Online and Real-Time Trajectory Generation Method for Unforeseen Events Using a Modified Spline Approach

Mehmet Altan Toksöz, Mehmet Önder Efe, *Senior Member, IEEE*, and Veysel Gazi

***Abstract*—This letter focuses on the problem of online and instantaneous generation of smooth trajectories and their analytical time derivatives, where future reference signals provided by a trajectory planner or a user are a-priori unknown. For this purpose, we propose a modified cubic spline polynomial which transforms the instantaneous piecewise references into smooth and continuous functions for digital control systems. The resulting reference signals are not only smooth and continuous in time but also analytically differentiable. The inferred derivative values could then be used in a variety of linear and non-linear controllers such as, Proportional-Integral-Derivative (PID), sliding mode, and backstepping. We also provide optimum and boundary values for the parameters used in the proposed technique. In addition, we present a lightweight implementation (a few lines of code) of the algorithm by which real-time computations are rapidly performed using a microcontroller. Both simulation and real experiments demonstrate that for the plants having different degrees of complexities, the proposed approach with a PD / PID controller outperforms the conventional filtered derivative-based one in terms of overshoot in the step response, robustness against noise, and trajectory tracking error.**

***Index Terms*—Analytical time derivatives, modified cubic spline, online trajectory generation, PID control.**

I. Introduction

ROBOTIC agents must be able to quickly respond to the unforeseen events in order to achieve the goals in dynamic environments. These events can be triggered by a sensor or a user by means of a joystick or gas pedal (e.g., a single or sequence of step signals is generated autonomously or manually by a pilot when a drone is avoiding an obstacle in an unknown environment). If the reference command signals generated by these events are directly fed into the controllers, the robotic plant under control may not properly respond. Also, improper commands result in prolonged rise time and settling time in some cases or they may even lead to eventual instability of the total system. These problems arise a need for online generation of proper control commands or achievable trajectories. In addition, the generated outputs must also provide derivative(s) of the command signal since most controllers (PID, sliding mode, backstepping, etc.) require the derivative information. Online command generation is a challenging problem and there are valuable research outcomes in this field [1], [2], [3], [4], [5], [6], [7]. The main difficulties are: (i) only the present or instantaneous reference is available which limits the usage of polynomial-based approaches since they require future waypoints; (ii) real-time operation strictly limits the computational time.

One may attempt to solve online trajectory generation problem using differential flatness (DF)-based techniques. A dynamic system is called differentially flat if there exists a flat output such that the system variables (inputs, states) can be expressed in terms of flat output and its derivatives. Recently, DF-based online trajectory generation methods have been proposed in [8], [9] which utilize constrained optimization. Also, a continuous time DF-based trajectory generation method has been proposed for unmanned aerial vehicles (UAVs) [4]. Similarly, a real-time trajectory generation method for UAVs has been presented in [5], which generates feasible paths in the order of ten microseconds on a laptop computer. Clearly, most systems do not have DF property and in general, those methods require solving a constrained optimization problem having significant amount of iterations.

Setpoint filtering could be considered as another type of online command generation. It is commonly used in classical PID controllers. For instance, in order to reduce the overshoots due to the step-wise changes, low pass filters have been proposed in [10], [11]. Since those techniques do not provide the derivative information, the use of numerical derivatives such as backward difference method is inevitable. However, realistic scenarios are subject to noisy measurements, which typically require the use of first or second order filters. Selecting the correct bandwidth for each filter then emerges as a new problem to be addressed. Even if we carefully adjust those parameters, at the end, some performance degradation due to information loss at the numerical differentiation stage are inevitable. Command filtered backstepping is another approach for constructing the proper reference signal and its derivatives. For instance, a

Manuscript received 4 August 2022; accepted 13 November 2022. Date of publication 24 November 2022; date of current version 30 November 2022. This letter was recommended for publication by Associate Editor K. Leahy and Editor A. Bera upon evaluation of the reviewers' comments. *(Corresponding author: Mehmet Altan Toksoz.)*

Mehmet Altan Toksöz is with the Department of Robotics and Autonomous Systems, HAVELSAN A.Ş., 06510 Ankara, Turkey (e-mail: matoksoz@gmail.com).

Mehmet Önder Efe is with the Department of Computer Engineering, Hacettepe University, 06800 Ankara, Turkey (e-mail: onderefe@gmail.com).

Veysel Gazi is with the Department of Control and Automation Engineering, Yıldız Technical University, 34220 İstanbul, Turkey (e-mail: vgazi@yildiz.edu.tr).

Digital Object Identifier 10.1109/LRA.2022.3224658

command filtering technique has been proposed in [12] in order to eliminate the complexity of the implementation of backstepping due to computing derivatives. Another method proposed in [13] focuses on the elimination of the problems in backstepping under parameter uncertainties. In [14], the presented command filtering technique guarantees the finite-time convergence. Although those approaches simply provide the derivative information, they require the use of backstepping and initially the first derivative.

In this letter, we propose an online and instantaneous trajectory generation mechanism in which the mathematical properties of a modified cubic spline function are exploited to transform the desired reference signals/setpoints into smooth and differentiable analytic functions. The resulting reference signals and their derivatives can then be used in a wide range of controllers demanding position and velocity information. We demonstrate an efficient implementation of the proposed approach for the conventional controllers in general class of systems. The contributions of this work are fourfold:

- The proposed method is online, which does not require any future way-points in order to generate feasible trajectories.
- By applying our method, the effects of the sudden changes in the reference signals are absorbed and the destabilizing effects of the sensor noises are minimized.
- Our method is applicable to general class of systems since it does not require any specific property such as differential flatness.
- Since the proposed approach is a polynomial-based one, it does not require any high performance hardware and generates the output instantaneously using a simple micro-controller.

The rest of the letter is organized as follows: The proposed spline-based online trajectory generation method is explained in Section II. Simulations for a quadrotor application are presented in Section III, and real experiments using a 1-DOF helicopter are demonstrated in Section IV. Finally, Section V concludes the letter.

II. Online Trajectory Generation Method

Cubic splines are commonly used in computer graphics, image processing, control systems, path planning, etc. in order to obtain smooth motion curves from discontinuous or discrete data [15], [16], [17], [18], [19]. If the reference data is only approximately known, the smoothing spline method may be preferred [20], [21]. The main idea of cubic spline technique is to fit some data points $(t_1, x(t_1))$, $(t_2, x(t_2))$, ..., $(t_n, x(t_n))$ to a continuous piecewise function of the form

$$s(t) = \begin{cases} s_1(t) & t_1 \le t \le t_2 \\ s_2(t) & t_2 \le t \le t_3 \\ & \dots \\ s_{n-1}(t) & t_{n-1} \le t \le t_n \end{cases} \tag{1}$$

where $s_i(t)$ is a third order polynomial

$$s_i(t) = a_i(t-t_i)^3 + b_i(t-t_i)^2 + c_i(t-t_i) + d_i, \quad i = 1, 2, \dots, n-1. \tag{2}$$

The first and second derivatives of $s_i(t)$ are given as

$$\dot{s}_i(t) = 3a_i(t-t_i)^2 + 2b_i(t-t_i) + c_i, \quad i = 1, 2, \dots, n-1 \tag{3}$$

$$\ddot{s}_i(t) = 6a_i(t-t_i) + 2b_i, \quad i = 1, 2, \dots, n-1. \tag{4}$$

The properties of the cubic spline method suggest that $s(t)$ is continuous on the interval $[t_1, t_n]$ and $s_i(t_i) = x(t_i)$ for $i = 1, 2, \dots, n$. Using these properties, and depending on the constraints/requirements, it is also possible to design the spline (i.e. to determine its parameters) so that it has continuous derivatives on the interval $[t_1, t_n]$. Requiring that $s(t)$ and its first derivative $\dot{s}(t)$ are continuous on $[t_1, t_n]$ and using the third order Taylor's formula ($x(t) = x(a) + \dot{x}(a)(t-a) + \frac{1}{2!}\ddot{x}(a)(t-a)^2 + \frac{1}{3!}x^{(3)}(a)(t-a)^3,\ a \in \mathbb{R}$), the solution for the coefficients for $i = 1, 2, \dots, n-1$ are obtained as

$$a_i = -\frac{2}{h_0^3}(x(t_{i+1}) - x(t_i)) + \frac{1}{h_0^2}(\dot{x}(t_{i+1}) + \dot{x}(t_i)) \tag{5}$$

$$b_i = \frac{3}{h_0^2}(x(t_{i+1}) - x(t_i)) - \frac{1}{h_0}(\dot{x}(t_{i+1}) + 2\dot{x}(t_i)) \tag{6}$$

$$c_i = \dot{x}(t_i) \tag{7}$$

$$d_i = x(t_i) \tag{8}$$

where $h_0 := t_{i+1} - t_i$ and we assume equidistant time intervals [22]. The parameters in (5)–(8) lead to continuous spline with continuous derivative. However, they do not guarantee that the second derivative would be continuous. In case continuous second derivatives are also desired, then one may use a fifth order polynomial, which requires six parameters instead of four for each segment. Alternatively, in case the derivative information is not available, solving tridiagonal system of equations, one can obtain the derivative information from the known positional data [23]. In the latter scenario, cubic spline with continuous first and second derivatives is computed. However, in order to compute it, all the data points must be available in advance. Therefore, it cannot be used in systems in which the data is provided sequentially one at each step. Moreover, as mentioned earlier, in some applications the derivative information might not be available or very difficult to obtain. In this letter, based on a modified spline we develop a methodology which generates smooth references with continuous first and second derivatives using the available current and past data as the control of a physical system is ongoing. It has the potential to be usefully applied in various settings.

Let the reference value for any state of a physical system at time t_{i+1} be represented by $(t_{i+1}, x^d(t_{i+1}))$. We might design a spline polynomial $s_i(t)$ such that it fits the consecutive setpoints $x^d(t_i)$ and $x^d(t_{i+1})$ for each time interval $[t_i,\ t_{i+1}]$. However, we assume that the future data $(t_{i+k}, x^d(t_{i+k}))$ for $k = 2, 3, \dots$ is not available. Also, notice that the original solution set in (5)–(8) for the coefficients cannot be directly used for our purposes since it contains derivatives of the reference data points which are assumed to be unavailable.

Consider the following third order polynomial

$$q_i(t) = \alpha_i(t-t_i)^3 + \beta_i(t-t_i)^2 + \gamma_i(t-t_i) + \delta_i, \quad i = 1, 2, \dots, n-1. \tag{9}$$

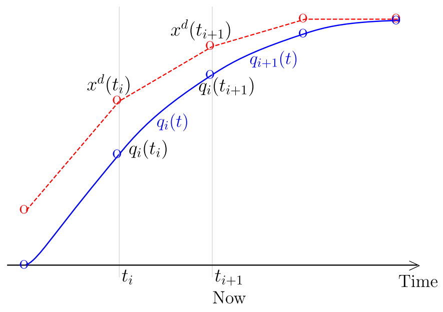

Fig. 1. Illustration of generated modified spline functions and the reference data points.

such that the coefficients are calculated using the modified set of equations

$$\alpha_i := -\frac{2}{h^3}(x^d(t_{i+1}) - q_i(t_i)) + \frac{2}{h^2}\dot{q}_i(t_i) \tag{10}$$

$$\beta_i := \frac{3}{h^2}(x^d(t_{i+1}) - q_i(t_i)) - \frac{3}{h}\dot{q}_i(t_i) \tag{11}$$

$$\gamma_i := \dot{q}_i(t_i) \tag{12}$$

$$\delta_i := q_i(t_i) \tag{13}$$

where $h \neq h_0$. Here, notice that the reference data at time t_i is replaced with $q_i(t_i)$. This means that the new consecutive reference data points for the modified spline are $q_i(t_i)$ and $x^d(t_{i+1})$. Also, the derivative of the new spline function at time t_{i+1} is assumed to be very close to the derivative of it at time t_i, that is, $\dot{q}_i(t_{i+1}) \approx \dot{q}_i(t_i)$. The generated modified spline functions and the reference data points for an arbitrary signal are illustrated in Fig. 1.

Now, we can state the following results about the properties of the new spline segment functions $q_i(t)$.

Lemma 1: For some $h > h_0 = t_{i+1} - t_i$, the coefficient set defined by (10)–(13) constructs a stable recursive filter at time t_{i+1} of the form

$$q_i(t_{i+1}) = \lambda_1 x^d(t_{i+1}) + (1-\lambda_1)q_i(t_i) + \lambda_2\dot{q}_i(t_i) \tag{14}$$

where $\lambda_1 \in (0,1)$ and $\lambda_2 \in \mathbb{R}$.

Proof: We know that for $h_0 = t_{i+1} - t_i$, the cubic spline, $q_i(t)$, at time t_{i+1} has the following form:

$$q_i(t_{i+1}) = \alpha_i h_0^3 + \beta_i h_0^2 + \gamma_i h_0 + \delta_i \tag{15}$$

If we substitute the coefficients defined in (10)–(13) into (15), we obtain

$$\begin{aligned} q_i(t_{i+1}) = &\left(\frac{3h_0^2}{h^2} - \frac{2h_0^3}{h^3}\right) x^d(t_{i+1}) \\ &+ \left(1 - \left(\frac{3h_0^2}{h^2} - \frac{2h_0^3}{h^3}\right)\right) q_i(t_i) \\ &+ \left(\frac{2h_0^3}{h^2} - \frac{3h_0^2}{h} + h_0\right)\dot{q}_i(t_i). \end{aligned} \tag{16}$$

Finally, replacing $\left(\frac{3h_0^2}{h^2} - \frac{2h_0^3}{h^3}\right)$ and $\left(\frac{2h_0^3}{h^2} - \frac{3h_0^2}{h} + h_0\right)$ by λ_1 and λ_2, respectively, we conclude the proof. Note that $\lambda_1 < 1$ since $h > h_0$.

Remark: The filter described in Lemma 1 is a smoothing filter. As λ_1 gets smaller, the smoothing effect gets stronger. This is implemented by increasing h while keeping the sampling rate, h_0, fixed.

Lemma 2: For $h_0/h = (1/2 \pm \sqrt{3}/6)$, the derivative of the modified spline, $\dot{q}_i(t)$, at time t_{i+1} produces approximate derivative of the reference data points $q_i(t_i)$ and $x^d(t_{i+1})$.

Proof: For $h_0 = t_{i+1} - t_i$, the derivative of $q_i(t)$ at time t_{i+1} has the form

$$\dot{q}_i(t_{i+1}) = 3\alpha h_0^2 + 2\beta h_0 + \gamma_i. \tag{17}$$

If we substitute the coefficients defined in (10)–(13) into (17), we obtain

$$\begin{aligned} \dot{q}_i(t_{i+1}) = &\left(\frac{6h_0}{h} - \frac{6h_0^2}{h^2}\right)\left(\frac{x^d(t_{i+1}) - q_i(t_i)}{h} - \dot{q}_i(t_i)\right) \\ &+ \dot{q}_i(t_i). \end{aligned} \tag{18}$$

If we equate the expression $(\frac{6h_0}{h} - \frac{6h_0^2}{h^2})$ to 1, we obtain that $h_0 = (1/2 \pm \sqrt{3}/6)h$, which provides the approximate numerical derivative as

$$\dot{q}_i(t_{i+1}) = \frac{x^d(t_{i+1}) - q_i(t_i)}{h} \tag{19}$$

which concludes the proof.

Lemma 3: For $h_0/h = 5/12$, the second derivative of the modified spline, $\ddot{q}_i(t)$, at time t_{i+1} produces approximate second derivative of the reference data points $q_i(t_i)$ and $x^d(t_{i+1})$.

Proof: For $h_0 = t_{i+1} - t_i$, the second derivative of $q_i(t)$ at time t_{i+1} has the form

$$\ddot{q}_i(t_{i+1}) = 6\alpha_i h_0 + 2\beta_i. \tag{20}$$

If we substitute the coefficients defined in (10)–(13) into (20), we obtain

$$\begin{aligned} \ddot{q}_i(t_{i+1}) &= \left(6 - \frac{12h_0}{h}\right)\left(\frac{1}{h}\right)\left(\frac{x^d(t_{i+1}) - q_i(t_i)}{h} - \dot{q}_i(t_i)\right) \\ &\approx \left(6 - \frac{12h_0}{h}\right)\left(\frac{\dot{x}^d(t_{i+1}) - \dot{q}_i(t_i)}{h}\right) \end{aligned} \tag{21}$$

If we equate the expression $(6 - \frac{12h_0}{h})$ to 1, we obtain that $h_0/h = 5/12$, which provides the approximate numerical second derivative as

$$\ddot{q}_i(t_{i+1}) \approx \frac{\dot{x}^d(t_{i+1}) - \dot{q}_i(t_i)}{h} \tag{22}$$

which concludes the proof.

Remark: For an appropriate computation of the second derivative, it is required that the term $(6 - \frac{12h_0}{h})$ in Lemma 3 is positive. As a result, we obtain $h_0/h < 0.5$. This result eliminates one of the solutions given as $h_0/h = (1/2 \pm \sqrt{3}/6)$ in Lemma 2. Therefore, the condition in Lemma 2 reduces to $h_0/h = (1/2 - \sqrt{3}/6)$, which is necessary for the first derivative.

Remark: The Lemma 2 and 3 present the optimal values and bounds for only the approximate first and second numerical derivatives. However, the optimal values for the real derivatives for the original reference signal may deviate. Practically, as the

Algorithm 1: Modified Cubic Spline (MCS).

```
Initialization:
$h_0 \leftarrow T$ (sampling period of the control system)
$h \leftarrow \sigma$ (a parameter which is larger than $2h_0$)
$q[0] \leftarrow x[0]$ (value of the measured state)
$q\_dot[0] \leftarrow 0$ (initialization of the derivative of the spline)
$k \leftarrow 0$ (time initialization)
Time loop $k$
 1:  read the external reference $x^d[k+1]$
 2:  $\alpha \leftarrow -\frac{2}{h^3}(x^d[k+1]-q[k]) + \frac{2}{h^2}\, q\_dot[k]$
 3:  $\beta \leftarrow \frac{3}{h^2}(x^d[k+1]-q[k]) - \frac{3}{h} q\_dot[k]$
 4:  $\gamma \leftarrow q\_dot[k]$
 5:  $\delta \leftarrow q[k]$
 6:  $q[k+1] \leftarrow \alpha h_0^3 + \beta h_0^2 + \gamma h_0 + \delta$
 7:  $q\_dot[k+1] \leftarrow 3\alpha h_0^2 + 2\beta h_0 + \gamma$
 8:  $q\_dot\_dot[k+1] \leftarrow 6\alpha h_0 + 2\beta$
 9:  Output : $q[k+1], q\_dot[k+1], q\_dot\_dot[k+1]$
10:  $k \leftarrow k+1$
```

ratio h_0/h gets smaller values than $5/12$, the second derivative fits better.

Theorem: Let $x^d(t_{i+1})$ be the reference data point at time t_{i+1}, h_0 be the sampling period of the control system, and $h > 0$ be a parameter. Let $q_i(t_i)$ and $\dot{q}_i(t_i)$ be the positional and velocity data at time t_i, respectively. The coefficient set in (10)–(13) generates a cubic spline of the form in (9) such that the generated cubic spline predicts the first order derivative information for $h_0/h = (1/2 - \sqrt{3}/6)$ in (17) and second order derivative information for $h_0/h = 5/12$ in (20).

Proof: See Lemma 1, Lemma 2, and Lemma 3.

Remark: Although the optimum values of h_0/h for the prediction of the first and second order derivatives are provided, we may need to use smaller h_0/h in order to smooth the discontinuous references such as step input. In this way, we provide that the first and second order derivatives are also smoothed.

The modified cubic spline-based procedure can be summarized in Algorithm 1. Here, at the next sampling instance, t_{i+1}, the controllers use $q_i(t_{i+1})$, $\dot{q}_i(t_{i+1})$, and $\ddot{q}_i(t_{i+1})$ instead of $x^d(t_{i+1})$ and its first and second order filtered derivatives.

As a use case, we consider a square wave reference signal since it is very suitable for representing sudden changes in the dynamic environments. For this purpose, we choose the sampling period (h_0) as 0.005 seconds and the parameter h as $h_0/(1/2 - \sqrt{3}/6)$. Notice that the analytic derivative of the reference signal is zero when the signal is constant and impulse when it changes its state. Using the proposed method, we obtain the modified spline fit and the derivative as shown in Fig. 2. We also included filtered numerical differentiation. This figure shows that the modified cubic spline method successfully generates smooth and differentiable results and converges to the reference.

III. Quadrotor Simulation Studies

In the previous sections, we have presented and analyzed the modified cubic spline-based online trajectory generation approach as a general framework for control system applications. In order to show the benefits of the mechanism, here we designed a set of quadrotor flight simulations. A quadrotor helicopter is an unstable and under-actuated 6-DOF plant. Control of this system has been extensively studied in the literature [24], [25], [26], [27], [28], [29], [30]. Here, we use the dynamics and kinematics model described in [24]. For the control approach, we use simple PD controllers with feedforward term for the attitude (ϕ, θ, ψ), altitude (z), and position (x, y) control as described in [24], [31].

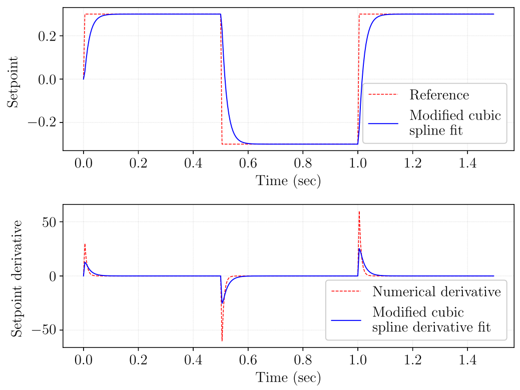

Fig. 2. A square wave reference signal, modified cubic spline fit and corresponding derivatives using numerical and modified spline derivative fitting.

TABLE I
Crazyflie 2.0 Nano-Quadrotor Parameters

Parameter	Value
Quadrotor mass (m)	0.033 kg
Arm length (l)	39.73e-3 m
Moment of inertia about x-axis (I_x)	1.395e-5 $kg\ m^2$
Moment of inertia about y-axis (I_y)	1.436e-5 $kg\ m^2$
Moment of inertia about z-axis (I_z)	2.173e-5 $kg\ m^2$
Inertia matrix (I)	$[I_x\ 0\ 0; 0\ I_y\ 0; 0\ 0\ I_z]$
Rotor's inertia (J_r)	0
Minimum rotor speed (ω_{min})	0 rad/s
Maximum rotor speed (ω_{max})	3000 rad/s
Motor constant (k_F)	2.8799e-08
Moment constant (k_M)	7.2385e-10
Motor gain (τ_m)	20 s^{-1}

A. Simulation Settings

In this part, we conducted several simulations using Crazyflie 2.0 nano-quadrotor parameters. The parameters are presented in Table I [32]. We compare the performances of a MCS (Modified Cubic Spline)-based quadrotor and a conventional one having filtered derivative-based control mechanism. Please note that the quadrotors are totally identical except the setpoints are processed and the derivatives are generated by the proposed techniqe for a MCS-based quadrotor. In addition, the derivate of the error is filtered in the conventional quadrotor when applying a PD controller. The PD controller with feedforward term in the conventional quadrotor can be written in the following Laplace transform form for attitude, altitude, and position control:

$$U^{pd}(\mathrm{s}) = k_p(X_d - X) + k_d \frac{\mathrm{s}(X_d - X)}{1 + \tau_1 \mathrm{s}} + \frac{\mathrm{s}^2 X_d}{1 + \tau_2 \mathrm{s}} \qquad (23)$$

TABLE II
CONTROLLER GAINS AND TIME CONSTANTS

Parameter	MCS-based	Optimized PD Controller
$k_{p,\phi}$	59.98	22.89
$k_{p,\theta}$	59.98	22.89
$k_{p,\psi}$	59.98	22.89
$k_{p,x}$	15.12	5.80
$k_{p,y}$	15.12	5.80
$k_{p,z}$	37.12	23.12
$k_{d,\phi}$	18.35	7.36
$k_{d,\theta}$	18.35	7.36
$k_{d,\psi}$	18.35	7.36
$k_{d,x}$	8.14	2.58
$k_{d,y}$	8.14	2.58
$k_{d,z}$	8.49	6.51
τ_1 for inner (attitude) controller	-	0.02
τ_2 for inner (attitude) controller	-	0.99
τ_1 for outer (position) controller	-	0.008
τ_2 for outer (position) controller	-	1.4e-4
$\frac{h_0}{h}$ for inner (attitude) controller	0.060	-
$\frac{h_0}{h}$ for outer (position) controller	0.025	-

In implementing the control law in (23), we need to optimize the values of k_p, k_d, τ_1, and τ_2. In order to get the best performance, we used Particle Swarm Optimization (PSO) approach. In the optimization stage, we implemented PSO method with default settings, that is, particle swarm size is 20 and 30 iterations for termination, [33]. In the MCS-based method, the search is performed for k_p, k_d, and the ratio, h_0/h, where h_0 is known. Note that the value of h_0/h can also be selected intuitively based on the approach given in this letter. In order to quantify the performance, we used the Integral of Square root of Time weighted Absolute Error (ISTAE) measure which can be given

$$\text{ISTAE} = \int_0^\infty \sqrt{t}|e(t)|\mathrm{d}t. \tag{24}$$

ISTAE measure was selected as it weighs the initial error better. To effectively determine the time constants of the derivative filter, we also added zero-mean sensor noise with standard deviation 0.01 to the system. For both quadrotors, the determined gains and time constants are given in Table II. For all simulations, we set the sampling frequency to 100 Hz, i.e. $h_0 = 0.01$ sec. In the simulations, where the controllers are sole PD ones, the maximum allowable values for the roll and pitch angles were ± 0.5 radians whereas these bounds are extended to ± 0.8 radians in the proposed technique. Beyond these values, it is observed that the closed loop system cannot maintain the stability and it is apparent that the proposed technique enlarges the maneuverability of the quadrotor.

B. Step Response

Using the configuration constructed in the previous part, we compared the step responses of two quadrotor models. For this, first, we tested the attitude performances of them. For both quadrotors, we set the reference points to 0.1 rd for all angles (yaw, pitch, roll) and simulated the scenarios for 2.5 seconds. The results for the conventional filtered derivative-based quadrotor are shown in Fig. 3(a). We observe that the rise time for all angles is about 0.1 seconds. We also observe about 40 percent overshoot for all of them. The amount of the overshoot for the pitch angle is slightly larger. For this quadrotor,

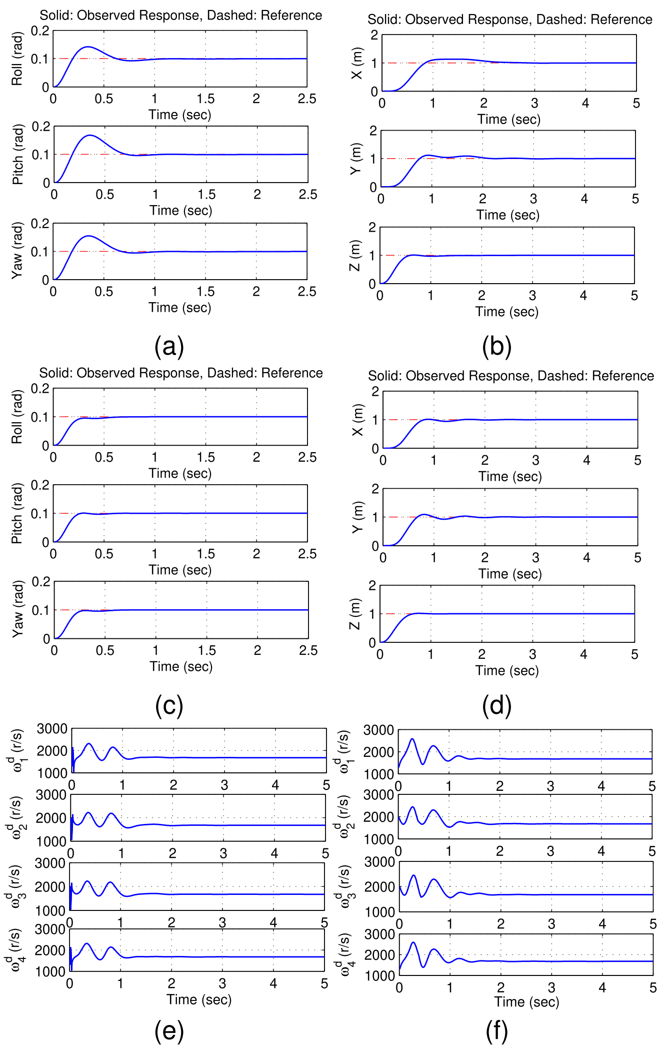

Fig. 3. (a) Attitude (roll, pitch, yaw) step response and (b) position (x, y, z) step response for conventional quadrotor. (c) Attitude step response and (d) position step response for MCS-based quadrotor. (e) Corresponding desired rotor speeds for conventional and for (f) MCS-based quadrotor.

it takes approximately 1 s to settle down to the desired attitude. The attitude step response results for the proposed MCS-based quadrotor is shown in Fig. 3(c). For this case, the observed rise time is about 0.25 seconds for all angles. We notice that the response of this quadrotor is slower. However, the amount of overshoot is nearly zero. Also, this quadrotor settles to its desired attitude angles within 0.5 seconds. In this part, in order to compare the altitude and position control performances of the two quadrotors, we conducted another set of experiments. For this, we set the reference positions and altitude to 1 m. Fig. 3(b) shows the results for the conventional quadrotor. We observe that the rise time for the position (x, y) is about 0.8 seconds and the amount of overshoot is 13 percent. The quadrotor settles its position within 2 seconds. The results for the proposed quadrotor is shown in Fig. 3(d). This time, the measured rise time is about 0.7 seconds which shows that it is slightly faster. The amount of overshoot is about 5 percent and the settling time is 1.5 seconds, approximately. The quadrotors settle their desired altitude values within 0.6 seconds with almost no overshoot. The corresponding desired rotor speeds are presented in Fig. 3(e) and Fig. 3(f) in order to show that both controllers produce proper control signals. The reason that the proposed method outperforms the conventional one in terms of overshoot is because the PD controller is directly exposed to the step signal in the latter case. However, it is smoothed and its smooth analytical derivatives are used in the proposed technique.

C. Step Response Under Noise

In this part, we studied another set of simulations in order to compare the performances of the quadrotors under both process and sensor noise. The setup for these simulations is the same as that of previous ones. While sensor noise was applied to all states of the quadrotors, the process noise was added to the rotor commands. We set the standard deviation of the sensor noise to 0.03 and process noise to 5. Both of the processes were generated using zero-mean Gaussian distribution. The results are shown in Fig. 4. We observe very large fluctuations in the attitude response of the conventional quadrotor (Fig. 4(a)). However, as shown in Fig. 4(c), there are very tiny ripples in the attitude response of the quadrotor employing the proposed technique. As far as the position step responses are considered, there are significant differences. As shown in Fig. 4(b), the applied noise destabilizes the conventional quadrotor. However, the effect of the noise is almost absorbed in the proposed MCS-based quadrotor (Fig. 4(d). In order to obtain quantative comparisons, we repeated the simulations 5 times, and in all cases, we observed that the conventional quadrotor was destabilized and the MCS-based quadrotor stayed stable. We also observe that the conventional PD controller produces improper control signals as shown in Fig. 4(e). However, the controller output of the MCS-based quadrotor (see Fig. 4(f)) is similar to the previous case.

D. Trajectory Tracking Performance

The last set of simulations were performed in order to compare the trajectory tracking capabilities of the quadrotors. For this, we generated two sine waves having 0.1 and 0.2 Hz frequencies, respectively. Each quadrotor initally starts at $(0, 0)$ in $x - y$ axis. While the amplitude of both references changes between 5 and -5 meters in x-axis, the amplitude in y-axis increases linearly from 0 to 50 meters. Fig. 5(a) and Fig. 5(b) show the results for the conventional quadrotor for both frequencies. While it tracks the reference successfully in the first case, the tracking performance significantly reduces if the frequency is increased. On the other hand, as shown in Fig. 5(c) and Fig. 5(d), for the proposed quadrotor, the successful tracking performance is maintained in the higher frequency case. We also repeated the simulations for agile frequencies (0.3, 0.4, and 0.5 Hz). For this case, we adjusted h_0/h to 0.50 and 0.012 for inner and outer controllers, respectively. The resulting mean square error values could be seen in Table III. We observe that the MCS-based quadrotor significantly outperforms the conventional one for agile trajectories.

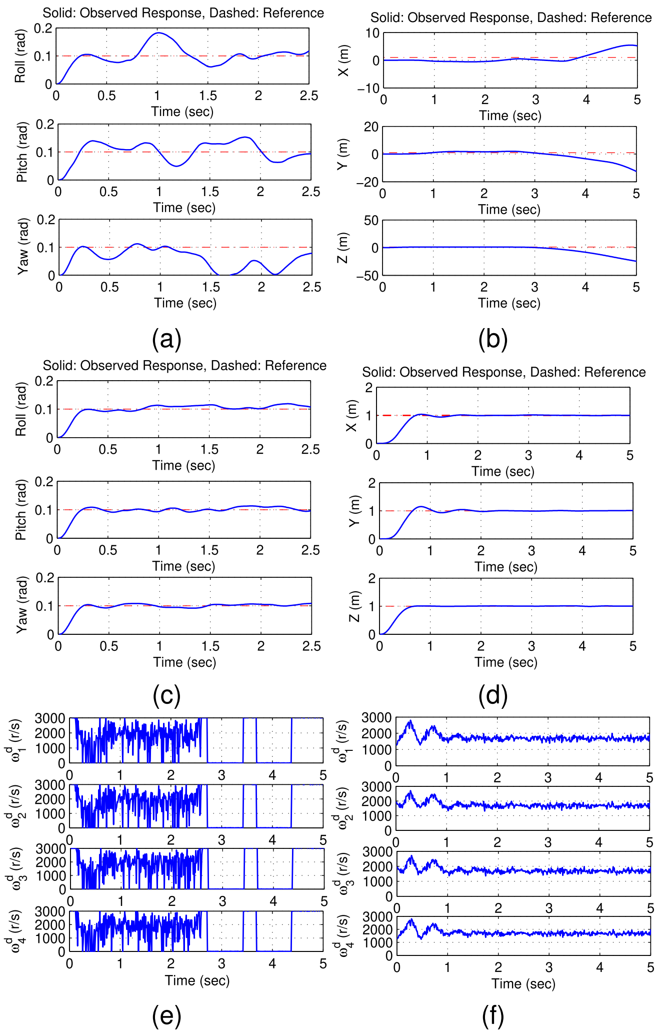

Fig. 4. Experiments under process and sensor noise. (a) Attitude (roll, pitch, yaw) step response and (b) position (x, y, z) step response for conventional quadrotor. (c) Attitude step response and (d) position step response for MCS-based quadrotor. (e) Corresponding desired rotor speeds for conventional and for (f) MCS-based quadrotor.

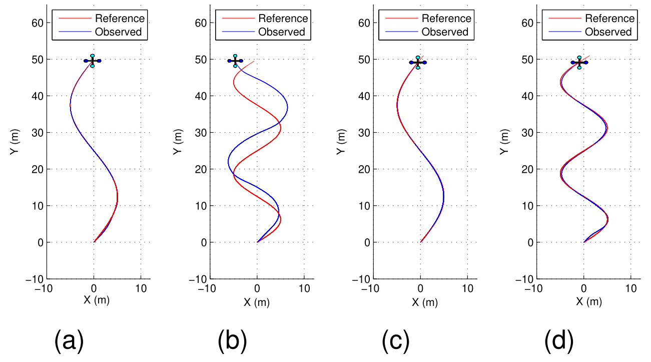

Fig. 5. Trajectories tracked by the conventional quadrotor when the trajectory frequency is (a) 0.1 Hz, (b) 0.2 Hz. Trajectories tracked by the MCS-based quadrotor when the trajectory frequency is (c) 0.1 Hz, (d) 0.2 Hz.

TABLE III
MEAN SQUARE ERRORS FOR DIFFERENT FREQUENCIES

Quadrotor	0.1 Hz	0.2 Hz	0.3 Hz	0.4 Hz	0.5 Hz
MCS-based	**0.0035**	**0.0623**	**2.3057**	**3.1485**	**5.2050**
Opt. PD-based	0.0081	9.4134	12.4438	9.8825	9.6740

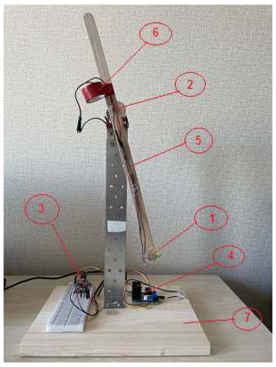

Fig. 6. 1-DOF Helicopter, components: 1. Brushed DC Motor 2. Potentiometer 3. Microcontroller 4. Motor Driver 5. Shaft 6. Balancing Load 7. Base of the System.

IV. Real Experiments Using a 1-DOF Helicopter

In order to validate the proposed technique, we designed an experimental setup to stabilize the angular position of a 1-DOF helicopter system, which can be seen in Fig. 6. The propulsion of the system is provided by a high speed (45000 rpm) brushed coreless DC motor which has an operation voltage range varying between 3 and 3.7 volts. The angle of the system is measured by a 10 k ohms analog potentiometer. The measured angle is read by a 12 bits analog to digital converter (ADC) pin of the microcontroller and it is normalized between 0 and 1. Please note that here the angle value is proportional to the output voltage of the potentiometer. We used a Raspberry Pi Pico as the microcontroller of the system on which the conventional and MCS-based PID controllers were implemented using MicroPython programming language. The sampling rate of the controller was set to 200 Hz ($h_0 = 0.005$). The output of the controller is limited between -1 and 1 to avoid large changes in the system. Because the DC motor requires pulse width modulation (PWM) signal to operate, the controller output (u) is converted to the PWM value using the following equation.

$$PWM = \frac{PWM_{\max} - PWM_{\min}}{u_{\max} - u_{\min}} u + PWM_{null} \quad (25)$$

where $PWM_{\max}$, $PWM_{\min}$, $u_{\max}$, $u_{\min}$, and PWM_{null} are set to 1, 0, 1,-1, and 0.5, respectively. Finally, the resulting PWM value is multiplied by $2^{12} - 1$ (max digital PWM signal) and send to the PWM output pin of the microcontroller. Note that the frequency of the PWM signal was set to 50 kHz. Normally, the microcontroller cannot generate enough power to drive the motor. Therefore, we used a L298 N motor driver connected to the PWM pin of the microcontroller and its input voltage value is adjusted to 6.4 Volts. The reason is because the driver itself also dissipates power, which causes 3 Volts voltage drop. The length of the long and short part of the rotating shaft are measured about

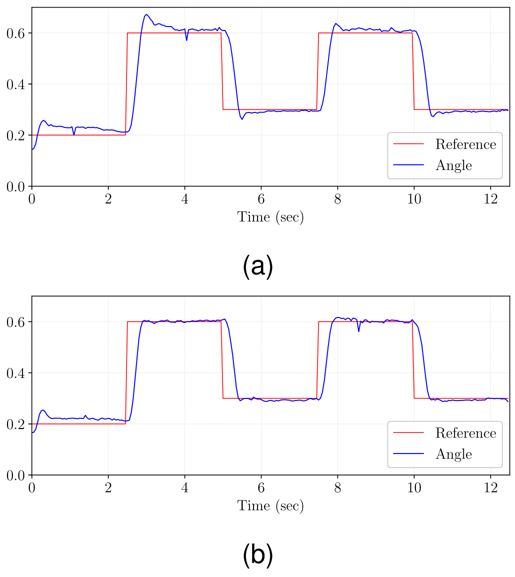

Fig. 7. (a) Step responses of the conventional PID-based helicopter. (b) Step responses of the MCS-based helicopter.

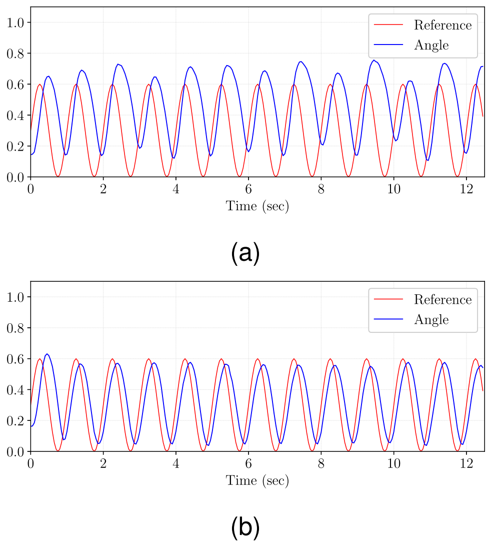

Fig. 8. (a) Trajectory (1 Hz) tracked by the conventional PID-based helicopter. (b) Trajectory tracked by the MCS-based helicopter.

24.5 cm and 15 cm, respectively. In the beginning of the short part, there is a small load to reduce the required thrust provided by the motor.

In the first set of physical experiments, we tested the step response of the system by applying sequential angle set points which are 0.2, 0.6, 0.3, 0.6, and 0.3, respectively. The coefficients of the PID controllers were adjusted using Zigler Nichols method and then fine tuned by minimizing the cumulative error. The resulting coefficients for the conventional PID controller are 28, 2.5, 15, respectively. In Fig. 7(a), we see the sequential step responses of the system and observe the expected overshoots, where the conventional PID controller is used. The coefficients of the MCS-based PID were found as 45, 4.5, 20, respectively. Also note that the parameter h was adjusted to 0.05. For this case, the resulting step responses can be seen in Fig. 7(b) where we see that the overshoots are almost eliminated. For the conventional and MCS-based system, we also observed the cumulative errors as 105.49 and 82.51, respectively between set points and system state for 12.25 seconds. We see that the MCS-based controller significantly reduces the error.

In the last set of experiments, we tested the trajectory tracking performances of both techniques. Please note that here we use the same PID coefficients as those of previous one. The reference trajectory is generated using a sinusoidal signal $0.3(\sin(2\pi ft)+1)$, where f is 1 Hz. The resulting trajectories tracked by the conventional and MCS-based controllers can be seen in Fig. 8. The corresponding cumulative errors were computed as 677.24 and 375.82 again for 12.25 seconds, respectively. We see that the MCS-based controller significantly outperforms the conventional one.

V. Conclusion

In this letter, we proposed a modified cubic spline-based online and instantaneous trajectory generation method for unforeseen events in order to smooth reference signals and generate their analytical derivatives. The proposed method is lightweight polynomial based and it requires only one parameter to be determined. The optimum and boundary values are presented for the sole parameter of the proposed method. Both simulation and real experiments for different class of systems showed that the controllers using the proposed technique significantly outperform the conventional filtered derivative-based ones, in terms of overshoot in the step response, robustness to noise, and trajectory tracking error. The approach presented here can be preferred in applications where the reference signals are unknown a-priori and the derivative information is essential. As a future work, we plan to test the robustness of the method under controller parameter uncertainty and sensor delays.

References

[1] S. Joos, M. Bitzer, R. Karrelmeyer, and K. Graichen, “Constrained online trajectory planning for nonlinear flat siso systems using a switched state variable filter,” *Automatica*, vol. 110, 2019, Art. no. 108583.

[2] B. Lindqvist, S. S. Mansouri, A.-A. Agha-mohammadi, and G. Nikolakopoulos, “Nonlinear MPC for collision avoidance and control of uavs with dynamic obstacles,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6001–6008, Oct. 2020.

[3] T. Kröger and F. M. Wahl, “Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events,” *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[4] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online uav replanning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5332–5339.

[5] M. Hehn and R. D. Andrea, “Real-time trajectory generation for quadrocopters,” *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 877–892, Aug. 2015.

[6] A. Gasparetto and V. Zanotto, “Optimal trajectory planning for industrial robots,” *Adv. Eng. Softw.*, vol. 41, no. 4, pp. 548–556, 2010.

[7] B. Li and Z. Shao, “Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots,” *Adv. Eng. Softw.*, vol. 87, pp. 30–42, 2015.

[8] F. Gao, W. Wu, W. Gao, and S. Shen, “Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments,” *J. Field Robot.*, vol. 36, no. 4, pp. 710–733, 2019.

[9] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss, “Observability-aware trajectory optimization for self-calibration with application to uavs,” *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1770–1777, Jul. 2017.

[10] T. Hägglund, “Signal filtering in pid control,” *IFAC Proc. Vol.*, vol. 45, no. 3, pp. 1–10, 2012.

[11] V. Vijayan and R. C. Panda, “Design of a simple setpoint filter for minimizing overshoot for low order processes,” *ISA Trans.*, vol. 51, no. 2, pp. 271–276, 2012.

[12] J. A. Farrell, M. Polycarpou, M. Sharma, and W. Dong, “Command filtered backstepping,” *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1391–1395, Jun. 2009.

[13] J. Yu, P. Shi, W. Dong, and H. Yu, “Observer and command-filter-based adaptive fuzzy output feedback control of uncertain nonlinear systems,” *IEEE Trans. Ind. Electron.*, vol. 62, no. 9, pp. 5962–5970, Sep. 2015.

[14] J. Yu, L. Zhao, H. Yu, C. Lin, and W. Dong, “Fuzzy finite-time command filtered control of nonlinear systems with input saturation,” *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2378–2387, Aug. 2018.

[15] K. Judd and T. McLain, “Spline based path planning for unmanned air vehicles,” in *Proc. AIAA Guid., Navig., Control Conf. Exhibit*, 2001, Art. no. 4238 .

[16] Y. Mezouar and F. Chaumette, “Path planning for robust image-based control,” *IEEE Trans. Robot. Automat.*, vol. 18, no. 4, pp. 534–549, Aug. 2002.

[17] B. Lau, C. Sprunk, and W. Burgard, “Kinodynamic motion planning for mobile robots using splines,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 2427–2433.

[18] H. Liu, X. Lai, and W. Wu, “Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints,” *Robot. Comput.- Integr. Manuf.*, vol. 29, no. 2, pp. 309–317, 2013.

[19] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, “Trajectory planning in robotics,” *Math. Comput. Sci.*, vol. 6, no. 3, pp. 269–279, 2012.

[20] C. H. Reinsch, “Smoothing by spline functions,” *Numerische Mathematik*, vol. 10, no. 3, pp. 177–183, 1967.

[21] H. J. Woltring, “A fortran package for generalized, cross-validatory spline smoothing and differentiation,” *Adv. Eng. Softw.*, vol. 8, no. 2, pp. 104–113, 1986.

[22] G. Wolberg, “Cubic spline interpolation: A review,” Dept. Comput. Sci. Columbia Univ., 1988.

[23] S. McKinley and M. Levine, “Cubic spline interpolation,” *College Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[24] S. Bouabdallah, A. Noth, and R. Siegwart, “PID vs LQ control techniques applied to an indoor micro quadrotor,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2004, pp. 2451–2456.

[25] G. Bo, L. Xin, Z. Hui, and W. Ling, “Quadrotor helicopter attitude control using cascade PID,” in *Proc. IEEE Chin. Control Decis. Conf.*, 2016, pp. 5158–5163.

[26] S. Bouabdallah and R. Siegwart, “Full control of a quadrotor,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2007, pp. 153–158.

[27] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, 2012.

[28] S. Bouabdallah and R. Siegwart, “Backstepping and sliding-mode techniques applied to an indoor micro quadrotor,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2005, pp. 2259–2264.

[29] M. A. Toksöz, S. Oğuz, and V. Gazi, “Decentralized formation control of a swarm of quadrotor helicopters,” in *Proc. IEEE 15th Int. Conf. Control Automat.*, 2019, pp. 1006–1013.

[30] M. Ö. Efe, “Neural network assisted computationally simple PID control of a quadrotor uav,” *IEEE Trans. Ind. Inform.*, vol. 7, no. 2, pp. 354–361, May 2011.

[31] Y. Kovryzhenko and E. Taheri, “Comparison of minimum-snap and finite fourier series methods for multi-copter motion planning,” in *Proc. AIAA SCITECH 2022 Forum*, 2022, Art. no. 1085.

[32] J. Förster, “System identification of the crazyflie 2.0 nano quadrocopter,” B.S. thesis, ETH Zurich, Inst. Dyn. Syst. Control, Zürich, Switzerland. 2015.

[33] M. I. Solihin, L. F. Tack, and M. L. Kean, “Tuning of pid controller using particle swarm optimization (pso),” *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 1, no. 4, pp. 458–461, 2011.