Materials Research Proceedings 69 (2026) 806-811 https://doi.org/10.21741/9781644904251-142

Learning-Based Quadrotor Tracking Control Using Recursive Gaussian Processes

Alessandro LUPATINI[1,a*], Giovanni GOZZINI[1,b], Alessandro NAZZARI[1,c], Roberto RUBINACCI[1,d] and Marco LOVERA[1,e]

[1]Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano, Milano, 20156 ITA

[a]alessandro.lupatini, [b]giovanni.gozzini, [c]alessandro.nazzari, [d]roberto.rubinacci, [e]marco.lovera}@polimi.it

Keywords: Recursive Gaussian Process, Feedback Linearization, Reference Tracking, UAV

Abstract. High-precision UAV tracking is hindered by hard-to-model aerodynamic effects. We propose a *learning-based controller* for a simplified quadrotor that uses Feedback Linearization (FL), to cancel modeled dynamics, and a sparse Bayesian inference method called *Recursive Gaussian Process* (RGP), to learn and compensate residual nonlinearities in real time. Simulations and experiments conducted on a real 1D quadrotor platform demonstrate reduced tracking error and practical feasibility for *real-world implementation*.

Introduction

High-precision reference tracking for Unmanned Aerial Vehicles (UAVs) is one of the main challenges in the aerospace control community, primarily due to aerodynamics effects. These effects are difficult to model accurately and, therefore, are often neglected. As a result, model-based control methods frequently fail to achieve the desired performance. With the growing popularity of machine learning techniques, a promising research direction is the integration of model-based control with data-driven approaches.).

In this context, a popular model-based controller is *Feedback Linearization*, which employs the system model to cancel its nonlinear dynamics and render the system into a linear one. Past works used FL tuned with reinforcement learning [1] or combined with neural networks [2].

However, these approaches lack rigorous stability analysis. In contrast, Gaussian Processes (GPs) offer probabilistic guarantees and have therefore attracted significant attention within the control community. In fact, Greeff and Schoellig [4] demonstrated that using GPs online alongside FL substantially improves tracking performance and enhances robustness. Yet the approach was validated only in simulation and the computational constraints of onboard GP inference were not addressed.

In this paper, we propose a similar approach to control a 1-D quadrotor system. Specifically, we model the residual of the FL using a *sparse GP*, which significantly reduces the computational load of inference. We validate the approach on a real platform.

Baseline controller

Model assumptions. We consider a *SISO control-affine* system with state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}$:

$$\dot{x} = f(x) + g(x)u \tag{1}$$

Materials Research Proceedings 69 (2026) 806-811 https://doi.org/10.21741/9781644904251-142

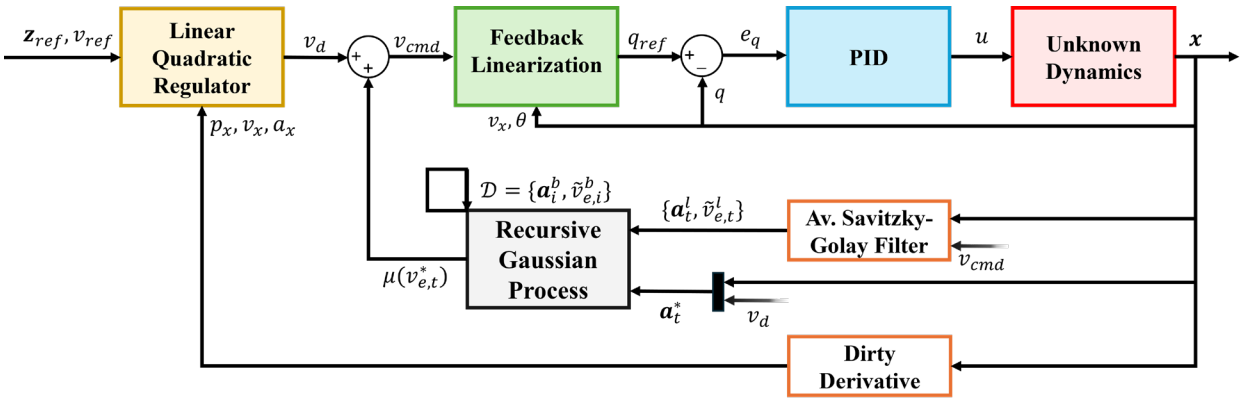

Figure 1: Controller scheme.

where $f(x)$ and $g(x)$ are unknown functions. *Assumption 1*: The system in 1 is differentially flat with respect to the known output $y = h(x) \in \mathbb{R}$. *Assumption 2*: A SISO nominal model of the system, also differentially flat in the output $y = h(x)$, is known:

$$\dot{x} = \hat{f}(x) + \hat{g}(x)u \tag{2}$$

Model specifications. The proposed controller (shown in Fig. 1) is formulated for a *2-DOF quadrotor system*, which dynamics is

$$\begin{cases} \dot{p}_x(t) = v_x(t) \\ \dot{v}_x(t) = -c v_x(t) - \bar{k} \sin(\theta(t)) \\ \dot{\theta}(t) = q(t) \\ \dot{q}(t) = -2\,\xi\,\omega_n q(t) - \omega_n^2 \theta(t) + \mu m_{c_x}(t) \end{cases} \tag{3}$$

where $c, \bar{k}, \xi, \omega_n$ and μ are obtained from grey-box identification. It is easy to show that the system is differentially flat. The angular velocity is controlled using a PID, which is a standard practice in quadrotor control. The resulting closed-loop is assumed fast enough to neglect it in the proposed control design. This allows us to consider the system a third-order one by approximating the pitch dynamics as

$$\dot{\theta} \simeq q_{ref} \tag{4}$$

where q_{ref} is the PID imposed dynamics.

Feedback Linearization and LQR. Using the above model, FL takes the form

$$\begin{aligned} q_{ref} &= \hat{\alpha}(\boldsymbol{z}) + \hat{\beta}(\boldsymbol{z}) v_{cmd} \\ &= \frac{c^2 v_x + c\,\bar{k} \sin\theta - v_{cmd}}{\bar{k} \cos\theta} \end{aligned} \tag{5}$$

where $v_{cmd} = \frac{u - \hat{\alpha}(\boldsymbol{z})}{\hat{\beta}(\boldsymbol{z})}$ is the imposed dynamics, *i.e.*, the third derivative of the output, and $\hat{\alpha}$ and $\hat{\beta}$ indicate approximations, obtained from the identified model, of the unknown true functions α and β. $\boldsymbol{z}$ is the states vector that contains the horizontal velocity v_x and acceleration a_x. $u = m_{c_x}$ is the system input, *i.e.*, the normalized pitch torque. The imposed dynamics v_{cmd}, *i.e.*, the desired third derivative, is calculated via the LQR from the setpoints and measurements.

Materials Research Proceedings 69 (2026) 806-811 https://doi.org/10.21741/9781644904251-142

Learning-based controller

We define the *nonlinear mismatch* v_e as the residual nonlinear dynamics that remain after the application of the FL, namely:

$$v_e(z,v) = v_{cmd} - v = \frac{\alpha(z) - \hat{\alpha}(z)}{\hat{\beta}(z)} + \frac{\beta(z) - \hat{\beta}(z)}{\hat{\beta}(z)} \tag{6}$$

Gaussian Process. The Gaussian Process is a Bayesian machine learning technique that approximates a function $f(x)$ at a new input x^*, using a dataset of stored observations $D = \{(x_i, \hat{y}_i) | i = 1, \dots, n\}$ [3]; this process is called inference and is performed under the assumption that all observations follow a multivariate Gaussian distribution. A GP is fully characterized by its prior distribution, which depends on the input values. This prior is specified by a mean function $m(x) = \mathbb{E}[f(x)]$ and a covariance function (or kernel) $k(x,x') = \mathbb{E}\big[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\big]$. This structure makes GP a *nonparametric model*, offering a versatile and flexible properties.

Recursive algorithm. To reduce GP computational complexity for real-time implementation, a sparsity method introduced by Huber [5], called *Recursive Gaussian Process*, is used: this method completely *avoids inversion* of the data-points covariance matrix by fixing a priori a set of basis vectors, *i.e.*, the stored dataset inputs. Their outputs are updated iteratively as the new observations of v_e are obtained, using an algorithm which reminds the Kalman filter. This modification reduces the computational complexity of GP inference *from cubic to quadratic*.

Controller integration. The RGP operates in two phases: *running and update*.

In the running phase, using the states of the systems z and v_d, the RGP infers the nonlinear mismatch v_e from the RGP basis vectors dataset and adds it to v_d, *i.e.*, $v_{cmd} = v_d + v_e(z, v_d)$.

While in the update phase, previously collected observations of the input $[z, v]$ and the output $\tilde{v}_e$ are refined and used to update the RGP basis-vector outputs, improving the predictions. Refinement employs a *Savitzky–Golay filter* [6], a non-causal signal-processing technique that enables high-precision and efficient smoothing of signals.

The basis vectors are initialized in a *grid structure*, where the user defines for each input dimension x_i the number of samples n_i in the selected range $[x_{i,min}, x_{i,max}]$. The total number of basis vectors is $\prod n_i$.

To tune the kernel hyperparameters, for this application we choose the Squared Exponential (SE) kernel function, we use a gradient based minimization algorithm using data collected during a previous experiment, aiming at reducing the prediction error.

Implementation

The RGP part is run on a ground station that communicates via ROS1 with the drone during the experiments, providing also the setpoints for following the trajectory. This is done to limit the computational effort carried by the drone, which CPU and memory is very limited. The whole controller is running at $250Hz$, while the RGP component at $10Hz$ to avoid interference with the PID component that controls the angular velocity.

The experimental platform is the *ANT-X 2-DoF drone* (Fig. 2) produced by ANT-X (https://antx.it/about-us-antx/); a simulator that accurately mimics the behavior of the platform is also available. All experiments were performed in the Aerospace Systems Control Laboratory (ASCL) at Politecnico di Milano.

Parameters values. On the platform, the parameters useful for the FL control law are set to $c = 1.168\ \mathrm{s}^{-1}$ and $\bar{k} = 4{,}748\ \mathrm{m/s}^2$. To prove the efficiency of the controller, in the simulation

Materials Research Proceedings 69 (2026) 806-811 https://doi.org/10.21741/9781644904251-142

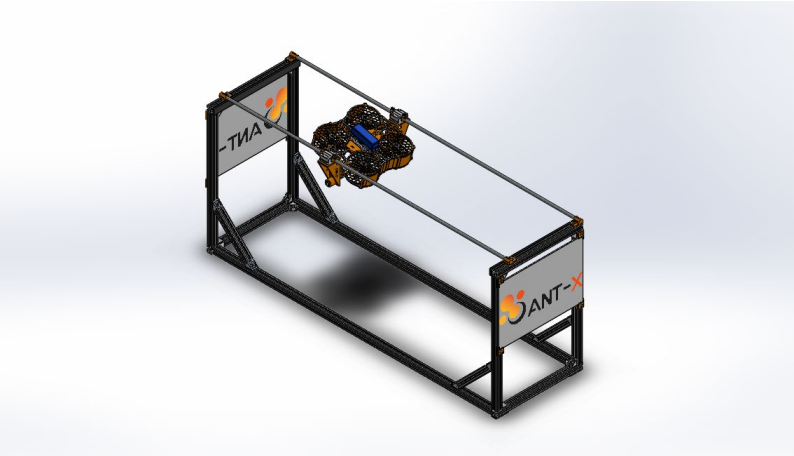

Figure 2: ANT-X 2-DoF drone.

moreover the measurement noise is added and in FL control action a slightly modified parameters respect to the identified model (*i.e.*, $\hat{c} = 2\ \mathrm{s}^{-1}$ and $\hat{\eta} = 12.5\ \mathrm{m/s}^2$) are used.
In both implementations, the grid of basis vector is initialized over the ranges $v_x \in [-0.3, 0.3]$, $\theta \in [-0.2, 0.2]$, and $v_{cmd}/v \in [-5, 5]$. For the physical platform, v_x, θ, and v_{cmd}/v are discretized into 4, 4, and 6 points, respectively, while for the simulator, they are discretized into 6, 6, and 8 points, respectively. The tuning algorithm produced different hyperparameter values for the RGP: for the platform, $l_{v_x} = 0.5\ \mathrm{m/s}$, $l_\theta = 0.5\ \mathrm{rad}$, $l_v = 1\ \mathrm{m/s}^3$, $\sigma_f = 1$, and $\sigma_n = 0.01$; for the simulator, $l_{v_x} = 0.4113\ \mathrm{m/s}$, $l_\theta = 0.4379\ \mathrm{rad}$, $l_v = 1.0272\ \mathrm{m/s}^3$, $\sigma_f = 0.9798$, and $\sigma_n = 0.016$.
The LQR parameters for the platform are set as $Q = [200, 1, 1]$ and $R = 1$, while for the simulation $Q = [100, 1, 1]$ and $R = 1$. The SGF parameter for the platform is $w_{size} = 51$; for the simulation $w_{size} = 21$.
The PID controller, pre-tuned by ANT-X, uses identical parameters for both the platform and simulation: $K_P = 0.009\ \mathrm{s/rad}$, $K_I = 0.21\ \mathrm{rad}^{-1}$ and $K_D = 0.0016\ \mathrm{s}^2/\mathrm{rad}$.

Results

To evaluate the performance, we used the *root-mean-square error (RMSE)* of the response with respect to the setpoint. The result shown for each reference signal is the *average of 5 experiments*. We performed two batches of simulated experiments: in the first batch the reference signal is a *sinusoid with constant amplitude* $r(t) = 0.2\sin(\omega t)$, while in the second one the signal is a *sinusoid with linear-varying amplitude* $r(t) = \frac{0.2}{60} t \sin(\omega t)$. The experiments are conducted varying the angular velocity: $\omega = 0.5, 0.8, 1, 1.5\ \mathrm{rad/s}$ for the first batch and $\omega = 1, 1.5\ \mathrm{rad/s}$ for the second one. We used sinusoidal signals because the time-domain results show the RGP's learning effect and the resulting reduction in error over time. Results in Fig. 3a and Fig. 3b show that *our proposed controller outperforms the baseline* (the same controller with no RGP contribution) with both types of signals.
In the real-world experiments, the reference signal is the *sinusoid with constant amplitude* $r(t) = 0.2\sin(\omega t)$ and the system is tested with angular velocities $\omega = 0.5, 0.8, 1.5\ \mathrm{rad/s}$. Improvements are significant in Fig. 3c at $\omega = 0.8\ \mathrm{rad/s}$, while at lower angular velocities the static friction can hinder the RGP's ability to accurately map v_e and at higher ones unmodeled nonlinear dynamics may adversely affect the RGP inference. In Fig. 4, a comparison between baseline and the full controller for reference tracking experiment with signal $r(t) = 0.2\sin(0.8\,t)$ is shown.

Materials Research Proceedings 69 (2026) 806-811 https://doi.org/10.21741/9781644904251-142

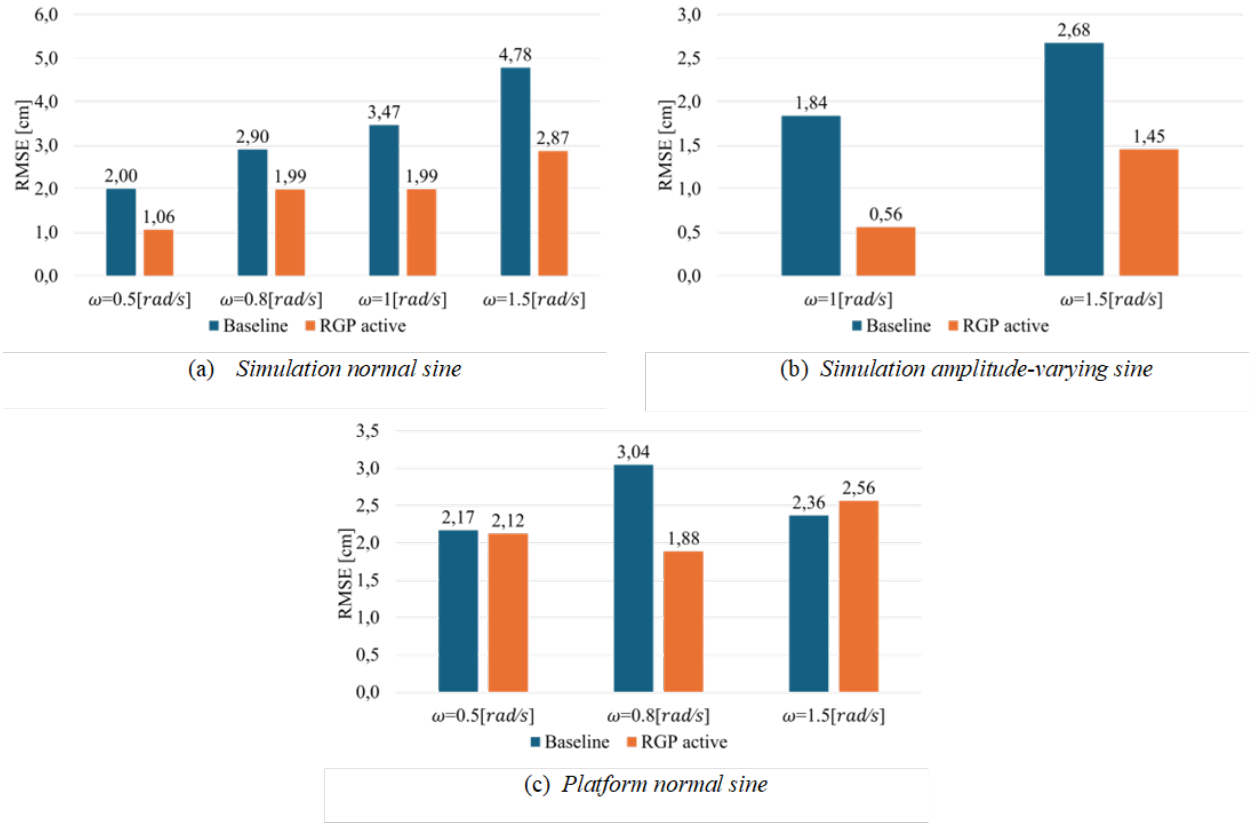

Figure 3: RMSE values for different angular velocities

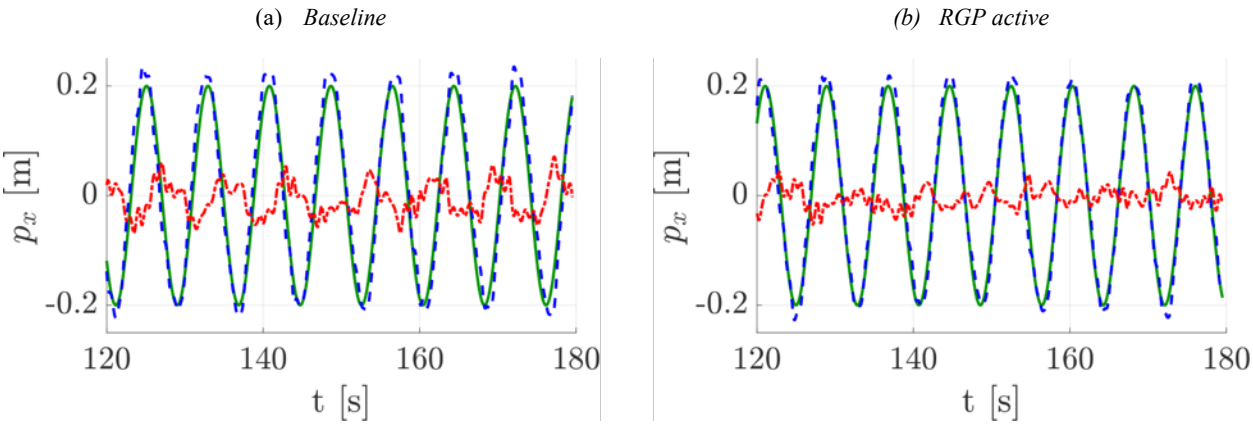

Figure 4: Platform tracking error for $r(t) = 0.2\,sin(0.8\,t)$

Conclusions

This paper aimed to design a learning-based tracking controller for a differentially flat system, which is proven to work effectively both in simulation and on the experimental platform. The computational issues associated with standard GP formulations were addressed using a sparsity method.

Future work includes the extension to an unconstrained drone, controlling all 3 axis, and the implementation of the RGP component directly onboard.

References

[1] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, "Feedback Linearization for Unknown Systems via Reinforcement Learning," Apr. 2020. arXiv:1910.13272 [math] https://doi.org/10.1109/ICRA40945.2020.9197158

[2] A. Yeşildirek and F. L. Lewis, "Feedback linearization using neural networks," Automatica, vol. 31, pp. 1659-1664, Nov. 1995. https://doi.org/10.1016/0005-1098(95)00078-B

[3] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning. Adaptive computation and machine learning", Cambridge, Mass.: MIT Press, 3. print ed., 2008.

[4] M. Greeff and A. P. Schoellig, "Exploiting Differential Flatness for Robust Learning-Based Tracking Control Using Gaussian Processes," IEEE Control Systems Letters, vol. 5, pp. 1121-1126, Oct. 2021. https://doi.org/10.1109/LCSYS.2020.3009177

[5] M. F. Huber, "Recursive Gaussian process: On-line regression and learning," Pattern Recognition Letters, vol. 45, pp. 85-91, Aug. 2014. https://doi.org/10.1016/j.patrec.2014.03.004

[6] R. Schafer, "What Is a Savitzky-Golay Filter? [Lecture Notes]," IEEE Signal Processing Magazine, vol. 28, pp. 111-117, July 2011. https://doi.org/10.1109/MSP.2011.941097