POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

# Drone Modeling and Data-Driven PX4 Controller Tuning via SIL/HIL

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

**Author:** Francesco Cataffo

**Advisor:** Prof. Marco Lovera

**Co-advisor:** Marta Manzoni

**Academic year:** 2024-2025

## 1. Introduction

Over the past decades, the development of Unmanned Aerial Vehicle (UAV) platforms has progressed from a primarily research field to a commercially promising technology. This advancement is enabled by intrinsic advantages such as operational flexibility, access to remote or hazardous locations, and cost-effective data acquisition. Among UAV configurations, quadrotor platforms are widely adopted for infrastructure inspection, precision agriculture, traffic monitoring, and surveillance. Their growing deployment demands high-performance, high-fidelity solutions while reducing overall development time. Scalable and accessible technologies combined with data-driven autopilot optimization can improve performance in complex operational scenarios and shorten development cycles; however, their effectiveness relies on systematic, high-fidelity validation procedures that bridge numerical simulation and real-world flight.

In this context, the present work proposes an integrated methodology that couples Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) testing philosophies with Bayesian Optimization (BO), applied directly to the PX4 open-source autopilot [3].

## 2. Modeling

To this purpose, the first step is to develop a high-fidelity, scalable model of the quadrotor dynamics, including external disturbances. Since the simulator must communicate with the autopilot at a fixed frequency, the computational cost represents a key metric in selecting the most suitable models.

Four quadrotor configurations are implemented, based on real drone data, selected to represent a wide spectrum of existing vehicles in terms of mass and operational capabilities: the DJI Matrice 400, Matrice 350 RTK, Matrice 30 Enterprise, and Mavic 3 Enterprise.

The environment model simulates three effects: ground interaction, aerodynamic drag, and wind disturbance. The first is handled through a linear and rotational mass-spring-damper model together with a smooth Coulomb friction model. The simplified aerodynamic drag of the fuselage is computed assuming a parallelepiped geometry, with drag forces proportional to the squared velocity components along each body axis and the corresponding cross area. The wind model is developed according to the *MIL-F-8785C* military standard, accounting for both turbulence and wind shear effects. Turbulence is modeled as a stochastic process using the Dryden spectral representation, which provides a good balance

between accuracy and computational efficiency. The wind vector shear model, which describes the variation of wind magnitude and direction with altitude, is adapted from its original version, linking the wind intensity to the first six intervals of the Beaufort scale to better suit the drone operational environment and increase the scalability and versatility of the model.

## 3. Software Description and Architecture

PX4 is an open-source, multi-purpose autopilot governed by the Dronecode Foundation and widely adopted across industrial and research applications. A key feature of PX4 is its native support for SIL and HIL simulation, where the drone dynamics are computed externally, and sensor data are exchanged with the autopilot through standardized MAVLink messages. This makes PX4 an accessible, flexible, and well-documented platform for model development and controller testing.
While PX4-supported simulators such as Gazebo or JMAVSim offer straightforward communication interfaces with the autopilot, they lack the flexibility needed to implement custom high-fidelity models. Therefore, despite the more complex interface with PX4, Simulink is selected for its ability to handle complex system modeling.
QGroundControl (QGC) is the open-source Ground Control Station (GCS) software used for mission planning, airframe configuration, and firmware flashing, thanks to its compatibility with the MAVLink protocol.

### SIL setup

The SIL simulation environment integrates three modules: the Simulink simulator, the PX4 autopilot, and the GCS. The simulator computes the drone dynamics, generates the simulated sensor data, and sends them to the autopilot via MAVLink messages. In this architecture, PX4 runs on a Windows Subsystem for Linux 2 (WSL 2) instance, receives the simulated measurements, estimates the vehicle state, and returns the motor Pulse Width Modulation (PWM) commands to the simulator. QGC provides the position and attitude setpoints. All inter-module communication is handled via the MAVLink protocol. The Simulink-PX4 link is established over TCP port 4560, while QGC communicates with the WSL 2 instance over UDP port 18570. On the Simulink side, the interface is implemented through an S-function, which converts the drone state into sensor measurements, packages them into standardized MAVLink messages, and routes them to PX4. The PWM signal from PX4 is received through the same interface and fed into the motor model.

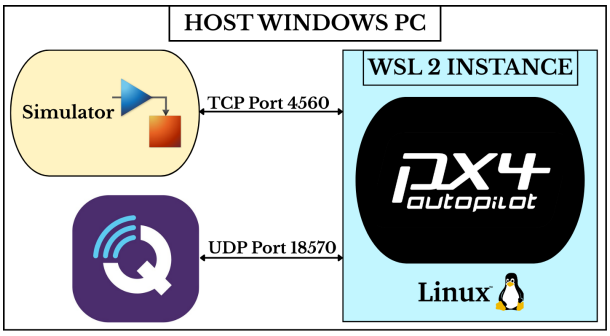


Figure 1: SIL architecture.

### HIL setup

The HIL simulation environment shares the same logic as SIL, with a main difference: the PX4 autopilot no longer runs on the host computer, but is embedded in a real Flight Controller (FC) board, the Pixhawk 4 mini. This allows the actual PX4 firmware, executed on the real FC board, to be tested in closed-loop with the simulator, increasing the results fidelity.
The Simulink simulator computes the drone dynamics and generates fictitious sensor measurements. These are packaged into MAVLink messages by a dedicated S-function and transmitted to the Pixhawk via serial connection. The FC processes the sensor data, runs its estimation and control algorithms, and returns the motor PWM commands to the simulator through the same interface. A second key difference with respect to the SIL architecture concerns the GCS. In this case, a Python GCS, exploiting the MAVSDK library, replaces QGC, offering greater flexibility in mission generation and connection handling. Since the serial port is in use by the S-function, the GCS cannot reach the Pixhawk directly. Therefore, the S-function acts as a bridge, forwarding MAVLink messages between the GCS and PX4 over UDP port 14550.

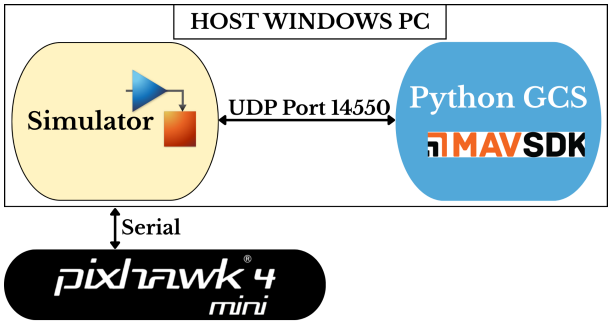


Figure 2: HIL architecture.

## 4. Illustrative missions

Four representative missions, depicted in Figure 3, are implemented to assess nominal PX4 autopilot performance. Each mission is evaluated in SIL and HIL for all six Beaufort wind intensity levels.

**Healthcare Delivery (DJI Matrice 400):** In SIL the vehicle completes all missions, with position Root Mean Square Error (RMSE) below 0.4 m for the first five disturbance levels. In HIL, the vehicle fails even at the lowest disturbance: during acceleration toward the first waypoint, the PWM signals saturate and the vehicle crashes. Central Processing Unit (CPU) and Rapid Access Memory (RAM) usage remain below 50%, indicating actuator saturation as the root cause.

**Urban Delivery (DJI Matrice 350 RTK):** In SIL the mission is completed up to Beaufort 5; at Beaufort 6 the vehicle cannot counteract wind disturbance. For successful cases, the position RMSE ranges approximately 0.76–1.24 m. In HIL, the vehicle crashes at all wind levels: control inputs approach saturation shortly after takeoff and actuators saturate as wind intensity increases with altitude, while board CPU and RAM loads remain within acceptable bounds.

**Area Surveillance (DJI Matrice 30 Enterprise):** In SIL the vehicle completes the first five wind levels; the sixth level fails due to the wind disturbance. In HIL, the platform performs better than the previous two, flying for ≈3.5 min before crashing. The failure occurs during a demanding survey phase requiring high speed, wind compensation, yaw rotation, and a slant attitude for imaging; the combined demands lead to PWM saturation and vehicle crash.

**Bridge Inspection (DJI Mavic 3 Enterprise):** As the lightest platform, this vehicle is most sensitive to wind. In SIL the mission is completed only for the first four disturbance levels, with position RMSE below 0.6 m. In HIL the vehicle completes the nominal mission up to Beaufort 4, but performance degrades compared to SIL: position RMSE rises from 0.1 m at Beaufort 1 to 1.8 m at Beaufort 4.

Across all missions, SIL results indicate that the nominal PX4 autopilot can fly the platforms correctly, but that improved tuning may yield better performance. HIL tests highlight additional criticalities as most vehicles experience actuator saturation, leading to crashes.

## 5. Controller Optimization

**PX4 Controller Structure:** The PX4 controller has a cascaded architecture, reported in Figure 4, composed of four layers per each axis. The outermost layer is a position P controller and a velocity PID controller. Then, a block that converts the resulting acceleration setpoint into the new thrust required follows. This signal enters the inner loop, composed of a P attitude controller and a PID rate controller, whose output is routed to the mixer to generate the PWM commands for the actuators. Assuming that the $x$ and $y$ gains are equal due to vehicle symmetry, the full controller has 20 tunable parameters, reducing to 8 when focusing on the outer position and velocity loops.

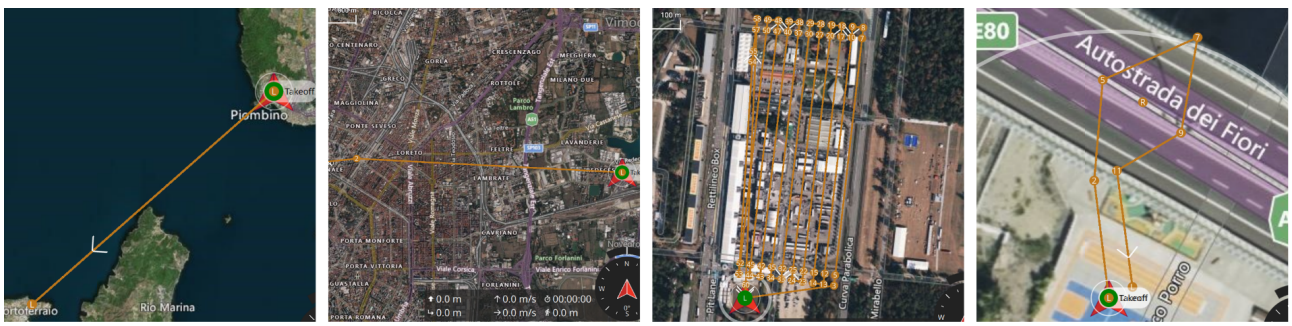


Figure 3: Path preview for the four illustrative missions.


Figure 4: PX4 Controller Schematic.

**Controller Optimization Problem:** The controller optimization problem is:

$$\text{find} \quad \mathbf{x}^* \quad \text{such that} \quad \mathbf{x}^* = \arg\max_{\mathbf{x}\in\Omega} f(\mathbf{x}) \ ,$$

where the Performance Function (PF) $f(\mathbf{x})$ summarizes system closed-loop performance, $\Omega$ is the search space of admissible controller gains, and $\mathbf{x}^*$ is the set of gains corresponding to the best performing controller. Several techniques exist to address this problem, ranging from classical model-based methods to data-driven approaches such as Genetic algorithms, Reinforcement Learning, and Iterative Feedback Tuning. However, given the high dimensionality of the parameter space and the non-negligible time required for running a single simulation, sampling efficiency becomes the primary selection criterion, making BO the most suitable approach.

## Bayesian Optimization

BO is a sequential, probabilistic method, suitable for global optimization of expensive black-box functions. This method seeks to maximize the PF using the least number possible of function evaluations (i.e., experiments). Its key idea is to build and iteratively refine a Probabilistic Surrogate Model (PSM) of the PF $f$, and to use the current estimate of $f$ to select the next query point [1].

The BO method has three main ingredients. The first is the PSM, which aims to estimate $f$ over the search space. In this work, Gaussian Process (GP) regression is used because its features make it well-suited for continuous search spaces and it allows closed-form posterior update. A GP is a stochastic process, specified by its mean and covariance function (also known as kernel), that defines a prior. Conditioning this prior on noisy observations of the PF yields the GP posterior, which provides posterior mean and covariance, allowing to estimate the PF with explicit uncertainty quantification [4]. The second is the dataset $\mathbf{D}$, which collects all noisy observations of the objective function $y_i = f(\mathbf{x}_i) + \varepsilon$ retrieved by experiments, where $\varepsilon$ is a zero-mean Gaussian noise and the index $i$ stands for a generic iteration of the BO loop. The dataset is exploited to update the kernel hyperparameters to minimize the prediction error of the PSM at the observed points via Maximum-a-Posteriori (MAP) or Maximum Likelihood Estimation (MLE) methods during the BO loop. The last ingredient is the Acquisition Function (AF) $\alpha(\mathbf{x})$. This tool determines where to sample next, balancing exploration, sampling regions of high uncertainty, and exploitation, focusing on regions where high performance is expected. At each iteration, the next query point is chosen as

$$\mathbf{x}_{next} = \arg\max_{\mathbf{x}\in\Omega}(\alpha(\mathbf{x})) \ ,$$

a new experiment is performed at that point, the dataset is updated with the new observation $y_{next}$, and the PSM is refined. The loop continues until convergence or a maximum number of iterations is reached, returning the best gain configuration found.

**High-dimensional BO:** Standard BO suffers from the curse of dimensionality: as the number of parameters increases, AF maximization becomes prohibitively expensive and is typically impractical beyond three to four dimensions. To overcome this limitation, tuning is limited to the PX4 outer loop (position and velocity), reducing the search space to 8 dimensions, and the LineBO algorithm is adopted to handle high-dimensional BO [2].

The key idea of LineBO is to restrict the maximization of the AF problem to a one-dimensional affine subspace of the original search space $\mathcal{L}(q, m) \in \Omega$, defined at each iteration as a the line passing through the best point found so far $q$ and whose direction $m$ is provided by a direction oracle $\Pi$. This choice

reduces by far the computational burden associated with the solution of the AF maximization problem. The direction oracle is a tool that provides the direction for the next sub-space according to some logic. For example, the CoordinateLineBO direction oracle explores the search space by changing the gains of the controller singularly in a cyclical manner.
The basic LineBO algorithm is reported in Algorithm 1, where $\hat{f}$ is the best estimate of the PF at the end of the loop.

Algorithm 1 LineBO algorithm

**1) Algorithm Initialization**
Select GP prior, initial point $\mathbf{x}_0$, $\Pi$, and $\Omega$
Initialize the dataset $\mathbf{D}_0 = \varnothing$
**2) Initial Design**
Select $\mathbf{x}_{1:n_0}$ random locations, $\mathbf{x}_i \in \Omega$
**for** $i = 1 : n_0$ **do**
  Experiment: $y_i = f(\mathbf{x}_i) + \varepsilon$
  $\mathbf{D}_i = \mathbf{D}_{i-1} \cup \{(\mathbf{x}_i, y_i)\}$
**end for**
Update GP model on $\mathbf{D}_{n_0}$
**3) BO Loop (LineBO)**
$\mathbf{x}_{next} = \mathbf{x}_0$; $j = n_0 + 1$
**while** termination criterion is not met **do**
  Experiment: $y_{next} = f(\mathbf{x}_{next}) + \varepsilon$
  $\mathbf{D}_j = \mathbf{D}_{j-1} \cup \{(\mathbf{x}_{next}, y_{next})\}$
  Update the GP model on $\mathbf{D}_j$
  Retain the direction $m$ from $\Pi$
  $\mathbf{x}_{next} = \arg\max_{\mathbf{x} \in \mathcal{L}(q,m)} (\alpha(\mathbf{x}))$
  $j = j + 1$
**end while**
**return** $(\mathbf{x}^+, y^+) : y^+ = \hat{f}(\mathbf{x}^+) = \max_{\mathbf{x} \in \Omega} \hat{f}(\mathbf{x})$

## BO for PX4 Controller Tuning

The LineBO algorithm is applied to tune the 8 parameters of the PX4 outer control loop. The PF is computed as:

$$P_{idx}(\delta_{Pos}, CE) = -\left(w_1\, \delta_{Pos} + w_2\, CE\right) .$$

It combines tracking error $\delta_{Pos}$ and control effort $CE$ into a single scalar index, with weights prioritizing tracking accuracy. Gain combinations that lead to vehicle crashes are assigned a fixed penalty value, calibrated to avoid sharp discontinuities in the PF while limiting the exploration of low-performing regions. The reference trajectory used is a three-dimensional figure-eight path with varying altitude, which ensures that the controller is uniformly excited in all axes. Regarding implementation, a preliminary test campaign is conducted to select the most suitable algorithm settings. The Matérn 3/2 kernel, whose hyperparameters are estimated through an initial design, is adopted for the PSM. The Upper Confidence Bound (UCB) AF and the CoordinateLineBO direction oracle are selected. The initial design uses 200 points, and the optimization loop runs for 600 iterations.
The results are summarized in Table 1 for all four vehicles for both SIL and HIL architecture, comparing position error, control effort, and performance function value before and after the optimization. In the table, N/A is reported when the mission fails.

# 6. Monte Carlo Analysis

The Monte Carlo (MC) analysis is employed to assess optimized controller robustness on an illustrative mission under parametric uncertainty

| | ID | $\delta_{Pos}$ [cm] | | | CE [-] | | $P_{idx}$ [-] | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Nom. | Opt. | Delta | Nom. | Opt. | Nom. | Opt. | Delta |
| SIL | 1 | N/A | 30 | N/A | N/A | 0.14 | N/A | -4.49 | N/A |
| | 2 | 30 | 27 | -10% | 0.10 | 0.10 | -4.06 | -3.78 | +7% |
| | 3 | 28 | 23 | -18% | 0.18 | 0.18 | -4.63 | -4.17 | +10% |
| | 4 | 29 | 23 | -21% | 0.16 | 0.16 | -4.55 | -3.99 | +12% |
| HIL | ID | $\delta_{Pos}$ [cm] | | | CE [-] | | $P_{idx}$ [-] | | |
| | | Nom. | Opt. | Delta | Nom. | Opt. | Nom. | Opt. | Delta |
| | 1 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| | 2 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| | 3 | N/A | 33 | N/A | N/A | 0.22 | N/A | -5.53 | N/A |
| | 4 | 21 | 16 | -24% | 0.19 | 0.19 | -3.96 | -3.58 | +10% |

Table 1: Nominal and optimized controller performance - SIL and HIL architectures comparison.

and compare its performance with the nominal controller. Since the first three vehicles require between 8 and 15 minutes to complete their illustrative mission, the fourth vehicle is employed for its shorter mission duration of approximately 2 minutes, which keeps the computational cost of the analysis acceptable.
The analysis is performed for the nominal and the optimized controllers on both SIL and HIL architectures, with 600 samples each. The uncertain parameters included in the analysis are drone mass, motor delay, barycenter shift, and wind intensity, all modeled as Gaussian random variables. The $3\sigma$ bounds are set to 10% of the nominal value for mass and motor delay, 10% of the largest vehicle dimension for the barycenter shift, and 20% of the nominal value for wind intensity. Wind direction is additionally varied as a uniform random variable over the full 360° range.

### Results

MC analysis results are summarized in Table 2. In SIL, both controllers fly the 600 simulations correctly, with the optimized controller demonstrating better performance.
In HIL configuration, the nominal controller achieves only a 47% success rate, with 315 missions failing due to actuator saturation or loss of stability caused by parameter variations. Conversely, the optimized controller maintains a 95% success rate. Among successful runs, the nominal controller exhibits a lower $\delta_{Pos}$ and $CE$. However, the position error distribution for the optimized controller in HIL shows that approximately 90% of samples achieve tracking errors below 0.20 m, with a long tail extending to 7 m for the most challenging cases involving high wind and unfavorable motor delays.

| | SIL | | HIL | |
|---|---|---|---|---|
| | **Nom.** | **Opt.** | **Nom.** | **Opt.** |
| $\boldsymbol{\delta_{Pos}}$ **[cm]** | 45 | 15 | 33 | 50 |
| **CE [-]** | 0.17 | 0.16 | 0.14 | 0.17 |
| % **Success** | 100 | 100 | 47 | 95 |

Table 2: MC results.

## 7. Conclusions

This work presents a compact framework that integrates high-fidelity simulation, systematic SIL and HIL testing, and data-driven optimization to improve PX4 autopilot performance across four representative quadcopter platforms. A Simulink-based drone model was interfaced with the PX4 firmware; four illustrative missions were used to evaluate nominal controller behavior under increasing wind disturbance; and BO was employed to tune position and velocity controller gains. The optimized controllers were validated through MC analysis to assess robustness to parametric uncertainty and compare the performance with the nominal controllers in SIL and HIL. The BO strategy proved sample-efficiency, delivering better performance within 100–200 iterations and offering a scalable approach for rapid controller prototyping in industrial development cycles. A significant performance gap between SIL and HIL configurations highlights the importance of HIL-based tuning to increase final-solution fidelity.
Future work should transition to real-world flight testing and explore multi-fidelity BO (i.e., combining real-drone experiments with HIL simulations to raise fidelity while controlling cost), using the SafeBO algorithm to enable safety constraints and reduce crash risk. Addressing actuator saturation via advanced anti-windup techniques is also recommended to improve stability and performance.
To sum up, coupling high-fidelity simulation with SIL/HIL testing and BO provides an effective, systematic methodology for autopilot tuning that accelerates development and increases the fidelity of the final solution.

## References

[1] A. Candelieri. A Gentle Introduction to Bayesian Optimization. In *Proceedings of the 2021 Winter Simulation Conference*, 2021.

[2] Kirschner et al. Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces. In *Proceedings of Machine Learning Research*.

[3] PX4 Development Team. *PX4 Autopilot User Guide*. PX4 Project, 2025. Accessed: 2025-26-09.

[4] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.