

Lightweight Tracking Control for Computationally Constrained Aerial Systems with the Newton-Raphson Method

Evanns Morales-Cuadrado, Luke Baird, Yorai Wardi, and Samuel Coogan

Abstract—We investigate the performance of a lightweight tracking controller, based on a flow version of the Newton-Raphson method, applied to a miniature blimp and a mid-size quadrotor. This tracking technique admits theoretical performance guarantees for certain classes of systems and has been successfully applied in simulation studies and on mobile robots with simplified motion models. We evaluate the technique through real-world flight experiments on aerial hardware platforms subject to realistic deployment and onboard computational constraints. The technique’s performance is assessed in comparison with established baseline control frameworks of feedback linearization for the blimp, and nonlinear model predictive control for both the quadrotor and the blimp. The performance metrics under consideration are (i) root mean square error of flight trajectories with respect to target trajectories, (ii) algorithms’ computation times, and (iii) CPU energy consumption associated with the control algorithms. The experimental findings show that the Newton-Raphson-based tracking controller achieves competitive or superior tracking performance to the baseline methods with substantially reduced computation time and energy expenditure.

Index Terms—Tracking control, quadrotor applications, blimp applications, model predictive control, feedback linearization.

I. Introduction

THE past two decades have witnessed a significant shift in the nature of hardware research for trajectory control of aerial platforms like quadrotors. Initially, testing and verification of novel techniques relied heavily on numerical simulators, later transitioning to real-world deployments that depended on ground station computers and simplified models; see [1]. Later developments of powerful single-board computers have enabled research to shift towards onboard execution even for computationally intensive control techniques [2]–[4]. The use of such computers can be credited for the maturation of several control techniques, of which Nonlinear Model Predictive Control (NMPC) has emerged as the state-of-the-art framework for quadrotor control [5].

The NASA University Leadership Initiative (grant #80NSSC20M0161) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity.

The authors are with the School of Electrical and Computer Engineering at the Georgia Institute of Technology, {egm,lbaird38,ywardi,sam.coogan}@gatech.edu. S. Coogan is also with the School of Civil and Environmental Engineering.

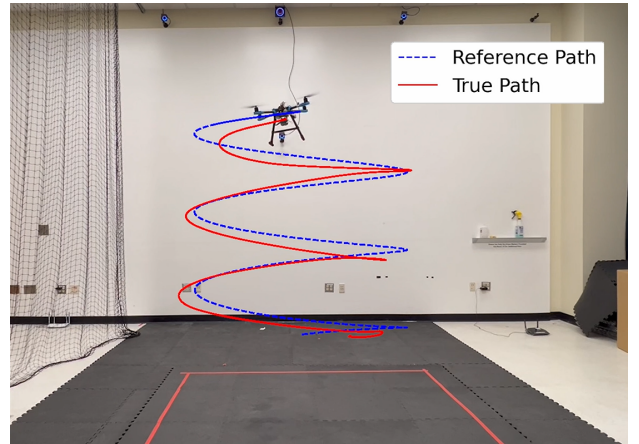


Fig. 1. Helix trajectory tracking with the Newton-Raphson-based tracking technique.

Advances in computing have also impacted the development of controllers for other kinds of Unmanned Aerial Vehicles (UAVs). In the realm of blimp research, older methods [6] were computationally limited to simplified linear models in order to perform model predictive control, but more recent works have reported on implementations of NMPC with high-fidelity models [7]. Reinforcement learning-based controllers that improve upon existing PID tracking controllers have also been implemented for blimps [8].

However, much of the state-of-the-art research has become dependent on exploiting powerful but relatively costly computing platforms. This renders them unrealistic for certain applications. As noted in [4], it may be “impractical to run NMPC on some miniature aerial vehicles with a limited computational budget.”

All of this suggests an opening for exploration of effective tracking-control techniques that are computationally lightweight enough so as not to require top-of-the-line hardware for their onboard implementations. In particular, such techniques ought to be sufficiently general to be easily deployable on various hardware platforms with minimal alterations, and be capable of low-level control at high frequencies.

With this in mind, authors of this paper have investigated a recently-proposed concept for tracking control that

is based on efficient computations. It is comprised of three elements: the Newton-Raphson method for computing the roots of equations, a single-instance lookahead for predictions, and a simple mechanism for speeding up the resulting controller's action. All three elements will be explained in detail in the sequel, where we argue for the efficiency and effectiveness of their combination as defined in [9], [10], for suitable applications.

The main objective of this paper is to investigate comparative performance tradeoffs between the tracking technique proposed in [10] and established tracking controllers which have had widespread success in applications: NMPC for quadrotors and blimps, and additionally, feedback linearization for blimps. The considered comparative performance metrics are Root Mean Square (tracking) Errors (RMSE), CPU times required by the controllers, and the energy spent by them during common control experiments. Results of several experiments have been tabulated and explained in the sequel, and they suggest that the tracking technique presented in [10] may be a viable alternative to the aforementioned established techniques under certain conditions. An illustration of the technique in action is provided in Fig. 1, which depicts quadrotor trajectory tracking using the proposed method.

The rest of the paper is organized as follows. Section II explains the tracking methodology proposed in [10], recounts relevant results from the literature, provides a statement of innovation in the present paper, and comments on implementation details for its experimental framework. Section III and Section IV present experimental results for a blimp and a quadrotor, respectively, and Section V summarizes the results and concludes the paper.¹

II. Methodology

The tracking-control technique presented in this paper was first introduced in [9] and further developed in [10]. Much of the initial discussion in this section follows [10], and consequent results are summarized in later citations.

We refer to the tracking control technique proposed in [10] by the “NR-based tracking technique”, or “NR-based technique” for brevity. The term “NR” highlights the fact that the tracking controller is founded on the Newton-Raphson method for finding roots of equations, as described next.

A. Fundamental ideas underscoring the NR-based tracking technique

The setting for the tracking technique is a continuous-time dynamical system defined by an ordinary differential equation as depicted in Figure 2, where $u(\cdot)$, $y(\cdot)$ and $r(\cdot)$ denote the input to the plant, output of the system, and reference signal, respectively. We assume that $t \in [0, \infty)$, and that for every $t \geq 0$, $u(t)$, $y(t)$ and $r(t)$ are in a Euclidean space $\mathbb{R}^m$ of a common dimension $m > 0$. The

state variable of the plant, $x(t) \in \mathbb{R}^n$, is internal to the plant, and its dimension, $n > 0$, need not be equal to m . The objective of the controller is to ensure that the output $y(\cdot)$ tracks the reference signal $r(\cdot)$ in a suitable sense defined below.

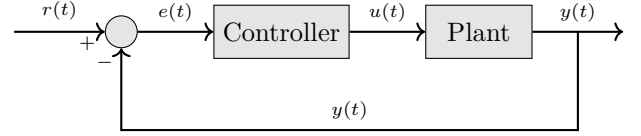


Fig. 2. Basic feedback control system.

We start the discussion with the following simple scenario in order to highlight the basic ideas behind the NR-based tracking technique. Suppose that the plant subsystem in Figure 2 is defined by a memoryless nonlinearity of the form

$$y(t) = g(u(t)), \quad (1)$$

where the function $g : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is continuously differentiable. Suppose also that the target trajectory is a constant, namely $r(t) \equiv r$ for a given $r \in \mathbb{R}^m$. Then the tracking controller is defined as

$$\dot{u}(t) = \left(\frac{dg}{du}(u(t)) \right)^{-1} (r - g(u(t))), \quad (2)$$

and we recognize it as a continuous-time flow version of the traditional iterative Newton-Raphson method for solving the equation $r - g(u) = 0$ in the variable $u \in \mathbb{R}^m$,

$$u_{i+1} = u_i - \left(\frac{\partial g}{\partial u}(u_i) \right)^{-1} (r_i - g(u_i)),$$

$i = 0, 1, 2, \dots$. For further discussion on the relationship between iterative algorithms and their “fluid-flow” variants, please see [10].

In the present discussion we implicitly assume that the Jacobian $\frac{dg}{du}(u)$ is nonsingular along the trajectory $u(\cdot)$ and other regularity conditions are satisfied, as pointed out in [10].

Define the function $V : \mathbb{R}^m \rightarrow \mathbb{R}^+$ by

$$V(u) := \frac{1}{2} \|r - g(u)\|^2. \quad (3)$$

Under the regularity assumptions in [10], convergence of $u(t)$ to a root of the equation $r - g(u) = 0$ is proven by taking the derivative $\dot{V}(u(t))$ in (3). To see this point, combine (2) – (3) to obtain

$$\begin{aligned} \dot{V}(u(t)) &= -\langle r - g(u(t)), \frac{dg}{du}(u(t)) \dot{u}(t) \rangle \\ &= -\langle r - g(u(t)), r - g(u(t)) \rangle \\ &= -\|r - g(u(t))\|^2 = -2V(u(t)), \end{aligned} \quad (4)$$

where the function $\langle \cdot, \cdot \rangle$ stands for the standard inner product in $\mathbb{R}^m$. This establishes the fact that $V(u(t))$ is a Lyapunov function, and a convergence of the output to the (constant) reference has the form

$$\lim_{t \rightarrow \infty} y(t) = r. \quad (5)$$

¹Our code and video demonstrations can be found at: https://github.com/gtfactslab/MoralesCuadrado_TCST2025

Next, we relax the assumption that $r(\cdot)$ is a constant and assume instead that $r(\cdot)$ is a bounded, continuous, and piecewise-continuously differentiable function of time t with bounded derivative $\dot{r}(t)$ over $t \in [0, \infty)$. We revise the definitions in (2) and (3) by replacing r by $r(t)$, but the corresponding results in (4) and (5) are no longer guaranteed. However, the following weaker conclusion is in force:

$$\limsup_{t \rightarrow \infty} \|r(t) - y(t)\| \leq \limsup_{t \rightarrow \infty} \|\dot{r}(t)\|. \quad (6)$$

Observe that (5) is the special case of (6) where $r(\cdot) \equiv r$, a constant.

A tightening of the Right-Hand Side (RHS) in (6) can be obtained by scaling the RHS of the control equation (2) by a constant $\alpha > 1$.

With this scaling the controller has the following form,

$$\dot{u}(t) = \alpha \left(\frac{dg}{du}(u(t)) \right)^{-1} (r(t) - g(u(t))), \quad (7)$$

and consequently, the conclusion expressed by (6) is generalized to the inequality

$$\limsup_{t \rightarrow \infty} \|r(t) - y(t)\| \leq \alpha^{-1} \limsup_{t \rightarrow \infty} \|\dot{r}(t)\|. \quad (8)$$

It is noted in [10] that the constant α acts as a speedup factor, not a gain of the controller subsystem, since it is a multiplicative factor of $\dot{u}(t)$, not $u(t)$.

The final stage of the discussion in this subsection considers the most general case of the plant subsystem, modeled by a differential equation of the form

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0, \quad (9)$$

and an output function

$$y(t) = h(x(t)). \quad (10)$$

In the state equation (9), $x(t) \in \mathbb{R}^n$ is the state variable and $u(t) \in \mathbb{R}^m$ is the plant's control input. The function $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is assumed to have sufficient smoothness and boundedness properties for the existence of unique solutions of the differential equation (9) for all $t \in [0, \infty)$, for every $u(\cdot)$ and x_0 in respective sets of interest (see [10] for details). The system's output function $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is assumed to be continuous and piecewise-continuously differentiable with bounded first derivatives on compact sets in $\mathbb{R}^n$.

Regarding the effects of this dynamic formulation on the NR-based tracking controller, the main difference between the memoryless system (1) and the dynamic model (9) – (10) is that in the latter case $y(t)$ is not a function of $u(t)$, therefore it is impossible to use (7) to define the controller. However, for given $t \geq 0$ and $T > 0$, both $x(t+T)$ and $y(t+T)$ are functions of $x(t)$ and $\{u(\tau) : \tau \in [t, t+T]\}$. Therefore it is reasonable to define, at time t , a predictor of $y(t+T)$ with the objective of having it track $r(t+T)$.

Since the future inputs $\{u(\tau) : \tau \in (t, t+T]\}$ are unknown at time t , we approximate them by assuming that the control input is held constant over the prediction

horizon, i.e., a zero-order hold approximation is used. Under this assumption we consider a predicted output, denoted by $\tilde{y}(t+T)$, as a function of $x(t)$ and $u(t)$. That is, we have that

$$\tilde{y}(t+T) = g(x(t), u(t)) \quad (11)$$

for a continuous function $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m$, assumed to be continuously differentiable in (x, u) and whose partial Jacobian $\frac{\partial g}{\partial u}(x(t), u(t))$ is locally Lipschitz continuous in (x, u) . Note that for this Jacobian to be invertible and for tracking error to be computed, the reference, system inputs, and outputs must all be of the same dimension.

Given a fixed controller-speedup factor $\alpha \geq 1$ and prediction horizon $T > 0$, the NR-based tracking control technique is defined by the following extension of (7),

$$\dot{u}(t) = \alpha \left(\frac{\partial g}{\partial u}(x(t), u(t)) \right)^{-1} (r(t+T) - g(x(t), u(t))). \quad (12)$$

The objective of this controller is to enable $\tilde{y}(t+T)$ to track the future input $r(t+T)$ ². The effectiveness of the NR-based tracking technique is characterized by the following inequality,

$$\begin{aligned} \limsup_{t \rightarrow \infty} \|r(t+T) - \tilde{y}(t+T)\| \\ \leq \alpha^{-1} \limsup_{t \rightarrow \infty} \|\dot{r}(t)\|, \end{aligned} \quad (13)$$

where we note the role of α in scaling down the RHS of (13), which is independent of $T > 0$.

By combining (9) and (12), the closed-loop system can be viewed as a dynamical system with the compounded state variable $z := (x^\top, u^\top)^\top \in \mathbb{R}^n \times \mathbb{R}^m$ and input $r(\cdot)$. Unlike the situation with the memoryless plant system defined by (1), Bounded Input Bounded State (BIBS) stability cannot be taken for granted or proved from verifiable assumptions. Furthermore, [10] defines a uniform variant of BIBS stability, labeled α stability, and proves from it the following limit,

$$\lim_{\alpha \rightarrow \infty} \limsup_{t \rightarrow \infty} \|r(t) - \tilde{y}(t)\| = 0. \quad (14)$$

(14) provides a definition for uniform asymptotic tracking of the NR-based technique. The theoretical question of tracking now hinges on whether α stability can be proven from verifiable assumptions for a particular system or a class of systems. That has been achieved for linear, time-invariant systems in [10], but proofs for nonlinear systems remained elusive until recently, as discussed in the sequel.

B. Additional Results

While the theoretical question of α stability has been slow to yield provable results, investigations of the NR-based technique focused primarily on simulation studies

²The future reference $r(t+T)$ is defined as the tracking-target at time $t+T$, regardless of how it is computed. For instance, in a simple case $r(\cdot)$ may be an exogenous process that is explicitly known to the system at the initial time $t_0 := 0$. In a more involved scenario, $r(t+T)$ may be a predicted value of an endogenous process of the system.

of various systems. Of a particular interest are dynamic models of autonomous vehicles, arrayed in platoons and other formations, with distributed motion controls by the NR-based tracking technique.

Simulation results of experiments on platoons of the dynamic bicycle model [11] are contained in [12], [13]. [14] extended these ideas to heterogeneous multi-agent networks with graph-Laplacian-based motion control, while [15] studied a model-free setting in which agent dynamics are learned directly from data. Over the course of these simulation studies, various ideas have been proposed to enhance the core NR-based tracking method. These include the incorporation of differential flatness for some vehicles [16], the use of machine learning methods to improve model accuracy, and the development of a barrier function approach for dynamic control laws [17].

In these simulation experiments we make the following observations.

- 1) At a given $\alpha > 0$, systems became unstable when the prediction horizon T was taken to be too short.
- 2) At a given $T > 0$, increasing α serves to stabilize an otherwise unstable system.
- 3) Given $T > 0$ and $\alpha > 0$ such that the closed-loop system is stable, if a system's predicted output is far off the target trajectory at the initial time, namely $\|\tilde{y}(T) - r(T)\|$ is large, then the closed-loop system may exhibit substantial overshoots and oscillations in the input $u(t)$, state variable $x(t)$, and/or output $y(t)$.

The first and second observations led us to the following process for choosing T and α (see [10]):

- Choose a small $T > 0$ so as to achieve a small asymptotic tracking error if the closed-loop system were stable. If the closed-loop system is unstable, increase α to stabilize it.

This approach of choosing T and α has been used in most of the aforementioned simulation experiments and shown to be effective.

The third observation above led to the use of a particular version of Control Barrier Functions (CBF), suitable for dynamic controllers [17]. Labeled Integral CBF, or I-CBF, it extends the scope of CBF to include systems whose plant's dynamics are not control affine, as for the dynamic bicycle.³ Furthermore, its application to dynamic controllers often requires lighter computing loads than the standard CBF.

A notable challenge in tracking control arises in the context of nonholonomic systems, i.e., systems subject to velocity constraints that cannot be integrated into positional constraints. Such systems are common in robotics, for instance wheeled vehicles whose motion is restricted to their instantaneous heading direction. Additionally, for many of these systems the dimension of the input is strictly smaller than that of the output of interest, which violates

the equal-dimension requirement of the NR-based technique. A specific instance of this difficulty is addressed for the NR-based technique in [18], where Dubins-car pursuers in a pursuit–evasion game are controlled by collapsing the two-dimensional position output into a scalar distance metric, thereby resolving the dimensional mismatch in that particular setting. However, extending the NR-based technique to a broader class of nonholonomic systems remains an open direction for future research. Notably, recent developments have extended NMPC to hardware applications on nonholonomic systems [19].

An approach to the NR-based tracking controller by the concept and technique of differential flatness is also investigated in [16]. Under broad verifiable assumptions, a quasi NR-based controller, operating on the flat system, can perform the prediction, I-CBF, and tracking control with greatly reduced computational burden as compared to their implementations on the physical system. In contrast, [20] uses the differential flatness only for the prediction but not the I-CBF or NR-based controller. Thus, it preserves the NR-based technique and not approximations thereof, but it requires more computing efforts than the aforementioned quasi NR-based technique. While proofs of α -stability are established in [10] for linear systems, [20] provides a proof of α -stability for differentially flat systems for the case where the target trajectory $r(\cdot)$ is a constant $r \in \mathbb{R}^m$.

The earliest implementation of the NR-based tracking technique on quadrotors was reported in [21], where the quadrotors are the same as those considered in this paper. The experiments in [21] tested a simple output prediction method, comprised of a linear, time-invariant prediction model, and used it to test the effectiveness of the tracking technique. The resultant tracking was not as precise as in this paper (see Subsection II.D, below), which uses a more sophisticated prediction model, and it did not include tests on other aerial vehicles. Also, [21] does not perform any comparisons with powerful, established tracking control alternatives, which is the main objective of this paper.

C. Contributions of this paper

Earlier in this section we mentioned a number of system models which have been simulated for the purpose of testing the NR-based tracking technique. At this point, we believe that the main challenge in the investigation of the NR-based tracking technique is to conduct a comparative study on hardware platforms of its performance and that of well-known tracking controllers that are commonly used in research and applications. Specifically, we are interested in comparative assessments when target trajectories require large computing loads. To this end, we perform the comparisons against variants of feedback linearization-based control and nonlinear model predictive control. The resulting comparisons were conducted in laboratory settings and the results are presented in the sequel.

³A dynamic control directly computes $\dot{u}(t)$, not $u(t)$. This requires an integrator to generate the input to the plant, $u(t)$, hence the I-CBF label.

D. Implementation Details

We close this section with a statement about the predictor used in most of the works on the NR-based technique. Our previous work on quadrotors [21] focused on efficiency of computation to meet real-time requirements. To this end, it implemented the output prediction step of the control algorithm with a linearized closed-form solution to the system dynamics. By pre-computing the linearized system matrices at Euler angles of zero and assuming small deviations in applications, we saved time in the loop. This led to inaccuracies in output prediction and hence in tracking performance, as well as preventing us from considering more aggressive trajectories such as those containing yawing components.

In this work, we focus on achieving competitive tracking performance with powerful, established techniques on more aggressive and complex trajectories. Thus, we implement Runge-Kutta (RK4) and the forward Euler integration methods of the dynamics associated with the state equations of the blimp and quadrotor, respectively defined by (15) and (23), below. The challenge in implementing such precise prediction methods is in ensuring that a complete iteration of the control algorithm is computed with a minimum update rate of 100 Hz, even under the computational limitations of a Raspberry Pi.

To accomplish this, we employed two equally effective computational approaches using established software tools: (i) compilation of the integration and Jacobian computation routines into C via the Cython [22] toolchain and use as a Python shared object, and (ii) JIT compilation using the JAX [23] numerical computing library. In the Cython-based approach, Jacobians are computed by finite differences, whereas the JAX-based approach uses JIT-compiled forward-mode automatic differentiation. All experimental results showcased in this work use the JAX-based implementation for consistency.

This implementation allows for fast prediction with high accuracy, permitting us to compare results of the NR-based technique to those obtained from NMPC and feedback linearization. Importantly, it also enables us to perform rotations in yaw decoupled from the motion of the quadrotor. This is key for applications that require the quadrotor to face a particular direction while it follows a path.

In addition, to assess the sensitivity of the NR-based technique to the speedup factor α , we conduct a sweep over a wide range of values while holding all other parameters fixed. As shown in Fig. 3, increasing α beyond a moderate level yields only marginal improvements in tracking accuracy. Over a broad interval, the RMSE remains nearly constant, indicating that the controller performance is largely insensitive to the precise choice of α within this region. Outside the range of α depicted in the figure—both for sufficiently small and sufficiently large values—the closed-loop system becomes unstable. These tests were carried out in a numerical simulator of the quadrotor model utilized in this work, with standard ratios

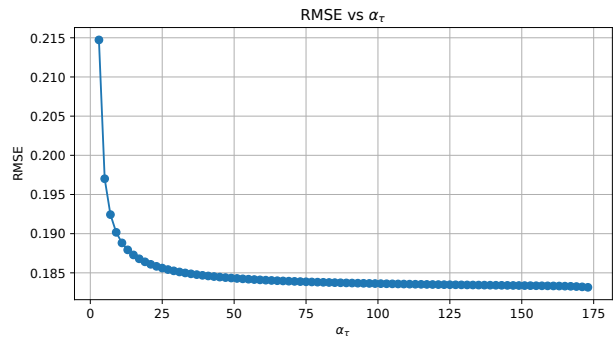


Fig. 3. Empirical sensitivity of tracking error to the controller-speedup factor α associated with the thrust input u_τ .

between the thrust speedup parameter and the remaining input parameters.

III. Miniature Blimp

Miniature blimps are an aerial platform that pose interesting challenges for control. Blimps are constructed with a wide variety of actuation capabilities and envelope shapes, leading to a wide array of dynamic models. Unlike quadrotors, blimps' neutral buoyancy allows them to stay aloft without any actuation, and thus they can be flown at a far cheaper energy cost for long-term missions compared to quadrotors. Moreover, the dynamics of a blimp and a quadrotor are fundamentally different. Unlike the quadrotor, the blimp model considered in this paper has lightly damped zero dynamics in the rolling and pitching motions when input-output feedback linearized with positional outputs.

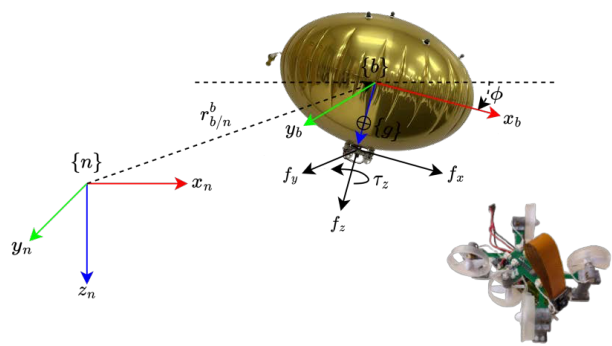


Fig. 4. The radially symmetric blimp with relevant coordinate frames. The gondola includes six motors—two for vertical actuation, and four for horizontal actuation and yawing.

Several tracking control strategies have been proposed for miniature blimps. The paper [6] models a blimp as

a simple chain of double integrators with time delay and tracks a desired yaw angle with model predictive control, but it is limited to only yaw tracking and ignores nonlinear dynamic effects. Sliding mode control has been demonstrated to achieve good tracking performance [24], but it requires the selection of the correct manifold onto which to project the dynamics, and it is computationally expensive.

In this work, we specifically consider a radially symmetric blimp with undermounted thrusters for horizontal holonomic control [25], for which several tracking controllers have been previously proposed. The paper [26] presents a controller that tracks a desired horizontal velocity by tracking a fixed roll or pitch angle first in a multi-loop control architecture; however, this method is specifically useful for velocity tracking. The paper [7] inverts the dynamics to derive a feedback linearization (FBL)-based tracking controller. By including high-order control barrier functions (CBFs) to limit the roll and pitch angle—the residual zero dynamics—large roll and pitch angles are empirically mitigated. The blimp and relevant coordinate frames are pictured in Figure 4.

A. Dynamic Model and Tracking Controllers for the Blimp

The radially symmetric blimp presented in [25] is modeled as a six-DOF underactuated system with holonomic horizontal control. We briefly summarize the model—a more complete representation is given in [7], [27]. Let $\nu_{b/n}^b = (v_{b/n}^b, \omega_{b/n}^b)$ represent translational and angular velocity in the body frame, and let $\eta_{b/n}^n = (p_{b/n}^n, \Theta_{b/n}^n)$ represent position in the world frame and orientation of the blimp (expressed in Euler angles) relative to the world frame, respectively. We then define the state vector for the blimp as $x = (\nu_{b/n}^b, \eta_{b/n}^n) = (v_{b/n}^b, \omega_{b/n}^b, p_{b/n}^n, \Theta_{b/n}^n) \in \mathbb{R}^{12}$. The inputs are forces along each axis in the body frame and torque about the body frame z -axis, $u = [f_x f_y f_z \tau_z]^\top$.

The dynamics about the center of gravity (CG) are originally derived from a submarine physics model [28] and are given here as

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -(M^{CB})^{-1}C^{CB}(x) - (M^{CB})^{-1}D^{CB} & 0 \\ \text{diag}(R(\Theta), T(\Theta)) & 0 \end{bmatrix} \begin{bmatrix} \nu \\ \eta \end{bmatrix} \\ &\quad - (M^{CB})^{-1}G^{CB}(x) + (M^{CB})^{-1}u \\ y &= Wx \end{aligned} \quad (15)$$

where M^{CB} is the mass/inertia matrix, $C^{CB}(x)$ is the Coriolis-centripetal matrix, D^{CB} is the diagonal aerodynamic damping matrix, $R(\Theta)$ is a rotation matrix, $T(\Theta)$ is a transformation matrix, W is a binary matrix to select the position variables p_x, p_y, p_z, ψ , and the gravity vector is

$$G^{CB}(x) = - \begin{bmatrix} 0_{3 \times 1} \\ r_{g/b}^b \times f_{g/b}^b \end{bmatrix}, \quad f_g^b = (R_b^n(x))^{-1} \begin{bmatrix} 0 \\ 0 \\ f_{z,g}^n \end{bmatrix}$$

where $f_{z,g}^n$ is the constant downward force of gravity and $r_{g/b}^b = [0 \ 0 \ r_{z,g/b}^b]^\top$ is the position vector of the CG in the

body frame, i.e. the vector from the center of buoyancy (CB) to the CG. Although translational acceleration due to gravity is zero due to the blimp's buoyancy, gravity induces non-zero restoring torque about the CB.

In the case of the miniature blimp, we compare our proposed NR-based tracking technique against the FBL-based and NMPC control strategies developed in [7]. To maintain consistency with the experimental setup in [7] and to enable a fair comparison with previously published results, we utilize the same hardware, motion-capture software, and implementation of these miniature blimp controllers. Our objective in this work is to evaluate the NR-based technique against established baseline controllers under their standard configurations as reported in the literature.

The FBL-based controller selects outputs $\sigma = [p_x p_y p_z \psi]^\top$ and inputs $u = [f_x f_y f_z \tau_z]^\top$. The blimp dynamics can be written in the control-affine form

$$\dot{\nu}_{b/n}^b = N(x)\nu_{b/n}^b + F(x) + Ku \quad (16)$$

with matrices $N(x)$, $F(x)$ and K given in [7, Equation 5–7].

Under appropriate geometric conditions, by [7, Theorem 1] the blimp is input-output feedback linearizable such that the second derivative of the outputs can be expressed in canonical form as

$$\ddot{\sigma} = a(x) + B(x)u := q \quad (17)$$

for state-dependent vector $a(x) \in \mathbb{R}^4$ and invertible matrix $B(x) \in \mathbb{R}^{4 \times 4}$.

This system is cast as a double integrator system with a virtual input q after performing a dynamic inversion of the form

$$u(x) = B^{-1}(x)(q - a(x)). \quad (18)$$

Now, given a reference trajectory $(r(t), \dot{r}(t), \ddot{r}(t))$ and corresponding error signal $e(t) := \sigma(t) - r(t)$, the feedback linearization controller tracks the reference via a proportional-derivative controller with feed-forward acceleration as the virtual input,

$$q = -k_1 e - k_2 \dot{e} + \ddot{r}. \quad (19)$$

Additionally, the feedback linearization controller is augmented with high-order CBFs that empirically mitigate the worst-case roll and pitch. For details on the implementation as a quadratic program, see [7].

The nonlinear model predictive controller that we compare to is also the baseline controller in [7]. Given the blimp dynamics in the form $\dot{x} = f(x, u)$, let $(r(t), \dot{r}(t))$ be the reference trajectory, and let $e(t) = [\sigma(t)^\top - r(t)^\top \quad \dot{\sigma}(t)^\top - \dot{r}(t)^\top]^\top$ be the tracking error signal and its derivative. After discretizing the dynamics with a zero-order hold, the optimization program

$$\begin{aligned} u_k^* &= \underset{u_k}{\operatorname{argmin}} \sum_{j=k}^N e_j^\top Q e_j + u_j^\top R u_j \\ \text{s.t. } x_{j+1} &= x_j + \Delta t f(x_j, u_j) \quad j = k, \dots, N-1 \\ u_j &\in \mathcal{U} \quad j = k, \dots, N-1 \end{aligned} \quad (20)$$

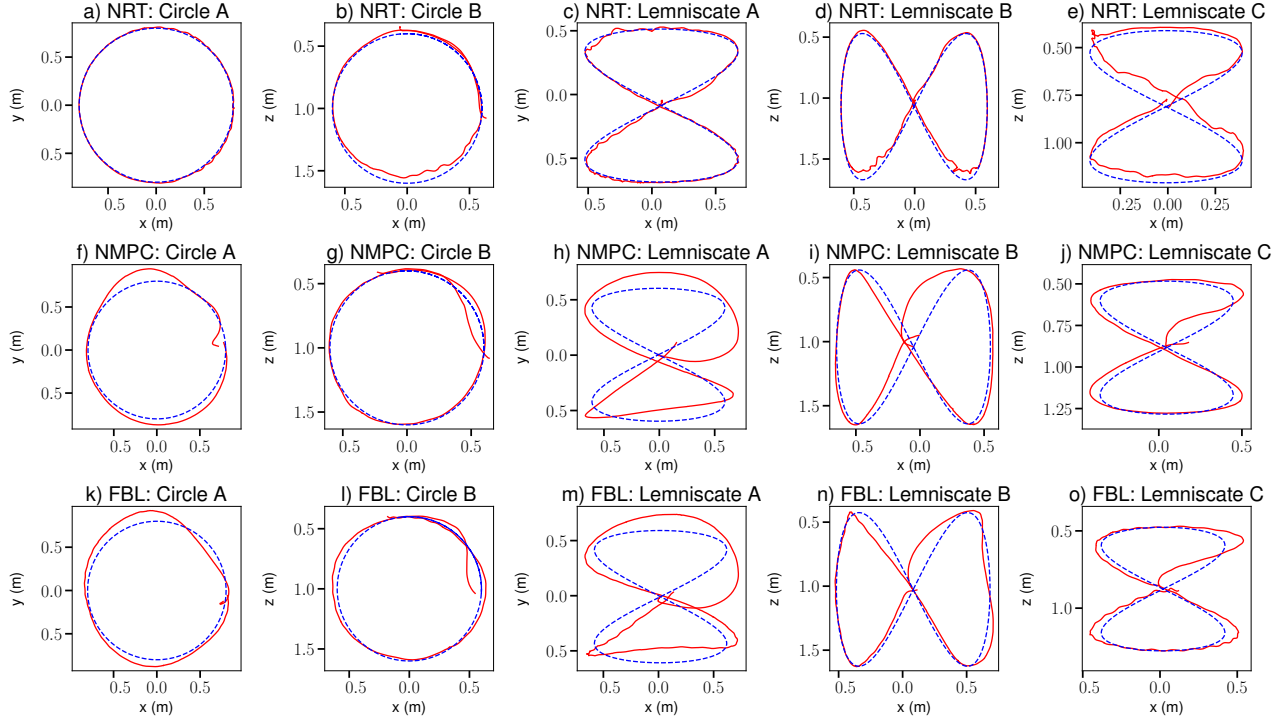


Fig. 5. Blimp: Comparison of five standard flight trajectories. Flight data in red, trajectory reference in blue. Rows from top to bottom are NR-based technique (labeled by the acronym NRT), NMPC, and FBL-based controller.

is repeatedly solved in CasADi [29], where Δt is the discretization step, $Q \in \mathbb{R}^{8 \times 8}$ and $R \in \mathbb{R}^{4 \times 4}$ are cost matrices, and $\mathcal{U}$ represents the actuation limits. For both the blimp and quadrotor, the cost matrices were identified through empirical tuning in which candidate matrices were tested and the final selection was made to minimize the tracking error.

B. Integral Control Barrier Functions for Smooth Actuator Saturation

Integral control barrier functions (I-CBFs) [17] are an extension of CBFs for dynamically defined control laws. I-CBFs can enforce forward invariance for both states and inputs by treating the input as an extended state variable. We utilize I-CBFs for smooth actuator saturation when utilizing the NR-based tracking technique. Denoting the right hand side of (12) by $\Psi(x(t), u(t), t)$, we modify the control law to

$$\dot{u}(t) = \Psi(x(t), u(t), t) + \eta(t), \quad (21)$$

where $\eta(t)$ is a minimal I-CBF intervention term that smoothly limits actuation. These developments yield a continuous-time Newton-Raphson-based dynamic control law with smooth input saturation to enhance flight safety. See [17], [30] for more information on I-CBFs and general CBFs, respectively.

Figure 6 displays the control inputs for the blimp in terms of forces and torque about the z-axis to track the horizontal circle trajectory. These are prescribed by the NR-based technique and kept within safe limits with I-CBFs, outlined by the red horizontal lines. This saturation

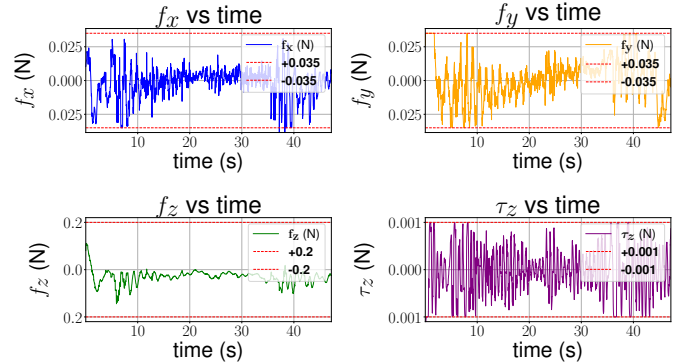


Fig. 6. Force and torque inputs from the NR-based technique smoothly saturated to within safe limits (red horizontal lines) via I-CBFs.

is smooth in the sense that it considers the speed at which the system approaches violations of the safety constraint and begins making adjustments before the safe limits are reached, unlike hard saturation methods that abruptly clip the control input once a bound is exceeded.

Although CBFs guarantee forward invariance of the safe set in theory, practical hardware implementations can introduce violations of this property. Contributing factors include model mismatch, finite-precision computation, sensor noise, and unmodeled disturbances. Most critically, standard CBF forward invariance proofs assume continuous-time implementation, whereas digital control imposes discretization and sampling effects that may cause temporary excursions outside the safe set. However, as

shown in Figure 6, the inputs are always immediately pushed back into the safe region by the I-CBFs when this occurs.

C. Blimp Experimental Setup and Results

The blimp gondola is comprised of six counter-rotating motors (two vertical and four horizontal), an inertial measurement unit (IMU), a motor board, and a Raspberry Pi Zero W. IMU data as well as motion-capture data from an OptiTrack system are streamed over a WiFi connection to a Dell Precision 5570 laptop with a 12th Gen Intel Core i7-12700H $\times$ 20 CPU. The laptop first fuses IMU and motion-capture data to estimate states, and utilizes this information for control synthesis, from which appropriate motor commands are calculated. These motor commands are sent to the motor board via a base station over a non-WiFi 2.4 GHz connection.

We compare the NR-based technique, NMPC, and FBL-based controllers on the miniature blimp. The tracking results are plotted in Figures 5, 7 with error and computation time given in Tables I, II, respectively. Across all tables, entries typeset in bold indicate the best-performing result in the corresponding performance category. For brevity, the NR-based technique is denoted by the acronym ‘NRT’ in all figures and tables. Computation time is always measured by the real-world time taken for the function call for each controller to return its value at every iteration, measured with the time Python module. In all tables, bold values indicate the best performance for each trajectory.

We selected five standard trajectories for the miniature blimp. These are Horizontal Circle (Circle A), Vertical Circle (Circle B), Horizontal Lemniscate (Lemniscate A), Vertical Short Lemniscate (Lemniscate B), Vertical Tall Lemniscate (Lemniscate C). We additionally selected three aggressive trajectories for the blimp: Regular Helix (Helix A), Yawing Helix (Helix B), and Yawing Horizontal Circle (Circle C). For brevity, we sometimes shorten trajectory names or use the abbreviated names in parentheses (e.g., Horizontal Circle as Circle A).

TABLE I: Blimp - NR-based technique vs. NMPC vs. FBL RMSE

Trajectory	NRT [m]	NMPC [m]	FBL [m]
Circle A	0.07866	0.14075	0.19332
Circle B	0.05137	0.10991	0.14255
Lemniscate A	0.10441	0.18517	0.22955
Lemniscate B	0.05389	0.10291	0.14565
Lemniscate C	0.05594	0.11793	0.13073
Helix A	0.07187	0.07732	0.11291
Helix B	0.08767	0.21534	0.22001
Circle C	0.10124	0.40327	0.38360

D. Blimp Analysis and Discussion

The NR-based tracking technique is an order of magnitude faster than the FBL-based controller, which is itself about half-an order of magnitude faster than the

TABLE II: Blimp - Computation Time Comparison

Trajectory	NRT [ms]	NMPC [ms]	FBL [ms]
Circle A	0.84 $\pm$ 0.046	58.67 $\pm$ 15.67	10.42 $\pm$ 9.69
Circle B	0.82 $\pm$ 0.031	51.47 $\pm$ 12.23	10.86 $\pm$ 10.43
Lemniscate A	0.82 $\pm$ 0.019	52.67 $\pm$ 12.49	10.51 $\pm$ 10.66
Lemniscate B	0.87 $\pm$ 0.23	54.18 $\pm$ 6.84	10.84 $\pm$ 10.12
Lemniscate C	0.86 $\pm$ 0.018	54.84 $\pm$ 11.93	10.72 $\pm$ 10.42
Helix A	0.94 $\pm$ 0.041	53.15 $\pm$ 6.78	9.89 $\pm$ 9.49
Helix B	0.95 $\pm$ 0.022	60.76 $\pm$ 12.78	10.27 $\pm$ 10.05
Circle C	0.96 $\pm$ 0.034	73.49 $\pm$ 20.92	10.18 $\pm$ 9.88

NMPC controller, see Table II. We note that the FBL-based controller calls on CBFs to mitigate oscillations in its internal dynamics, which factor into its computation time.

Given a prescribed control update rate of 40 Hz, the NR-based and FBL-based controllers consistently meet the corresponding 25 ms deadline necessary for low-level control of our particular blimp platform. However, due to computational constraints, the NMPC controller does not meet its deadline and consequently tracking performance is degraded. Note that when performing a simulation with real-time computational requirements removed and a 3 s lookahead, the NMPC controller achieves the lowest tracking error; however, this may not be realizable within realistic computational constraints.

We highlight an additional advantage of the NR-based technique. For both the FBL-based controller as well as NMPC to achieve competitive tracking performance, we found it was necessary to enhance them with reference trajectory derivative information. On the other hand, the NR-based technique was only given the desired trajectory with no derivative information. Nevertheless, the RMSE for the NR-based technique was between one-half and one-quarter of that of the other controllers, as seen in Table I.

IV. Quadrotor

Quadrotor research is led by work on small, agile indoor racing drones with significant onboard computing capabilities. In particular, quadrotors small enough to be classified as Micro-unmanned Aerial Vehicles (MAVs) of a size between 0.1 m to 0.5 m in diagonal length and 0.1 kg to 0.5 kg of mass [31], [32] are commonly used.

While smaller quadrotors are desirable for upkeep and for multi-agent research in small flight spaces, the primary reason for their expansive use lies in the physics of quadrotor flight. A quadrotor’s translational motion is controlled by tilting its body, meaning rotational dynamics directly influence translational agility. Consequently, because angular acceleration is inversely proportional to the size of the quadrotor [32], MAVs feature prominently in present-day research.

However, while less maneuverable, medium-sized and larger quadrotors have the inertia necessary to withstand unfavorable real-world weather conditions and can carry larger payloads [32], [33]. This makes research exploiting their strengths and addressing their limitations both useful for real-world deployment and less commonly addressed in

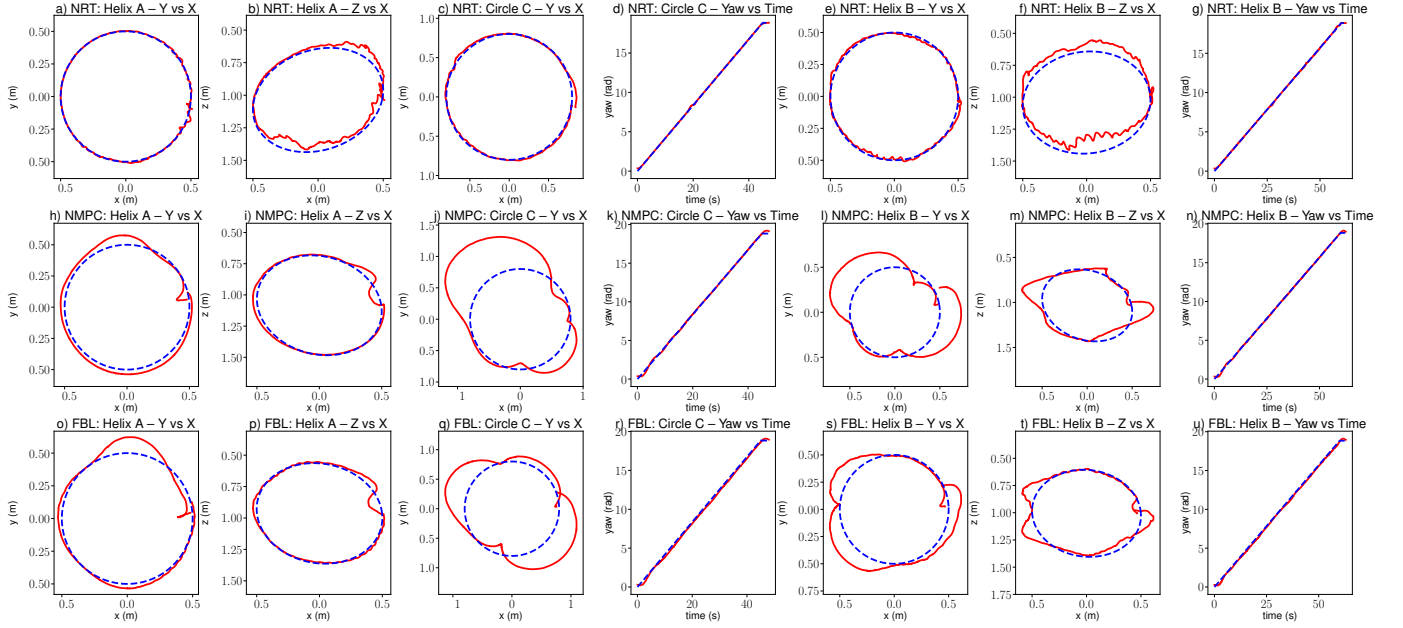


Fig. 7. Blimp: Comparison of three aggressive trajectories. Flight data in red, trajectory reference in blue. Rows from top to bottom are from the NR-based technique, NMPC, and FBL-based controller.

work that focuses on small racing drones. In contrast to the dominant focus on MAV agility, our work investigates the complementary challenges posed by larger platforms. We begin with the challenges of computational constraints and limited vehicle agility, and seek to maximize trajectory speed and tracking accuracy despite these constraints. The quadrotor used in our flight experiments is depicted in Figure 8.



Fig. 8. The Holybro X500 V2 quadrotor used for hardware experiments. It is fitted with tracking markers, a radio receiver, and an on-board Raspberry Pi with ROS2 for control computations. The quadrotor weighs 2.1 kg, measures 0.6 m diagonally, and 0.25 m vertically.

A. Dynamic Model and Tracking Controllers for the Quadrotor

We model our quadrotor as a 9-state, 4-input system. We take the system state as

$$x = [p_x \ p_y \ p_z \ V_x \ V_y \ V_z \ \phi \ \theta \ \psi]^T \in \mathbb{R}^9 \quad (22)$$

where (p_x, p_y, p_z) is the system's 3D position in a fixed north, east, down (NED) world frame, (V_x, V_y, V_z) is the system's velocity within the frame, and (ϕ, θ, ψ) are Euler

angles describing the orientation of the system within the same coordinate framework.

The system input is $u = [u_\tau, u_p, u_q, u_r]^T \in \mathbb{R}^4$, where u_τ is the net thrust through the quadrotor's body-frame z -axis and $\omega^b = (u_p, u_q, u_r)$ are the angular body rates of the quadrotor. The nonlinear dynamics are

$$\begin{aligned} (\dot{p}_x, \dot{p}_y, \dot{p}_z) &= (V_x, V_y, V_z), \\ \dot{V}_x &= -\frac{u_\tau}{m} (\sin \phi \sin \psi + \cos \phi \cos \psi \sin \theta), \\ \dot{V}_y &= -\frac{u_\tau}{m} (\cos \phi \sin \psi \sin \theta - \cos \psi \sin \phi), \\ \dot{V}_z &= g - \frac{u_\tau}{m} (\cos \phi \cos \theta), \\ (\dot{\phi}, \dot{\theta}, \dot{\psi}) &= T \omega^b, \end{aligned} \quad (23)$$

where T is the matrix for angular velocity transformations [34], namely

$$T = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix}.$$

For the quadrotor, we compare the NR-based technique against a compiled NMPC algorithm built with the *acados* [35] fast embedded optimal control toolbox. Across all methods for both the quadrotor and the blimp, we employ conservative tuning approaches that favor out-of-the-box configurations and minimal solver-level customization. For example, although more specialized adjustments to NMPC—such as modified termination criteria, customized tolerances, or tailored warm-start schemes—could reduce computation times for certain trajectories, such enhancements typically require substantial modification and may obscure the source of the performance gains. In contrast, the NR-based technique exhibits predictable behavior under minimal tuning. For these reasons, we evaluate

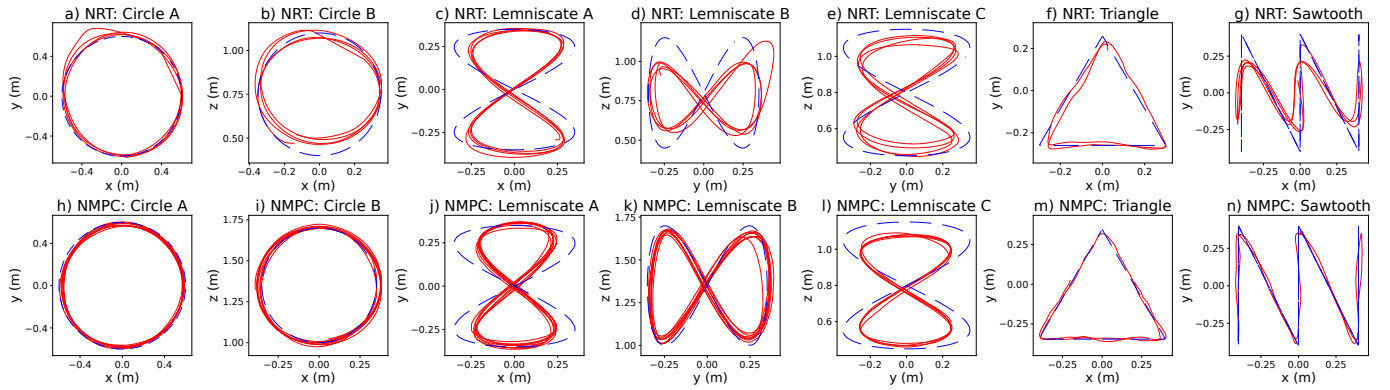


Fig. 9. Quadrotor: Flight data in red, trajectory reference in blue. Rows from top to bottom are the NR-based technique and NMPC.

all methods under well-tuned yet conservative numerical configurations.

In particular, the NMPC scheme deployed for the quadrotor is configured using a Real-Time Iteration (RTI) approach [36], which performs a single SQP linearization and QP solve at each sampling instant, rather than iterating to convergence. While this reduces the computational burden, the average solve time still exceeds one sampling period on the onboard computer (see Table IV). This is a well-known concern in modern NMPC implementations. To handle this, we follow a computational delay compensation strategy [37] to ensure consistent command timing despite variable solver execution times. In addition, implementing the tracking error within the CasADi quadrotor model, rather than within the acados NMPC formulation, was found to improve yaw tracking performance. Collectively, these implementation choices were necessary to achieve stable tracking across all trajectories, avoiding undesirable behaviors such as shaking and jitter.

B. Quadrotor Experimental Setup and Results

Leveraging the cascaded control architecture of the PX4 flight stack [38], we send rate control commands to the innermost loop of the stack. This control structure makes the system highly sensitive to small input variations, requiring continuous micro-adjustments. Consequently, tracking controllers deployed on such a system must publish commands at rates of at least 100 Hz to ensure flight stability.

In our hardware experiments, we use a Holybro X500 quadrotor equipped with a Pixhawk 6X flight controller as the flight management unit. Onboard control computations are performed using a Raspberry Pi 4 Model B transmitting control commands to the flight controller via a serial connection. Communication between the Raspberry Pi and the flight controller is facilitated through the ROS 2 API available within the PX4 flight stack. The μ XRCE-DDS middleware serves as a bridge between PX4’s native uORB topics and ROS 2 topics. For motion capture, we employ the OptiTrack system and use a

relay to convert OptiTrack messages into visual odometry messages. These are subsequently fused with the flight controller’s onboard sensor data via the flight stack’s extended Kalman filter.

Despite the significant differences between the blimp and quadrotor platforms, the NR-based tracking technique exhibits a modular structure that supports cross-platform deployment. Porting the controller requires (i) substituting the appropriate vehicle dynamics within the internal prediction model, and (ii) updating the input evolution law $u_{\text{next}} = u + \dot{u} \Delta t$ to match the platform’s control-rate and actuation conventions. Additionally, incorporating platform-specific safe control limits into the I-CBF constraints ensures compatibility with each vehicle’s actuator capabilities and desired flight envelope. The remaining effort consists of tuning the speedup factor α and lookahead horizon T . While these steps are not trivial, the controller’s structure makes the adaptation process systematic and relatively straightforward.

We again utilize the five standard trajectories from the Blimp experiments: Horizontal Circle (Circle A), Vertical Circle (Circle B), Horizontal Lemniscate (Lemniscate A), Vertical Short Lemniscate (Lemniscate B), and Vertical Tall Lemniscate (Lemniscate C). We also include the aggressive trajectories: Regular Helix (Helix A), Yawing Helix (Helix B), and Yawing Horizontal Circle (Circle C). Due to the higher maneuverability of quadrotors compared to blimps, we implement two additional aggressive trajectories: Sawtooth and Triangle. Figure 9 plots tracking results and provides a visual for the trajectories which can be plotted in two dimensions.⁴

Comparative results of the five standard trajectories and the five aggressive trajectories are summarized in Tables III and IV, providing quantitative differences between the NR-based technique and the NMPC approach.

C. Quadrotor Analysis and Discussion

We note that the flight trajectories are designed such that the quadrotor hovers at the origin for five seconds

⁴As for the case of the blimp, we use the acronym “NRT” for “NR Technique” in Figure 8 and Tables III and IV.

TABLE III: Quadrotor - NR-based technique vs. NMPC, RMSE Comparison (Clipped vs. Non-Clipped Transients)

Trajectory	Clipped Transients		With Transients	
	NRT [m]	NMPC [m]	NRT [m]	NMPC [m]
Circle A	0.10695	0.03932	0.12266	0.10653
Circle B	0.11887	0.03246	0.12276	0.15162
Lemniscate A	0.12328	0.09738	0.12249	0.11311
Lemniscate B	0.15915	0.04683	0.15203	0.10479
Lemniscate C	0.14554	0.09938	0.14532	0.17329
Helix A	0.14879	0.13327	0.14867	0.16865
Helix B	0.19540	0.14977	0.18851	0.18044
Circle C	0.17398	0.14782	0.17073	0.17715
Sawtooth	0.04014	0.02331	0.04505	0.06032
Triangle	0.05979	0.01803	0.08136	0.08387

TABLE IV: Quadrotor - Average Computation Time Comparison for the NR-based technique and NMPC

Trajectory	NRT [ms]	NMPC [ms]
Circle A	2.381 $\pm$ 0.206	13.036 $\pm$ 0.173
Circle B	2.426 $\pm$ 0.201	13.352 $\pm$ 0.139
Lemniscate A	2.438 $\pm$ 0.239	13.088 $\pm$ 0.092
Lemniscate B	2.486 $\pm$ 0.200	13.288 $\pm$ 0.246
Lemniscate C	2.397 $\pm$ 0.200	9.216 $\pm$ 5.646
Helix A	2.480 $\pm$ 0.222	13.307 $\pm$ 0.196
Helix B	2.471 $\pm$ 0.271	13.294 $\pm$ 0.110
Circle C	2.454 $\pm$ 0.255	13.705 $\pm$ 0.222
Sawtooth	2.507 $\pm$ 0.254	13.329 $\pm$ 0.140
Triangle	2.417 $\pm$ 0.196	13.182 $\pm$ 0.291

before initiating each maneuver, and returns to the origin to hover again at the end. These transitions introduce discontinuities in the reference, resulting in a transient portion of flight. To account for this, we report performance both with and without the transient segment, using data from the best runs of each method on each trajectory to enable fair comparisons that reflect the most favorable performance of each approach.

If we clip these transient portions of flight to consider strictly the trajectory tracking abilities of the NMPC and NR-based techniques, as seen in the “Clipped Transients” column of Table III, NMPC outperforms the NR-based technique in terms of RMSE for all successful flights. We note again that without computational delay compensation strategies, NMPC was susceptible to shaking behavior and crashed while attempting Helix B and Circle C. Further, as noted previously, the tracking performance of both NMPC and the NR-based technique can be degraded when the initial position and initial target are far apart. When the transient flight portions are considered (“With Transients” column in Table III), the NR-based technique is capable of outperforming the NMPC technique on some trajectories.

Next, Table IV shows that the computation times for NMPC are generally between 5.32 and 5.59 times larger than those of the NR-based technique. Moreover, while the computation times of the NR-based technique are largely independent of the trajectory being tracked, the computation time for NMPC with full SQP solves exhibited significant variability. For the SQP-RTI configuration reported in Table IV, this variability is less pronounced, with the exception of Lemniscate C.

TABLE V: Average CPU Energy Expenditure Comparison per Method on Blimp and Quadrotor

Method	Avg. CPU Energy Expenditure [μ J]
Blimp NRT	1.25 $\times 10^4 \pm 4.29 \times 10^3$
Blimp MPC	$2.04 \times 10^5 \pm 3.80 \times 10^4$
Blimp FBL	$3.06 \times 10^4 \pm 2.22 \times 10^4$
Quad NRT	1.73 $\times 10^4 \pm 7.78 \times 10^3$
Quad MPC	$6.15 \times 10^4 \pm 2.41 \times 10^4$

Noting that Helix B and Circle C differ from Helix A and Circle A by including yaw variations, it was found that for full SQP solves, this seems to increase the required computing times for NMPC beyond the 100Hz rate constraint. Consequently, each attempt to track the yawing trajectories led to crashes. Even on the standard trajectories with SQP-RTI, NMPC computation on average takes more than the 10ms allotted for each iteration. The decoupled compute-and-publish architecture described above circumvents this issue by ensuring that command publication remains consistently timed at 100Hz regardless of solver runtime. This allows NMPC to perform all trajectories, often achieving lower RMSE than the NR-based controller. Lastly, average energy expenditures across all tracking controllers for both quadrotor and blimp are shown in Table V. The NR-based technique has the lowest CPU energy expenditure across all controllers used for both the blimp and the quadrotor.

V. Conclusions

This work compares the proposed Newton-Raphson-based tracking control technique against well-established alternatives from the literature on two aerial hardware platforms under realistic deployment and computational constraints. The comparison is based on three primary metrics: tracking accuracy, computation time, and CPU energy expenditure. The proposed NR-based controller outperforms the FBL-based control method from the blimp literature across all three metrics. Compared to the more computationally demanding NMPC controller, the NR-based technique demonstrates shorter computing times and less CPU energy usage while maintaining competitive tracking accuracy across most trajectories, and in some cases, achieving greater accuracy.

When given ample time, computational power, and agile hardware like micro-UAVs, NMPC achieves exceptional performance for aggressive trajectory tracking, as is well documented in the literature. However, its suitability may be limited in deployment scenarios where computational constraints prevent meeting real-time requirements for low-level control. As demonstrated in this study, the NR-based technique provides a lightweight, adaptable, and theoretically grounded alternative, making it a viable choice for resource-constrained applications across a range of systems.

Future work on the NR-based tracking technique may explore enhancements to the prediction mechanism, including the use of adaptive or learning-based predictors, as well as uncertainty-aware prediction models. In addition,

the speedup parameter α has thus far been restricted to a tuned constant vector in all existing implementations of the NR-based technique. Future research could investigate adaptive schemes for α , as well as generalized formulations in which α is promoted to a full matrix of speedup parameters, thereby incorporating cross-terms and enabling an examination of their impact on tracking performance and stability.

References

- [1] M. Bangura and R. Mahony, “Real-time model predictive control for quadrotors,” IFAC Proceedings Volumes, vol. 47, no. 3, pp. 11773–11780, 2014. 19th IFAC World Congress.
- [2] M. Kamel, K. Alexis, M. Achtelek, and R. Siegwart, “Fast nonlinear model predictive control for multicopter attitude tracking on $so(3)$,” in 2015 IEEE Conference on Control Applications (CCA), pp. 1160–1166, 2015.
- [3] F. Nan, S. Sun, P. Foehn, and D. Scaramuzza, “Nonlinear mpc for quadrotor fault-tolerant control,” IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 5047–5054, 2022.
- [4] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, “A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight,” IEEE Transactions on Robotics, vol. 38, no. 6, pp. 3357–3373, 2022.
- [5] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, “Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors,” IEEE Robotics and Automation Letters, vol. 7, p. 690–697, Apr. 2022.
- [6] H. Fukushima, K. Kon, Y. Hada, F. Matsuno, K. Kawabata, and H. Asama, “State-predictive control of an autonomous blimp in the presence of time delay and disturbance,” in 2007 IEEE International Conference on Control Applications, pp. 188–193, IEEE, 2007.
- [7] M. Kasmalkar, L. Baird, and S. Coogan, “Feedback linearization of an underactuated miniature blimp with zero dynamics mitigation using high order control barrier functions,” IEEE Control Systems Letters, 2024.
- [8] Y. T. Liu, E. Price, M. J. Black, and A. Ahmad, “Deep residual reinforcement learning based autonomous blimp control,” in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 12566–12573, IEEE, 2022.
- [9] Y. Wardi, C. Seatzu, M. Egerstedt, and I. Buckley, “Performance regulation and tracking via lookahead simulation: Preliminary results and validation,” in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 6462–6468, 2017.
- [10] Y. Wardi, C. Seatzu, J. Cortés, M. Egerstedt, S. Shivam, and I. Buckley, “Tracking control by the newton–raphson method with output prediction and controller speedup,” International Journal of Robust and Nonlinear Control, vol. 34, no. 1, pp. 397–422, 2024.
- [11] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” IEEE Transactions on Control Systems Technology, vol. 15, no. 3, pp. 566–580, 2007.
- [12] S. Shivam, I. Buckley, Y. Wardi, C. Seatzu, and M. Egerstedt, “Tracking control by the newton–raphson flow: Applications to autonomous vehicles,” in 2019 European Control Conference (ECC), pp. 1562–1567, 2019.
- [13] S. Shivam, Y. Wardi, M. Egerstedt, A. Kanellopoulos, and K. G. Vamvoudakis, “Intersection-traffic control of autonomous vehicles using newton–raphson flows and barrier functions,” IFAC-PapersOnLine, vol. 53, no. 2, pp. 15733–15738, 2020.
- [14] K. Niu, Y. Wardi, C. T. Abdallah, and M. Hayajneh, “Consensus controller for heterogeneous multi-agent systems using output prediction,” IEEE Control Systems Letters, vol. 7, pp. 673–678, 2023.
- [15] K. Niu, Y. Wardi, C. T. Abdallah, and M. Hayajneh, “A model-free tracking controller based on the newton–raphson method and feedforward neural networks,” in 2022 American Control Conference (ACC), pp. 3254–3259, 2022.
- [16] G. Notomista and Y. Wardi, “A safe and computationally efficient tracking control algorithm for autonomous vehicles,” in 2024 American Control Conference (ACC), pp. 3734–3739, 2024.
- [17] A. D. Ames, G. Notomista, Y. Wardi, and M. Egerstedt, “Integral control barrier functions for dynamically defined control laws,” IEEE Control Systems Letters, vol. 5, no. 3, pp. 887–892, 2021.
- [18] S. Shivam, A. Kanellopoulos, K. G. Vamvoudakis, and Y. Wardi, “A predictive deep learning approach to output regulation: The case of collaborative pursuit evasion,” in 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 853–859, 2019.
- [19] M. Rosenfelder, H. Ebel, J. Krauspenhaar, and P. Eberhard, “Model predictive control of non-holonomic systems: Beyond differential-drive vehicles,” Automatica, vol. 152, p. 110972, June 2023.
- [20] K. Niu, Y. Wardi, and C. T. Abdallah, “Stability analysis of the newton–raphson controller for a class of differentially flat systems,” in Proc. IEEE Conference on Decision and Control (CDC), 2025.
- [21] E. Morales-Cuadrado, C. Llanes, Y. Wardi, and S. Coogan, “Newton–raphson flow for aggressive quadrotor tracking control,” in 2024 American Control Conference (ACC), pp. 3879–3884, 2024.
- [22] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, “Cython: The best of both worlds,” Computing in Science & Engineering, vol. 13, no. 2, pp. 31–39, 2011.
- [23] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs, 2018.
- [24] M. Wasim, A. Ali, M. A. Choudhry, I. Shaikh, and F. Saleem, “Robust design of sliding mode control for airship trajectory tracking with uncertainty and disturbance estimation,” Journal of Systems Engineering and Electronics, vol. 35, no. 1, pp. 242–258, 2024.
- [25] Q. Tao, M. Hou, and F. Zhang, “Modeling and identification of coupled translational and rotational motion of underactuated indoor miniature autonomous blimps,” in 16th International Conference on Control, Automation, Robotics and Vision, pp. 339–344, IEEE, 2020.
- [26] Q. Tao, J. Tan, J. Cha, Y. Yuan, and F. Zhang, “Modeling and control of swing oscillation of underactuated indoor miniature autonomous blimps,” Unmanned Systems, vol. 9, no. 01, pp. 73–86, 2021.
- [27] Q. Tao, J. Wang, Z. Xu, T. X. Lin, Y. Yuan, and F. Zhang, “Swing-reducing flight control system for an underactuated indoor miniature autonomous blimp,” IEEE/ASME Transactions on Mechatronics, vol. 26, no. 4, pp. 1895–1904, 2021.
- [28] T. I. Fossen, Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons, 2011.
- [29] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” Mathematical Programming Computation, vol. 11, 07 2018.
- [30] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in 2019 18th European Control Conference (ECC), pp. 3420–3431, 2019.
- [31] X. Liu, S. W. Chen, G. V. Nardari, C. Qu, F. Cladera, C. J. Taylor, and V. Kumar, “Challenges and opportunities for autonomous micro-uavs in precision agriculture,” IEEE Micro, vol. 42, no. 1, pp. 61–68, 2022.
- [32] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, “Towards a swarm of agile micro quadrotors,” Autonomous Robots, vol. 35, no. 4, pp. 287–300, 2013.
- [33] C. Powers, D. Mellinger, and V. Kumar, Quadrotor Kinematics and Dynamics, pp. 307–328. Dordrecht: Springer Netherlands, 2015.
- [34] F. Sabatino, “Quadrotor control: modeling, nonlinear control design, and simulation,” Master’s thesis, KTH, Automatic Control, 2015.
- [35] R. Verschuuren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, acados: a modular open-source framework for fast embedded optimal control, 2020.

- [36] M. Diehl, H. G. Bock, and J. P. Schlöder, “A real-time iteration scheme for nonlinear optimization in optimal feedback control,” *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.
- [37] R. Findeisen, *Nonlinear Model Predictive Control : A Sampled Data Feedback Perspective*. Phd thesis, University of Stuttgart, 2005.
- [38] PX4, *Controller Diagrams*. Dronecode Project, 2023. Online; accessed 2023-09-21].