

Methods for Multi-objective Optimization PID Controller for quadrotor UAVs

Andrea Vaiuso^{a,*}, Gabriele Immordino^{a,2}, Ludovica Onofri^{a,c,3}, Giuliano Coppotelli^{c,4}, Marcello Righi^{a,5}

^a*School of Engineering, Zurich University of Applied Sciences ZHAW, Winterthur, Switzerland*

^b*Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, United Kingdom*

^c*Department of Mechanical and Aerospace Engineering, "La Sapienza" University of Rome , Italy*

Abstract

Integrating unmanned aerial vehicles into daily use requires controllers that ensure stable flight, efficient energy use, and reduced noise. Proportional integral derivative controllers remain standard but are highly sensitive to gain selection, with manual tuning often yielding suboptimal trade-offs. This paper studies different optimization techniques for the automated tuning of quadrotor proportional integral derivative gains under a unified simulation that couples a blade element momentum based aerodynamic model with a fast deep neural network surrogate, six degrees of freedom rigid body dynamics, turbulence, and a data driven acoustic surrogate model that predicts third octave spectra and propagates them to ground receivers. We compare three families of gradient-free optimizers: metaheuristics, Bayesian optimization, and deep reinforcement learning. Candidate controllers are evaluated using a composite cost function that incorporates multiple metrics, such as noise footprint and power consumption, simultaneously. Experiments are run in a complete set of missions, rather than studying a single response, spanning wide, unconstrained searches and bounded refinements, using a manually tuned Ziegler–Nichols baseline for reference. Metaheuristics improve performance consistently, with Grey Wolf Optimization producing optimal results. Bayesian optimization is sample efficient

*Corresponding Author

Email address: vaiu@zhaw.ch (Andrea Vaiuso)

¹Research Associate

²Post-doc

³PhD Student

⁴Associate Professor, AIAA Associate Fellow

⁵Professor, AIAA Senior Member, Lecturer at Federal Institute of Technology Zurich ETHZ

but carries higher per iteration overhead and depends on the design domain. The reinforcement learning agents do not surpass the baseline in the current setup, suggesting the problem formulation requires further refinement. On unseen missions the best tuned controller maintains accurate tracking while reducing oscillations, power demand, and acoustic emissions. These results show that noise aware proportional integral derivative tuning through black box search can deliver quieter and more efficient flight without hardware changes.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly recognized as versatile, multi-functional, and cost-effective tools, with applications in transportation, communication, agriculture, disaster response, and environmental monitoring [1]. Their integration into civil and commercial sectors continues to expand, offering sustainable alternatives that can reduce greenhouse gas emissions and mitigate other environmental impacts [2]. However, their widespread adoption in urban and suburban environments faces challenges related to efficiency, reliability, and public acceptance related to acoustic disturbance [3], that affects both humans and wildlife [4]. These limitations highlight the need for advanced UAV design techniques and controller tuning strategies [5], that can simultaneously account for different, and often conflicting, requirements. As a result, operating UAVs in densely populated environments poses an inherently multi-objective problem where competing performance criteria must be optimized and balanced simultaneously.

A large body of research has focused on hardware-oriented solutions to address the problem of noise pollution, including aerodynamic refinements, propeller geometry optimization, and the use of noise-absorbing materials [6, 7, 8, 9]. Although effective, such modifications often involve significant costs and may compromise aerodynamic performance. Therefore, control strategies are often preferred as a more flexible and less invasive alternative. Proportional–Integral–Derivative (PID) controllers remain the most widely adopted solution for UAV stabilization due to their simplicity, robustness, and ease of implementation [10, 11, 12].

Optimization-based control approaches have been extensively studied in literature, as a means to enhance the performance of existing UAV platforms through systematic controller tuning rather than structural redesign [13]. Classical PID tuning approaches, such as the Ziegler–Nichols technique [14] and other rule-based heuristics like Cohen-Coon and Gain and Phase methods [15], provide simple but straightforward procedures to determine gain values. However, they typically rely on empirical tests, are sensitive to model uncertainties, and tend to produce aggressive control

actions that may lead to overshoot, oscillations, or inefficient energy use especially in nonlinear dynamic models [16]. Lastly, these methods do not explicitly account for multi-objective trade-offs, including noise reduction and power efficiency.

One of the most widely explored methods for PID controls tuning is optimization based on metaheuristics [17, 16]. There exists a large variety of metaheuristic algorithms, ranging from evolutionary approaches to swarm intelligence and physics-inspired methods, each with its own strengths and trade-offs. Among them, Particle Swarm Optimization (PSO) [18] remains nowadays the most widely adopted technique in PID controller tuning [17], thanks to its balance between convergence speed and solution quality [19, 20]. Genetic Algorithms (GA) [21] are among the earliest metaheuristic optimization techniques. Despite years of use, they remain widely applied in PID-controls problem [22, 23, 24] due to their versatility and robustness against local minima, making them a useful reference point for illustrating how metaheuristic methods have evolved over time. On the other hand, Grey Wolf Optimization (GWO) [25], a metaheuristic inspired by the social hierarchy and co-operative hunting strategy of grey wolves, has recently demonstrated competitive or superior performance in various applications [26]. Its simplicity, efficient formulation, low computational cost, and adaptability have contributed to its growing use in engineering optimization and controller tuning problems [27, 28]. These methods are gradient-free and well suited to non-linear, multi-modal cost landscapes. However, finding near-optimal solutions for high-dimensional problems with metaheuristics often requires many iterations of population updates or iterative searches, resulting in long computation times. In addition, they may need to be adapted for each specific problem [29].

Another family of optimization methods is represented by black-box optimization methods, such as Bayesian Optimization (BO) [30]. Rather than evolving populations of candidate solutions as metaheuristics do, BO builds a probabilistic surrogate of the objective function and guides the search through an acquisition function that balances exploration and exploitation. This strategy is particularly effective when evaluating candidate solutions is computationally expensive [31], as in UAV simulations involving aerodynamics and acoustics. BO has shown strong performance in controller-tuning problems with a moderate number of decision variables, although its scalability to high-dimensional spaces is more limited [32, 33]. A key drawback is its strong dependence on the definition of the optimization domain: a poor choice may cause instability or slow convergence, thereby increasing the experimental cost. In practice, the domain is usually set using prior knowledge from simulations, experiments, or expert intuition. This reliance on manual tuning or trial-and-error approaches could limit the applicability of BO to automatic multi-

loop PID controller tuning in multivariable processes where defining an appropriate design space is particularly difficult [34].

More recently, Reinforcement Learning, and in particular Deep Reinforcement Learning, has emerged as a promising option for multiobjective tuning in continuous and high dimensional settings [35, 36, 37]. DRL has been applied to PID controller design, with studies reporting competitive results against classical methods and proposing various Markov Decision Process (MDP) formulations for gain tuning, including one-shot selection, online adaptive updating, and multi-phase schedules [38, 39, 40, 41].

In adaptive gain tuning, the agent is a gain updater that adjusts PID parameters online; the state collects error signals and short history terms; the action is a new set of gains or gain increments; the reward measures tracking quality and control effort over a finite reference task [42, 43]. Other works keep the agent online but focuses the state on error based features such as the error, its integral, and its rate of change; the action is an updated gain vector; the reward penalizes overshoot and oscillations during the run [44]. A different line embeds the PID relation in the decision itself: the agent outputs the control signal at each step, the state is the triplet of error, integral, and derivative, and the policy parameters play the role of effective gains learned across episodes [45]. In contrast, our study frames DRL tuning as a one step evolution: the agent proposes a complete PID vector, the environment returns the negative of a composite cost after a single simulation, and the episode ends immediately. This design is straightforward, essentially making the RL agent a black-box optimizer. This allows for a fair comparison with black-box optimizers while avoiding adaptive control during flight. However, it could also restrict the expressive power of the RL formulation. Since the resulting policies are static and lack temporal interaction, the agent may lose the ability to learn adaptive strategies that adjust gains in response to changing flight conditions.

This study presents a systematic comparison of three optimization approaches, metaheuristics, BO, and DRL, for tuning UAV PID controllers under realistic flight conditions. The proposed framework addresses multiple, often competing objectives, including flight stability, trajectory tracking accuracy, energy efficiency, and acoustic emissions. This contrasts with much of the existing literature, which typically evaluates optimization under narrow initial conditions or demonstrates controller performance through isolated responses or partial PID tuning [19, 20, 22, 13]. To this end, we developed a comprehensive simulation environment that captures the coupled effects of aerodynamics, rigid-body dynamics, and environmental disturbances. The framework integrates several key components: a reduced-order aerodynamic model coupled with data-driven surrogates for computational efficiency,

six degree-of-freedom rigid-body dynamics to simulate both translational and rotational motion, a stochastic turbulence model to replicate atmospheric disturbances, and an acoustic emission model to evaluate noise performance. The test scenario involves a quadcopter executing trajectory tracking maneuvers in turbulent conditions, with PID gains optimized through repeated closed-loop simulations. Building on this methodological foundation, the paper proceeds to evaluate and compare the three optimization approaches within the established simulation framework. Section 2 provides comprehensive details on each framework component: UAV dynamics and aerodynamics, controller architecture, acoustic modeling, and the implementation of the three optimization algorithms. Section 3 presents a comparative analysis of the optimization strategies, examining their performance trade-offs and relative strengths across all objectives. Section 4 synthesizes the key findings, acknowledges study limitations, and identifies promising avenues for future research.

2. Methodology

Our methodology aims to build a simulation environment that can cyclically and automatically evaluate the UAV performance by varying the PID configuration while generating useful synthetic data. The framework is specifically designed to incorporate acoustic emission metrics, alongside traditional flight dynamics, enabling the evaluation of control strategies not only in terms of stability, but also in terms of acoustic footprint and power efficiency. To achieve this, the following framework is structured into four principal modules, each conceived as an independent and interchangeable component: (1) a physical model, introduced in Section 2.1, that implements the dynamics and aerodynamics of the UAV in a simulated physical space, reproducing its rigid-body dynamics and rotor aerodynamics while incorporating turbulence effects; (2) a cascade PID-based controller responsible for translating navigation objectives into actuator commands, introduced in Section 2.2; (3) an acoustic model predicting noise emissions during the simulation, introduced in Section 2.3; and (4) an optimization layer, supporting metaheuristic algorithms (Section 2.5), BO (Section 2.6) and DRL (Section 2.7) methods, tasked with tuning the PID gains to achieve an optimal trade-off between flight performance and noise mitigation. A flowchart of the entire architecture is reported in Figure 1.

The simulation framework is designed to establish a balance between simulation fidelity and computational efficiency for in-loop optimization. Higher-fidelity simulations generate more reliable evaluations without requiring costly real-world experiments for every candidate control configuration, however, they can quickly become computationally expensive, especially when aerodynamic forces are determined by

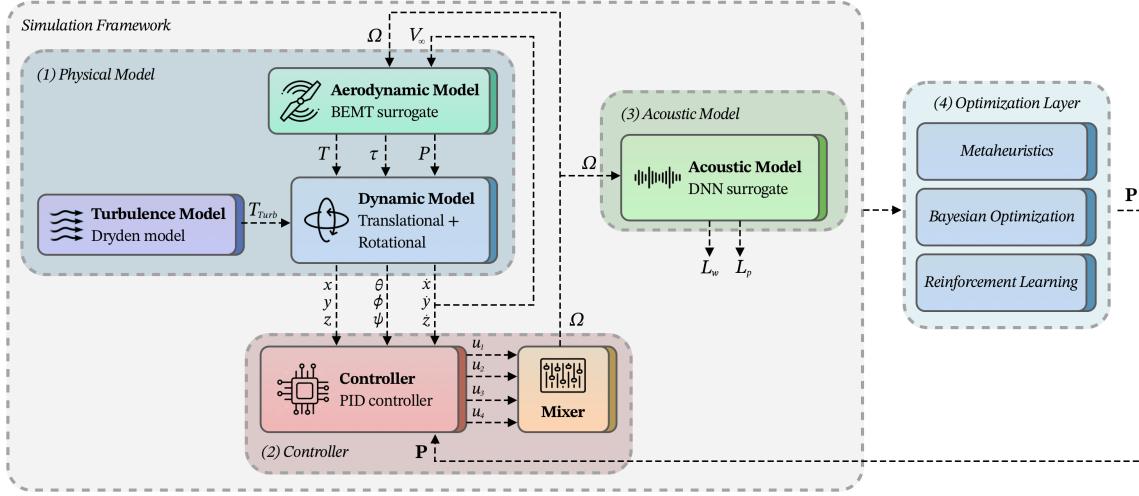


Figure 1: Architecture component flowchart. The optimization layer output \mathbf{P} represents a complete set of optimized PID.

complex simulation techniques, such as computational fluid dynamics (CFD). At the same time, a single PID-dependent simulation must be efficient enough to be effectively integrated into an optimization cycle. This trade-off is particularly important for noise-aware control as well, since the acoustic footprint is influenced not only by steady-state rotor speeds but also by transient events, such as turbulence-induced load fluctuations. For this reason, the aerodynamic model needs to combine speed with accuracy, which is why data-driven surrogate models are often employed in real-time or accelerated simulation scenarios [46, 47, 48]. To support reproducibility and adaptability to different fidelity requirements, the simulation environment has been implemented in Python as an open-source library. This allows for the integration of new aerodynamic models, acoustic models, controllers, and optimization algorithms that can easily be re-implemented or replaced with different reduced order models (ROM) or surrogates.

2.1. Physical Model

Blade Element Momentum Theory. While CFD can provide precise force estimation using iterative Navier-Stokes equation-based solvers, ROMs and low-order methods can offer a good balance of fidelity and efficiency. The aerodynamic model in our framework is based on the Blade Element Momentum Theory (BEMT) [49], which combines blade element theory with momentum theory to compute rotor thrust and torque. These models are widely employed in aeroacoustic research [50] to construct flight-mechanics representations of drones, efficiently accounting for spanwise

variations in angle of attack, chord length, and blade twist while maintaining low computational cost compared to traditional CFD. In our simulator, BEMT is used to generate rotor performance data (thrust T_i , torque τ_i , power P_i and corresponding coefficients) for different rotor speeds ω_i (see Appendix A) and freestream velocity.

Despite its efficiency, BEMT remains a low-order method with important limitations. It does not capture three-dimensional flow effects such as tip vortices, dynamic stall, and strong rotor-wake interactions, which can strongly influence performance and noise predictions in realistic flight conditions [51]. Its use therefore represents a pragmatic compromise: it enables fast data generation within a reasonable time frame, but the results must be interpreted with an awareness of these simplifications.

BEMT Surrogate. While BEMT provides good low-order predictions of rotor performance, repeated evaluation in an optimization loop can still be computationally expensive. To address this, a surrogate model based on a fully connected Deep Neural Network (DNN) has been trained offline using data generated from the implemented BEMT solver. The inputs to the network are the rotor angular velocity Ω_i (RPM) and freestream velocity V_∞ (m/s), and the outputs are six aerodynamic quantities: thrust T_i , torque τ_i , power P_i , and their respective non-dimensional coefficients C_T , C_Q , and C_P . The dataset was generated by sweeping rotor speed between 0 and 4,000 RPM and velocity from 1 to 20 m/s, yielding 8,000 samples divided into training and validation sets in an 80/20 split. All targets were normalized prior to training to improve convergence, and early stopping was employed to avoid overfitting.

To ensure robust evaluation, k-fold cross-validation with $k = 5$ was performed, and the network architecture along with its hyperparameters was optimized using `optuna` within the PyTorch framework. After training, the DNN replaces the BEMT in the simulations, achieving a substantial reduction in computation time while maintaining adequate accuracy. Performance was evaluated using Symmetric Mean Absolute Percentage Error (SMAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) metrics. The model exhibits a speedup of approximately 1,900 \times with a SMAPE validation error of around 5.75%. The surrogate performance is outlined in Table 1.

Table 1: DNN surrogate architecture, training configuration, and performance comparison with the BEMT solver.

	Parameter	Value
Architecture	Input features	2 (Ω_i, V_∞)
	Hidden layers	3 (16-32-16 units)
	Output features	6 ($T_i, \tau_i, P_i, C_T, C_Q, C_P$)
	Activation function	ReLU
Training setup	Loss function	MSE
	Optimizer	Adam
	Learning rate	1×10^{-4}
	Training-set size	6,400
Performance	Validation-set size	1,600
	Training time	≈ 40 s per fold
	DNN Prediction time	1.46×10^{-4} s
	BEMT Prediction time	2.7×10^{-1} s
Speedup (BEMT/DNN)	MAE	3.18
	RMSE	8.99
	SMAPE	5.75%
		$\approx 1,900 \times$

An important feature of the surrogate implementation is model interchangeability. The simulator queries rotor forces and moments through a unified interface, meaning that the DNN can be retrained with datasets obtained from higher-fidelity simulations or experimental rotor measurements. Alternatively, it can be substituted with more advanced neural architectures without altering the rest of the simulation framework. This design choice directly supports the integration of Navier-Stokes surrogates based on promising new ML approaches, which have recently demonstrated strong potential and applicability [52, 48, 53, 54]. Modularity enables the aerodynamic model to evolve in fidelity without compromising the efficiency necessary for large-scale optimization, thus achieving an adaptable balance between computational cost and physical accuracy.

Translational dynamics. Translational and rotational UAV dynamics are simulated using a rigid-body six degree-of-freedom model, represented in Figure 2, with roll (ϕ), pitch(θ), and yaw (ψ) correspond to rotations about the body x, y, and z axes. We follow the formulation in [10], adapted to use thrust and torque values

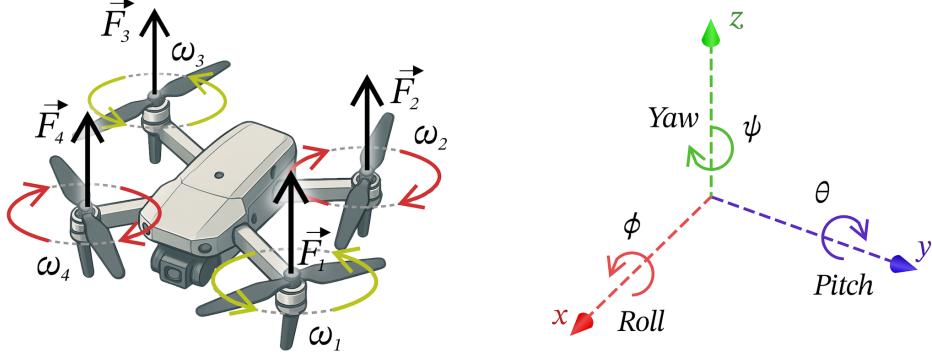


Figure 2: Physical quadricopter drone model representation

computed from the BEMT surrogate implementation. The translational motion law is:

$$m \dot{\mathbf{v}} = \mathbf{F}_T + \mathbf{F}_D - m \mathbf{g}, \quad (1)$$

where m is the UAV mass, $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^\top$ is the velocity in the world frame, and \mathbf{g} is the gravity acceleration vector.

The total thrust force in the world frame is:

$$\mathbf{F}_T = \mathbf{R}(\phi, \theta, \psi) [0, 0, T_\Sigma]^\top$$

where $T_\Sigma = \sum_{i=1}^4 T_i$ is the sum of the four rotor thrusts and $\mathbf{R}(\phi, \theta, \psi)$ is the body-to-world rotation matrix computed with a $z-y-x$ Euler sequence, using roll ϕ , pitch θ , and yaw ψ .

The aerodynamic drag is modeled as:

$$\mathbf{F}_D = -\text{diag}(C_{d,x}, C_{d,y}, C_{d,z}) \mathbf{v},$$

where $C_{d,x}$, $C_{d,y}$, and $C_{d,z}$ are linear drag coefficients along each axis.

Expanding Equation (1) for each component yields:

$$\ddot{x} = \frac{T_\Sigma}{m} (\cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi) - \frac{C_{d,x}}{m} \dot{x}, \quad (2)$$

$$\ddot{y} = \frac{T_\Sigma}{m} (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi) - \frac{C_{d,y}}{m} \dot{y}, \quad (3)$$

$$\ddot{z} = \frac{T_\Sigma}{m} (\cos \theta \cos \phi) - \frac{C_{d,z}}{m} \dot{z} - g. \quad (4)$$

In-time state propagation uses a fixed-step, fourth-order Runge–Kutta scheme [55] (see Appendix C). This integrator offers a good trade-off between accuracy and cost for closed-loop optimization runs and is standard in flight-dynamics simulation [56].

Rotational dynamics. The UAV rotational motion is governed by the Euler rigid-body equations with gyroscopic coupling. Let $\mathbf{I} = \text{diag}(I_x, I_y, I_z)$ be the inertia matrix, $p = \dot{\phi}$, $q = \dot{\theta}$, $r = \dot{\psi}$ the angular velocity components in the body frame, and J_r the combined spinning inertia of the rotor and shaft.

Control moments are decomposed along the body axes as:

$$u_1 = T_\Sigma \quad (\text{total thrust}), \quad (5)$$

$$u_2 = l(T_4 - T_2) \quad (\text{roll moment}), \quad (6)$$

$$u_3 = l(T_3 - T_1) \quad (\text{pitch moment}), \quad (7)$$

$$u_4 = \tau_1 - \tau_2 + \tau_3 - \tau_4 \quad (\text{yaw moment}), \quad (8)$$

where l is the arm length from the center to each rotor, and τ_i are the rotor reaction torques computed from the BEMT surrogate.

Let $\Omega_{\text{diff}} = (\omega_1 - \omega_2 + \omega_3 - \omega_4)$ be the signed sum of rotor angular speeds (counter-rotating pairs). The rotational dynamics then read:

$$I_x \dot{p} = u_2 - C_{a,x} \text{sgn}(p) p^2 - J_r \Omega_{\text{diff}} q - (I_z - I_y) q r, \quad (9)$$

$$I_y \dot{q} = u_3 - C_{a,y} \text{sgn}(q) q^2 + J_r \Omega_{\text{diff}} p - (I_x - I_z) p r, \quad (10)$$

$$I_z \dot{r} = u_4 - C_{a,z} \text{sgn}(r) r^2 - (I_y - I_x) p q, \quad (11)$$

where $C_{a,x}$, $C_{a,y}$, $C_{a,z}$ are quadratic rotational damping coefficients.

Dryden turbulence thrust augmentation. Atmospheric turbulence is modeled using the Dryden spectral model [57], a stochastic process representation frequently adopted in flight simulation and control system evaluation (see Appendix B). Stochastic wind components $(u_{\text{turb}}, v_{\text{turb}}, w_{\text{turb}})$ represent, respectively, the turbulent velocities along the x , y , and z axes of the inertial frame. These components are projected onto the rotor disk to obtain the local relative velocity:

$$U_{\text{disk}}(\theta, r) = \omega r + u_{\text{turb}} \cos \theta + v_{\text{turb}} \sin \theta$$

where ω is the rotor angular velocity (rad/s), r is the radial position along the blade span, and θ is the azimuthal angle around the disk.

The instantaneous load density over the disk is expressed as:

$$f(\theta, r) = \frac{2\pi w_{\text{turb}}}{\omega r} U_{\text{disk}}^2(\theta, r),$$

Integrating over the rotor annulus, with R_1 and R_2 denoting the inner and outer radii (root and tip) of the blade Section considered, yields:

$$\int_0^{2\pi} \int_{R_1}^{R_2} f(\theta, r) dr d\theta = 2\pi^2 w_{\text{turb}} \omega (R_2^2 - R_1^2) + \frac{2\pi^2 w_{\text{turb}}}{\omega} (u_{\text{turb}}^2 + v_{\text{turb}}^2) \ln \frac{R_2}{R_1}. \quad (12)$$

The turbulence-induced thrust increment is then:

$$T_{\text{turb}} = \frac{1}{2} \rho \left[\int_0^{2\pi} \int_{R_1}^{R_2} f(\theta, r) dr d\theta \right] \quad (13)$$

$$T_{\text{tot}} = T_{\Sigma} + T_{\text{turb}} \quad (14)$$

Equation (12) is evaluated for each rotor individually and its contribution is added to the BEMT-computed thrust following Equation (14).

2.2. Controller

The UAV is stabilised using a cascade PID architecture inspired by open-source flight stacks PX4 [58]. The structure consists of three main control loops:

- **Outer loop (position & altitude control):** Computes desired horizontal and vertical velocities from position and altitude errors.
- **Middle loop (attitude control):** Converts desired velocities into roll and pitch angle setpoints, and generates angular rate setpoints for roll, pitch, and yaw.
- **Inner loop (rate control):** Converts the angular rate setpoints to produce torque and thrust commands, which are finally distributed to the individual rotors via the mixer.

Layout. In total, the system implements five independent PID controllers: position in the x and y axes (sharing the same gain set), altitude control, roll and pitch and yaw attitude control (sharing the same gain set), horizontal speed control, and vertical speed control. Each PID block has proportional, integral, and derivative gains $\{K_p, K_i, K_d\}$, resulting in 15 possible tunable parameters. The whole layout is represented in Figure 3. The outer loop regulates position and altitude, generating

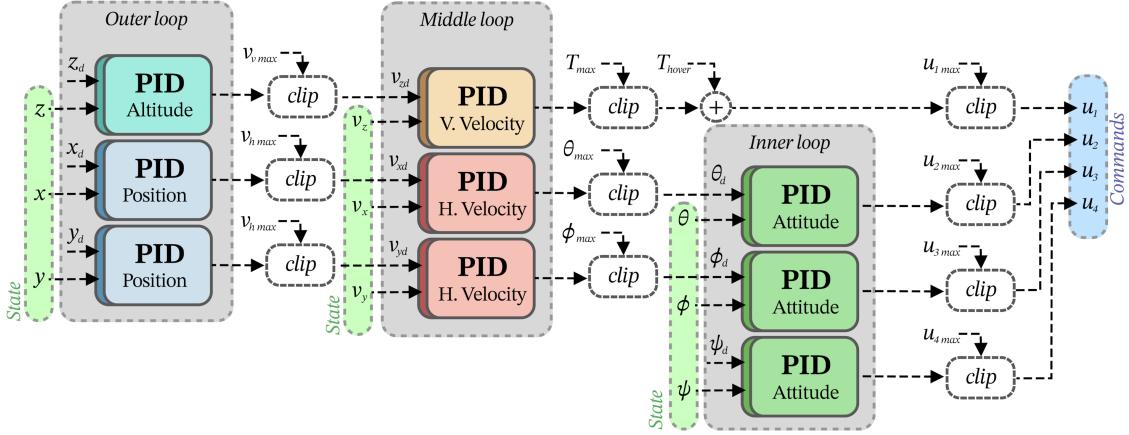


Figure 3: Representation of the quadcopter controller layout. Dashed green boxes indicate if a parameter is part of the drone current state, while z_d , x_d , y_d , v_{zd} , v_{xd} , v_{yd} , θ_d , ϕ_d and ψ_d are the desired values calculated based on target position.

desired velocities. The middle loop maps velocities to desired thrust and attitudes. The inner loop tracks attitudes, outputting per-rotor commands u_1 – u_4 . Some PID controllers share the same gains to reduce the dimensionality of the optimization problem and to reflect inherent system symmetries. For instance, the UAV dynamics in the x and y directions are nearly identical under nominal conditions, which justifies adopting a common gain set for both position axes. Similarly, roll and pitch and yaw exhibit comparable dynamic behavior due to the geometric symmetry of the quadcopter frame, allowing them to be stabilized effectively with the same set of attitude gains. This parameter grouping not only simplifies tuning but also improves optimization efficiency by reducing the search space without sacrificing control performance.

Several mechanisms are included to improve stability and robustness. The controllers use anti-windup [59] by clamping the integral term to a limit proportional to the control authority, avoiding integral runaway during large or sustained errors. Command saturation is applied individually to thrust, roll, pitch, and yaw outputs, ensuring they remain within physical limits. Desired horizontal and vertical speeds, as well as roll and pitch angles, are bounded to prevent aggressive maneuvers that could destabilize the UAV. A feed-forward hover compensation term adjusts thrust commands according to the instantaneous roll and pitch, maintaining equilibrium during maneuvers [58].

Mixer and thrust/torque mapping. The mixer converts the desired collective thrust and body-axis moments into individual rotor speeds, taking into account quadrotor geometry, rotor spin directions, and the thrust–torque characteristics of each propeller–motor unit.

The control inputs to the mixer are:

u_1 (collective thrust), u_2 (roll moment), u_3 (pitch moment), u_4 (yaw moment).

For a quadrotor in the “+” configuration (where one rotor is aligned with the forward axis of the UAV), the mapping from rotor thrusts $\mathbf{T} = [T_1, T_2, T_3, T_4]^\top$ to control inputs, as defined in [10], is:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ \frac{d}{C_T} & -\frac{d}{C_T} & \frac{d}{C_T} & -\frac{d}{C_T} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix},$$

where C_T is the thrust coefficient and d is the drag factor. The inverse mapping distributes the control inputs into individual rotor thrusts:

$$\mathbf{T} = \mathbf{A}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}.$$

Using the relation $T_i = C_T \omega_i^2$, this can be expressed directly in terms of rotor angular speed squared:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \frac{1}{4C_T} \begin{bmatrix} 1 & 0 & -2/l & d/C_T \\ 1 & -2/l & 0 & -d/C_T \\ 1 & 0 & 2/l & d/C_T \\ 1 & 2/l & 0 & -d/C_T \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}.$$

The mixer ensures physical feasibility by clipping ω_i^2 to the range $[0, \omega_{\max}^2]$. The rotor speeds are then obtained from square roots and converted to revolutions per minute (RPM). The aerodynamic model then maps (RPM $_i$) to thrust T_i and torque τ_i for each rotor, coupling the control inputs to the rigid-body dynamics of the UAV.

2.3. Acoustic Model

The acoustic emissions of the multi-rotor UAV are modeled as a function of the radiation angle ζ and the rotor rotational speed Ω , since multicopter noise depends both on rotor RPM which control blade passing frequency (BPF), and on a characteristic directivity pattern with stronger radiation downward than sideways. In general, as the RPM increases, the overall sound power rises and the spectral content shifts toward higher frequencies. For a given operating condition, the sound power level spectrum $L_w(f, \zeta, \Omega)$ is expressed in discrete 1/3-octave bands f_i [60]. The use of 1/3-octave bands provides sufficient resolution to capture the relevant tonal components of rotor noise while avoiding the complexity and data requirements of narrowband analyses such as FFT [61], though it is not adequate for detailed psychoacoustic evaluation [62]. This representation offers a practical balance between spectral fidelity and computational tractability for UAV noise modeling.

To integrate an acoustic model into the simulation framework, different approaches were evaluated, with the goal of ensuring high accuracy while maintaining computational efficiency for use in optimization loops. Data-driven methods were selected as the most suitable compromise, as they can capture complex acoustic dependencies while remaining faster than high-fidelity physics-based models [63, 64]. Among existing approaches, the empirical regression model proposed by Wunderli [62] was initially implemented. This method assumes that the directivity pattern of the UAV noise is nearly independent of rotor speed and flight maneuver, expressing L_w as a polynomial function of the deviations in radiation angle $\Delta\zeta$ and rotor speed $\Delta\Omega$ from a reference condition (e.g., hover), plus a maneuver-dependent correction term C_{proc} .

A second approach employed a fully connected DNN to directly regress the frequency-domain spectrum. Both models were trained on a real data measurement dataset [65], acquired according to ISO 5305 [66] standards during a campaign on a DJI Matrice 300 UAV [67], yielding 210 samples containing log data on Ω (RPM), ζ (rad), pitch, roll, yaw, and C_{proc} , calculated following [62], and the third-octave-band L_w spectrum used as supervised label.

For the polynomial model, after normalizing the RPM and C_{proc} values, the regression converged in 55 iterations with a runtime of about 3 minutes.

The surrogate archived a SMAPE of 9.89%, RMSE of 7.56, and MAE of 5.59 on a separated test set of 32 samples. The DNN, consisting of three fully connected layers with LeakyReLU activation, was trained using the Adam optimizer with a learning rate of 10^{-4} . Early stopping and hyperparameter optimization (`optuna`) were used to prevent overfitting, with performance evaluated via k-fold cross-validation with $k = 5$. The dataset was split into 147 training, 31 validation, and 32 test samples.

The optimized DNN outperformed the empirical model, achieving a test SMAPE of 3.29%, RMSE of 2.5, and MAE of 1.9, with an average training time of 30 seconds per fold and ≈ 27 times faster prediction time.

For simulation, the trained DNN is combined with a lookup table of angle-dependent corrections to compute the instantaneous sound power level for each rotor at its current orientation. The contribution of a single rotor is isolated by subtracting 6 dB from the reference hovering spectrum.

Table 2: Comparison of the Wunderli regression model [62] and the optimized DNN for UAV acoustic prediction.

	Parameter	Wunderli Model [62]	DNN Model
Architecture	Input features	6 (Ω , ζ , C_{proc})	5 (Ω , ζ)
	Hidden layers	2 nd order poly.	3 (13-40-63 units)
	Output features	63 (L_w)	63 (L_w)
	Activation function	–	LeakyReLU
Training setup	Loss function	MSE	MSE
	Optimizer	L-BFGS-B	Adam
	Learning rate	–	1×10^{-4}
	Training-set size	178	147
	Validation-set size	–	31
Performance	Test-set size	32	32
	Training time	206 s	≈ 30 s per fold
	Prediction time	3.5×10^{-3}	1.3×10^{-4}
	MAE	5.59	1.90
	RMSE	7.56	2.50
Speedup (Wunderli/DNN)	SMAPE	9.89%	3.29%
			$\approx 27 \times$

Propagation Model

In order to calculate the step-by-step Sound Pressure Level (SPL) signal in the ground, L_w is propagated to ground receivers using a free-field spherical spreading model with optional ISO 9613-1 [68] atmospheric absorption. The simulation space domain is discretized into a spatial $N \times N$ grid, with each grid cell g represented by its horizontal centroid at height $z = 0$. At each simulation time step t_j , the UAV position and orientation determine the instantaneous distance $d_g(t_j)$ and radiation

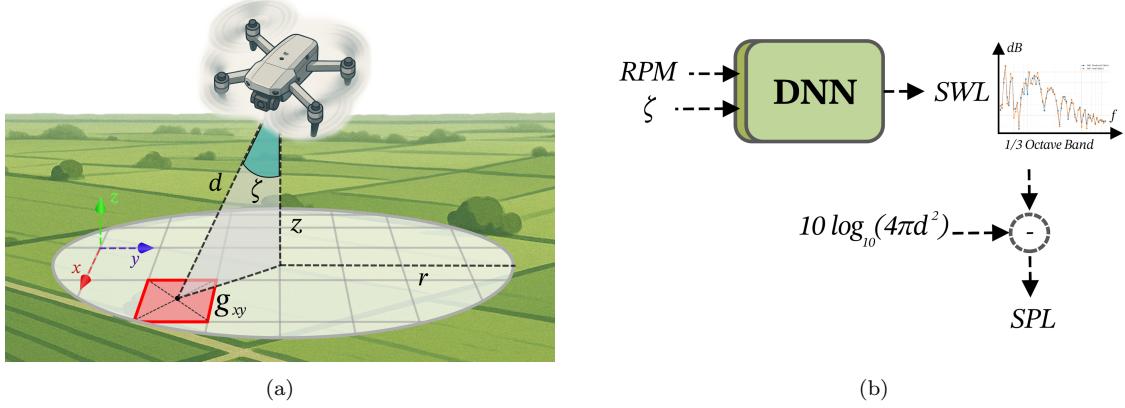


Figure 4: Figure (a) Discretized simulation space domain in cells. Figure (b) Schematic of the noise model that generates the final SPL value.

angle $\zeta_g(t_j)$ for each cell, as shown in Figure 4a. The sound pressure level signal is aggregated step by step within each cell at a radius limit of r , using the radiation angle ζ and the distance from the cell centroid to the noise source of d . Using the propagation model above, the 1/3-octave-band SPL spectrum $L_p(f, t_j, g)$ is computed for all grid points, following the approach illustrated in Figure 4b.

Propagation from the UAV to each receiver point is computed on a per-band basis. The free-field spherical spreading attenuation is:

$$A_{sp}(d) = 10 \log_{10} (4\pi d^2) \quad (15)$$

Atmospheric absorption per frequency band is computed according to ISO 9613-1 [68]: $Atm_{abs}(f, f_{rO}, f_{rN}, p_r, T, T_0)$ as a function of f_{rO} and f_{rN} that are the oxygen and nitrogen relaxation frequencies respectively, p_r is the relative pressure, T is the absolute temperature (K), and $T_0 = 293.15$ K. An optional directivity index $DI(f, \theta)$ can be added if not already embedded in the reference emission spectra (see the complete formulation in Appendix D).

The received SPL per band at a receiver point is then:

$$L_p(f, t) = L_w^{\text{total}}(f, t) - A_{sp}(d(t)) - Atm_{abs}(f) d(t) + DI(f, \theta(t)) \quad (16)$$

To reduce computational complexity, spectral updates are skipped for cells whose predicted broadband SPL is outside a pre-defined radius r from the projected UAV position in the ground (for reproducibility, simulation parameters are reported in Appendix E).

2.4. Optimization Framework

The optimization framework is designed to evaluate candidate PID gain configurations by simulating the UAV response in predefined mission scenarios and computing a composite cost function that incorporates multiple performance metrics.

To determine the optimal PID controller gains for UAV control, we employ three main families of gradient-free optimization approaches:

- **Metaheuristic search algorithms**, which explore the parameter space by evolving a population of candidate solutions according to heuristic rules inspired by natural processes, which iteratively update a population of candidate solutions based on exploration–exploitation heuristics.
- **Bayesian Optimization (BO)**, which builds a probabilistic surrogate of the objective function and selects new candidates by maximizing an acquisition function that balances exploration and exploitation.
- **Reinforcement Learning (RL) methods**, where an agent learns an optimal control policy by interacting with the simulation environment and receiving scalar feedback (rewards) that encode performance objectives.

All strategies share a common evaluation framework: each candidate set of gains is tested in the UAV simulation, the resulting trajectory, energy consumption, and acoustic profile are recorded, and a composite cost function is computed, ensuring a fair comparison between algorithms under identical conditions.

The evaluation of each candidate PID configuration is performed by simulating the UAV in a fixed representative mission scenario using the simulation framework, and computing the cost function. To avoid overfitting the gains to a single route or a short path, we construct a single composite mission $\tilde{s} = (s_1, \dots, s_J)$ that concatenates heterogeneous segments (short hops, long transits, S-turns, climb/descents, hover–translate–hover) with varied speeds and altitudes. Additionally, the introduction of the Dryden turbulence, can generate noise, making the overall optimization more robust and generalizable.

This approach diverges from existing literature, where optimization performance is typically evaluated under limited initial conditions or demonstrates controller effectiveness through single responses or individual PID component tuning [19, 20, 22, 13]. Given the inherent nonlinear coupling characteristics of the quadrotor system, comprehensive evaluation requires challenging the controller with complex, multi-dimensional trajectories that exercise the full range of system dynamics. The selected trajectory design employs heuristic principles that systematically explore the

vehicle flight envelope, ensuring that the optimization algorithms are tested across representative operational conditions covering most of the flight envelope.

The cost function is designed to penalize long mission times, large final position errors, excessive oscillations in attitude and thrust, high energy consumption, waypoint overshoot, incomplete missions, and elevated acoustic levels. Each term is weighted to reflect its relative importance in the specific application, with gains determined through an heuristic tuning process. From a scientific perspective, the exact choice of these weights is not critical, as it is application-specific and primarily serves to balance competing objectives. The cost function weights were calibrated by first obtaining a manually tuned baseline controller using the Ziegler–Nichols method [14]. To prevent unnecessary computation, simulations were terminated early when PID gains produced clearly unfeasible behavior. Specifically, runs were aborted if the UAV rapidly diverged from the initial waypoint, failed to initiate movement, or required excessive time to reach the first target. In such cases, a high penalty cost was assigned. Individual cost terms were then balanced to comparable magnitudes under this reference configuration, ensuring that no single metric disproportionately dominated the optimization. The resulting cost function promotes fast, accurate, stable, energy-efficient, and acoustically unobtrusive flight behavior.

The total cost is represented by the following equation, where each term is explained in Table 3.

$$J(\mathbf{P}) = w_t C_t + w_d C_d + w_o C_o + w_c C_c + w_{os} C_{os} + w_p C_p + w_n C_n + C_{nm} \quad (17)$$

where $\mathbf{P} \in \mathbb{R}^d$ stacks the controller gains to be optimized.

Table 3: Cost function components

Penalty term	Formula	Description
Mission time cost (C_t)	T_f	Total simulation time required to reach the final waypoint. Lower values indicate faster mission completion.
Final position error (C_d)	$\ \mathbf{p}_f - \mathbf{p}^*\ ^{\gamma_d}$	Euclidean distance between the UAV's final position \mathbf{p}_f and the target \mathbf{p}^* , shaped by exponent $\gamma_d \in (0, 1]$ to control sensitivity to large errors in missions where the final target is not reached.
Attitude oscillation (C_o)	$\sum_k (\Delta\phi_k + \Delta\theta_k)$	Sum of absolute changes in roll (ϕ) and pitch (θ) over all time steps k , representing unnecessary or unstable attitude variations.
Thrust oscillation (C_{to})	$\sum_k \Delta T_k $	Sum of absolute variations in total thrust command T_k over the mission, penalizing rapid fluctuations in rotor speed demand.
Completion cost (C_c)	$\begin{cases} 0, & \text{mission completed} \\ P_c, & \text{otherwise} \end{cases}$	Large increasing penalty P_c applied if the UAV fails to visit all waypoints within the allowed time.
Overshoot (C_{os})	$\sum_{j=1}^{N_w} \max(0, \ \mathbf{p}_{\max,j} - \mathbf{p}_j^*\)$	Penalizes overshooting each waypoint j by measuring the maximum excess distance $\mathbf{p}_{\max,j}$ beyond the target \mathbf{p}_j^* .
Power consumption (C_p)	$\sum_k P_k$	Total electrical energy expenditure computed as the sum of instantaneous power draws P_k at each time step.
Noise cost (C_n)	$\ \mathbf{s}\ _p^p + \max(\mathbf{s})$	Combination of the p -norm of the broadband SWL vector s , obtained by power-summing the per-band L_w values across frequency, and its maximum value $\max(\mathbf{s})$.
No-movement cost (C_{nm})	$\begin{cases} P_{nm}, & \text{if disp.} < \epsilon \\ 0, & \text{otherwise} \end{cases}$	Large penalty P_{nm} if the UAV displacement is below the threshold ϵ , preventing and explicitly skipping PID solutions that underestimate the minimum power required to initiate and sustain UAV motion.

The w . coefficients are application-specific weights obtained heuristically to balance the influence of each term in the aggregated objective.

2.5. Metaheuristic Optimization

We cast PID tuning as the minimization of the black-box objective $J(\mathbf{P})$ in (17). A metaheuristic maintains a population $\{\mathbf{P}_i\}_{i=1}^N$ of candidate gain vectors, evaluates $J(\mathbf{P}_i)$ by closed-loop simulation, and updates the population using exploration-exploitation rules that do not require gradients. The formulations below follow established practice for controller tuning [17].

Genetic Algorithm

In a generational Genetic Algorithm (GA) [21], each candidate \mathbf{P}_i is assigned a fitness F_i monotonically related to performance, e.g. $F_i = -J(\mathbf{P}_i)$. Stochastic

selection can be implemented via fitness-proportionate sampling,

$$p_i = \frac{F_i}{\sum_{j=1}^N F_j}, \quad (18)$$

where p_i is the probability that individual i is chosen as a parent, F_i is its fitness, and N is the population size. New offspring are generated by crossover, for example the arithmetic form

$$\mathbf{P}^{(\text{child})} = \lambda \mathbf{P}^{(\text{parent1})} + (1 - \lambda) \mathbf{P}^{(\text{parent2})}, \quad \lambda \in [0, 1], \quad (19)$$

where λ is the mixing coefficient weighting the two parents. A mutation step perturbs offspring components,

$$P_k^{(\text{child})} \leftarrow P_k^{(\text{child})} + \sigma_k \xi_k, \quad \xi_k \sim \mathcal{N}(0, 1), \quad (20)$$

where P_k is the k -th parameter, $\sigma_k > 0$ is the mutation scale, and ξ_k is standard Gaussian noise. Elitism optionally copies the best individuals unchanged to the next generation, improving stability on rugged landscapes.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) [18] represents each candidate as a particle with position \mathbf{x}_i^t and velocity \mathbf{v}_i^t at iteration t . Let \mathbf{p}_i denote the particle's personal best and \mathbf{g} the global best in the swarm. The standard update reads

$$\mathbf{v}_i^{t+1} = \chi \mathbf{v}_i^t + c_1 \mathbf{r}_1 \odot (\mathbf{p}_i - \mathbf{x}_i^t) + c_2 \mathbf{r}_2 \odot (\mathbf{g} - \mathbf{x}_i^t), \quad (21)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \quad (22)$$

where $\chi \in [0, 1]$ is the inertia weight, $c_1, c_2 > 0$ are the cognitive and social acceleration coefficients, $\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}[0, 1]^d$ are independent identically distributed (i.i.d) vectors of uniform random numbers (drawn componentwise), and \odot denotes the Hadamard (elementwise) product. Equations (21)–(22) bias motion toward the personal and global best positions while retaining momentum through χ , yielding fast convergence with modest hyperparameter sensitivity.

Grey Wolf Optimizer

Grey Wolf Optimization (GWO) [25] abstracts the encircling and hunting strategy of wolves with a leadership hierarchy. Let \mathbf{X}_α , \mathbf{X}_β , and \mathbf{X}_δ be the three best

solutions at iteration t , and let \mathbf{X} denote the position of a generic wolf. Encircling and guidance are defined by

$$\mathbf{D}_\alpha = |\mathbf{C}_1 \odot \mathbf{X}_\alpha - \mathbf{X}|, \quad \mathbf{X}_1 = \mathbf{X}_\alpha - \mathbf{A}_1 \odot \mathbf{D}_\alpha, \quad (23)$$

$$\mathbf{D}_\beta = |\mathbf{C}_2 \odot \mathbf{X}_\beta - \mathbf{X}|, \quad \mathbf{X}_2 = \mathbf{X}_\beta - \mathbf{A}_2 \odot \mathbf{D}_\beta, \quad (24)$$

$$\mathbf{D}_\delta = |\mathbf{C}_3 \odot \mathbf{X}_\delta - \mathbf{X}|, \quad \mathbf{X}_3 = \mathbf{X}_\delta - \mathbf{A}_3 \odot \mathbf{D}_\delta, \quad (25)$$

with the position update

$$\mathbf{X}^{t+1} = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3}, \quad \mathbf{A}_j = 2a \mathbf{r}_j - a, \quad \mathbf{C}_j = 2 \mathbf{r}_j. \quad (26)$$

Here $\mathbf{r}_j \sim \mathcal{U}[0, 1]^d$, \odot denotes elementwise multiplication, and a decreases linearly from 2 to 0 over iterations. Large $|\mathbf{A}_j|$ encourages exploration early on; as $a \rightarrow 0$ the search contracts around $\mathbf{X}_\alpha, \mathbf{X}_\beta, \mathbf{X}_\delta$, effecting exploitation.

2.6. Bayesian Optimization

Bayesian Optimization (BO) is employed to minimize the black-box objective $J(\mathbf{P})$ in (17) under simple box constraints $\Pi = \{\mathbf{P} \in \mathbb{R}^{15} \mid \boldsymbol{\ell} \leq \mathbf{P} \leq \mathbf{u}\}$. The parameter vector \mathbf{P} stacks five PID triplets (K_p, K_i, K_d) . At each iteration t , BO maintains a probabilistic surrogate of the cost conditioned on the evaluated designs $\mathcal{D}_t = \{(\mathbf{P}_i, J(\mathbf{P}_i))\}_{i=1}^t$, and proposes the next candidate by maximizing an acquisition function α that balances exploration and exploitation:

$$\mathbf{P}_{t+1} = \arg \max_{\mathbf{P} \in \Pi} \alpha(\mathbf{P}; \mathcal{D}_t).$$

This sequential design enables BO to locate near-optimal solutions in relatively few evaluations, making it particularly well suited when each simulation is expensive. Each function evaluation executes the full composite mission in the simulator (aero-dynamics, noise model, and, when enabled, Dryden turbulence), returning the scalar cost $J(\mathbf{P})$. This setup exploits BO's sample efficiency when each rollout is relatively expensive, while keeping the dimensionality moderate so that surrogate modeling remains effective.

2.7. Reinforcement Learning

We conceptualized our RL implementation with a question: can a DRL agent capture the essence of an optimization problem when framed with the most elementary action–return formulation? Our curiosity emerged from the possibility that modern DRL algorithms could serve as general-purpose optimizers for tasks that cannot be

easily broken down into multi-step episodes with structured intermediate rewards. This perspective is especially relevant for classes of control and design problems, such as PID controller tuning or aerodynamic shape optimization, where the evaluation of a candidate solution often provides only a single terminal reward without meaningful intermediate feedback.

In this context, we formulated PID tuning as a continuous, bound-constrained one step MDP. There is no notion of state transitions or long-term planning within an episode: each episode consists of a single decision and immediate terminal feedback. In the other hand, a fully multi-state RL formulation would treat the PID gains as adaptive variables that evolve with the flight state and environment, yielding a dynamic, context-dependent controller, i.e. adaptive control. While such an approach is promising, it is beyond our current scope and strays from previous considerations. Consequently, the problem reduces to learning a sampling policy that maximizes expected immediate reward, without modeling state transitions or intra-episode adaptation. Framing the problem in this way allows us to investigate whether general-purpose RL algorithms can effectively solve static parameter selection under noisy returns, and to directly compare the learned sampling policy against traditional optimization methods without the need to redesign the control architecture.

Thus, the MDP schema is modeled as follows. Let $\mathbf{P} \in \Pi \subset \mathbb{R}^{15}$ denote the vector of tunable PID gains for the six optimized loops: x and y position, altitude, roll–pitch attitude, horizontal speed, and vertical speed, while yaw gains are kept fixed to reduce dimensionality. The feasible set is the hyper-rectangle

$$\Pi = \{ \mathbf{P} \in \mathbb{R}^{15} \mid \boldsymbol{\ell} \leq \mathbf{P} \leq \mathbf{u} \}, \quad (27)$$

with bounds $\boldsymbol{\ell}, \mathbf{u} \in \mathbb{R}^{15}$ chosen consistently with the ranges used both in the meta-heuristics and BO.

An action is the gain vector $a \equiv \mathbf{P} \in \Pi$, which is decoded into grouped (K_p, K_i, K_d) triplets for the six loops. Executing a triggers a closed-loop simulation on the composite mission with those gains and returns the scalar objective $J(\mathbf{P})$ defined by the cost function. The (undiscounted) reward is

$$r(\mathbf{P}) = -J(\mathbf{P}). \quad (28)$$

Each episode has horizon $H = 1$: the agent proposes \mathbf{P} , the simulator returns $r(\mathbf{P})$, and the episode terminates. The observation (context) s is a bounded vector in \mathbb{R}^{15} used only to condition the policy. In practice, it is either fixed to a nominal initialization or independently resampled at reset from a prescribed distribution. Since episodes terminate immediately, the transition kernel is trivial and the value

function reduces to expected immediate reward. The learning objective is therefore

$$\max_{\pi} \mathbb{E}_{s \sim \rho, \mathbf{P} \sim \pi(\cdot|s)} [r(\mathbf{P})] = - \min_{\pi} \mathbb{E}_{s \sim \rho, \mathbf{P} \sim \pi(\cdot|s)} [J(\mathbf{P})], \quad (29)$$

where ρ denotes the reset distribution over contexts.

Optional stochasticity is introduced by enabling Dryden turbulence and waypoint randomization during rollouts, so the return is noisy even for fixed \mathbf{P} . From the agent's standpoint, this corresponds to i.i.d. bandit pulls with stochastic rewards, provided resets and seeds are independent across episodes [69].

We employ off-policy actor-critic methods with replay to learn a proposal (sampling) policy over Π . Although such algorithms are generally designed for multi-step control, in the single-step setting their Bellman targets collapse to the immediate reward, so they act as entropy-regularized derivative-free optimizers with learned proposal distributions.

Soft Actor-Critic

Soft Actor-Critic (SAC) is an off-policy method that normally learns a stochastic policy by maximizing expected return together with an entropy bonus that promotes exploration [70]. In our one-step formulation, the agent proposes a gain vector \mathbf{P} , receives an immediate reward $r(\mathbf{P})$, and the episode terminates. Hence, the usual soft Bellman recursion collapses to a direct target

$$y = r(\mathbf{P}), \quad (30)$$

so each critic Q_{ψ_j} is simply trained to regress onto the observed reward:

$$\mathbb{E}\left[(Q_{\psi_j}(s, \mathbf{P}) - y)^2\right], \quad j \in \{1, 2\}. \quad (31)$$

The actor update retains its entropy-regularized form,

$$J_{\pi}(\phi) = \mathbb{E}_{s \sim \mathcal{D}, \mathbf{P} \sim \pi_{\phi}(\cdot|s)} \left[\alpha \log \pi_{\phi}(\mathbf{P} | s) - \min_j Q_{\psi_j}(s, \mathbf{P}) \right], \quad (32)$$

where the temperature α is adapted to match a target entropy. Actions are rescaled to lie inside Π . Because there are no future rewards, SAC here learns a probability distribution over Π that places higher density on low-cost regions while maintaining randomness to keep exploration active.

Twin Delayed DDPG

Twin Delayed DDPG (TD3) is an off-policy algorithm that learns a deterministic actor $\mu_\phi(s)$ with two critics to counteract value overestimation [71]. In a standard multi-step setting, targets would include bootstrapped estimates of future value. Here, with horizon one, the target is purely the observed reward:

$$y = r(\mathbf{P}), \quad \mathbf{P} = \mu_{\bar{\phi}}(s) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \text{ (clipped)}, \quad (33)$$

where small Gaussian noise is added for target policy smoothing. Each critic is trained by minimizing

$$\mathbb{E}\left[\left(Q_{\psi_j}(s, \mathbf{P}) - y\right)^2\right], \quad (34)$$

and the actor parameters are updated less frequently using the deterministic policy gradient

$$\nabla_\phi J(\phi) \approx \nabla_\phi Q_{\psi_1}(s, \mu_\phi(s)).$$

All proposed actions are projected into Π . The update delay and smoothing mechanisms stabilize training even in this simplified one-step setting.

2.8. Considerations

To ensure the optimization loop remains computationally feasible while still capturing the main aerodynamic and acoustic mechanisms, we adopt a layered modeling strategy. Aerodynamics are evaluated through a surrogate trained on BEMT to avoid costly high-fidelity sweeps, and acoustic emissions are estimated per rotor in 1/3-octave bands, propagated on an $N \times N$ ground grid with radius-based culling so that only cells above a relevant threshold are updated. In practice, N is chosen so that SPL maps change by less than about 1 dB upon refinement, while the active radius is derived online from instantaneous source level, spherical spreading, and optional ISO 9613-1 absorption to skip far-field cells with negligible contribution. This design keeps computational load manageable and enables the large number of rollouts required by optimization algorithms.

Within the optimization loop, the cost function is defined using proxies that balance accuracy and efficiency. For acoustic performance, the cost uses a SWL-based measure that combines a high-order norm of the sound power level time series with its maximum value, penalizing both sustained energy and peaks. This formulation captures the main characteristics of rotor noise while remaining computationally inexpensive, but neglects detailed frequency-domain features such as tonal harmonics and fine-grained modulation effects, which may be relevant for high-fidelity acoustic

characterization, or psychoacoustic evaluations. Similarly, the aerodynamic surrogate abstracts away wake interactions and coherence effects between rotors. However, for controller optimization the chosen formulation is adequate, as it enables thousands of rollouts with consistent ranking of candidate controllers, while more detailed models would render the loop intractable.

A final point concerns the RL formulation. In standard multi-step settings, SAC and TD3 rely on bootstrapping to propagate value estimates across time, but in our one-step formulation each trial terminates immediately, so their Bellman targets collapse to the observed reward. This makes the critics essentially supervised regressors of $r(\mathbf{P})$, while the actors shape how new candidate gains are sampled. SAC maintains a stochastic policy with an entropy bonus, so in this context it acts as a learned probability distribution that gradually concentrates mass on promising regions while still exploring alternatives. TD3 instead maintains a deterministic actor corrected by noise injection and delayed updates, effectively functioning as a learned point estimator refined by controlled perturbations. Both algorithms can therefore be understood as adaptive proposal mechanisms: SAC as a stochastic sampler and TD3 as a deterministic optimizer with smoothing. From a practical perspective, this places them conceptually close to classical optimizers: SAC behaving like an adaptive random search biased toward low-cost regions, and TD3 like a gradient-driven descent with jitter added for robustness; yet with the distinction that their sampling strategies are learned online from experience rather than hand-crafted. In our setting, these methods are not expected to yield the most efficient optimization compared to specialized algorithms; rather, they are included to examine how general-purpose RL behaves when cast as a static parameter selection problem with noisy rewards, and to assess whether their adaptive sampling provides any advantage in a one-step formulation.

3. Results

This section presents how the optimization campaign was set up. The section first introduces the simulation setup and test case, then reports results under progressively constrained conditions, and finally discusses the computational overhead and the performance of the optimized controllers on unseen missions.

3.1. Test Cases

The optimization framework was evaluated using a simulated quadrotor model based on the DJI Matrice 300 platform, characterized by a mass of 5.2 kg and arm length of 0.32 m. Controller-imposed speed constraints limited maximum horizontal

and vertical velocities to 13.89 m/s and 5.56 m/s, respectively. Each simulation ran for a fixed mission duration of 150 seconds with a sampling frequency of 125 Hz (0.008-second intervals), resulting in approximately 15.0 seconds of wall clock computation time on our workstation with Intel Xeon W-2135 3.70Ghz, 6 core, 12 threads CPU.

All optimization algorithms were tested under varying initial conditions, specifically by changing the PID search bounds (maximum and minimum explorable values), by enabling or disabling wind turbulence simulation, and by using an initial PID set obtained through manual tuning. We present the results in three steps:

1. We start by using wide bounds for the search space of gains value. This allows us to test each method's ability to explore the space without an initial guess.
2. Then, based on manual tuning, we restrict bounds, and we use warm start on manual tuned gains themselves, without turbulence.
3. Third, the same setting as second step, with Dryden turbulence for evaluating robustness and generalization.

In the final discussion, we compare the computation time per iteration and analyze the best controller for an unseen mission in terms of trajectory, attitude, noise, power, and cost decomposition.

The manual tuning followed the Ziegler–Nichols procedure [14]: for each loop under test we set $K_i = 0$ and $K_d = 0$, applied a step in the corresponding reference (hover-to-altitude step for altitude; small position hops for horizontal motion), and increased K_p until a sustained oscillation with nearly constant amplitude was observed. The associated ultimate gain K_u and ultimate period T_u were then mapped to PID gains via the Ziegler–Nichols formulas $K_p = 0.6 K_u$, $T_i = T_u/2$ (so $K_i = K_p/T_i$), and $T_d = T_u/8$ (so $K_d = K_p T_d$). This baseline produced a total cost $J = 218.2$ in the composite mission with no turbulence, and is used as a reference line in the following analyses.

3.2. Optimization performance

(1) No initial guess/large bounds. The algorithms were initially tested with a fixed evaluation budget of 6,000 steps, using wide parameter bounds and no initial guesses. The aim was not to outperform manual tuning, but rather to examine the extent to which each method could explore the search space and identify viable gain regions without any prior information. All PID search bounds were set uniformly in the range $[0, 1 \times 10^5]$. Since this range defines an extremely large search space, we cannot expect better results compared to manual tuning.

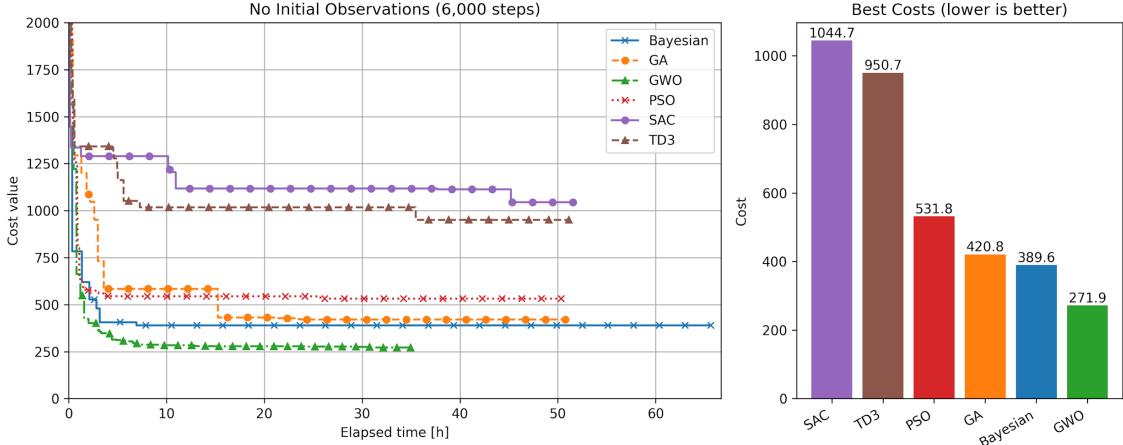


Figure 5: Wide-bound, no-warm-start study (6,000 evaluations). Left: minimum cost versus wall time; Right: best cost achieved at the step budget limit.

Figure 5 illustrates that metaheuristic methods and BO outperform DRL methods, both in terms of convergence speed and final cost reduction.

The left panel reports the temporal evolution of the minimum observed cost, while the right panel presents a bar chart comparing the best final results after 6,000 evaluations. Among the tested algorithms, GWO achieved the best performance, converging to a final cost of 271.9, which remains slightly higher than the baseline obtained from manual Ziegler–Nichols tuning. BO ranked second, followed by GA and PSO.

DRL methods, in contrast, exhibited slower progress and higher variance across runs. In this configuration the algorithms spent a significant portion of the budget maintaining exploration rather than exploiting promising regions, which hindered convergence within the 6,000-step limit. Under such conditions, RL can identify feasible regions but struggles to match the efficiency of specialized metaheuristics or BO.

(2) Small bounds, warm start. After restricting the bounds based on information from manual tuning, a second set of experiments was performed with a fixed wall-time budget of 14 hours. The aim was to assess the extent to which optimization methods could improve the performance of the manually tuned PID configuration. The search space bounds were defined heuristically by inspecting the sensitivity of each parameter to overshoot and oscillatory behavior during the manual tuning. In addition, all algorithms were initialized with an initial observation corresponding to the specific set of PID gains obtained with manual tuning (warm start). For DRL

methods, the first action was forced to evaluate this initial set. Optimizations were conducted first in a disturbance-free environment and then with Dryden turbulence enabled.

Figure 6 shows the results for the bounded search with warm start and no turbulence. All metaheuristic algorithms were able to improve upon the manual baseline within the given time budget, with GWO again achieving the lowest final cost, followed by PSO, GA, and BO. RL, in the one-step-like setup adopted here, did not yield noticeable improvement over the baseline. While the GWO achieved the best performance in this study, reducing the cost by about 23% compared to manual tuning, this does not imply it is universally superior: metaheuristic performance is inherently problem-dependent and varies with mission scenarios or cost formulations [29].

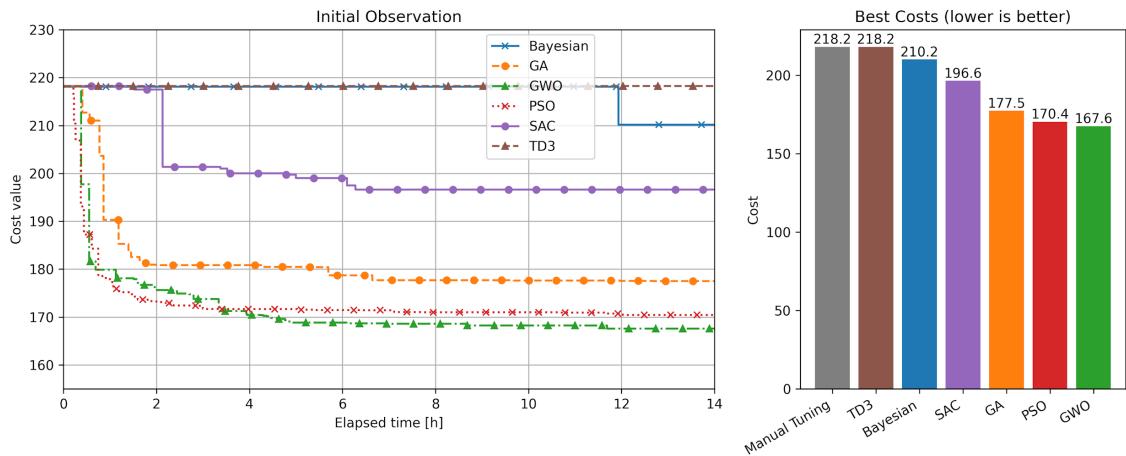


Figure 6: Bounded search with warm start, fixed 14-hour budget, no turbulence. Left: incumbent cost versus wall time; Right: best cost at time budget limit.

(3) Small bounds, warm start with wind turbulence. When Dryden turbulence is enabled, the additional disturbances increase the ability of the optimizers to discriminate between candidate solutions, leading to more robust and generalizable tuning outcomes (Figure 7). GWO again delivered the best performance, reducing the cost by 27% relative to the manual baseline, with PSO and GA achieving similar but slightly weaker results. BO converged more slowly, while DRL methods continued to show no measurable improvement. These findings indicate that, in a one-shot action-reward setting, DRL is poorly suited for static PID gain optimization.

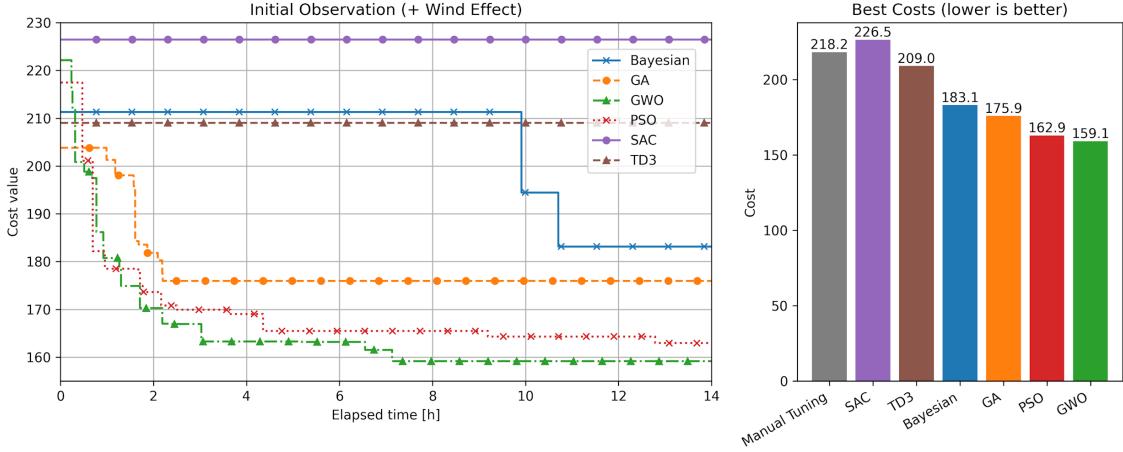


Figure 7: Bounded search with warm start, fixed 14-hour budget, with Dryden turbulence enabled.

Overall, these experiments confirm that optimization algorithms can substantially improve PID tuning compared to manual heuristics. Among the tested approaches, GWO consistently achieved the best performance, while BO proved effective at exploring wide parameter ranges but was less successful in fine-tuning around specific regions. In contrast, DRL under the one-step formulation showed limited effectiveness, suggesting that such methods may be better suited to multi-step, dynamic control tasks, such as learning adaptive or state-dependent PID policies, rather than static parameter selection.

3.3. Discussion

This section provides an interpretation of the outcomes of the optimization campaign. The optimal cost values obtained through the different optimization algorithms are summarized in Table 4. In particular, Test Case 1 refers to a wide parameter search without an initial guess, Test Case 2 corresponds to a bounded search with a warm start in the absence of turbulence, and Test Case 3 denotes a bounded search with a warm start under Dryden turbulence conditions. Then, we quantified the additional computation introduced by each method relative to the baseline simulator, assessed the best configuration for a different mission, and analyzed the improvements compared to manual tuning.

Table 4: Optimal costs obtained with different optimization algorithms across the three test cases. Lower values indicate better performance.

	GWO	PSO	GA	SAC	TD3	BO
Test Case 1	271.9	531.8	420.8	1044.7	950.7	389.6
Test Case 2	167.6	170.4	177.5	196.6	218.2	210.2
Test Case 3	159.1	162.9	175.9	226.5	209.0	183.1

Computational overhead

Table 5 reports the average computation time per optimization step for all tested algorithms during the fixed wall-time experiments. This is to reduce the occurrences of short simulations that are interrupted prematurely due to highly inaccurate PID configuration proposals. Results show that metaheuristic methods exhibit nearly identical runtimes, with GWO, PSO, and GA requiring approximately 17 s per iteration. Reinforcement Learning methods, SAC and TD3, are slightly slower, with TD3 in particular showing a noticeable increase in computational demand. BO resulted in the highest runtime, averaging more than 34 s per iteration, due to the additional cost of surrogate model fitting and acquisition function maximization at each step.

Table 5: Average computation time per iteration for all evaluated optimization algorithms. Algorithm overhead represents the additional computational cost imposed by each optimization method beyond the baseline single-step simulation time of 15.0 seconds.

Algorithm	Average time per iteration [s]	Algorithm Overhead [%]
GWO	17.04	13.6
PSO	17.37	15.8
GA	17.66	17.7
SAC	17.95	19.6
TD3	22.56	50.4
BO	34.39	102.6

Performance analysis

The final stage of the analysis evaluates the performance of the optimized controller on relevant cost components, comparing the PID configuration obtained with GWO under turbulent conditions against the manually tuned baseline. To assess generalization, this evaluation was performed on a mission trajectory different from

the one used during training, thereby testing whether the optimized gains transfer effectively to new flight conditions rather than overfitting to the composite path employed during optimization. The training and test trajectories are illustrated in Figure 8.

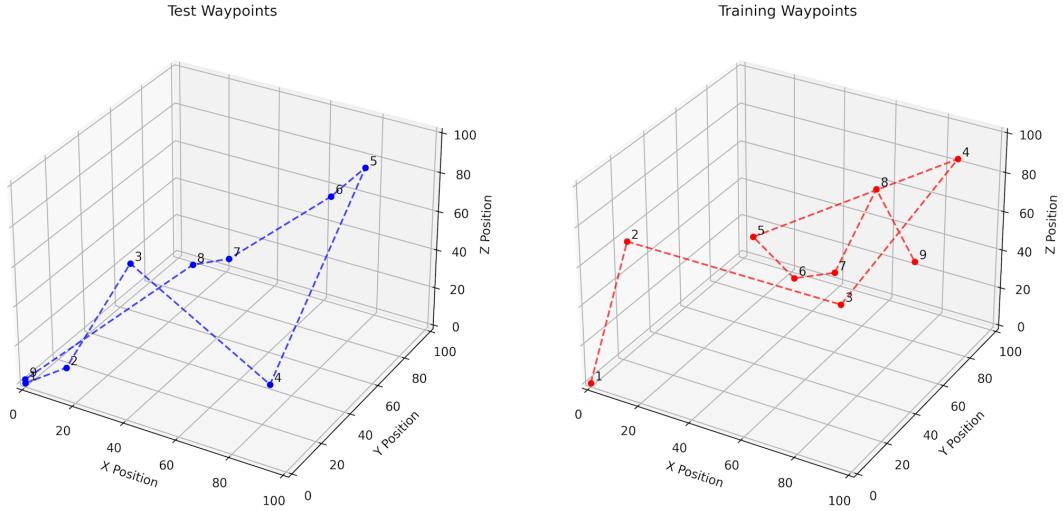


Figure 8: Comparison between the training and testing waypoint trajectories used for evaluating generalization of the optimized controllers.

Before turning to cost decomposition, we first compare the main dynamic signals to illustrate how optimization affects stability, noise, and power. These qualitative trends provide context for the quantitative improvements reported later.

As shown in Figure 9, the optimized controller converges more rapidly to the reference trajectory. In particular, GWO approach completes the mission in 55.79 seconds, compared to 64.62 seconds obtained with the manually tuned controller. For clarity, the figure illustrates a single target-reaching maneuver rather than the entire mission. Beyond this faster convergence, no substantial trajectory improvements are observed, since the Ziegler–Nichols method already yields satisfactory stabilization of the spatial response. The main benefits of optimization instead emerge in the indirect objectives included in the cost function, reduced oscillations, lower acoustic emissions, and improved power efficiency.

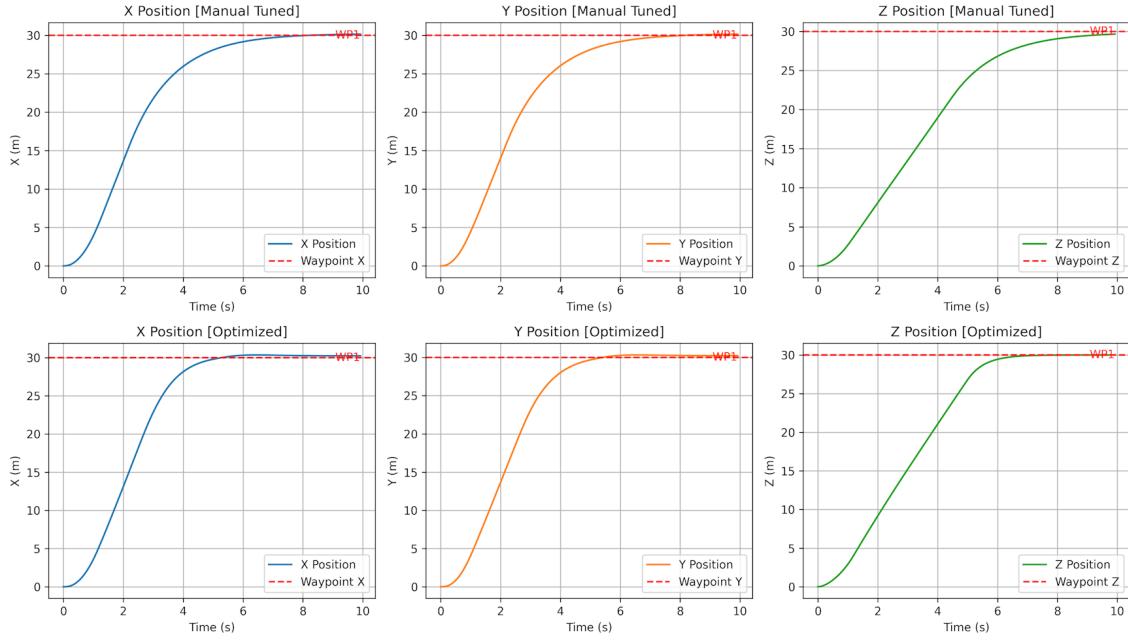


Figure 9: Trajectory tracking during the test mission. Top row: manual baseline (Ziegler–Nichols). Bottom row: GWO-optimized.

The evolution of attitude angles is shown in Figure 10. Compared with the manually tuned baseline, the GWO-optimized controller yields visibly smoother profiles with reduced oscillatory behaviour across pitch, roll, and yaw. This indicates that the optimization substantially improves closed-loop stability, with the controller avoiding unnecessary corrective actions that typically amplify oscillations and contribute to both excess power consumption and tonal noise generation.

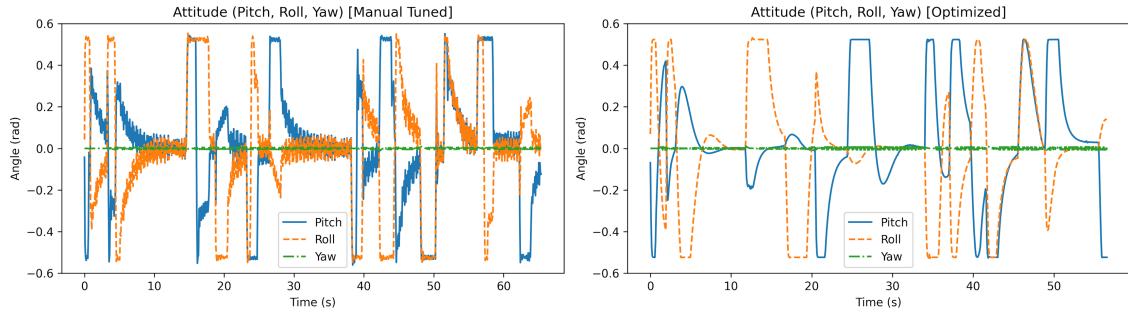


Figure 10: Time histories of pitch, roll, and yaw during the test mission. Left: manual baseline. Right: GWO-optimized.

The acoustic results are presented in Figure 11, which compares broadband averaged SPL at receivers and SWL at the source. Averaged SPL is calculated as the mean for each cell area for each timestep. In both cases, the optimized controller consistently achieves lower acoustic levels, confirming that the improvements in stability and smoother thrust commands translate directly into a reduced acoustic footprint. Importantly, the reductions are not transient but persist throughout the mission, which suggests that the optimization improves the overall noise profile rather than isolated peaks.

Figure 12 reports the electrical power demand over time. The optimized controller exhibits smoother consumption with fewer fluctuations, leading to an average reduction in power usage. This not only reflects greater efficiency in trajectory execution but also indicates reduced stress on the propulsion system, which could translate into improved endurance and component lifetime in real-world operations.

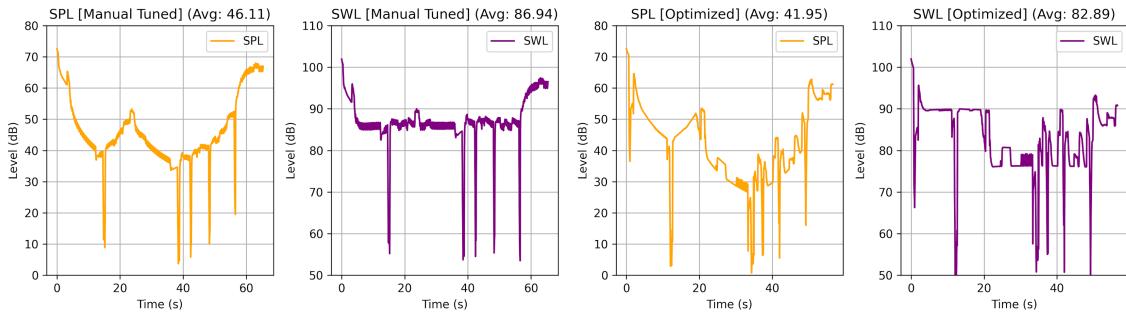


Figure 11: Comparison of SPL (receivers) and SWL (source) time series. First and second images: manual baseline. Third and forth images: GWO-optimized.

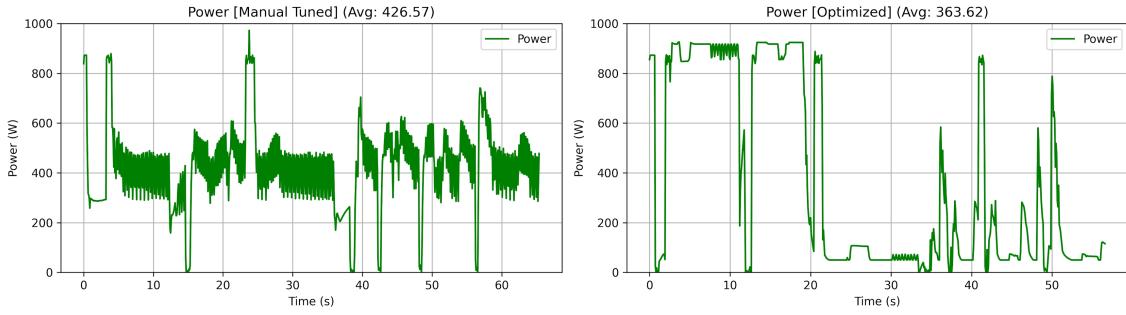


Figure 12: Time evolution of electrical power demand. Left: manual baseline. Right: GWO-optimized.

The quantitative breakdown in Table 6 highlights where the most significant improvements occur. The GWO-optimized configuration reduces the overall cost by 42.7%, with the largest relative improvements obtained in attitude oscillations (−77.8%), noise emissions (−35.9%), and power consumption (−25.7%). Mission time is also improved (−13.6%), while overshoot remains essentially unchanged, indicating that the optimization focuses primarily on stabilizing dynamics and improving efficiency without sacrificing trajectory accuracy. These results underline the limitations of manual tuning: although it can yield acceptable trajectory tracking, it does not provide systematic control over secondary objectives such as energy usage or acoustic footprint, which are critical in urban UAV operations.

Table 6: Cost decomposition on the unseen mission: manual baseline versus GWO-optimized controller (with turbulence). Percentage change is computed as (baseline – optimized)/baseline.

Term	Baseline	Optimized	Change
Total cost	289.59	165.93	−42.7%
Mission time cost	64.62	55.79	−13.6%
Attitude oscillation cost	119.23	26.47	−77.8%
Overshoot cost	35.34	34.46	−2.5%
Power cost	34.52	25.65	−25.7%
Noise proxy cost (SWL-based)	34.03	21.79	−35.9%

Complementary indicators are reported in Table 7. On average, SPL at receivers decreases by 4.16 dB, while SWL at the source is reduced by 4.05 dB. In parallel, average electrical power drops by 14.8%. These reductions confirm that the benefits of optimization are not limited to abstract cost terms but also manifest as tangible improvements in acoustic footprint and energy efficiency. In the psychoacoustics literature, reductions in the order of 3–5 dB are typically regarded as perceptually significant for human listeners, especially in outdoor urban contexts where background levels are variable but relatively low [72, 73]. This indicates that the optimized controllers not only improve technical performance but also achieve noise reductions that are likely to be noticeable to end users and communities.

Table 7: Acoustic and energy indicators on the unseen mission (arithmetic means over time).

Indicator	Baseline	Optimized	Change
Avg SPL at receivers [dB]	46.11	41.95	-4.16 dB
Avg source SWL [dB]	86.94	82.89	-4.05 dB
Avg electrical power [W]	426.57	363.62	-14.8%

4. Conclusions

This study presented a systematic comparison of metaheuristic algorithms, BO, and RL for noise-aware PID tuning of a quadrotor UAV within a unified simulation framework. Among the tested approaches, metaheuristics, particularly GWO, consistently provided the most effective and robust improvements under fixed computational budgets, yielding significant reductions in cost and noise-related terms. BO achieved competitive performance but required substantially higher evaluation times per iteration. By contrast, in the present one-step formulation, SAC and TD3 did not outperform the manually tuned baseline, suggesting that RL in its static form is ill-suited for this problem. Nevertheless, the ability of RL to adapt policies over time makes it a promising direction if multi-step episodes or phase-dependent gain scheduling are introduced. Also, controllers tuned with the proposed pipeline generalized well to unseen missions, indicating that the use of a composite training trajectory and turbulence randomization successfully promoted robust solutions that extend beyond the specific optimization scenario. The test simulation demonstrates that optimization produces controllers that not only generalize to unseen scenarios but also deliver meaningful practical benefits: smoother dynamics, lower noise, and improved efficiency. These improvements directly address key barriers to UAV adoption in urban contexts, where both acoustic acceptability and energy constraints are critical. Despite the promising achieved results, a number of challenges remain, opening directions for future work. First, noise optimization was constrained to simplified proxies based on SWL history and peaks. A more sophisticated pipeline including audio-rate auralization and compliant psychoacoustic models would allow direct integration of perceptual metrics into the optimization loop, potentially improving noise-aware control. Second, only PID gain values were optimized, while anti-windup limits, command saturations, and yaw gains remained fixed; expanding the search space to include these parameters could further improve performance. Third, the aerodynamic surrogate relied on a BEMT-generated dataset; extending the surrogate with higher-fidelity data, or applying data fusion and transfer-learning methods would enhance physical fidelity without compromising efficiency [54]. Fi-

nally, RL approaches could be reformulated to exploit their full potential: instead of treating the problem as a one-step action, where the action provides the complete PID vector, multi-phase formulations could assign gain sets to flight phases (e.g., takeoff, cruise, landing), with state summaries including error norms, wind estimates, and saturation time, and rewards shaped by phase-wise trajectory tracking, control effort, and SPL.

Acknowledgments

This research was funded by the Swiss Federal Office of Civil Aviation (FOCA) under the Spezialfinanzierung Luftverkehr programme, measure SFLV 2022-047, "DRACONIAN: DRone trAjectory and CONtrol optImisAtioN for noise emissions reduction".

Appendix A. BEMT

From momentum theory, for a hovering rotor of radius R , the thrust T_Σ is related to the uniform induced velocity at the disk $v_{\text{ind},0}$ by

$$T_\Sigma = 2\rho Av_{\text{ind},0} (V_\infty + v_{\text{ind},0}), \quad (\text{A.1})$$

where ρ is air density, $A = \pi R^2$ is rotor disk area, and V_∞ is the freestream velocity. This provides a global constraint linking thrust and induced velocity but does not resolve blade-level forces.

From blade element theory, each blade of chord c is discretized into radial segments of length dr , and the elemental thrust and torque are:

$$dT = \frac{1}{2}\rho U_{\text{rel}}^2(r) c C_L(\alpha) dr, \quad (\text{A.2})$$

$$dQ = \frac{1}{2}\rho U_{\text{rel}}^2(r) c C_D(\alpha) r dr, \quad (\text{A.3})$$

where $U_{\text{rel}}(r)$ is the local relative velocity (combination of rotational and inflow components), α is the local angle of attack, and $C_L(\alpha)$, $C_D(\alpha)$ are airfoil lift and drag coefficients.

BEMT couples the two: momentum theory provides $v_{\text{ind}}(r)$, the induced velocity distribution, while blade element theory provides C_L , C_D from airfoil data and calculates the elemental forces. An iterative procedure ensures that the thrust computed from the integrated blade element loads matches the momentum-theory prediction. This model captures both induced and profile power losses and allows for the inclusion of rotor-specific geometric parameters.

Appendix B. Dryden Response model

The Dryden model provides continuous-time shaping filters that, when driven by Gaussian white noise, generate turbulence velocity components (u_{turb} , v_{turb} , w_{turb}) with statistical properties matching the Dryden power spectral densities (PSDs). The one-sided PSDs for longitudinal (u_{turb}), lateral (v_{turb}), and vertical (w_{turb}) turbulence components are:

$$\Phi_u(\Omega_{\text{sp}}) = \sigma_u^2 \frac{2L_u/\pi}{1 + (L_u\Omega_{\text{sp}})^2}, \quad (\text{B.1})$$

$$\Phi_v(\Omega_{\text{sp}}) = \sigma_v^2 \frac{L_v/\pi}{1 + (L_v\Omega_{\text{sp}})^2} \left(1 + \frac{4L_v^2\Omega_{\text{sp}}^2}{1 + (L_v\Omega_{\text{sp}})^2} \right), \quad (\text{B.2})$$

$$\Phi_w(\Omega_{\text{sp}}) = \sigma_w^2 \frac{L_w/\pi}{1 + (L_w\Omega_{\text{sp}})^2} \left(1 + \frac{3L_w^2\Omega_{\text{sp}}^2}{1 + (L_w\Omega_{\text{sp}})^2} \right), \quad (\text{B.3})$$

where $\sigma_{\{\cdot\}}$ are turbulence standard deviations, $L_{\{\cdot\}}$ are turbulence scale lengths, and Ω_{sp} is the spatial frequency in the aircraft frame.

From these PSDs, transfer functions generate wind velocity fluctuations from white-noise inputs. The turbulence components are then expressed in the inertial frame and transformed into the rotor disk frame.

Appendix C. Runge-Kutta Scheme

Given $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ (collecting translational and rotational dynamics) and step h :

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{x}_n), \quad (\text{C.1})$$

$$\mathbf{k}_2 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_1\right), \quad (\text{C.2})$$

$$\mathbf{k}_3 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_2\right), \quad (\text{C.3})$$

$$\mathbf{k}_4 = \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k}_3), \quad (\text{C.4})$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \quad (\text{C.5})$$

Appendix D. Propagation model

Atmospheric absorption is modeled according to ISO 9613-1:1993 [68], which expresses the attenuation per unit distance as a frequency-dependent coefficient $\alpha(f)$ in dB/m. The per-band attenuation over a propagation distance d is given by:

$$A_{\text{atm}}(f, d) = \alpha(f) \cdot d \quad (\text{D.1})$$

The absorption coefficient $\alpha(f)$ is computed as:

$$\begin{aligned}\alpha(f) = & 8.686 f^2 \left[1.84 \times 10^{-11} \frac{1}{p_r} \left(\frac{T}{T_0} \right)^{1/2} \right. \\ & \left. + \left(\frac{T}{T_0} \right)^{-5/2} \left(\frac{0.01275 e^{-2239.1/T}}{f_{rO} + \frac{f^2}{f_{rO}}} + \frac{0.1068 e^{-3352/T}}{f_{rN} + \frac{f^2}{f_{rN}}} \right) \right]\end{aligned}\quad (\text{D.2})$$

Here, f is the center frequency of the 1/3-octave band measured in [Hz], p_r is the relative atmospheric pressure with respect to 101.325 [kPa], T is the absolute temperature in [K] with $T_0 = 293.15$ K, and f_{rO} and f_{rN} are the oxygen and nitrogen relaxation frequencies.

$$f_{rO} = p_r \left[24.0 + \frac{4.04 \times 10^4 h (0.02 + h)}{0.391 + h} \right] \quad (\text{D.3})$$

$$f_{rN} = p_r \left(\frac{T}{T_0} \right)^{-1/2} \left[9.0 + 280.0 h e^{-4.170((T/T_0)^{-1/3}-1)} \right] \quad (\text{D.4})$$

The molar humidity ratio h is given by:

$$h = \frac{H_r P_{\text{sat}}(T)}{p_{\text{atm}}} \quad (\text{D.5})$$

where H_r is the relative humidity (0–1), p_{atm} is the ambient pressure in kPa, and $P_{\text{sat}}(T)$ is the saturation vapor pressure at temperature T .

When source directivity is not already embedded in the reference sound power data, a Directivity Index (DI) correction term can be applied per band. This correction accounts for directional radiation characteristics as a function of observation angle θ and frequency f .

The DI is defined as:

$$DI(f, \theta) = 10 \log_{10} \left[\frac{I(f, \theta)}{\bar{I}(f)} \right] \quad (\text{D.6})$$

where $I(f, \theta)$ is the radiated acoustic intensity in the direction θ , and $\bar{I}(f)$ is the average intensity over the full sphere:

$$\bar{I}(f) = \frac{1}{4\pi} \int_0^{2\pi} \int_0^\pi I(f, \theta) \sin \theta d\theta d\phi \quad (\text{D.7})$$

The directivity term is added to the per-band propagation equation for the sound pressure level at the receiver:

$$L_p(f) = L_w(f) - A_{\text{sp}}(d) - A_{\text{atm}}(f, d) + DI(f, \theta) \quad (\text{D.8})$$

where $A_{\text{sp}}(d)$ is the free-field spherical spreading attenuation, $A_{\text{atm}}(f, d)$ is the atmospheric absorption, and $DI(f, \theta)$ is expressed in decibels.

Appendix E. Simulation parameters

The simulation setup is defined by a set of parameters that specify the temporal resolution, mission duration, vehicle dynamics, and environmental constraints. Table E.8 summarizes all the values adopted in this study.

The simulation time step is set to $dt = 0.008$ s, with a total mission duration of 150 s. Thresholds for trajectory switching and simulation termination are set to 2 m. A circular region of 14 m radius is considered for evaluating noise annoyance.

The quadrotor model assumes four rotors, each with a maximum speed of 3000 RPM. Maximum roll, pitch, and yaw angles are constrained to 30° . Velocity limits are set to 50 km/h horizontally and 20 km/h vertically. Yaw target is fixed to 0. The vehicle mass is 5.2 kg, with inertia and aerodynamic coefficients listed in Table E.8.

Noise predictions rely on pre-trained surrogate models, with the rotor speed normalized to a reference of 2500 RPM.

Finally, several optimization algorithms are employed for controller tuning. GA, PSO, and GWO are configured with common settings such as population size, number of iterations, and rates for crossover, mutation, and elitism. Their specific configurations are also summarized in Table E.8.

Table E.8: Simulation parameters used in the experiments.

Parameter	Value
Time step dt	0.008 s
Simulation time	150 s
Termination threshold	2 m
Target shift threshold	2 m
Noise annoyance radius	14 m
Max RPM	3000
Number of rotors	4
Number of blade per rotor	2
Rotor diameter	0.6m
Rotor radius hub	0.02m
BEMT number of sections	8
Max roll/pitch/yaw angle	30°
Max horizontal speed	50 km/h
Max vertical speed	20 km/h
Mass	5.2 kg
Inertia	[3.8e-3, 3.8e-3, 7.1e-3] kg m ²
Arm length l	0.32 m
Rotor drag coefficient d	7.5×10^{-7}
Traslational drag coefficients. C_d	[0.1, 0.1, 0.15]
Aerodynamic friction coefficients. C_a	[0.1, 0.1, 0.15]
Rotor inertia J_r	6×10^{-5} kg m ²
Reference RPM (noise)	2500
GA settings	30 pop., $c_r = 0.8$, $m_r = 0.1$, elitism 0.1
PSO settings	30 swarm, $w = 0.7$, $c_1 = 1.5$, $c_2 = 1.5$
GWO settings	30 wolves

References

- [1] D. Floreano, R. J. Wood, Science, technology and the future of small autonomous drones, *nature* 521 (2015) 460–466.
- [2] A. Goodchild, J. Toy, Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry, *Transportation Research Part D: Transport and Environment* 61 (2018) 58–67.

- [3] S. Watkins, J. Burry, A. Mohamed, M. Marino, S. Prudden, A. Fisher, N. Kloet, T. Jakobi, R. Clothier, Ten questions concerning the use of drones in urban environments, *Building and Environment* 167 (2020) 106458.
- [4] D. Raya Islam, A. Stimpson, M. Cummings, Small UAV Noise Analysis, Technical Report, Technical Report, 2017.
- [5] M. Maaruf, M. S. Mahmoud, A. Ma'arif, A survey of control methods for quadrotor uav, *International Journal of Robotics and Control Systems* 2 (2022) 652–665.
- [6] B. Uragun, I. N. Tansel, The noise reduction techniques for unmanned air vehicles, in: 2014 international conference on unmanned aircraft systems (ICUAS), IEEE, 2014, pp. 800–807.
- [7] O. Gur, A. Rosen, Design of quiet propeller for an electric mini unmanned air vehicle, *Journal of propulsion and power* 25 (2009) 717–728.
- [8] V. H. Trinh, D. Vu, Designs of sound absorbers for noise reduction of unmanned aerial vehicles, in: Innovations and Developments in Unmanned Aerial Vehicles, IGI Global Scientific Publishing, 2025, pp. 245–278.
- [9] M. V. Mane, P. D. Sonawwanay, M. Solanki, V. Patel, A comprehensive review on advancements in noise reduction for unmanned aerial vehicles (uavs), *Journal of Vibration Engineering & Technologies* 12 (2024) 1375–1397.
- [10] S. Abdelhay, A. Zakriti, Modeling of a quadcopter trajectory tracking system using pid controller, *Procedia Manufacturing* 32 (2019) 564–571.
- [11] N. H. Sahrir, M. A. Mohd Basri, Modelling and manual tuning pid control of quadcopter, in: Control, instrumentation and mechatronics: Theory and practice, Springer, 2022, pp. 346–357.
- [12] I. Lopez-Sanchez, J. Moreno-Valenzuela, Pid control of quadrotor uavs: A survey, *Annual Reviews in Control* 56 (2023) 100900.
- [13] T. George, V. Ganesan, Optimal tuning of pid controller in time delay system: A review on various optimization techniques, *Chemical Product and Process Modeling* 17 (2022) 1–28.

- [14] P. Meshram, R. G. Kanajiya, Tuning of pid controller using ziegler-nichols method for speed control of dc motor, in: IEEE-international conference on advances in engineering, science and management (ICAESM-2012), IEEE, 2012, pp. 117–122.
- [15] M. W. Foley, R. H. Julien, B. R. Copeland, A comparison of pid controller tuning methods, *The Canadian Journal of Chemical Engineering* 83 (2005) 712–722.
- [16] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, B. M. Khammas, Meta-heuristic algorithms for pid controller parameters tuning: Review, approaches and open problems, *Heliyon* 8 (2022).
- [17] A. Abushawish, M. Hamadeh, A. Nassif, Pid controller gains tuning using meta-heuristic optimization methods: A survey, *International Journal of Computers* 14 (2020) 87–95.
- [18] J. Kennedy, R. Eberhart, Particle swarm optimization, *Proceedings of ICNN'95 - International Conference on Neural Networks* 4 (2002) 1942–1948 vol.4. URL: <https://api.semanticscholar.org/CorpusID:3114196>.
- [19] T.-H. Kim, I. Maruta, T. Sugie, Robust pid controller tuning based on the constrained particle swarm optimization, *Automatica* 44 (2008) 1104–1110.
- [20] M. I. Solihin, L. F. Tack, M. L. Kean, Tuning of pid controller using particle swarm optimization (pso), *International Journal on Advanced Science, Engineering and Information Technology* 1 (2011) 458–461.
- [21] D. E. Goldberg, Genetic algorithms in search optimization and machine learning, 1988. URL: <https://api.semanticscholar.org/CorpusID:38613589>.
- [22] A. Y. Jaen-Cuellar, R. de J. Romero-Troncoso, L. Morales-Velazquez, R. A. Osornio-Rios, Pid-controller tuning optimization with genetic algorithms in servo systems, *International Journal of Advanced Robotic Systems* 10 (2013) 324.
- [23] D. Meena, A. Devanshu, Genetic algorithm tuned pid controller for process control, in: 2017 International Conference on Inventive Systems and Control (ICISC), IEEE, 2017, pp. 1–6.

- [24] E. W. Suseno, A. Ma’arif, Tuning of pid controller parameters with genetic algorithm method on dc motor, International Journal of Robotics and Control Systems 1 (2021) 41–53.
- [25] S. M. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61. URL: <https://api.semanticscholar.org/CorpusID:15532140>.
- [26] M. H. Hashem, H. S. Abdullah, K. I. Ghathwan, Grey wolf optimization algorithm: A survey, Iraqi Journal of Science (2023). URL: <https://api.semanticscholar.org/CorpusID:265650990>.
- [27] A. Madadi, M. M. Motlagh, Optimal control of dc motor using grey wolf optimizer algorithm, Tech. J. Eng. Appl. Sci 4 (2014) 373–379.
- [28] Q. Li, H. Chen, H. Huang, X. Zhao, Z. Cai, C. Tong, W. Liu, X. Tian, An enhanced grey wolf optimization based feature selection wrapped kernel extreme learning machine for medical diagnosis, Computational and mathematical methods in medicine 2017 (2017) 9512741.
- [29] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (2002) 67–82.
- [30] P. I. Frazier, Bayesian optimization, in: Recent advances in optimization and modeling of contemporary problems, Informs, 2018, pp. 255–278.
- [31] M. Diessner, J. O’Connor, A. Wynn, S. Laizet, Y. Guan, K. Wilson, R. D. Whalley, Investigating bayesian optimization for expensive-to-evaluate black box functions: Application in fluid dynamics, Frontiers in Applied Mathematics and Statistics 8 (2022) 1076296.
- [32] M. L. Santoni, E. Raponi, R. D. Leone, C. Doerr, Comparison of high-dimensional bayesian optimization algorithms on bbob, ACM Transactions on Evolutionary Learning 4 (2024) 1–33.
- [33] M. Malu, G. Dasarathy, A. Spanias, Bayesian optimization in high-dimensional spaces: A brief survey, in: 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), IEEE, 2021, pp. 1–8.
- [34] J. P. Coutinho, L. O. Santos, M. S. Reis, Bayesian optimization for automatic tuning of digital multi-loop pid controllers, Computers & Chemical Engineering 173 (2023) 108211.

- [35] B. Recht, A tour of reinforcement learning: The view from continuous control, *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019) 253–279.
- [36] K. Li, T. Zhang, R. Wang, Deep reinforcement learning for multiobjective optimization, *IEEE transactions on cybernetics* 51 (2020) 3103–3114.
- [37] F. Zou, G. G. Yen, L. Tang, C. Wang, A reinforcement learning approach for dynamic multi-objective optimization, *Information Sciences* 546 (2021) 815–834.
- [38] I. Carlucho, M. De Paula, A. H. R. Costa, et al., Incremental q-learning strategy for adaptive pid control of mobile robots, *Expert Systems with Applications* 80 (2017) 183–199. doi:10.1016/j.eswa.2017.03.020.
- [39] Q. Shi, et al., Adaptive pid controller based on q-learning algorithm, *IET Cyber-Systems and Robotics* 1 (2018) 13–22. doi:10.1049/trit.2018.1007.
- [40] Y. Ding, et al., Multi-phase focused pid adaptive tuning with deep deterministic policy gradient, *Electronics* 12 (2023) 3925. doi:10.3390/electronics12183925.
- [41] M. A. Chowdhury, Q. Lu, Entropy-maximizing td3 based reinforcement learning for automatic pid tuning, *Computers & Electrical Engineering* 110 (2023) 108843. doi:10.1016/j.compeleceng.2023.108843.
- [42] Y.-h. CHENG, S. Wei, et al., A proposal of adaptive pid controller based on reinforcement learning, *Journal of China University of Mining and Technology* 17 (2007) 40–44.
- [43] Z. Guan, T. Yamamoto, Design of a reinforcement learning pid controller, *IEEJ transactions on electrical and electronic engineering* 16 (2021) 1354–1360.
- [44] T. Shuprajhaa, S. K. Sujit, K. Srinivasan, Reinforcement learning based adaptive pid controller design for control of linear/nonlinear unstable processes, *Applied Soft Computing* 128 (2022) 109450.
- [45] G. Bujgoi, D. Sendrescu, Tuning of pid controllers using reinforcement learning for nonlinear systems control (2024).
- [46] R. Yondo, K. Bobrowski, E. Andrés, E. Valero, A review of surrogate modeling techniques for aerodynamic analysis and optimization: current limitations and future challenges in industry, *Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences* (2018) 19–33.

- [47] G. Sun, S. Wang, A review of the artificial neural network surrogate modeling in aerodynamic design, *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233 (2019) 5863–5872.
- [48] G. Immordino, A. Vaiuso, A. Da Ronch, M. Righi, Predicting transonic flow-fields in non-homogeneous unstructured grids using autoencoder graph convolutional networks, *Journal of Computational Physics* 524 (2025) 113708.
- [49] B. Davoudi, K. Duraisamy, A hybrid blade element momentum model for flight simulation of rotary wing unmanned aerial vehicles, in: *AIAA Aviation 2019 Forum*, 2019, p. 2823.
- [50] B. Pacini, A. Yildirim, B. Davoudi, J. R. Martins, K. Duraisamy, Towards efficient aerodynamic and aeroacoustic optimization for urban air mobility vehicle design, in: *AIAA AVIATION 2021 FORUM*, 2021, p. 3026.
- [51] U. Boatto, P. Bonnet, F. Avallone, D. Ragni, Assessment of a bemt-based rotor aerodynamic model under uniform aligned steady inflow, in: *AIAA SCITECH 2023 Forum*, 2023, p. 0609.
- [52] D. Massegur, A. Da Ronch, Graph convolutional multi-mesh autoencoder for steady transonic aircraft aerodynamics, *Machine Learning: Science and Technology* 5 (2024) 025006.
- [53] G. Immordino, A. Vaiuso, A. Da Ronch, M. Righi, Spatio-temporal graph convolutional autoencoder for transonic wing pressure distribution forecasting, *Aerospace Science and Technology* (2025) 110516.
- [54] A. Vaiuso, G. Immordino, M. Righi, A. Da Ronch, Multi-fidelity transonic aerodynamic loads estimation using bayesian neural networks with transfer learning, *Aerospace Science and Technology* (2025) 110301.
- [55] A. Jameson, W. Schmidt, E. Turkel, Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes, 1981. URL: <https://api.semanticscholar.org/CorpusID:6948802>.
- [56] A. ando, N. Cangiotti, M. Sensi, Exploring exponential runge-kutta methods: A survey, ArXiv abs/2507.04024 (2025). URL: <https://api.semanticscholar.org/CorpusID:280049506>.
- [57] H. L. Dryden, A review of the statistical theory of turbulence, *Quarterly of Applied Mathematics* 1 (1943) 7–42.

- [58] L. Meier, D. Honegger, M. Pollefeyt, Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms, in: 2015 IEEE international conference on robotics and automation (ICRA), IEEE, 2015, pp. 6235–6240.
- [59] S. Galeani, S. Tarbouriech, M. C. Turner, L. Zaccarian, A tutorial on modern anti-windup design, 2009 European Control Conference (ECC) (2009) 306–323. URL: <https://api.semanticscholar.org/CorpusID:783713>.
- [60] I. E. Commission, et al., Iec 61260-1: 2014, electroacoustics—octave-band and fractional-octave-band filters—part 1: Specifications, IEC: London, UK (2014) 88.
- [61] C. Ramos-Romero, N. Green, S. Roberts, C. Clark, A. J. Torija, Requirements for drone operations to minimise community noise impact, International Journal of environmental research and public health 19 (2022) 9299.
- [62] J. M. Wunderli, J. Meister, O. Boolakee, K. Heutschi, A method to measure and model acoustic emissions of multicopters, International Journal of Environmental Research and Public Health 20 (2022) 96.
- [63] C. Thurman, D. D. Boyd, B. M. Simmons, Comparison of prediction modeling methodologies for aeroacoustic characterization of hovering suas rotors, in: AIAA SciTech 2023 Forum, 2023, p. 0027.
- [64] C. Poggi, M. Rossetti, J. Serafini, G. Bernardini, M. Gennaretti, U. Iemma, Neural network meta-modelling for an efficient prediction of propeller array acoustic signature, Aerospace Science and Technology 130 (2022) 107910.
- [65] M. Righi, M. Apicella, O. Coretti, A. Vaiuso, Neaptide dataset, 2024. URL: <https://doi.org/10.5281/zenodo.10512044>. doi:10.5281/zenodo.10512044.
- [66] International Standard ISO 5305, Noise measurements for uas (unmanned aircraft systems), 2024.
- [67] A. Vaiuso, M. Righi, O. Coretti, M. Apicella, Drone acoustic analysis for predicting psychoacoustic annoyance via artificial neural networks, arXiv preprint arXiv:2410.22208 (2024).
- [68] International Standard ISO 9613, Acoustics—attenuation of sound during propagation outdoors, 1993.

- [69] D. Silver, S. Singh, D. Precup, R. S. Sutton, Reward is enough, *Artificial intelligence* 299 (2021) 103535.
- [70] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, ArXiv abs/1801.01290 (2018). URL: <https://api.semanticscholar.org/CorpusID:28202810>.
- [71] S. Dankwa, W. Zheng, Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent, *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing* (2019). URL: <https://api.semanticscholar.org/CorpusID:218871553>.
- [72] H. Fastl, Psychoacoustics, sound quality and music, 2007. URL: <https://api.semanticscholar.org/CorpusID:69722637>.
- [73] E. Zwicker, H. Fastl, *Psychoacoustics: Facts and models*, volume 22, Springer Science & Business Media, 2013.