Robust Sequential Convex Programming for Quadrotor Trajectory Planning with Obstacle Avoidance Under Wind Disturbances

Kutay Demiralay

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington
2025

Committee:
Behçet Açıkmeşe
Amirhossein Taghvaei

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Robust Sequential Convex Programming for Quadrotor Trajectory Planning with Obstacle Avoidance Under Wind Disturbances

Kutay Demiralay

Chair of the Supervisory Committee:
Behçet Açıkmeşe
Aeronautics and Astronautics

In this work, we address the problem of trajectory planning for a 3D quadrotor navigating around three spherical obstacles under bounded wind disturbances. The goal is to reach a desired final state from a given initial state within a fixed time, while minimizing fuel consumption and avoiding obstacles.

We first compute a nominal trajectory in a no-wind setting using Sequential Convex Programming (SCP) applied to a 6-state, 3-DoF quadrotor model. This trajectory satisfies all state, control, and obstacle constraints while minimizing fuel use.

However, in the presence of constant wind, the nominal path becomes unreliable due to unmodeled drift, leading to potential obstacle violations—unacceptable in safety-critical applications.

To improve robustness without relying on any low-level controller, we implement three strategies: (i) LQR-based tracking of the wind-free nominal trajectory, (ii) a receding-horizon SCP approach that re-solves the optimization at each node using updated state feedback, convex half-space approximations, and Wind-Adaptive Residual Correction (WARC), and (iii) a tracking-style method using smaller SCP subproblems to increase responsiveness. Finally, we apply funnel synthesis along the nominal trajectory to certify allowable deviations at each node. This framework enables safe, fuel-efficient flight despite bounded disturbances.

Kutay Demiralay

1 Introduction

Trajectory optimization using Sequential Convex Programming (SCP) has become a widely used framework for generating dynamically feasible and collision-free trajectories in nonlinear systems such as quadrotors. SCP works by iteratively solving a sequence of convex approximations to an originally non-convex problem. The process begins by linearizing the nonlinear dynamics and constraints around a current nominal trajectory, also known as the trust point, using a first-order Taylor expansion based on Jacobians. The continuous-time system is then discretized to obtain a finite set of nodes, and the linearized discrete dynamics are used to propagate the state between these nodes. To ensure the validity of the linear approximations, a penalized trust region is imposed, restricting updates to remain close to the nominal trajectory. At each time step, convex constraints and a convex cost function are applied, allowing the problem to be formulated as a convex optimization problem. This problem is solved iteratively, with the nominal trajectory updated after each solution, until convergence is achieved.

However, by design, it assumes perfect knowledge of the system dynamics and environment. As a result, SCP lacks robustness to unmodeled disturbances, such as wind, when these are not encoded directly into the system's dynamics.

This work builds on foundational ideas presented in [3, 4] by Michael Szmuk and colleagues, where successive convexification is applied to aggressive quadrotor flight through constrained environments. Our SCP implementation is adapted from these works and extended to 3D environments with obstacles and disturbances. Furthermore, our quadrotor model uses a simplified 3-DoF translational dynamics model treated as a point mass.

In this study, we implement a full SCP-based trajectory optimization pipeline in Python for a 6-state, 3-DoF quadrotor navigating in a 3D environment with three spherical obstacles. The quadrotor is modeled as a point mass, meaning its state vector consists of 6 elements: position (x, y, z) and velocity (v_x, v_y, v_z). The control input is a 3-dimensional force vector (f_x, f_y, f_z), representing the total force applied in each Cartesian direction. Our initial SCP formulation generates a fuel-efficient trajectory from a specified start point to a desired endpoint in a fixed final time. After optimization, total fuel consumption and obstacle constraint violations are evaluated separately.

However, when a bounded but unmodeled wind disturbance is introduced, the quadrotor drifts away from the planned path. This leads to obstacle collisions and terminal state deviation, revealing SCP's fundamental vulnerability to unknown disturbances.

To improve robustness, a low-level controller is typically used alongside a trajectory planner to correct deviations from the planned path. However, in this thesis, we introduce several assumptions that make the problem more challenging:

- No low-level controllers are used for feedback or stabilization.
- The drone's position and state error information can only be fed into the SCP algorithm at discrete, predefined time steps—not continuously.
- Wind is bounded but unknown; there is no prior knowledge of future wind conditions.

With these assumptions in place, we deliberately avoid relying on a low-level controller and instead explore several strategies—rooted in tracking-based and MPC-style control philosophies—integrated directly into our main SCP algorithm to improve robustness against wind disturbances.

- Several tracking-based strategies, where the wind-free nominal SCP trajectory is used as a reference, and deviations are corrected using various local feedback or trajectory correction techniques.
- Multiple MPC-style strategies, where the full SCP problem is re-solved either at every node or within a shifting window, using the current (drifted) state as the new initial condition.
- Additional robustness is introduced by expanding obstacle constraints into convex half-space approximations with safety margins, helping to reduce violation risk under drift.

Beyond these control-based methods, we incorporate a more formal robustness guarantee by adopting the funnel synthesis framework from Taewan Kim et al. [1], titled *Optimization-Based Constrained Funnel Synthesis for Systems with Lipschitz Nonlinearities via Numerical Optimal Control*. This method constructs Lyapunov-based funnels around a nominal trajectory, providing local regions of guaranteed safety under bounded perturbations. Since our drone dynamics are Lipschitz continuous and the trajectory is smooth, we are able to safely apply this method on top of our SCP method.

All modeling and optimization are implemented in Python, with symbolic derivatives computed using SymPy and convex optimization performed using CVXPY. We adopt the same notation and symbolic formulation style presented in Danylo Malyuta's tutorial [2], which served as a key reference for encoding the dynamics and constraints in our SCP implementation.

Ultimately, this work evaluates and compares several robustification techniques for trajectory planning under bounded wind disturbances. We aim to balance obstacle avoidance, terminal accuracy, and fuel efficiency while using known SCP methods, practical engineering tools, and convex optimization principles.

All Python and Simulink code used for the simulations in this thesis is openly available at: https://github.com/kutaydemiralay/Robust-SCP-for-Quadrotor-Obstacle-Avoidance-and-Controller-Design

2 Dynamics and Optimization Setup

In this section, we describe the point-mass model used for quadrotor trajectory optimization. The quadrotor is modeled as a point mass with a 6-dimensional state vector comprising 3D position and velocity, and 3 control inputs representing force components in the inertial frame. This simplification allows for efficient optimization while capturing the essential translational dynamics of the vehicle. The linearization and discretization techniques employed are adapted from the methodologies in [6] and [7].

2.1 Point-Mass Dynamics (Continuous-Time)

The state vector $x_k \in \mathbb{R}^6$ is composed of the position $p_k \in \mathbb{R}^3$ and velocity $v_k \in \mathbb{R}^3$. The control input $u_k \in \mathbb{R}^3$ corresponds to the total force applied in the world frame. The dynamics are given

by:

$$\dot{p}_k = v_k \tag{1}$$

$$\dot{v}_k = \frac{1}{m} u_k + g \tag{2}$$

here:

- $p_k = [x, y, z]^\top$ is the discrete position,
- $v_k = [\dot{x}, \dot{y}, \dot{z}]^\top$ is the discrete velocity,
- $u_k = [f_x, f_y, f_z]^\top$ is the discrete applied force,

2.2 Linearization

At each time step, we linearize the nonlinear point-mass dynamics about the current nominal trajectory $\bar{x}(t), \bar{u}(t)$ using a first-order Taylor expansion:

$$\dot{x}(t) \approx A(t)x(t) + B(t)u(t) + z(t) \tag{3}$$

here:

$$A(t) = \left.\frac{\partial f}{\partial x}\right|_{\bar{x}(t),\bar{u}(t)} \tag{4}$$

$$B(t) = \left.\frac{\partial f}{\partial u}\right|_{\bar{x}(t),\bar{u}(t)} \tag{5}$$

$$z(t) = f(\bar{x}(t), \bar{u}(t)) - A(t)\bar{x}(t) - B(t)\bar{u}(t) \tag{6}$$

- $A \in \mathbb{R}^{6\times 6}$ is the linearized dynamics matrix,
- $B \in \mathbb{R}^{6\times 3}$ are control input matrices from FOH interpolation,
- $z \in \mathbb{R}^6$ is the affine shift vector.

2.3 First-Order Hold (FOH) Interpolation

To capture the effect of control variations between discretization nodes, we use First-Order Hold (FOH) interpolation. For $t \in [t_k, t_{k+1}]$, the control input is defined as:

$$u(t) = \sigma_k^-(t)u_k + \sigma_k^+(t)u_{k+1} \tag{7}$$

where the interpolation weights are given by:

$$\sigma_k^-(t) = \frac{t_{k+1} - t}{t_{k+1} - t_k} \tag{8}$$

$$\sigma_k^+(t) = \frac{t - t_k}{t_{k+1} - t_k} \tag{9}$$

2.4 Discretized Dynamics (Euler Integration)

We discretize the dynamics using Euler integration with a time step Δt. The discrete-time affine dynamics become:

$$x_{k+1} = A_k x_k + B_k u_k + z_k \tag{10}$$

with:

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \in \mathbb{R}^6, \quad u_k \in \mathbb{R}^3$$

$$A_k = \begin{bmatrix} I_3 & \Delta t \cdot I_3 \\ 0_{3\times3} & I_3 \end{bmatrix}, \quad B_k \begin{bmatrix} 0_{3\times3} \\ \frac{\Delta t}{m} I_3 \end{bmatrix}, \quad z_k = \begin{bmatrix} 0_{3\times1} \\ \Delta t \cdot g \end{bmatrix} \tag{11}$$

here:

- I_3 is the 3×3 identity matrix,
- $0_{3\times3}$, $0_{3\times1}$ are zero matrices of appropriate dimensions,

2.5 Continuous Deviation Propagation Over Time Interval

Instead of just using Euler, we can do a more accurate integration by propagating the full linearized system over the interval. To compute how the state evolves from time t_k to t_{k+1}, we integrate the linearized system dynamics over that interval:

$$x(t_{k+1}) = \Phi(t_{k+1}, t_k)\, x(t_k) + \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau)\, [B(\tau)\, u(\tau) + z(\tau)]\, d\tau$$

Here, $\Phi(t_{k+1}, t_k)$ is the state transition matrix that propagates the state from time t_k to t_{k+1}, and $z(\tau)$ is the affine term resulting from linearization.

This integral is computed numerically using a high-accuracy Runge-Kutta 4 method (e.g., DOP853). As a result, the following affine discrete-time dynamics are obtained:

We assumed a first-order hold (FOH) on the control input $u(t)$, which interpolates linearly between control values at two adjacent time nodes:

$$u(\tau) = \sigma_k^-(\tau)\, u_k + \sigma_k^+(\tau)\, u_{k+1}$$

Substituting the FOH expression into the integral allows us to express the influence of both u_k and u_{k+1} on the next state x_{k+1}, leading to the final discrete-time affine dynamics used in optimization.

2.6 Break the Integral into Influence Terms

By substituting the FOH expression for $u(t)$ into the integral form of the deviation dynamics, and reorganizing the terms, we can group terms to obtain the following discrete-time approximation:

$$x_{k+1} = A_k x_k + B_k u_k + B_k^- u_{k+1} + z_k$$

Where:

- $A_k = \Phi(t_{k+1}, t_k)$: the state transition matrix computed over the time interval $[t_k, t_{k+1}]$, typically using numerical integration methods such as Runge-Kutta.
- $B_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) B(\tau) \sigma_k^-(\tau)\, d\tau$: the control influence from u_k, weighted by the FOH interpolation factor σ_k^-.
- $B_k^- = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) B(\tau) \sigma_k^+(\tau)\, d\tau$: the control influence from u_{k+1}, due to FOH linear interpolation with σ_k^+.
- $z_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) z(\tau)\, d\tau$: the accumulated effect of the affine term from linearization over the interval.

2.7 Final Discrete-Time Affine Dynamics

The resulting discrete-time affine dynamics used in the SCP formulation are:

$$x_{k+1} = A_k x_k + B_k u_k + B_k^- u_{k+1} + z_k$$

This compact form captures:

- The influence of both current and next-step control inputs, made possible by the FOH approximation.
- The time-varying, linearized dynamics propagated over a finite time step.
- The residual nonlinear behavior encoded in the affine term z_k, which corrects for linearization errors.

This affine dynamic model is key to enabling convex optimization within the SCP framework while preserving a high level of fidelity to the original nonlinear system.

This linearization and discretization framework follows the approaches described in [6] and [7].

2.8 Wind Effect

In the presence of optional wind, the affine term z_k is modified as:

$$z_k \leftarrow z_k + \Delta t \cdot w \tag{12}$$

We begin by defining a constant wind vector $w \in \mathbb{R}^3$, which perturbs only the position components of the state trajectory. This wind is treated as a worst-case constant disturbance. Since wind is assumed to be bounded, injecting the maximum allowable wind vector throughout the trajectory simulates the most adverse scenario. Designing and verifying robustness against this worst-case disturbance provides a conservative yet practical guarantee of safety and is a well-established technique in robust control and trajectory planning.

Since each node in the trajectory corresponds to a fixed time interval of 0.4 seconds in our simulations, applying a constant wind vector introduces a physically meaningful drift in position. A constant wind can be interpreted as a steady external velocity disturbance acting over time. Because the wind vector is constant, it causes a consistent shift in the position at each

node—resulting in equal drift increments across the trajectory, which is physically reasonable. For example, a wind of one unit in the + y-direction results in a positional drift of approximately one unit every 0.4 seconds. Over a 20-step horizon (i.e., 8 seconds), this leads to a cumulative drift of 20 units in the +y-direction. This demonstrates how even a simple, constant disturbance can significantly impact system performance when not modeled in the dynamics, emphasizing the need for robust trajectory planning techniques.

3 Non-Convex Trajectory Optimization Formulation

In this section, we formulate the trajectory optimization problem for a point-mass quadrotor model which is naturally a non-convex, nonlinear optimal control problem. The quadrotor is treated as a point mass subject to Newtonian dynamics, with thrust as the control input. The goal is to generate a dynamically feasible and fuel-efficient trajectory that avoids obstacles and satisfies boundary conditions.

3.1 Objective

The objective is to minimize the total control effort, measured as the integral of the Euclidean norm of the thrust vector:

$$\min_{T(t)} \int_0^{t_f} \|T(t)\|_2 \, dt$$

Here:

- $T(t) \in \mathbb{R}^3$: Thrust vector (control input),

This corresponds to minimizing the total fuel consumption over the trajectory.

3.2 Boundary Conditions

The initial and final conditions on position, velocity, and thrust are specified as:

$$\begin{aligned} r(0) = r_i, \quad v(0) = v_i, \quad T(0) = T_i \\ r(t_f) = r_f, \quad v(t_f) = v_f, \quad T(t_f) = T_f \end{aligned}$$

Here:

- $r(t) \in \mathbb{R}^3$: Position vector,
- $v(t) \in \mathbb{R}^3$: Velocity vector,

3.3 Dynamics

The point-mass dynamics of the system are governed by Newton's second law:

$$\dot{r}(t) = v(t)$$
$$\dot{v}(t) = \frac{1}{m}T(t) + g$$

Here:

- $g = [0, 0, g_z]^\top$: Gravity vector,
- m: Mass of the quadrotor.

3.4 Thrust Magnitude Constraints

To model physical limitations of the motors, we constrain the norm of the thrust vector:

$$T_{\min} \leq \|T(t)\|_2 \leq T_{\max}$$

This ensures that thrust remains within allowable bounds at all times.

3.5 Thrust Direction Constraint (Tilt Angle constraint)

The thrust vector is typically constrained to lie within a tilt cone relative to the vertical axis $e_1 = [0, 0, 1]^\top$. This is enforced as:

$$\|T(t)\|_2 \cos(\theta_{\max}) \leq e_1^\top T(t)$$

where $\theta_{\max}$ is the maximum allowed tilt angle.

3.6 Obstacle Avoidance Constraints

We model each obstacle j as a sphere defined by its center $r_j(t) \in \mathbb{R}^3$ and radius R_j. To enforce avoidance, we require:

$$\|H_j(t)(r(t) - r_j(t))\|_2 \geq 1 \quad \forall j$$

For spherical obstacles, we let $H_j(t) = \frac{1}{R_j} I_3$, where I_3 is the 3×3 identity matrix. This constraint ensures that the quadrotor remains outside the boundary of each spherical obstacle at all times.

3.7 Remarks

This formulation is nonlinear and non-convex due to the thrust norm, tilt cone constraint, and the nonlinear obstacle avoidance constraints. In later sections, we introduce convexification strategies using sequential convex programming to solve this problem efficiently.

4 Discrete Convex Subproblem Formulation

Fortunately, we can reformulate our originally nonconvex, nonlinear problem into a convex, discretized subproblem that can be solved iteratively to generate a feasible trajectory, as shown in [3], [4]. At each SCP iteration, we solve this convex optimization problem using the ECOS solver.

4.1 Objective

The objective is to find a trajectory and control sequence that closely tracks the nominal path, minimizes fuel usage, avoids obstacles, respects dynamics, and reaches the desired final state, while penalizing any soft constraint violations to maintain robustness and feasibility.

$$\begin{aligned}\min_{x_k,u_k,\nu_{j,k},\nu_{\text{dyn},k}} \quad & \sum_{k=1}^{N}\left(w_{\text{tr}}\|x_k-\bar{x}_k\|_2^2+w_{\text{fuel}}\|u_{k-1}\|_2+w_\nu\sum_{j=1}^{n_{\text{obs}}}\nu_{j,k-1}^2+w_{\text{dyn}}\|\nu_{\text{dyn},k-1}\|_1\right)\\ & +w_{\text{terminal}}\|x_N-x_{\text{final}}\|^2\end{aligned}$$

Here:

$$\begin{aligned} x_k &\in \mathbb{R}^6 \quad \text{(position and velocity at time step } k\text{)}\\ u_k &\in \mathbb{R}^3 \quad \text{(control input / force vector)}\\ \nu_{j,k} &\in \mathbb{R}_+ \quad \text{(obstacle avoidance slack variable for obstacle } j\text{)}\\ \nu_{\text{dyn},k} &\in \mathbb{R}^6 \quad \text{(dynamics violation slack variable)}\end{aligned}$$

Since this objective function is composed of a combination of quadratic terms, ℓ_2-norms, sums of squares of scalar variables, and ℓ_1-norms—all of which are convex—a nonnegative weighted sum of these terms preserves convexity, making the entire objective function convex.

4.2 Constraints

- **Linearized Dynamics (for all $k = 1, \ldots, N$):**

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + B_{k-1}^{-}u_k + z_{k-1} + \nu_{\text{dyn},k-1}$$

- **Obstacle Avoidance (for all obstacles j and steps k):**

  A linearized approximation of the nonconvex obstacle avoidance constraint is constructed around the previous iteration's state using a first-order Taylor expansion. This linearization

leverages the fact that for any convex function f, the first-order Taylor expansion is a *global underestimator*:

$$f(x) \;\geq\; f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) \quad \forall x.$$

This property ensures that replacing the original nonconvex constraint with its affine underestimator yields a conservative convex constraint. The resulting constraint is implemented as a *soft constraint*, with a nonnegative slack variable $\nu_{j,k-1}$ added to the objective function and penalized to discourage violations.

(a) Signed Distance Linearization: The (scaled) signed distance to obstacle j at step $k-1$ is given by

$$f_{j,k-1} \;=\; \left\| H_{j,k-1} \left(x_{k-1}^{\text{pos}} - c_j \right) \right\|_2,$$

with gradient

$$\nabla f_{j,k-1} \;=\; \frac{H_{j,k-1}^\top H_{j,k-1} \left(x_{k-1}^{\text{pos}} - c_j \right)}{\left\| H_{j,k-1} \left(x_{k-1}^{\text{pos}} - c_j \right) \right\|_2}.$$

Applying the first-order Taylor underestimator, the unit-clearance constraint $f_j(x_k^{\text{pos}}) \geq 1 - \nu_{j,k-1}$ is conservatively enforced by the affine inequality

$$1 - f_{j,k-1} - \nabla f_{j,k-1}^\top \left(x_k^{\text{pos}} - x_{k-1}^{\text{pos}} \right) \;\leq\; \nu_{j,k-1}.$$

(b) Barrier Function Linearization: The barrier function for obstacle j is defined as

$$g_{j,k-1} \;=\; 1 - \frac{\left\| \bar{x}_{k-1}^{\text{pos}} - c_j \right\|_2^2}{r_j^2},$$

with gradient

$$\nabla g_{j,k-1} \;=\; -\frac{2}{r_j^2} \left(\bar{x}_{k-1}^{\text{pos}} - c_j \right).$$

Using the same underestimator property, the linearized barrier constraint

$$g_{j,k-1} + \nabla g_{j,k-1}^\top \left(x_k^{\text{pos}} - \bar{x}_{k-1}^{\text{pos}} \right) \;\leq\; \nu_{j,k-1}$$

is enforced. The slack variable $\nu_{j,k-1} \geq 0$ is penalized in the objective to maintain feasibility while encouraging obstacle clearance.

Trust Region Constraint:

To ensure stability and convergence during each SCP iteration, we constrain the change in the solution between iterations using an ℓ_1-norm trust region:

$$\|\mathbf{r}_k^i - \mathbf{r}_k^{i-1}\|_1 \leq \Delta^i \quad \forall k \in \mathbb{K}$$

This limits how much the trajectory is allowed to deviate from the previous iteration, preventing large steps that could lead to infeasibility or divergence.

Thrust Magnitude and Tilt Angle Constraints:

To ensure that the control input (acceleration) remains within feasible physical limits and the tilt of the thrust vector does not exceed the allowable cone around the vertical axis, we reformulate the thrust magnitude and direction constraints into a convex form using the following set of inequalities:

$$\|\mathbf{a}_k^i\|_2 \leq \sigma_k^i \tag{13}$$

- This constrains the magnitude (Euclidean norm) of the control vector $\mathbf{a}_k^i$ to be less than or equal to a slack variable σ_k^i, which decouples the thrust direction from its magnitude in a convex way.
- This is a second-order cone (SOC) constraint, and therefore convex.

$$a_{\min} \leq \sigma_k^i \leq a_{\max} \tag{14}$$

- This imposes upper and lower bounds on the auxiliary variable σ_k^i, which acts as a proxy for the thrust magnitude and ensures it remains within the physically allowable range $[a_{\min}, a_{\max}]$.
- This is a pair of linear inequality constraints, and therefore convex.

$$\sigma_k^i \cos(\theta_{\max}) \leq \mathbf{e}_1^\top \mathbf{a}_k^i \quad \forall k \in \mathbb{K} \tag{15}$$

- This enforces a tilt angle constraint, limiting the angle between the thrust vector $\mathbf{a}_k^i$ and the vertical axis $\mathbf{e}_1 = [0, 0, 1]^\top$, by ensuring that the vertical component of thrust is sufficiently large relative to its magnitude.
- This is a linear inequality constraint, and therefore convex.

Here:

- $\mathbf{a}_k^i \in \mathbb{R}^3$: acceleration (control input) at node k and SCP iteration i, representing the direction and magnitude of applied thrust.
- $\sigma_k^i \in \mathbb{R}$: an auxiliary scalar variable introduced to bound the norm of $\mathbf{a}_k^i$, enabling convex enforcement of both upper and lower thrust magnitude limits.
- $a_{\min}, a_{\max}$: user-defined lower and upper bounds on total thrust magnitude (expressed via acceleration).
- $\theta_{\max}$: the maximum allowed tilt angle between the thrust vector and the vertical axis.
- $\mathbf{e}_1 = [0, 0, 1]^\top$: the unit vector in the vertical direction.

Control and State Bounds:

$$x_k \in [x_{\min}, x_{\max}], \quad u_k \in [u_{\min}, u_{\max}]$$

This defines a hyperrectangle (box) in $\mathbb{R}^n$, which is a convex set.

Initial and Final State Conditions:

$$x_0 = x_{\text{init}}, \quad x_N = x_{\text{final}} \quad \text{(hard constraint)}$$

An affine equality constraint defines a convex set.

5 High-Level Algorithm: SCP-Based Robust Trajectory Optimization for a Point-Mass Quadrotor Under Wind

Step 1: Initialize Parameters
Set total time horizon T_f, number of nodes N, initial state $x_i \in \mathbb{R}^6$, and final desired state $x_f \in \mathbb{R}^6$. Generate initial state and control trajectory guesses using straight-line interpolation. The initial trajectory is a straight line from x_i to x_f with an initial control input of $[0, 0, 9.81]^\top$, corresponding to hover thrust.

Step 2: Compute Symbolic Jacobians
Derive linearized dynamics:

$$A(x,u) = \frac{\partial f}{\partial x}, \quad B(x,u) = \frac{\partial f}{\partial u}$$

for the point-mass model:

$$\dot{x} = \begin{bmatrix} v \\ \frac{1}{m}u + g \end{bmatrix}$$

Step 3: Iterative SCP Loop
Repeat until convergence or maximum iteration $k_{\max}$ is reached:

- **Linearize Dynamics:**
  For each node k, use numerical integration (e.g., Runge-Kutta) to compute A_k, B_k, next-step control influence B_k^-, and affine term z_k.
- **Linearize Obstacle Constraints:**
  For each obstacle j and node k, compute signed distance $g_{j,k}$ and gradient $\nabla g_{j,k}$. Apply first-order approximation:
  $$g_{j,k} + \nabla g_{j,k}^\top \delta x_k \leq \nu_{j,k}$$
- **Solve Convex Subproblem:**
  Minimize the following cost function:
  $$\sum_k \left(\|x_k - \bar{x}_k\|^2 + \|u_k - \bar{u}_k\|^2 + \|u_k\| + \|\nu_k\|^2 + \|\nu_{\text{dyn},k}\|_1 \right)$$
  subject to all linearized constraints, using a modeling tool such as `cvxpy` and a solver such as `ECOS`.
- **Update Trajectory:**
  Set $x \leftarrow x^{\text{opt}}, \quad u \leftarrow u^{\text{opt}}$. Also Set $\bar{x} \leftarrow x^{\text{opt}}, \quad \bar{u} \leftarrow u^{\text{opt}}$ for future iterations.
- **Check Convergence:**
  Terminate if relative change in x and u is below a predefined threshold.

Step 4: Post-Processing — Apply Wind Disturbance
For each node k, perturb position states to simulate wind drift:

$$x_k^{\text{pos}} \leftarrow x_k^{\text{pos}} + w \cdot t_k$$

Step 5: (Optional) Funnel Synthesis
At each node, solve local Lyapunov subproblems to certify robustness margins.

Output: The algorithm returns the optimized state trajectory $x(t)$, control inputs $u(t)$, total fuel usage, obstacle violation metric, terminal error, and relevant plots.

Total fuel usage is computed as the sum of the magnitudes of control inputs over the entire trajectory. Formally, if u_k denotes the control input at time step k, then the total fuel cost is given by:

$$\text{Fuel Usage} = \sum_{k=0}^{N-1} \|u_k\|_2,$$

where $\|\cdot\|_2$ denotes the Euclidean norm of the control input vector at each time step.

Total obstacle violation quantifies how much the state trajectory enters the unsafe (obstacle) region. Let g_k represent the clearance from the obstacle at time step k, such that $g_k \geq 0$ implies no violation. The cumulative obstacle violation is then computed as:

$$\text{Obstacle Violation} = \sum_{k=0}^{N} \max(0, -g_k),$$

where any negative clearance value contributes positively to the violation cost.

Terminal error is defined as the Euclidean distance between the final state x_N and the desired final state x_f:

$$\text{Terminal Error} = \|x_N - x_f\|_2.$$

6 Simulation Runs

6.1 Baseline Trajectory Without Disturbance

Now that the SCP-based trajectory optimization for point-mass quadrotor obstacle avoidance is complete, we run the simulation and visualize the resulting trajectory. The drone is modeled as a 1 kg vehicle subject to realistic gravitational acceleration, and its task is to move from an initial to a final state while avoiding obstacles.

We define the following parameters for the simulation:

- **Mass (m)**: 1.0 kg
- **Gravitational acceleration (g)**: 9.81 m/s^2

- **Initial state (x_{init})**: $[-10,\ 0,\ 0,\ 0,\ 0,\ 0]$
  The drone starts at position $x = -10$, $y = z = 0$, initially at rest.
- **Final state (x_{final})**: $[10,\ 0,\ 0,\ 0,\ 0,\ 0]$
  The drone ends at position $x = 10$, $y = z = 0$, also at rest.
- **Time step between nodes (Δt)**: 0.4 seconds
- **Wind vector (w)**: $[0.0,\ 0.0,\ 0.0]$
  No external wind disturbance is applied in any direction.
- **Obstacle centers**:

  Obstacle 1: $[-5.0,\ -1.1,\ 0.0]$
  Obstacle 2: $[0.0,\ 0.0,\ 0.0]$
  Obstacle 3: $[5.0,\ 0.9,\ 0.0]$

- **Obstacle radii**: $[1.7,\ 1.2,\ 1.8]$
- **Maximum control input ($u_{\max}$)**: $[20,\ 20,\ 20]$
- **Minimum control input ($u_{\min}$)**: $[-20,\ -20,\ -20]$
- **Maximum state values ($x_{\max}$)**: $[10,\ 10,\ 10,\ 50,\ 50,\ 50]$
- **Minimum state values ($x_{\min}$)**: $[-10,\ -10,\ -10,\ -50,\ -50,\ -50]$

The scaling methods, weighting parameters, and CVXPY modelling shape optimization dimensions used in this work were guided by Chris Hayner's CTCS tutorial presented and shared during the AA548 Spring 2023 course at UW, instructed by Professor Karen Leung [8]. Some key Cvxpy programming related definitions were deducted from this work.

These parameters define a clean baseline scenario in which the quadrotor moves in a straight line under ideal conditions, allowing us to evaluate the effectiveness of the SCP-based trajectory planner in a disturbance-free environment.

As shown in the figure 1, the optimized trajectory successfully navigates through the spherical obstacles without any collisions. The absence of obstacle constraint violations confirms that the generated path remains entirely within the feasible safe space. The vehicle also reaches the specified final state with a total fuel cost of approximately 214.04,

6.2 Simulation Run With Constant Wind

To evaluate the effect of wind on the nominal trajectory, we introduce a constant wind vector $w = [0.0,\ 1.0,\ 0.0]$, which applies a unit drift in the positive y-direction at each time step. This simulates a worst-case scenario where wind steadily pushes the quadrotor away from its intended path.

Although the planned trajectory remains the same as in the wind-free case, the accumulated effect of wind leads to a growing deviation from the nominal path. Specifically, because the wind

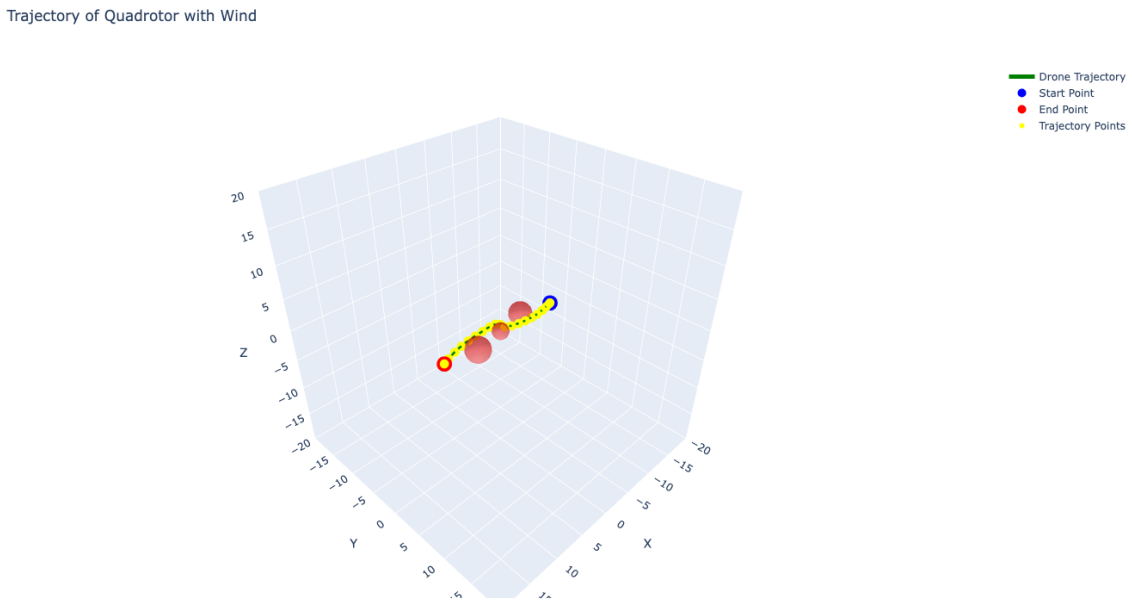

Figure 1: Trajectory of quadcopter plot found with non-robust SCP without wind. Total fuel cost for Simulation is 214.04, Total Obstacle Violation: 0.0

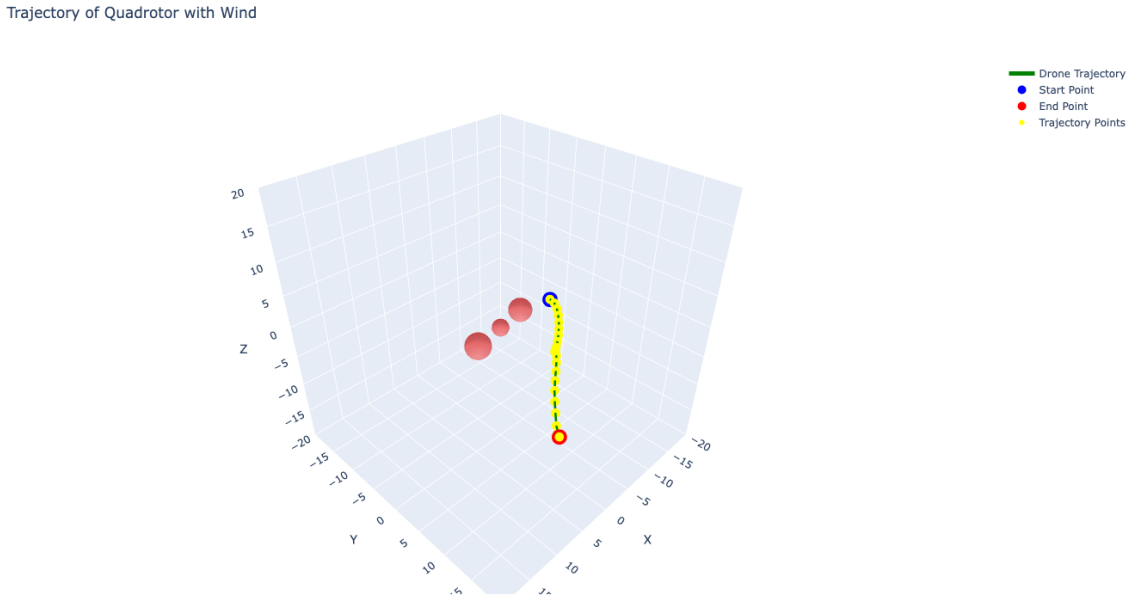

Figure 2: Trajectory of quadcopter plot found with non-robust SCP in the presence of constant wind.Simulation is 214.04, Total Obstacle Violation: 0.0

introduces a drift of one unit per step, after 20 steps the quadrotor has deviated by approximately 20 units in the $+y$-direction.

Since our Sequential Convex Programming (SCP) formulation is not inherently robust to dis-

turbances and the dynamics model does not account for wind, the optimizer remains unaware of this cumulative drift. As a result, while the fuel cost remains nearly identical to the no-wind case, the actual trajectory incurs significant obstacle violations, highlighting the need for additional robustification strategies.

6.3 LQR Feedback Control for Wind Compensation

Given the motivation established in the previous section—namely, that constant wind disturbances cause significant trajectory drift and obstacle violation due to the SCP planner's lack of robustness—we explore a first step toward compensating for such disturbances via feedback control. Specifically, we implement a Linear Quadratic Regulator (LQR) as a tracking controller to mitigate the effects of wind.

We begin by computing a nominal trajectory using the SCP planner in the absence of wind. This trajectory serves as the reference path to be tracked. Then, under wind-disturbed dynamics, we apply an LQR controller to track this nominal trajectory as closely as possible.

We consider the linear time-varying discrete system:

$$x_{k+1} = A_k x_k + B_k u_k$$

where A_k, B_k are linearized dynamics matrices evaluated around the nominal trajectory at each timestep k.

The goal of LQR is to find a feedback control law that minimizes the infinite-horizon quadratic cost:

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_k^{\text{ref}})^\top Q (x_k - x_k^{\text{ref}}) + (u_k - u_k^{\text{ref}})^\top R (u_k - u_k^{\text{ref}}) \right] + (x_N - x_N^{\text{ref}})^\top P_N (x_N - x_N^{\text{ref}})$$

$$P_k = A_k^\top P_{k+1} A_k - A_k^\top P_{k+1} B_k (B_k^\top P_{k+1} B_k + R)^{-1} B_k^\top P_{k+1} A_k + Q$$

Here:

- Q penalizes state tracking error,
- R penalizes control effort,
- P_k represents the cost-to-go matrix at time k. The Discrete-time Algebraic Riccati Equation equation is the recursive relation used to compute P_k backward in time, capturing how future costs propagate through the system dynamics under optimal control.

We compute the optimal control gains by solving the Riccati recursion backward in time.

For $k = N-1, N-2, \ldots, 0$, recursively compute the Riccati equation using the terminal condition:

$$P_N = Q_f$$

At each time step k, compute the optimal feedback gain:

$$K_k = \left(B_k^\top P_{k+1} B_k + R \right)^{-1} B_k^\top P_{k+1} A_k$$

Then, the optimal control law is:

$$u_k = u_k^{\text{ref}} - K_k(x_k - x_k^{\text{ref}})$$

where K_k is the time-varying feedback gain computed by solving the Discrete-time Algebraic Riccati Equation .

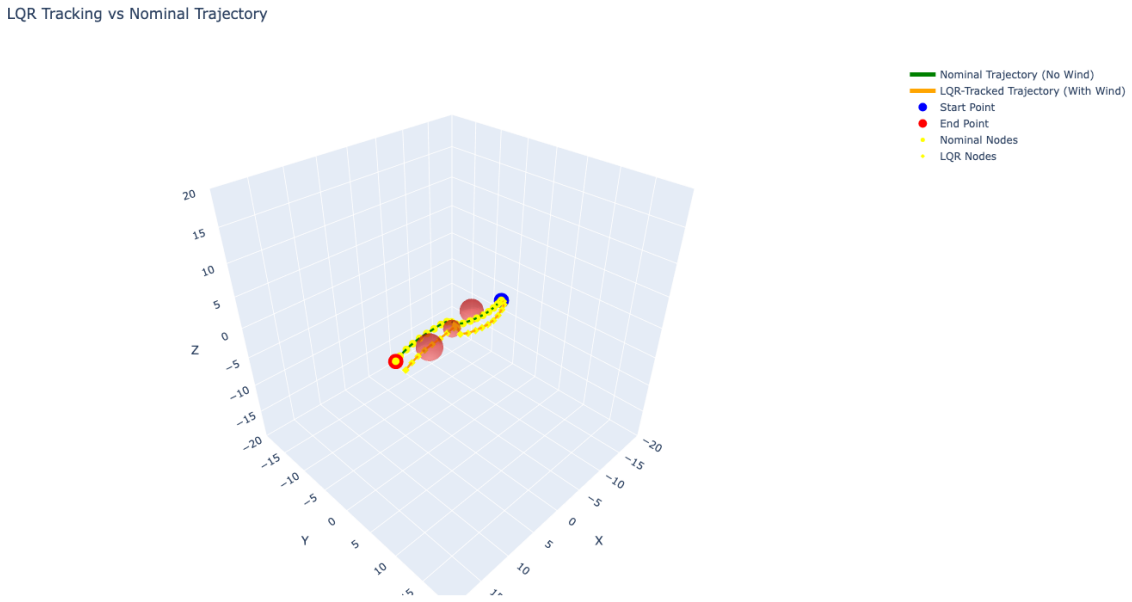

Figure 3: Trajectory of the quadcopter under constant wind using LQR, along with the nominal trajectory found via SCP in the absence of wind. Total fuel cost for LQR tracking is 214.04. Total obstacle violation is 2.99.

6.3.1 Limitations of LQR in Obstacle-Rich Environments

Although LQR provides a simple and efficient feedback controller for trajectory tracking, it lacks awareness of environmental constraints such as obstacle boundaries. As a result, even though LQR can keep the trajectory close to the nominal reference under wind disturbances, it still leads to significant obstacle violations if the deviation is large enough as can be seen on figure 3.

This limitation motivates the need for more advanced, constraint-aware robustification strategies. In the next section, we address this by integrating a receding-horizon Model Predictive Control (MPC) based SCP method, which allows re-optimization in real time and explicit consideration of obstacle constraints under disturbances.

6.4 MPC based Sequential Convex Programming for Robust Replanning

To address challenges in trajectory optimization under disturbances such as wind, we introduce an MPC based extension to the Sequential Convex Programming (SCP) framework. In this approach, the optimization problem is resolved at each time step using the current state as the initial condition. The remaining portion of the trajectory is re-optimized in a receding-horizon fashion, enabling the system to dynamically adapt to perturbations such as wind drift.

With this MPC-style replanning under a constant wind vector of $w = [0, 1, 0]$, we observe that the quadrotor avoids obstacle collisions successfully, even under persistent drift. However, the resulting trajectories no longer pass through the gaps between obstacles as in the undisturbed case. Instead, the planner directs the vehicle to take wider turns around the obstacles to maintain feasibility.

MPC-Style Sequential Convex Programming (SCP) Algorithm:

1. **Initialize:** Set initial state $x_{\text{init}} \leftarrow x_0$, desired final state x_{final}, total horizon length N, and time step Δt.
2. **While** the final state is not reached:
   (a) Solve the full-horizon SCP problem from current state x_{init} to target x_{final}.
   (b) Let the solution yield optimal state trajectory $\{x_0, x_1, \ldots, x_N\}$ and control inputs $\{u_0, u_1, \ldots, u_{N-1}\}$.
   (c) Simulate or apply one control step if needed. Then shift the initial state:
   $$x_{\text{init}} \leftarrow x_1$$
   i.e., the second state in the current optimal trajectory becomes the new initial condition.
   (d) Repeat the SCP optimization with updated x_{init}.
3. **Output:** Concatenate all local trajectories into a global state and control history.

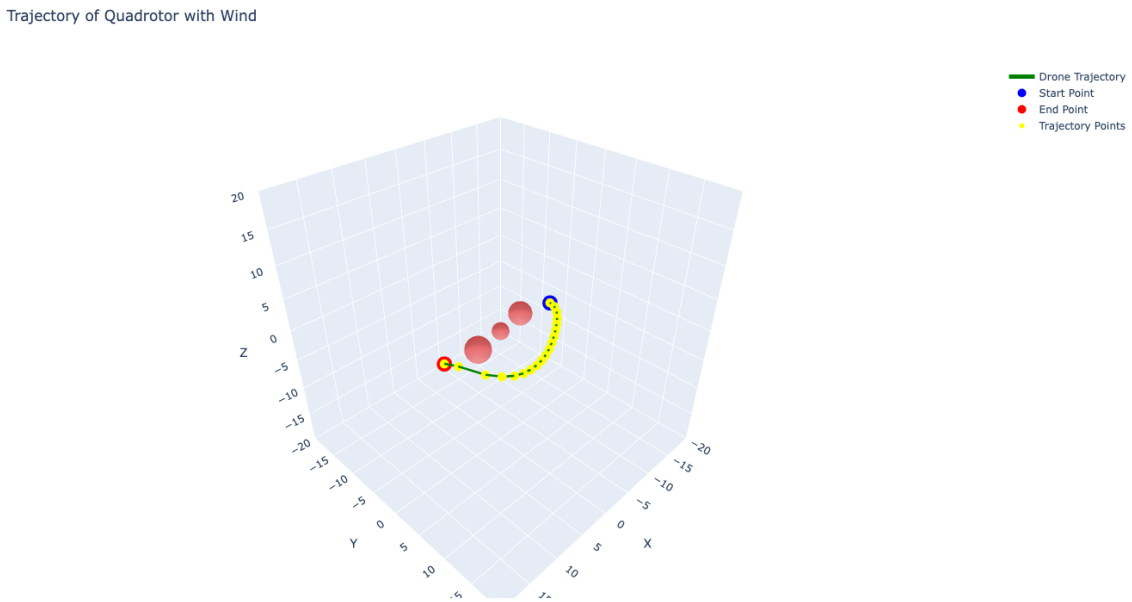

Figure 4: Trajectory of the quadcopter under constant wind using SCP solved iteratively in an MPC-like fashion at each node, w Total fuel cost for the stacked trajectory is 226.47. Total obstacle violation: 0.00.

This behavior results in a significant increase in fuel consumption, highlighting the trade-off between robustness and efficiency as can be seen from figure 4. The total fuel cost rises to 226.47, compared to 214.03 for the nominal trajectory, primarily due to an unnecessarily wide turn. Although the quadrotor successfully avoids collisions, the detour leads to energetically suboptimal behavior. This motivates further investigation into strategies that maintain safety under disturbances while promoting tighter, more efficient, funnel-shaped maneuvering around obstacles.

6.5 Wind-Adaptive Residual Correction and Convex Half-Space Integration

6.5.1 Convex Half-Space Integration for Directness

To mitigate excessive deviation and fuel cost caused by wide detours around obstacles, we introduce convex half-space constraints that encourage the quadrotor to follow the reference trajectory more directly.

Soft Half-Space Constraints:

Along the nominal path, a sequence of convex half-spaces is defined:

$$a_k^\top x_k \leq b_k + \varepsilon_k$$

where a_k and b_k define the separating plane, and $\varepsilon_k \geq 0$ is a slack variable penalized in the cost function.

This inequality is convex because the left-hand side, $a_k^\top x_k$, is an affine function of x_k (since a_k is a fixed vector). Affine functions are both convex and concave. The right-hand side, $b_k + \varepsilon_k$,

is also affine in ε_k, with b_k and ε_k treated as constants or decision variables depending on the context. The complete inequality compares one affine function to another, which defines a convex constraint. Therefore, the set of (x_k, ε_k) satisfying this inequality forms a convex set.

If we're trying to define a wall at $y = 2$, we use the half-space constraint as:

$$y_k \leq 2$$

In vector form:
Let the state vector be $x_k = [x,\, y,\, z,\, \dots]^\top \in \mathbb{R}^n$.
To extract the y-component, define a vector $a_k \in \mathbb{R}^n$ as:

$$a_k = [0,\ 1,\ 0,\ \dots,\ 0]^\top \quad \text{(1 in the y-position)}$$

Set the boundary value:

$$b_k = 2$$

Which ensures:

$$a_k^\top x_k = y_k \leq 2 + \varepsilon_k$$

Here, $\varepsilon_k \geq 0$ is a slack variable that allows for soft constraint violation and is penalized in the cost function to discourage violation while maintaining feasibility.

Cost Shaping: To enforce these half-spaces as soft constraints, we add a penalty in the objective:

$$J \leftarrow J + w_\varepsilon \sum_k \varepsilon_k$$

This encourages the optimizer to minimize deviation from the desired corridor without rendering the problem infeasible.

This mechanism discourages excessive lateral displacement and promotes more efficient paths through tight gaps or narrow obstacle corridors. The result is reduced fuel usage and improved trajectory compactness.

With the introduction of the soft half-space constraint, we were able to reduce the total fuel cost of our MPC-styled, iteratively solved robust SCP algorithm from 226.47 to 215.72, as cen be seen on figure 5. This improvement is primarily due to shortening the length of the drone's path within the same time horizon—requiring lower velocities and accelerations, which in turn reduces the overall thrust and fuel consumption. Importantly, the drone still avoids all obstacles; it simply takes tighter, less conservative turns rather than wide arcs. But even with the integration of convex half-space constraints near obstacles, our MPC-style SCP algorithm is unable to consistently guide the drone through narrow gaps between obstacles to minimize travel distance and fuel consumption.

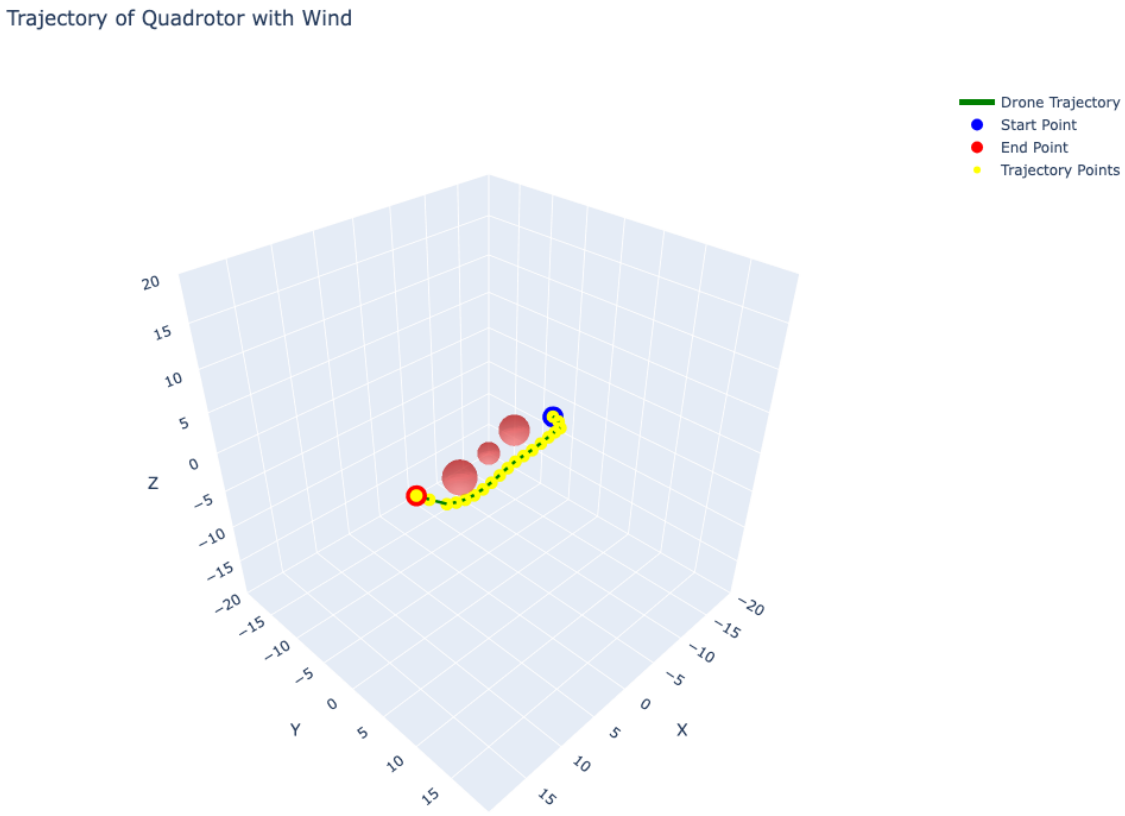

Figure 5: Trajectory of the quadcopter under constant wind using SCP solved iteratively in an MPC-like fashion at each node, with soft half-space constraints applied to a wall located at $y = 2.0$. Total fuel cost for the stacked trajectory is 215.72. Total obstacle violation: 0.000.

6.5.2 Wind-Adaptive Residual Correction

To enhance robustness at a lower level within the MPC-style Sequential Convex Programming (SCP) framework, we introduce the *Wind-Adaptive Residual Correction* technique. This method adaptively estimates and corrects for wind-induced deviations during flight, without requiring explicit modeling of wind in the dynamics.

At each node, after the control input u_k is applied, we record the quadrotor's actual position using onboard sensors. We compare this with the predicted (wind-free) position from the nominal trajectory:

$$\Delta x_k^{\text{wind}} = x_k^{\text{actual}} - x_k^{\text{nominal}}$$

This deviation represents the net effect of the wind at timestep k.

We compute the slope or rate of change of the wind effect as:

$$\text{slope}_k = \frac{\Delta x_k^{\text{wind}} - \Delta x_{k-1}^{\text{wind}}}{\Delta t}$$

This slope captures how the wind-induced deviation is evolving.

We modify the residual term z_k in the First-Order Hold (FOH) discretization:

$$x_{k+1} = A_k x_k + B_k u_k + B_k^+ u_{k+1} + z_k$$

The residual z_k is updated by the previous time steps to include a correction based on the observed wind slope:

$$z_k \leftarrow z_k + \Delta x_k^{\text{wind}}$$

This adjustment allows the system to implicitly compensate for wind without altering the underlying dynamics or re-linearizing the system at each step.

This wind-informed z_k is updated at every MPC step. The SCP solver then uses the adjusted linear dynamics to replan the trajectory, leading to better alignment with the reference path under wind perturbations.

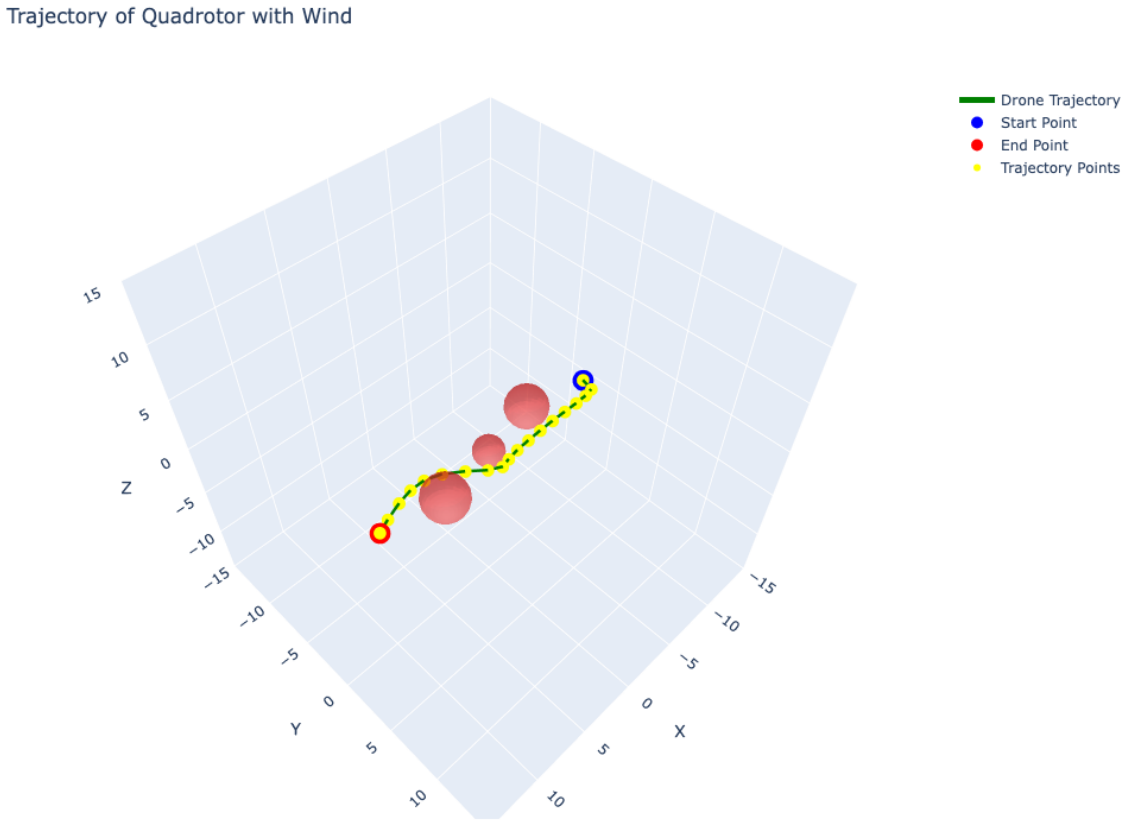

Figure 6: Trajectory of the quadcopter under constant wind using SCP solved iteratively in an MPC-like fashion at each node, with wind adaptive residual correction applied. Total fuel cost for the stacked trajectory is 214.61. Total obstacle violation: 0.000.

With the wind-adaptive residual correction (WARC) applied under a constant wind disturbance, we observed a fuel cost only slightly higher than in the nominal case without wind—without hitting any obstacles as can be seen on figure 6. The trajectory passed precisely between the obstacles, which explains the low fuel consumption. At first glance, this result might appear promising and suggest that we have achieved our goal. However, there is an important caveat.

This approach worked well only because the wind was constant and represented a known worst-case scenario—specifically, a constant wind vector of $[0, 1, 0]$ throughout the entire trajectory. After the first iteration of our MPC loop, the SCP dynamics effectively learned the wind's direction and magnitude from the previous slope. Since the wind remained unchanged, the algorithm had perfect knowledge of it for all subsequent nodes. As a result, it solved the remainder of the trajectory nearly perfectly, with only a transient spike between the first two nodes.

However, if the wind were non-constant or, worse, changed direction at each node, this WARC technique would introduce more problems than it solves. Because our system is completely unaware of future wind behavior, we do not consider this approach a viable or robust solution. The WARC method relies on a strong assumption: the wind remains constant over time. This assumption is unrealistic in our case, where wind conditions are uncertain and potentially time-varying. Therefore, we now turn to alternative techniques that can offer robustness against unknown and time-varying wind disturbances.

6.6 Two-Stage Tracking-Style Robust SCP Approach

The previous wind-robustified approach, while successful in avoiding obstacles, resulted in excessive fuel consumption due to wide turns around obstacle clusters. To address this, we introduce a more efficient method that combines SCP with a tracking-style correction strategy.

In this new approach, we first solve the entire trajectory once using Sequential Convex Programming (SCP) under nominal (wind-free) conditions. This yields an optimal reference path that typically threads efficiently between obstacles.

Then, as a second stage, we iteratively solve smaller trajectory segments between successive control nodes from the original SCP solution. At each iteration, the drone's current state (possibly affected by wind) becomes the new starting point, and the next control node from the nominal path becomes the temporary target. This local replanning continues step-by-step until the drone reaches the final goal state at $(10, 0, 0)$.

Tracking-Style SCP with Stepwise Local Replanning Algorithm:

1. **Step 1: Solve Nominal Full-Horizon Trajectory**
   - Use SCP to compute a full trajectory $\{x_0, x_1, \ldots, x_N\}$ and control sequence $\{u_0, u_1, \ldots, u_{N-1}\}$ under wind-free dynamics.
   - This solution will serve as the nominal reference path for tracking.
2. **Step 2: Initialize Tracking**
   - Set the current state $x_{\text{current}} \leftarrow x_0$ (initial condition).
   - Set iteration index $k = 0$.
3. **Step 3: Iterative Local Replanning**
   - **While** $k < N$ and goal not reached:
     (a) Define the next target state as $x_{\text{target}} \leftarrow x_{k+1}^{\text{ref}}$ from the nominal solution.

(b) Solve a short-horizon SCP subproblem from x_{current} to x_{target}, over a short horizon (e.g., 1 step or few nodes).

(c) Apply or simulate the first control step of the solution.

(d) Update the current state x_{current} to the resulting state (wind-perturbed).

(e) Increment index: $k \leftarrow k + 1$.

4. **Step 4: Output**

- Concatenate all locally optimized segments to form the final trajectory and control sequence.

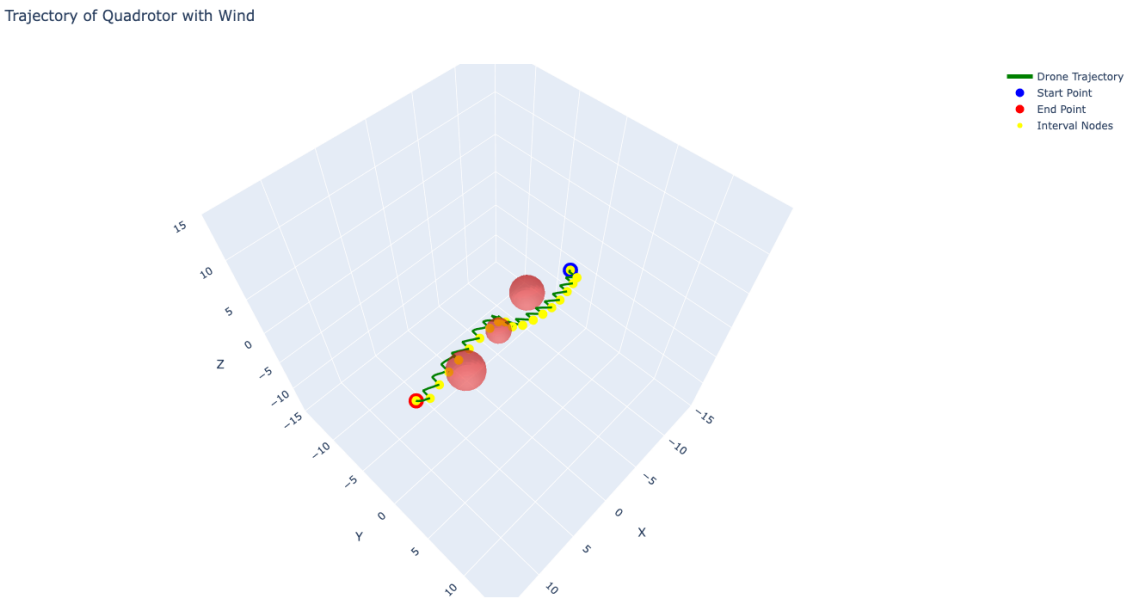

Figure 7: Trajectory of the quadcopter under constant wind, using a precomputed nominal trajectory. At each node, wind is applied, and a subproblem SCP is solved to steer the drone toward the next nominal node, resulting in a total of 20 sequential subproblems. Total fuel cost for the stacked trajectory is 214.60 Total obstacle violation: 4.978.

This method significantly reduces fuel consumption compared to the previous robust MPC-style approach. Notably, the drone is once again able to navigate through narrow gaps between obstacles, closely resembling the behavior observed in the wind-free scenario as can be seen on figure 7 below. However, this tighter maneuvering introduces a trade-off: a small number of node violations are observed, reflected in the slight zig-zagging of the trajectory—an expected outcome under these conditions.

These violations arise because, although the trajectory is more fuel-efficient, the drone travels closer to obstacles. In regions with strong wind disturbances, this proximity increases the risk of collision. Thus, although the method improves efficiency, it also heightens vulnerability in tightly constrained environments under disturbance. Managing this trade-off is crucial for safe and reliable trajectory design in real-world deployments. While using a finer step size between nodes could

help the drone react more quickly and avoid obstacles, this approach is computationally expensive. Moreover, since we have already assumed a fixed sensing and control update rate based on the drone's onboard capabilities, alternative strategies must be considered for improving safety without exceeding computational or real-time constraints.

6.7 Safety Margin in Tracking-Style Approach

While the tracking-style approach offered significantly improved fuel efficiency, it was still prone to occasional node violations when the drone attempted to pass through very tight spaces. These violations were due to the fact that, under strong wind disturbances, even small deviations could push the drone into obstacle boundaries.

To mitigate this, a safety margin of 1 unit was introduced around each obstacle. This margin corresponds to the maximum norm of the wind disturbance vector, effectively modeling the worst-case deviation that wind could cause in any direction.

By incorporating this safety margin into the obstacle avoidance constraints, the drone is guaranteed to maintain a buffer zone around each obstacle. As a result, even in the presence of the worst-case wind scenario, the drone cannot collide with obstacles. This technique ensures zero node violations without requiring complex online disturbance estimation or adaptation.

Although the safety margin causes the drone to follow a slightly longer trajectory—since it must steer farther from obstacles—it still achieves a much more direct path than previous conservative approaches, creating a favorable balance between safety and fuel efficiency. By further decreasing the node time step length—at the expense of increased computation time—we can obtain even more robust solutions. This finer temporal resolution allows for more accurate modeling of system dynamics and disturbances, and enables the use of more refined funnel scalings and safety margins, ultimately leading to more optimal and reliable trajectory optimization results.

7 Funnel Synthesis

As a final layer of robustness, we implement a funnel synthesis framework based on the optimization formulation introduced in Taewan Kim's work ([1]). The goal is to compute ellipsoidal sets or funnel along the precomputed trajectory obtained from our sequential convex programming (SCP)-based obstacle avoidance solution.

Funnel synthesis provides a formal measure of robustness by computing time-varying ellipsoids:

$$\mathcal{E}_k = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; (x - \bar{x}_k)^\top Q_k^{-1} (x - \bar{x}_k) \le c_k \right\}$$

These set of ellipsoids represent regions of admissible deviation around the nominal trajectory $\bar{x}_k$, within which the drone can safely operate without violating obstacle constraints or dynamic limits.

In our implementation, we:

- Transcribe the continuous-time funnel synthesis problem into a discrete-time convex program
- Use finite-difference approximations to model $\dot{Q}(t)$
- Inject worst-case wind disturbances into the nominal trajectory

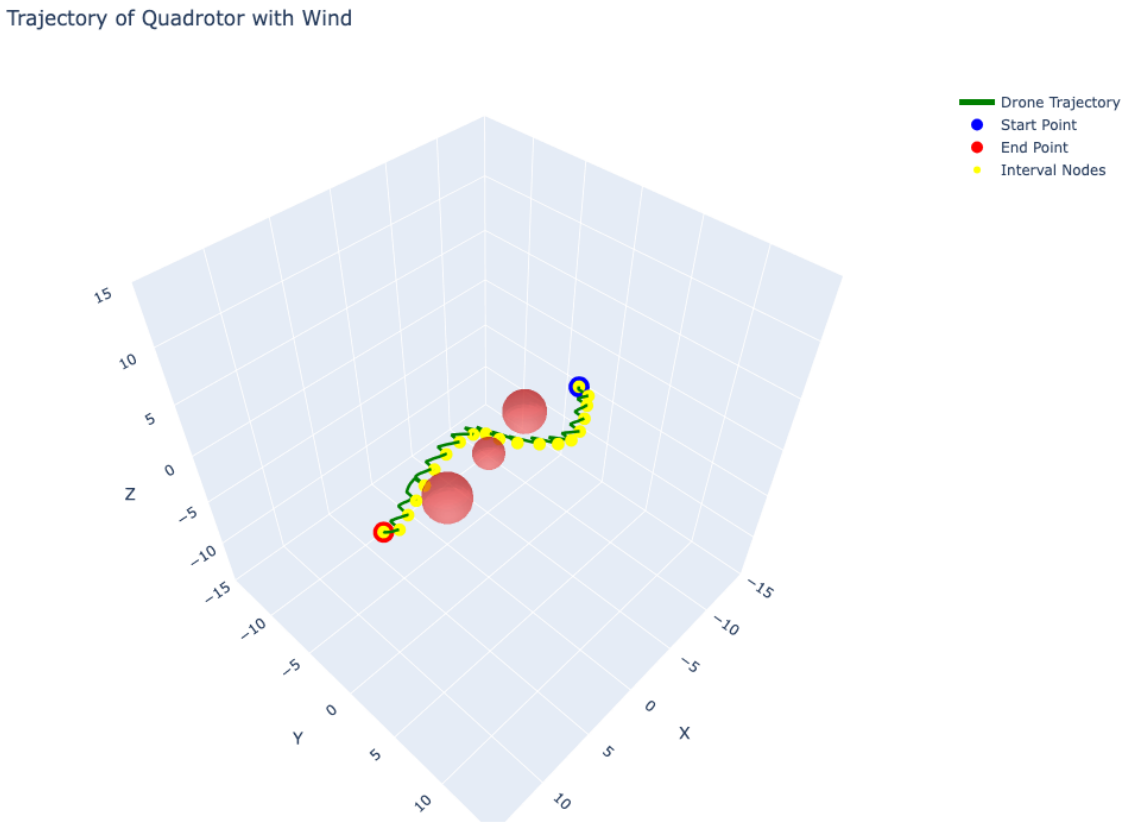

Figure 8: Trajectory of the quadcopter under constant wind, using a precomputed nominal trajectory. At each node, wind is applied, and a subproblem SCP is solved to steer the drone toward the next nominal node, resulting in a total of 20 sequential subproblems. Safety margin of 1 added around each obstacle. Total fuel cost for the stacked trajectory is 215.81. Total obstacle violation: 0.0.

- Encode Lipschitz nonlinearity and disturbance bounds via matrix inequalities and slack variables
- Solve the resulting convex optimization problem using the MOSEK solver
- Plot and visualize the ellipsoid set, funnel, around our nominal trajectory we were dealing with.

This approach allows us to robustly quantify the system's tolerance to state deviations caused by model uncertainty or external disturbances. By visualizing these sequence of ellipsoids indexed over time or funnel in short, we can see how much deviation the drone can safely tolerate while still avoiding obstacles and respecting system dynamics. This robustification technique serves as the final safety envelope around the previously optimized path.

7.1 Overview of Taewan Kim's Continuous-Time Funnel Synthesis

The funnel synthesis framework proposed by Taewan Kim [1] formulates a robust, convex optimization problem aimed at constructing sequence of ellipsoids indexed over time—referred to as *funnel*—around a nominal trajectory. These ellipsoidal set or funnel capture admissible state deviations due to disturbances and model nonlinearities, while ensuring that the system remains safe and within specified constraints.

The central goal is to compute a time-varying ellipsoidal set

$$E(t) := \left\{ \eta \in \mathbb{R}^{n_x} \,\middle|\, \eta^\top Q(t)^{-1} \eta \leq 1 \right\},$$

such that for all time $t \in [t_0, t_f]$, the system state deviation $\eta(t) := x(t) - \bar{x}(t)$ remains within this tube, even in the presence of bounded external disturbances and nonlinear modeling errors. Here, $Q(t) \in \mathbb{S}^{n_x}_{++}$ is a time-varying, positive definite matrix that defines the shape and orientation of the ellipsoidal set, while $\eta(t)$ represents the deviation of the actual system state $x(t)$ from the precomputed nominal trajectory $\bar{x}(t)$. The ellipsoid $E(t)$ serves as a certificate of robustness.

7.1.1 System Dynamics

Consider the following continuous-time nonlinear system:

$$\dot{x}(t) = f(t, x(t), u(t), w(t)), \quad t \in [t_0, t_f],$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is the control input, and $w(t) \in \mathbb{R}^{n_w}$ represents a bounded disturbance such that

$$\|w(\cdot)\|_\infty := \sup_{t \in [t_0, t_f]} \|w(t)\| \leq 1,$$

and t_0, t_f are the initial and final time, respectively.

We can linearize the nonlinear system around a nominal trajectory $(\bar{x}(t), \bar{u}(t), \bar{w}(t))$ to obtain a linear time-varying (LTV) system with a nonlinear remainder term. This results in the following Lur'e-type representation:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) + F(t)w(t) + Ep(t), \\ p(t) &= \phi(t, q(t)), \quad q(t) = Cx(t) + Du(t) + Gw(t), \end{aligned} \tag{2}$$

where:

- $p(t) \in \mathbb{R}^{n_p}$ is a lumped nonlinearity represented by the nonlinear function $\phi(t, \cdot)$,
- $q(t) \in \mathbb{R}^{n_q}$ is the input to the nonlinearity,
- $A(t)$, $B(t)$, and $F(t)$ are the time-varying Jacobian matrices (first-order approximations) of the nonlinear dynamics evaluated along the nominal trajectory,
- $E \in \mathbb{R}^{n_x \times n_p}$, $C \in \mathbb{R}^{n_q \times n_x}$, $D \in \mathbb{R}^{n_q \times n_u}$, and $G \in \mathbb{R}^{n_q \times n_w}$ are constant selector matrices.

7.1.2 Deviation Dynamics

We begin by defining the deviation variables:

- $\eta(t) := x(t) - \bar{x}(t)$: deviation of actual state from nominal
- $\xi(t) := u(t) - \bar{u}(t)$: deviation of actual control input
- $q(t) := Cx(t) + Du(t)$: input to a nonlinear function $\phi(t, q)$

From the full nonlinear dynamics $\dot{x}(t) = f(t, x, u, w)$, we define the deviation dynamics:

$$\dot{\eta}(t) = f(t, x, u, w) - f(t, \bar{x}, \bar{u}, 0).$$

Under linearization and Lipschitz assumptions, this becomes:

$$\dot{\eta}(t) = A(t)\eta(t) + B(t)\xi(t) + F(t)w(t) + E(t)\delta p(t),$$

where:

- $A(t), B(t)$: Jacobians of f with respect to x, u
- $F(t)$: models how disturbances $w(t) \in \mathbb{R}^{n_x}$ enter the dynamics
- $E(t)$: models sensitivity to nonlinearities
- $\delta p(t) := \phi(t, q(t)) - \phi(t, \bar{q}(t))$: nonlinear residual error

The signal $q(t)$ is defined as a linear combination of the system state, control input, and disturbance:

$$q(t) = Cx(t) + Du(t) + Gw(t),$$

where C, D, G are selector matrices that extract and scale the components of $x(t)$, $u(t)$, and $w(t)$ relevant to the system's nonlinearities. The resulting vector $q(t) \in \mathbb{R}^{n_q}$ serves as the input to the nonlinear function $\phi(t, q(t))$, which captures the remaining nonlinear dynamics.

Let $\bar{x}(t)$ and $\bar{u}(t)$ denote the nominal state and control trajectories. Define the deviations:

$$\xi(t) := u(t) - \bar{u}(t), \quad \eta(t) := x(t) - \bar{x}(t), \quad \bar{q}(t) := C\bar{x}(t) + D\bar{u}(t).$$

Then, the deviation in the nonlinear input space is:

$$\delta q(t) := q(t) - \bar{q}(t) = C\eta(t) + D\xi(t) + Gw(t),$$

- $q(t) \in \mathbb{R}^{n_q}$: The input to the nonlinear function, typically formed as a linear combination of the state, control input, and disturbance:

  It captures the parts of the system that feed into the nonlinearity.

- $p(t) \in \mathbb{R}^{n_p}$: The output of the nonlinearity, defined as:

  $$p(t) = \phi(t, q(t))$$

  which is then fed back into the dynamics.

- To ensure robustness and facilitate convex analysis, the nonlinearity $\phi(t, \cdot)$ is assumed to be Lipschitz continuous. This leads to the following inequality:

$$\|\delta p(t)\|_2 \leq \gamma(t)\|\delta q(t)\|_2, \quad \forall t \in [t_0, t_f]$$

and $\gamma(t) > 0$ is a time-varying Lipschitz constant bounding how much the nonlinearity can amplify input deviations.

7.1.3 Closed-Loop Dynamics with Feedback Control

By applying a linear state-feedback controller $\xi(t) = K(t)\eta(t)$, the deviation dynamics become:

$$\dot{\eta}(t) = (A + BK)\eta(t) + Fw + E\delta p(t),$$

$$\delta q(t) = C\eta + D\xi + Gw.$$

$$\|\delta p\|_2 \leq \gamma\|\delta q\|_2, \quad \|w(\cdot)\|_\infty \leq 1$$

This shows that the deviation $\eta(t)$ evolves under linear dynamics influenced by:

- feedback control ($BK\eta$),
- additive wind disturbance (where wind is bounded) (Fw),
- and nonlinear modeling error ($E\delta p$), which itself depends on η, ξ, and w.

7.1.4 Lyapunov-Based Invariance and Objective

To ensure the deviation remains bounded inside the funnel, a time-varying Lyapunov function is defined:

$$V(t, \eta) := \eta^\top Q(t)^{-1}\eta.$$

Then the Lyapunov conditions are imposed:

$$\begin{aligned} \dot{V}(t, \eta) &\leq -\alpha V(t, \eta), \quad \text{(exponential decay)} \\ V(t, \eta) &\geq \|w(t)\|_2^2, \quad \text{(robustness to disturbance)} \\ \|\delta p(t)\|_2 &\leq \gamma(t)\|\delta q(t)\|_2. \quad \text{(Lipschitz bounded nonlinearity)} \end{aligned}$$

Here, $\gamma(t) \in \mathbb{R}_+$ is a known time-varying Lipschitz constant. This condition ensures that nonlinear residuals do not grow too fast with respect to deviations in the input space.

With the Lyapunov condition

$$\dot{V}(t, \eta) \leq -\alpha V(t, \eta),$$

where $V(t, \eta) := \eta^\top Q^{-1}(t)\eta$ and $\alpha > 0$ is a user-defined decay rate, we can guarantee that the ellipsoidal set

$$E(t) = \left\{\eta \in \mathbb{R}^{n_x} \mid \eta^\top Q^{-1}(t)\eta \leq 1\right\}$$

is invariant under the dynamics using lemmas [9, Lemma B10] and [13, Lemma 1] in Taewan's paper [1]. That is, if $\eta(t_0) \in E(t_0)$, then $\eta(t) \in E(t)$ for all $t \in [t_0, t_f]$. Moreover, the following inequality holds for the value of the Lyapunov function along trajectories:

$$V(t, \eta(t)) \leq \max\left\{e^{-\alpha(t-t_0)}V(t_0, \eta(t_0)),\ 1\right\},$$

which implies that trajectories starting within or near the boundary of $E(t_0)$ will be attracted toward the interior of $E(t)$ over time, meaning the ellipsoid is attractive.

To allow a larger set of initial conditions while still ensuring eventual contraction to $E(t)$, we define a new, larger ellipsoidal set, the invariant set:

$$E_c(t) := \left\{\eta \in \mathbb{R}^{n_x} \mid \eta^\top Q^{-1}(t)\eta \leq \frac{1}{c(t)}\right\},$$

With the ellipsoid $E_c(t)$ having $1/c(t)$ as the support value, the invariance property of $E_c(t)$ is established in Lemma 2 of Taewan Kim's paper [1].

$$E_c(t) = \left\{x \in \mathbb{R}^{n_x} \,\middle|\, (x - \bar{x}(t))^\top Q(t)^{-1}(x - \bar{x}(t)) \leq \frac{1}{c(t)}\right\}$$

Where $c(t) \in (0, 1]$ is a continuous, scalar-valued function representing the inverse of the ellipsoid's support level. This is the actual ellipsoidal funnel in state space. It is centered at the nominal trajectory point $\bar{x}(t)$, and scaled by $c(t)$, representing the allowable deviation radius under bounded disturbances. Any state $x(t)$ within $\mathcal{E}(t)$ is guaranteed to remain safe.

To guarantee that $E_c(t)$ is also invariant (i.e., any state $\eta(t_0) \in E_c(t_0)$ remains inside $E_c(t)$ for all future t), we impose the following condition on $c(t)$:

$$\frac{1}{c(t)} \geq \max\left\{1,\ e^{-\alpha(t-t_0)} \cdot \frac{1}{c(t_0)}\right\},$$

with the constraint $0 < c(t_0) \leq 1$. This ensures that the outer funnel $E_c(t)$ shrinks at a rate compatible with the exponential decay in the Lyapunov function. The ellipsoid must not shrink faster than the exponential, It also must not go below 1 thus preserving invariance.

This all means Any solution $\eta(\cdot)$ of the closed-loop system starting at $E_c(t_0)$ remains in the state funnel $E_c(t)$ for all $t \in [t_0, t_f]$ due to the invariance condition. That is,

$$\eta(t_0) \in E_c(t_0) \quad \Rightarrow \quad \eta(t) \in E_c(t), \quad \forall t \in [t_0, t_f].$$

Interpretation:

- $E(t)$: The *attractive funnel*—a minimal ellipsoidal tube where the state is guaranteed to converge over time.

- $E_c(t)$: The *invariant funnel*—a larger ellipsoidal tube which captures the reachable region from an initial deviation, and guarantees convergence into $E(t)$ as time evolves.

- The function $c(t)$ controls how much larger $E_c(t)$ is compared to $E(t)$, while still preserving invariance.

This guarantees that the ellipsoidal set E(t) is forward invariant under the closed-loop system: if $\eta(t_0) \in E(t_0)$, then $\eta(t) \in E(t)$ for all $t \in [t_0, t_f]$.

7.1.5 Goal of the Optimization

The goal of the continuous-time funnel synthesis problem is to find variables $Q(t)$, $Y(t) := K(t)Q(t)$, $c(t)$, $\nu(t)$, and $v^Q(t)$, such that:

- The Lyapunov function $V(t, \eta)$ decays over time
- The tube $E(t)$ accounts for worst-case disturbances and nonlinearities
- Physical constraints (obstacle avoidance, actuator limits) are satisfied
- And the size of the funnel is as large as possible, especially at t_0

Where each variable has the following meaning:

- $Q(t) \in \mathbb{S}^{n_x}_{++}$: Symmetric positive-definite matrix defining the shape of the ellipsoidal tube.
- $K(t) \in \mathbb{R}^{n_u \times n_x}$: Linear state-feedback gain used in the controller $u(t) = \bar{u}(t) + K(t)(x(t) - \bar{x}(t))$.
- $Y(t) := K(t)Q(t)$: Decision variable used in convex formulation instead of $K(t)$ directly.
- $c(t) \in \mathbb{R}_+$: A scalar scaling factor that determines the size of the ellipsoidal funnel at time t.
- $\nu(t) \geq 0$: Slack variable for bounding the Lipschitz nonlinearity in the Lyapunov matrix inequality.
- $v^Q(t) \geq 0$: Scalar upper bound on the eigenvalues of $Q(t)$, enforcing ellipsoid size limits across time.

In the next section, we formulate this infinite-dimensional continuous-time problem into a tractable finite-dimensional discretized convex program that can be solved numerically, ensuring that the Lyapunov condition holds for the constructed funnel, the ellipsoid $E(t)$ in (6) is invariant and attractive, and the outer funnel $E_c(t)$ in remains invariant.

7.1.6 Objective Function

The proposed convex objective is:

$$\min_{Q(t),Y(t),c(t),\nu(t),v^Q(t)} \left[w_c \cdot c(t_0) - w_{Q_0} \log\det Q(t_0) + \int_{t_0}^{t_f} \bar{w}_Q \cdot v^Q(t)\, dt \right]$$

This cost function balances competing goals in funnel synthesis. Each term contributes as follows:

- $c(t_0)$: The initial scaling factor of the ellipsoidal tube. A larger value implies the system can tolerate more initial deviation from the nominal trajectory. The term $w_c \cdot c(t_0)$ encourages robustness at the start of the trajectory.
- $\log\det Q(t_0)$: Represents the volume of the initial ellipsoid. Since $\det(Q)$ increases with the ellipsoid size, minimizing $-\log\det Q(t_0)$ discourages overly large initial funnels that would otherwise degrade tight tracking performance. The coefficient w_{Q_0} balances this size penalty against the robustness gain from $c(t_0)$.

- $v^Q(t)$: A scalar upper bound on the eigenvalues of the positive definite matrix $Q(t)$, satisfying $Q(t) \preceq v^Q(t)I$. This enforces shape control on the ellipsoid at each time t. The integral term $\int_{t_0}^{t_f} \bar{w}_Q \cdot v^Q(t)\, dt$ encourages ellipsoids to remain small across the entire time horizon, tightening the funnel over time.

- $w_c, w_{Q_0}, \bar{w}_Q$: Tunable weights that control the trade-off between initial robustness (large $c(t_0)$), ellipsoid compactness (small $Q(t_0)$), and uniform tightness of funnels over time (small $v^Q(t)$).

Overall, this objective function promotes safety and efficiency by creating tight, well-scaled ellipsoidal tubes that track the nominal trajectory and reject disturbances, without being overly conservative.

7.1.7 Constraints

The optimization is subject to the following convex constraints:

1. **Funnel Decay (9):**

$$c(t) \geq \max\left(1, \frac{e^{-\alpha(t-t_0)}}{c(t_0)}\right)$$

   This constraint enforces an exponential decay condition on the funnel scaling factor $c(t)$ over time.

   Where:

   - $\alpha \in \mathbb{R}_+$: The Lyapunov decay rate. A larger α forces the funnel to shrink faster.
   - $c(t_0)$: The funnel size at the initial time t_0.
   - The inequality ensures that the size of the funnel at time t does not shrink arbitrarily fast, and remains large enough to satisfy exponential convergence guarantees based on the chosen decay rate α.
   - The use of $\max(1, \cdot)$ ensures the funnel does not become smaller than a unit baseline, providing numerical stability and a minimum size threshold.

   In discrete time (as implemented), this becomes:

$$e^{-\alpha(t_k - t_0)} \cdot c_k \leq c_0 \quad \text{for each time step } k$$

   which ensures that the scaled Lyapunov function decreases over time, reflecting stability and robustness.

2. **Lyapunov Matrix Inequality (11):**

$$H(t) := \begin{bmatrix} M(t) - \dot{Q}(t) & \nu(t)E^\top & F^\top & (CQ(t) + DY(t))^\top \\ \nu(t)E & -\nu(t)I & 0 & 0 \\ F & 0 & -\lambda_w I & 0 \\ CQ(t) + DY(t) & 0 & 0 & -\frac{\nu(t)}{\gamma^2} I \end{bmatrix} \preceq 0$$

where $M(t) = A(t)Q(t) + Q(t)A(t)^\top + B(t)Y(t) + Y(t)^\top B(t)^\top + \alpha Q(t) + \lambda_w Q(t)$.

This is the central robust Lyapunov constraint in Taewan Kim's funnel synthesis framework. It guarantees that the Lyapunov function:

$$V(x - \bar{x}(t)) = (x - \bar{x}(t))^\top Q(t)^{-1}(x - \bar{x}(t))$$

decays over time, even under Lipschitz nonlinearities and bounded disturbances. for the closed-loop system with the feedback gain defined as $K(t) = Y(t)Q^{-1}(t)$.Thus, when $Q(t)$ and $K(t)$ satisfy this differential linear matrix inequality (DLMI), the ellipsoid $E(t)$ in (is invariant and attractive, and the outer funnel $E_c(t)$ in is invariant by lemmas 1 and 2 given in Taewan's paper.

Where:

- $\dot{Q}(t)$: Time derivative of $Q(t)$, approximated by finite difference in discretized implementation.
- $\lambda_w \in \mathbb{R}_+$: Slack variable representing the effect of worst-case disturbance energy.
- γ: Known Lipschitz constant of the system's nonlinearities.
- E, F, C, D: Structure matrices modeling how the nonlinear dynamics and disturbances influence the system. Typically:

$$E = F = C = I, \quad D = 0$$

- The upper-left block $M(t) - \dot{Q}(t)$ captures Lyapunov decay plus extra margin from control and disturbance effects.
- The full block matrix ensures a dissipation inequality that robustly contains nonlinearities and disturbances within the funnel defined by $Q(t), c(t)$.

Purpose: Enforcing $H(t) \preceq 0$ ensures that for all deviations $x - \bar{x}(t)$ inside the funnel, the worst-case evolution of the Lyapunov function is non-increasing—guaranteeing robust forward invariance of the trajectory tube.

3. **Obstacle Avoidance (12):**

$$\begin{bmatrix} (b_i^h - (a_i^h)^\top \bar{x})^2 c & (a_i^h)^\top Q \\ Q a_i^h & Q \end{bmatrix} \succeq 0$$

To ensure that the funnel remains collision-free, Taewan Kim proposes a semidefinite condition that guarantees the ellipsoidal tube does not intersect with predefined obstacle boundaries. This is formalized using the Schur Complement.

Where:

- $\bar{x} \in \mathbb{R}^{n_x}$ is the center of the ellipsoidal funnel at a given time step (i.e., the nominal state).

- $a_i^h \in \mathbb{R}^{n_x}$, $b_i^h \in \mathbb{R}$: Vectors and scalars that define the half-space representation of an obstacle constraint of the form:
$$(a_i^h)^\top x \leq b_i^h$$
The superscript h in a_i^h, b_i^h stands for "half-space." These parameters describe a hyperplane that bounds a convex obstacle region. By using a_i^h and b_i^h, we linearize the otherwise highly nonlinear obstacle avoidance constraints, representing them as half-space inequalities. The convex approximation shown below for all obstacle avoidance highly nonlinear constraints ensures that the optimization problem remains both feasible and tractable.

$$\mathcal{P}_x = \left\{ x \in \mathbb{R}^{n_x} \,\middle|\, (a_i^h)^\top x \leq b_i^h, \quad i = 1, \ldots, m_x \right\}$$

Now with this polyhedral constraint set $\mathcal{P}_x$ defined as: we aim to design the ellipsoidal funnel E_c in , using Q and K, such that:

$$\{\bar{x}\} \oplus E_c \subseteq \mathcal{P}_x,$$

These set containment conditions can be equivalently expressed as the following inequality for each $i = 1, \ldots, m_x$:

$$\left\| \left(\frac{Q}{c} \right)^{1/2} a_i^h \right\|_2 \leq b_i^h - (a_i^h)^\top \bar{x}.$$

Squaring both sides and applying the Schur complement, this leads to the our number 12 Linear Matrix Inequality Constraint.

$$\begin{bmatrix} (b_i^h - (a_i^h)^\top \bar{x})^2 & (a_i^h)^\top Q \\ Q a_i^h & Q \end{bmatrix} \succeq 0, \quad i = 1, \ldots, m_x.$$

This constraint ensures that the ellipsoidal set

$$\mathcal{E}_k := \left\{ x \in \mathbb{R}^{n_x} \mid (x - \bar{x})^\top Q^{-1} (x - \bar{x}) \leq \frac{1}{c} \right\}$$

remains entirely within the safe side of the halfspace $a_i^{h\top} x \leq b_i^h$.

Geometrically, the condition above verifies that the entire ellipsoid is contained within the halfspace by ensuring that the signed distance from the ellipsoid center $\bar{x}$ to the boundary $a_i^{h\top} x = b_i^h$ is strictly greater than the maximum extent of the ellipsoid in the direction of a_i^h. This is mathematically guaranteed by the semidefinite condition through the S-procedure.

The use of this formulation allows for efficient incorporation of obstacle avoidance into the convex optimization problem, since the inequality is convex with respect to the decision variables Q and c, and can be checked using semidefinite programming tools.

4. **Control Bounding (13):**

$$\begin{bmatrix} (b_j^g - (a_j^g)^\top \bar{u})^2 c & (a_j^g)^\top Y^\top \\ Y a_j^g & Q \end{bmatrix} \succeq 0$$

This Linear Matrix Inequality (LMI) ensures that the feedback control input remains within a predefined admissible set for all possible state deviations within the ellipsoidal funnel.

Where:

- $\bar{u} \in \mathbb{R}^{n_u}$: Nominal control input at time step k, precomputed from a separate trajectory optimization (e.g., SCP).
- $a_j^g \in \mathbb{R}^{n_u}$, $b_j^g \in \mathbb{R}$: Vectors and scalars defining linear input constraints on the control inputs, such that:

  $$(a_j^g)^\top u \leq b_j^g$$

  The superscript g in a_j^g, b_j^g refers to "gain" or "input" bounds. These represent actuator or input safety limits (e.g., maximum thrust, torque, or voltage).

By using a_j^g and b_j^g, we linearize the otherwise highly nonlinear feasibility constraints, representing them as half-space inequalities. The convex approximation shown below for all control-related highly nonlinear constraints ensures that the optimization problem remains both feasible and tractable.

$$\mathcal{P}_u = \left\{ u \in \mathbb{R}^{n_u} \,\middle|\, (a_j^g)^\top u \leq b_j^g, \quad j = 1, \ldots, m_u \right\}$$

Now with the defined polyhedral admissible control input set $\mathcal{P}_u$, To ensure that the ellipsoidal input deviation funnel E_u remains within $\mathcal{P}_u$, we require:

$$\left\| \left(K \frac{Q}{c} K^\top \right)^{1/2} a_j^g \right\|_2 \leq b_j^g - (a_j^g)^\top \bar{u}, \quad j = 1, \ldots, m_u.$$

This can be rewritten using the matrix variable $Y = KQ$, resulting in:

$$\left\| \left(\frac{Y Q^{-1} Y^\top}{c} \right)^{1/2} a_j^g \right\|_2 \leq b_j^g - (a_j^g)^\top \bar{u}.$$

Squaring both sides and applying the Schur complement leads to our number 13 constraint Linear Matrix Inequality:

$$\begin{bmatrix} (b_j^g - (a_j^g)^\top \bar{u})^2 & (a_j^g)^\top Y^\top \\ Y a_j^g & Q \end{bmatrix} \succeq 0, \quad j = 1, \ldots, m_u.$$

5. **Ellipsoid Size Bound (15):**

$$Q(t) \preceq v^Q(t) I$$

This constraint bounds the size of the ellipsoidal funnel by enforcing that the symmetric positive definite matrix $Q(t) \in \mathbb{S}^{n_x}_{++}$, which defines the shape of the ellipsoid, is less than or equal to a scalar multiple of the identity matrix. Specifically:

- $v^Q(t)$: a scalar variable representing the maximum eigenvalue bound on $Q(t)$. This serves as a control knob to keep the ellipsoid from expanding uncontrollably in any direction.
- I: the identity matrix of appropriate dimension $n_x \times n_x$, providing an isotropic bounding shape.

Geometrically, this condition ensures that all eigenvalues of $Q(t)$ are less than or equal to $v^Q(t)$, so the ellipsoid remains compact. Without this bound, the optimization could produce overly large ellipsoids that may no longer represent safe deviation regions under bounded disturbances. This constraint is crucial for maintaining a well-scaled, safe funnel around the nominal trajectory.

6. **Boundary Conditions (16d-e):**

$$Q(t_0) \succeq c(t_0) Q_i, \quad Q(t_f) \preceq c(t_f) Q_f$$

These constraints ensure that the ellipsoidal funnel:

- Starts large enough to cover a known initial set of uncertainties, and
- Ends small enough to fit inside a desired terminal goal region.

Where:

- Q_i, $Q_f \in \mathbb{S}^{n_x}_{++}$: Fixed positive-definite matrices that define the desired shape of the initial and final uncertainty sets.
- $Q(t_0) \succeq c(t_0) Q_i$: Ensures that the initial funnel contains the initial uncertainty set $\{x \mid (x - \bar{x}_0)^\top Q_i^{-1} (x - \bar{x}_0) \leq 1\}$. The funnel must be large enough at the start.
- $Q(t_f) \preceq c(t_f) Q_f$: Ensures that the terminal funnel is fully contained within a final goal set $\{x \mid (x - \bar{x}_f)^\top Q_f^{-1} (x - \bar{x}_f) \leq 1\}$. The funnel must shrink appropriately by the end.

This guarantees proper initialization and goal convergence, which is especially important for sequential replanning or funnel library approaches.

7.2 Our Implementation: Discretized Funnel Synthesis for Precomputed Trajectories

First and foremost our quadrotor dynamics are Lipschitz, because they consist of smooth nonlinear functions such as thrust-to-acceleration mappings and linear velocity-position relationships, which are differentiable and bounded over the feasible domain. This allows us to use Taewan's funnel syhthesis method([1]) for our case

In our trajectory planning framework, the nominal state $\bar{x}_k$ and control $\bar{u}_k$ trajectories are already computed using a Sequential Convex Programming (SCP) approach. These trajectories minimize fuel usage while satisfying dynamics, control bounds, and obstacle avoidance constraints in a wind-free setting. We use the exact same parameters and nominal trajectories as in the no-wind setup.

To robustify this nominal plan against disturbances, we now aim to construct ellipsoidal funnels at each timestep that certify safety under bounded deviations. This is done by implementing the continuous-time funnel synthesis method proposed by Taewan Kim in a discretized form.

The original formulation is continuous in time and infinite-dimensional, making it intractable for numerical optimization directly. Therefore, we discretize the funnel synthesis problem using finite differences over N time steps. For each timestep $k \in \{0, \ldots, N-1\}$, we approximate the time derivative of the funnel shape matrix $Q(t)$ as:

$$\dot{Q}(t_k) \approx \frac{Q_{k+1} - Q_k}{\Delta t}$$

This allows us to transcribe the continuous convex constraints—such as the Lyapunov decay condition, control and state bounds, and disturbance resilience—into a series of matrix inequalities that can be handled with convex solvers like MOSEK via CVXPY.

In this discretized implementation, we retain the precomputed trajectory from SCP and focus solely on synthesizing the funnel tube around it to analyze and certify robustness to disturbances such as wind.

7.2.1 Nominal Trajectory

In this work, the nominal state $\bar{x}_k$ and control input $\bar{u}_k$ trajectories are precomputed using a separate trajectory optimizer without wind disturbances. Therefore, constraint (13), which constrains the control input set, is not included—since no control synthesis or feedback gain is optimized.

7.2.2 Choice of Objective Function

In Taewan Kim's original formulation, the objective function balances three competing goals:

- Maximizing the initial funnel scale c_0
- Minimizing the volume of the initial ellipsoid via $-\log\det Q_0$
- Minimizing the cumulative upper bound on eigenvalues $v_q[k]$ over the trajectory

This leads to the composite cost:

$$\min \left\{ w_c \cdot c_0 - w_{Q_0} \cdot \log\det Q_0 + \sum_{k=0}^{N-1} \bar{w}_Q \cdot v_q[k] \right\}$$

However, in our implementation, we simplify this objective and only maximize c_0, i.e.,

$$\max \; c_0$$

The justification for this simplification is twofold:

1. **Numerical Stability:** The term $-\log\det Q_0$ introduces nonlinearity and complicates solver compatibility, often requiring special care (e.g., domain constraints or use of 'qcp=True'). By omitting this term, we maintain full convexity under DCP rules and simplify convergence.

2. **Practical Focus:** Since the precomputed trajectory already satisfies feasibility and safety margins, our primary goal is to maximize the initial tolerance of the system to disturbances (represented by c_0). The bounding of ellipsoid size (related to $v_q[k]$) is already enforced through the constraint:
$$Q_k \preceq v_q[k] \cdot I, \quad \forall k$$
thereby decoupling it from the objective.

This design choice leads to faster optimization and still provides robust, safe funnels that adapt to the modified (wind-injected) nominal trajectory.

7.2.3 Constraints Implemented

The following constraints are implemented in the discretized problem:

- **Funnel Decay (9):** $e^{-\alpha(t_k - t_0)} c_k \leq c_0$
- **Lyapunov Inequality (11):** Discretized via block matrix H_k with finite-difference $\dot{Q}_k$
- **Obstacle Avoidance (12):** Approximated for Cartesian-Aligned Bounding Ellipsoids.

  In Taewan Kim's original formulation, Constraint (12) is imposed using a semidefinite matrix inequality derived from the S-procedure, which certifies that the ellipsoidal funnel remains entirely within a given halfspace. This allows general obstacle shapes to be encoded via arbitrary hyperplanes.

  In our implementation, however, we simplify this process by focusing exclusively on axis-aligned spherical obstacles. Instead of using full semidefinite constraints, we conservatively approximate each spherical obstacle using six Cartesian-aligned halfspaces (positive and negative directions of x, y, and z). For each direction, we extract a scalar constraint based on the projection of the ellipsoid along that axis:

$$a_i^\top \bar{x}_k \pm \sqrt{\frac{a_i^\top Q_k a_i}{c_k}} \leq b_i$$

where $a_i \in \mathbb{R}^{13}$ is a unit vector aligned with the axis and b_i is derived from the obstacle center and radius.

This simplification avoids using a full matrix inequality and instead applies six convex scalar constraints per obstacle per timestep. While this limits us to spherical or axis-aligned obstacles, it significantly reduces computational complexity and retains convexity of the problem.

- **Ellipsoid Size Bound (15):** $Q_k \preceq v_q[k]I$, with lower bounds on $v_q[k]$
- **Boundary Conditions (16d-e):** $Q_0 \succeq c_0 Q_i$, $Q_N \preceq c_N Q_f$

7.2.4 Additional Constraints Added

The following additional constraints are added to improve numerical stability and realism:

- **Monotonic Shrinkage:** $c_{k+1} \leq c_k$
- **Funnel Smoothness:** $|c_{k+1} - c_k| \leq \epsilon$
- **Ellipsoid Contraction:** $Q_{k+1} \succeq \beta Q_k$ for $\beta < 1$
- **Minimum Funnel Volume:** $\text{trace}(Q_N) \geq \epsilon$

7.2.5 Why Constraint (13) is Omitted

Constraint (13) from Taewan Kim's formulation involves bounding the control input $u(t)$ via a matrix inequality:

$$\begin{bmatrix} (b_j^g - (a_j^g)^T \bar{u})^2 c & (a_j^g)^T Y^T \\ Y a_j^g & Q \end{bmatrix} \succeq 0$$

This constraint is relevant in settings where the control inputs $\bar{u}(t)$ are decision variables within the optimization. In our case, however, the control sequence $\{u_k\}_{k=0}^{N-1}$ is precomputed using a prior trajectory optimization and is treated as fixed. Since no feedback policy is being designed directly in this funnel synthesis, and Y_k appears only as a modeling term for disturbances in the Lyapunov matrix inequality, Constraint (13) is omitted in our implementation.

7.2.6 Plotting

We focus on plotting and computing the invariant funnels at each node and visualizing them in the x-y plane. Since the wind acts in the y-direction and the quadrotor primarily moves along the x-axis—from $x = -10$, $y = z = 0$ to $x = 10$, $y = z = 0$—a 2D slice of the full 3D x-y-z trajectory in the x-y plane provides meaningful insight into the effect of wind and the shape of the funnels.

7.3 Results for Funnel Synthesis Implementation on Precomputed Nominal Inputs and Outputs

The discretized convex optimization framework for funnel synthesis successfully preserves the key robustness properties of the original continuous-time formulation. By injecting worst-case wind

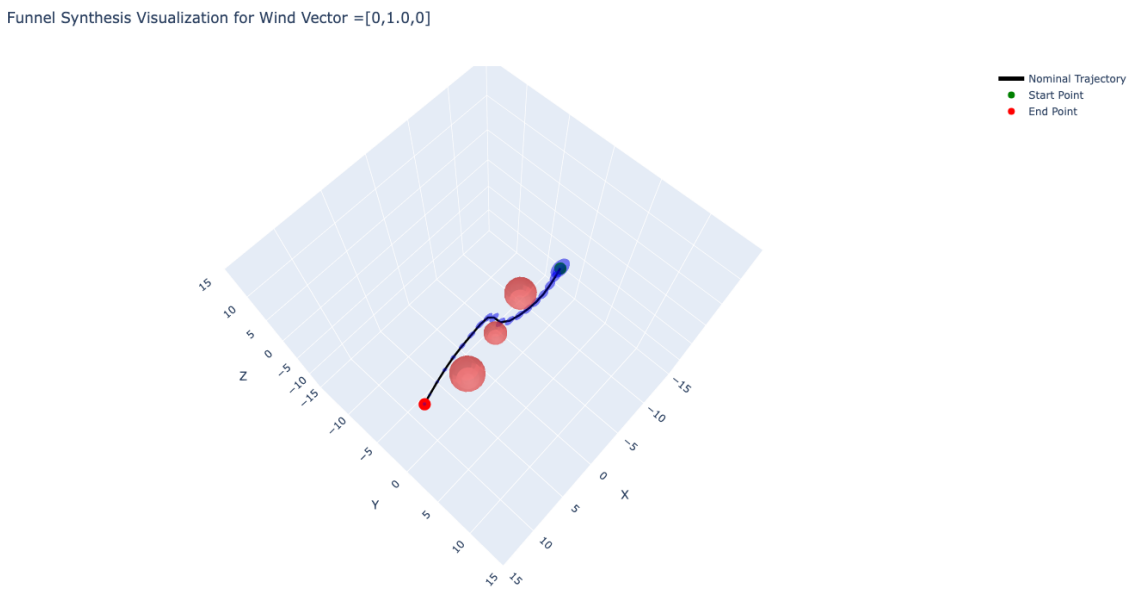

Figure 9: Nominal trajectory with invariant ellipsoids overlaid at each node, illustrating the allowable deviation from the nominal path at every time step.

perturbations into the precomputed nominal trajectory and utilizing fixed control inputs, the approach ensures conservative safety guarantees without significantly increasing the complexity of the optimization problem. This makes it an effective post-processing robustification technique for systems where the nominal trajectory has already been generated. The ellipsoids in the plot appear to shrink along the trajectory and do not intersect any obstacles.

8 Conclusions

In this work, we developed a Sequential Convex Programming (SCP) framework tailored for point-mass quadrotor trajectory optimization in cluttered environments. Inspired by existing literature, we constructed a baseline SCP algorithm capable of safely navigating through a corridor with three spherical obstacles under a fixed final time horizon.

To address real-world uncertainties, particularly constant wind disturbances, we implemented a series of robustification techniques. Each method was designed to improve fuel efficiency while minimizing obstacle violation. We began by evaluating the effect of constant wind on the nominal trajectory and observed significant drift and constraint violations due to SCP's lack of inherent robustness.

To mitigate these issues, we introduced multiple layers of robustness:

- **LQR Tracking Control**, which enables trajectory correction by applying feedback control laws based on linearized dynamics.
- **Receding-horizon SCP approach (MPC-style SCP)**, where the optimization is re-solved at each node to adapt to the wind's effect.

- **Wind-Adaptive Residual Correction** , which estimates wind drift from previous steps and incorporates it into the linearization residuals.
- **Convex Halfspace Constraints**, which discourage large deviations from the nominal path, promoting tighter, more fuel-efficient trajectories.
- **Tracking-Style Replanning**, which divides the reference trajectory into segments and replans locally with safety margins.

Finally, we integrated the *funnel synthesis framework* from Taewan Kim [1] to evaluate how far the system can deviate from the nominal path at each node while still remaining safe under disturbances. The resulting ellipsoidal funnels characterize robust invariance regions around the trajectory. By combining funnel synthesis with our previously developed techniques, we successfully ensured robust obstacle avoidance and reached the final goal state with no node violations, despite the presence of constant wind.

References

[1] T. Kim, P. Elango, T. P. Reynolds, B. Açıkmeşe, and M. Mesbahi, "Optimization-based Constrained Funnel Synthesis for Systems with Lipschitz Nonlinearities via Numerical Optimal Control," *arXiv preprint arXiv:2303.10504*, Jul. 2023. Presented at AIAA SciTech 2021. Published online: Jan. 4, 2021. Available: `https://doi.org/10.2514/6.2021-0504`

[2] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex Optimization for Trajectory Generation," *arXiv preprint arXiv:2106.09125*, Jun. 2021.

[3] M. Szmuk, C. A. Pascucci, and B. Açıkmeşe, "Real-Time Quad-Rotor Path Planning for Mobile Obstacle Avoidance Using Convex Optimization," in *Proc. of the 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018, pp. 7601–7608. doi: `https://doi.org/10.1109/IROS.2018.8594351`

[4] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açıkmeşe, "Convexification and Real-Time On-Board Optimization for Agile Quad-Rotor Maneuvering and Obstacle Avoidance," in *Proc. of the 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 4862–4868. doi: `https://doi.org/10.1109/IROS.2017.8206363`

[5] M. Szmuk and B. Açıkmeşe, "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time," in *Proc. of the 2018 AIAA Guidance, Navigation, and Control Conference*, AIAA 2018-0617, Jan. 2018. Available: `https://doi.org/10.2514/6.2018-0617`
Also available as: *arXiv preprint arXiv:1802.03827*, Feb. 2018. `https://doi.org/10.48550/arXiv.1802.03827`

[6] A. G. Kamath, P. Elango, T. Kim, S. McEowen, M. Mesbahi, and B. Açıkmeşe, "Customized Real-Time First-Order Methods for Onboard Dual Quaternion-based 6-DoF Powered-Descent Guidance," in *AIAA SciTech 2023 Forum*, AIAA 2023-2003, Jan. 2023. Available: `https://doi.org/10.2514/6.2023-2003`
Also available from ResearchGate: `https://www.researchgate.net/publication/367314617_Customized_Real-Time_First-Order_Methods_for_Onboard_Dual_Quaternion-based_6-DoF_Powered-Descent_Guidance`

[7] A. G. Kamath, P. Elango, Y. Yu, S. McEowen, G. M. Chari, J. M. Carson III, and B. Açıkmeşe, "Real-Time Sequential Conic Optimization for Multi-Phase Rocket Landing Guidance," arXiv preprint, arXiv:2212.00375 [math.OC], Dec. 2022. Available: `https://doi.org/10.48550/arXiv.2212.00375`

[8] C. R. Hayner, J. M. Carson III, B. Açıkmeşe, and K. Leung, "Continuous-Time Line-of-Sight Constrained Trajectory Planning for 6-Degree of Freedom Systems," *IEEE Robotics and Automation Letters*, pp. 1–8, 2025. doi: 10.1109/LRA.2025.3545299.

[9] T. Kim, P. Elango, and B. Açıkmeşe, "Joint Synthesis of Trajectory and Controlled Invariant Funnel for Discrete-time Systems with Locally Lipschitz Nonlinearities," *arXiv preprint arXiv:2209.03535*, Jan. 2024. Accepted to the International Journal of Robust and Nonlinear Control. Available: `https://doi.org/10.48550/arXiv.2209.03535`

I would like to acknowledge the use of OpenAI's ChatGPT, which assisted with grammar and style editing during the preparation of this thesis.