: *Journal of Spacecraft and Rockets* 49.3 (May 2012), pp. 535–546. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/

1.58789. URL: https://arc.aiaa.org/doi/10.2514/1.58789 (visited on 09/14/2022).

[48] Ehsan Taheri and Ossama Abdelkhalik. “Fast Initial Trajectory Design for Low-Thrust Restricted-Three-Body Problems”. en. In: *Journal of Guidance, Control, and Dynamics* 38.11 (Nov. 2015), pp. 2146–2160. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G000878. URL: https://arc.aiaa.org/doi/10.2514/1.G000878 (visited on 09/13/2022).

[49] Ehsan Taheri and Ossama Abdelkhalik. “Initial three-dimensional low-thrust trajectory design”. en. In: *Advances in Space Research* 57.3 (Feb. 2016), pp. 889–903. ISSN: 02731177. DOI: 10.1016/j.asr.2015.11.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117715008315 (visited on 09/14/2022).

[50] Ehsan Taheri, Ilya Kolmanovsky, and Ella Atkins. “Shaping low-thrust trajectories with thrust-handling feature”. en. In: *Advances in Space Research* 61.3 (Feb. 2018), pp. 879–890. ISSN: 02731177. DOI: 10.1016/j.asr.2017.11.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117717308013 (visited on 09/14/2022).

[51] Mingying Huo et al. “Initial Trajectory Design of Electric Solar Wind Sail Based on Finite Fourier Series Shape-Based Method”. In: *IEEE Transactions on Aerospace and Electronic Systems* 55.6 (Dec. 2019), pp. 3674–3683. ISSN: 1557-9603. DOI: 10.1109/TAES.2019.2906050.

[52] Zichen Fan et al. “Fast preliminary design of low-thrust trajectories for multi-asteroid exploration”. en. In: *Aerospace Science and Technology* 93 (Oct. 2019), p. 105295. ISSN: 12709638. DOI: 10.1016/j.ast.2019.07.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S1270963819303554 (visited on 09/14/2022).

[53] Wanmeng Zhou et al. “Low-Thrust Trajectory Design Using Finite Fourier Series Approximation of Pseudoequinoctial Elements”. en. In: *International Journal of Aerospace Engineering* 2019 (Nov. 2019), pp. 1–18. ISSN: 1687-5966, 1687-5974. DOI: 10.1155/

2019/1364834. URL: https://www.hindawi.com/journals/ijae/2019/1364834/ (visited on 09/13/2022).

[54] Andrea Caruso et al. “Optimal solar sail trajectory approximation with finite Fourier series”. en. In: *Advances in Space Research* 67.9 (May 2021), pp. 2834–2843. ISSN: 02731177. DOI: 10.1016/j.asr.2019.11.019. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117719308270 (visited on 09/14/2022).

[55] Nicholas P Nurre and Ehsan Taheri. “Multiple gravity-assist low-thrust trajectory design using finite Fourier series”. en. In: *AIAA/AAS Astrodynamics Specialist Conference, AAS 20-671*. 2020, p. 21.

[56] Yevhenii Kovryzhenko and Ehsan Taheri. “Comparison of Minimum-Snap and Finite Fourier Series Methods for Multi-Copter Motion Planning”. en. In: *AIAA SCITECH 2022 Forum*. San Diego, CA & Virtual: American Institute of Aeronautics and Astronautics, Jan. 2022. ISBN: 978-1-62410-631-6. DOI: 10.2514/6.2022-1085. URL: https://arc.aiaa.org/doi/10.2514/6.2022-1085 (visited on 09/13/2022).

[57] Jack W Langelaan et al. “The thesis of Zhenda Li was reviewed and approved by the following:” en. In: AIAA Scitech 2020 Forum, 2020. DOI: 10.2514/6.2020-0856.

[58] Yevhenii Kovryzhenko and Ehsan Taheri. “Quadcopter Trajectory Generation Using Finite Fourier Series and Polynomial Parameterizations”. en. In: *AIAA Journal of Guidance, Control, and Dynamics* under review (Sept. 2022).

[59] Jean Levine. *Analysis and Control of Nonlinear Systems: A Flatness-based Approach*. en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-642-00838-2 978-3-642-00839-9. DOI: 10.1007/978-3-642-00839-9. URL: https://link.springer.com/10.1007/978-3-642-00839-9 (visited on 05/24/2023).

[60] K.J. Kyriakopoulos and G.N. Saridis. “Minimum jerk path generation”. In: *1988 IEEE International Conference on Robotics and Automation Proceedings*. Apr. 1988, 364–369 vol.1. DOI: 10.1109/ROBOT.1988.12075.

[61] A. Piazzi and A. Visioli. "Global minimum-jerk trajectory planning of robot manipulators". In: *IEEE Transactions on Industrial Electronics* 47.1 (Feb. 2000), pp. 140–149. ISSN: 1557-9948. DOI: 10.1109/41.824136.

[62] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation". In: *IEEE Transactions on Robotics* 31.6 (Dec. 2015). Conference Name: IEEE Transactions on Robotics, pp. 1294–1310. ISSN: 1941-0468. DOI: 10.1109/TRO.2015.2479878.

[63] Yonghee Park, Woosung Kim, and Hyungpil Moon. "Time-Continuous Real-Time Trajectory Generation for Safe Autonomous Flight of a Quadrotor in Unknown Environment". en. In: *Applied Sciences* 11.7 (Apr. 2021), p. 3238. ISSN: 2076-3417. DOI: 10.3390/app11073238. URL: https://www.mdpi.com/2076-3417/11/7/3238 (visited on 09/14/2022).

[64] Behdad Davoudi et al. "Quad-Rotor Flight Simulation in Realistic Atmospheric Conditions". en. In: *AIAA Journal* 58.5 (May 2020), pp. 1992–2004. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J058327. URL: https://arc.aiaa.org/doi/10.2514/1.J058327 (visited on 09/14/2022).

[65] Z. Zuo. "Trajectory tracking control design with command-filtered compensation for a quadrotor". en. In: *IET Control Theory & Applications* 4.11 (Nov. 2010), pp. 2343–2355. ISSN: 1751-8644, 1751-8652. DOI: 10.1049/iet-cta.2009.0336. URL: https://digital-library.theiet.org/content/journals/10.1049/iet-cta.2009.0336 (visited on 09/14/2022).

[66] Xi Chen and Liuping Wang. *Cascaded model predictive control of a quadrotor UAV*. Journal Abbreviation: 2013 3rd Australian Control Conference, AUCC 2013 Pages: 359 Publication Title: 2013 3rd Australian Control Conference, AUCC 2013. Nov. 2013. ISBN: 978-1-4799-2497-4. DOI: 10.1109/AUCC.2013.6697298.

[67] Hao Lu et al. "Online optimisation-based backstepping control design with application to quadrotor". en. In: *IET Control Theory & Applications* 10.14 (2016). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cta.2015.0976, pp. 1601–1611. ISSN:

1751-8652. DOI: 10.1049/iet-cta.2015.0976. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-cta.2015.0976 (visited on 03/26/2023).

Appendices

Waypoints for all the cases are summarized. All dimensions are specified in meters or radians. Time per segment T_i notation is re-used to indicate a time-stamp of when a specific point i has to be reached. The exact same set of inputs was used to solve the fixed-time problems and are use as initial guesses for the time-allocation problems.

"Simple" trajectory: $T_1 = 0, X_1 = 1, T_2 = 3, X_2 = 3$.

"Figure-8" trajectory: $T_1 = 0, X_1 = 0, Y_1 = 0, Z_1 = 0, \psi_1 = 0.79$; $T_2 = 3.75, X_2 = 0.5, Y_2 = 1, Z_2 = 0.25, \psi_2 = 1.57$; $T_3 = 7.5, X_3 = 0, Y_3 = 2, Z_3 = 0.5, \psi_3 = 3.14$; $T_4 = 11.25, X_4 = -0.5, Y_4 = 1, Z_4 = 0.25, \psi_4 = 4.71$; $T_5 = 15, X_5 = 0, Y_5 = 0, Z_5 = 0, \psi_5 = 5.5$; $T_6 = 18.75, X_6 = 0.5, Y_6 = -1, Z_6 = -0.25, \psi_6 = 4.71$; $T_7 = 22.50, X_7 = 0, Y_7 = -2, Z_7 = -0.50, \psi_7 = 3.14$; $T_8 = 26.25, X_8 = -0.5, Y_8 = -1, Z_8 = -0.25, \psi_8 = 1.57$; $T_9 = 30, X_9 = 0, Y_9 = 0, Z_9 = 0, \psi_9 = 0.79$.

"Square" trajectory: $T_1 = 0, X_1 = 1, Y_1 = 1$; $T_2 = 1.25, X_2 = 0, Y_2 = 1$; $T_3 = 2.5, X_3 = -1, Y_3 = 1$; $T_4 = 3.75, X_4 = -1, Y_4 = 0$; $T_5 = 5, X_5 = -1, Y_5 = -1$; $T_6 = 6.25, X_6 = 0, Y_6 = -1$; $T_7 = 7.5, X_7 = 1, Y_7 = -1$; $T_8 = 8.75, X_8 = 1, Y_8 = 0$; $T_9 = 10, X_9 = 1, Y_9 = 1$.

"Circle" trajectory: $T_1 = 0, X_1 = 0, Y_0 = 2$; $T_2 = 1.43, X_2 = 1.56, Y_2 = 1.25$; $T_3 = 2.86, X_3 = 1.95, Y_3 = -0.45$; $T_4 = 4.29, X_4 = 0.87, Y_4 = -1.8$; $T_5 = 5.71, X_5 = -0.87, Y_5 = -1.8$; $T_6 = 7.14, X_6 = -1.95, Y_6 = -0.45$; $T_7 = 8.57, X_7 = -1.56, Y_7 = 1.25$; $T_8 = 10, X_8 = 0, Y_8 = 2$.

"3 Blocks" trajectory: $T_1 = 0, X_1 = 5.4, Y_1 = 3, Z_1 = -0.4$; $T_2 = 2.25, X_2 = 4.2, Y_2 = 1.6, Z_2 = -1.4$; $T_3 = 4.5, X_3 = 2.8, Y_3 = 1.6, Z_3 = -1.4$; $T_4 = 6.75, X_4 = 0.5, Y_4 = 1.8, Z_4 = -0.9$; $T_5 = 9, X_5 = 0.4, Y_5 = 0.8, Z_5 = -0.4$.