Multi-UAVs reference tracking using MPC

Mohammad Alikhani Najafabadi
Model Predictive Control
Universitat Politecnica de Catalunya
Barcelona, Spain
Mohammad.najafabadi@estudiantat.upc.edu

I. Introduction

In this project, the culminating Control Design assignment for the Model Predictive Control course, adhering to the guidelines in [1] [2] , will be presented. The objective is to consolidate the insights gained from lectures and laboratory sessions to develop and apply two distinct MPC control approaches—Linear MPC and Non-Linear MPC—for the same system. The ultimate goal is to evaluate the effectiveness of MPC in applications well-suited to this control methodology.

II. Problem Statement

The goal of this work is to be able to control three UAVs independently towards a control goal while maintaining safe distance and pitch angle to avoid collision and managing maneuvers.

A. System definition

Taking into account the complexity of the nonlinear UAV drone, we decided to chose a more simplified model for our study. The model employed will be as follows, where the state and control variables are determined by the robot's pose information, represented by x, y, and θ, along with its velocities. The control inputs are defined as two forces, each corresponding to a degree of freedom of the robot.

The quadrotor will have two inputs: motor thrusts from the left and the right motors. They cause a net force and torque applied to the quadrotor:

$$\begin{aligned} f &= f_L + f_R \\ \tau &= (f_R - f_L)L \end{aligned} \tag{1}$$

The net force and torque will be used as the two inputs to the quadrotor to simplify subsequent calculations. controls are redefined as the thrust forces produced by the propellers $\mathbf{u} = (u_1, u_2) = (f_R, f_L)$. The system has 6 states represented as: $x = \left[x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}\right]^T$, where:

- **States:**
  - x: Horizontal position.
  - $\dot{x}$: Horizontal velocity.
  - y: Vertical position.
  - $\dot{y}$: Vertical velocity.
  - θ: Pitch angle (orientation).
  - $\dot{\theta}$: Angular velocity.
- **Inputs:**
  - u_1: Thrust from the left motor.
  - u_2: Thrust from the right motor.
- **Parameters:**
  - m: Mass of the quadrotor.
  - g: Gravitational acceleration.
  - l: Distance from the center of mass to each motor.
  - I: Moment of inertia about the pitch axis.

The equations governing the system's motion are:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(u_1 + u_2)\sin(\theta) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{1}{m}(u_1 + u_2)\cos(\theta) - g \\ \dot{x}_5 &= x_6 \\ \dot{x}_6 &= \frac{L}{I_z}(u_1 - u_2) \end{aligned} \tag{2}$$

III. Controller Design

A. Linear MPC design

Assuming small heading angles and the following simplification the linearization would take place, all the assumpation were made such that keep the drone at it´s Hover mode:

Position and Velocity:

- $x = 0,\ \dot{x} = 0,\ y = 0,\ \dot{y} = 0.$

Angle and Angular Velocity:

- $\theta = 0,\ \dot{\theta} = 0.$

Thrusts: The net force must balance gravity, and the torque must be zero.

- $\theta = 0$, so $\cos(\theta) = 1$, $\sin(\theta) = 0$.
- $\dot{y} = 0$, leading to $\frac{1}{m}(u_1 + u_2)\cos(\theta) - g = 0$, hence $u_1 + u_2 = mg$.
- $\dot{\theta} = 0$, leading to $u_1 - u_2 = 0$, hence $u_1 = u_2$.
- Solving, $u_1 = u_2 = \frac{mg}{2}$.

Computing Jacobian Matrices:

- $\mathbf{A} = \left.\frac{\partial f}{\partial x}\right|_{(x_e, u_e)}, \quad \mathbf{B} = \left.\frac{\partial f}{\partial u}\right|_{(x_e, u_e)}$
- At equilibrium $x_e = [0, 0, 0, 0, 0, 0]^T$, $u_e = \left[\frac{mg}{2}, \frac{mg}{2}\right]^T$:

- $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{g}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

- $\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \\ 0 & 0 \\ \frac{L}{I} & -\frac{L}{I} \end{bmatrix}$

B. LMPC formulation

In the LMPC implementation control objectives are reference tracking and energy minimization for all three drones, the cost function which will be implemented separately for drones is defined as follows:

$$J_{\text{total}} = \sum_{k=0}^{k=H_p} J(k) \tag{3}$$

where,

$$J(k) = \tilde{e}(k)^T Q \tilde{e}(k) + \tilde{u}(k)^T R \tilde{u}(k) \tag{4}$$

$$e(k) = (r - x_0) - Cx(k) \tag{5}$$

where $r = (x_r, y_r)$ is the path reference to follow, x_0 is the equilibrium point, and

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The input and the error are normalized between the minimum and maximum values,

$$\tilde{u}(k) = \frac{u(k) - u_{\min}}{u_{\max} - u_{\min}}, \quad \tilde{e}(k) = \frac{e(k) - e_{\min}}{e_{\max} - e_{\min}} \tag{6}$$

In the cost function appear the 3 tuning parameters for the MPC which are H_p, Q, R, prediction horizon, reference tracking weight and energy minimization weight respectively.

The constraints are defined as follows,

$$u \in \mathcal{U} = \begin{cases} \delta u_1 \in (0, mg) \\ \delta u_2 \in (-mg, mg) \\ \delta u_3 \in (-0.1, 0.1) \end{cases} \quad x \in \mathcal{X} = \begin{cases} \delta x \in (x_{\min}, x_{\max}) \\ \delta \dot{x} \in (-2, 2) \\ \delta y \in (0, y_{\max}) \\ \delta \dot{y} \in (-2, 2) \\ \delta \theta \in (-0.1, 0.1) \\ \delta \dot{\theta} \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \end{cases} \tag{7}$$

The position can be limited but also can be left unbounded, the velocities are limited to 2 m/s. The pitch angle is limited to a small value due to the small angles assumption and the pitch rate is limited to $\frac{\pi}{2}$ rad/s.

3) Implementation: The implementation is performed using the Matlab parser `YALMIP`, and since the model has been linearized, the solver `quadprog` may be used.

C. NLMPC design

For non-linear design we added a wind disturbance, modeled as a drag force and will be taken into account defined by the following equations,

$$f_{\text{ext}} = \begin{bmatrix} f_{\text{ext},x} \\ f_{\text{ext},y} \end{bmatrix} = \begin{bmatrix} \beta_1 v_1 |v_1| \\ \beta_2 v_2 |v_2| \end{bmatrix} \tag{8}$$

Where w_1, w_2, β_1, β_2 are the wind speeds and the drag coefficients respectively. Finally, the nonlinear model or Simulation Oriented Model is defined as follows,

$$\ddot{x} = -\frac{1}{m} \left(\sin\theta (u_1 + u_2) - f_{\text{ext},x} \right) \tag{9}$$

$$\ddot{y} = \frac{1}{m} \left(\cos\theta (u_1 + u_2) - f_{\text{ext},y} \right) - g \tag{10}$$

$$\ddot{\theta} = \frac{L}{I_z} (u_1 - u_2) \tag{11}$$

The system remains unchanged, with only the control inputs redefined, while the disturbance, state variables, and dynamics remain consistent. Analysis of the equations reveals that the robot must tilt to achieve horizontal acceleration, as the thrust forces are oriented upward relative to the robot.

1) Model: The model is characterized by the state variables and their continuous-time derivatives. For use in the MPC, it must be discretized. The forward Euler method is applied, resulting in the following nonlinear discretized model:

$$\begin{cases} x(k+1) = x(k) + \Delta t\, \dot{x}(k) \\ \dot{x}(k+1) = \dot{x}(k) + \Delta t \left(-\frac{1}{m} \left(\sin\theta (u_1 + u_2) - f_{\text{ext},x} \right) \right) \\ y(k+1) = y(k) + \Delta t\, \dot{y}(k) \\ \dot{y}(k+1) = \dot{y}(k) + \Delta t \left(\frac{1}{m} \left(\cos\theta (u_1 + u_2) - f_{\text{ext},y} \right) - g \right) \\ \theta(k+1) = \theta(k) + \Delta t\, \dot{\theta}(k) \\ \dot{\theta}(k+1) = \dot{\theta}(k) + \Delta t \left(\frac{L}{I_z} (u_1 - u_2) \right) \end{cases} \tag{12}$$

2) NLMPC formulation: The MPC formulation stays the same for the drone1 (Leader) but we have employed a new approach for the second and the third drone to maintain safe distance and pitch angle to avoid collision witch each other and the leader drone.

$$J = \sum_{k=1}^{N} \Big[(r_k - Cx_k)^T Q (r_k - Cx_k) + u_k^T R_k u_k + \Delta u_k^T S \Delta u_k + x_k^T Q_{\text{state}} x_k \Big] \tag{13}$$

- **Tracking:** $(r_k - Cx_k)^T Q(r_k - Cx_k), \quad Q = 5 \cdot I_{3\times 3}$
- **Control Effort:** $R_k = I + w \cdot x_0^{0.5} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad w = 10$
- **Smoothness:** $\Delta u_k^T S \Delta u_k, \quad S = 5 \cdot I_{2\times 2}$
- **State Penalty:** $Q_{\text{state}} = \text{diag}(0, 2, 0, 2, 0, 5)$ (penalizes $\dot{x}, \dot{y}, \dot{\theta}$)

Constraints:

- **Inputs:** $0 \leqslant u_k(1), u_k(2) \leqslant mg, \quad u_k(1) + u_k(2) \leqslant mg \cdot (1 - \alpha |x_{0.5}|), \quad \alpha = 5$
- **States:** $-10 \leqslant x, y \leqslant 10, \quad 5 \leqslant \dot{x}, \dot{y} \leqslant 5, \quad 0.1 \leqslant \theta \leqslant 0.1, \quad -2 \leqslant \dot{\theta} \leqslant 2$
- **Torque:** $\left| \frac{l}{I}(u_k(1) - u_k(2)) \right| \leqslant 2$

Tracking Error

The tracking error focuses on ensuring the quadrotor's states—horizontal position (x), vertical position (y), and pitch angle (θ)—closely follow their desired reference values (r_x, r_y, r_θ). A high priority is placed on accurate positioning and orientation, meaning the system is designed to minimize deviations in these specific states to maintain precise control during flight.

Control Effort

The control effort addresses the management of the quadrotor's motor thrusts (u_1, u_2). It aims to prevent excessive thrust usage, ensuring the motors don't overwork, which could lead to inefficiency or instability. Additionally, it includes a dynamic component that limits rapid changes in thrust when the pitch angle (θ) is large, promoting smoother and more cautious adjustments to maintain stability and control.

Smoothness

Smoothness minimizes rapid thrust changes, reducing energy consumption and wear on actuators. It ensures the control inputs change gradually for more stable operation.

State Penalty

State penalty discourages high velocities (e.g., $\dot{x}, \dot{y}$) and angular velocity ($\dot{\theta}$), promoting energy efficiency through drag effects and stabilizing the quadrotor's motion.

Inputs

- Ensures thrusts ($u_k(1)$, $u_k(2)$) are positive and within motor limits, with a maximum of $mg = 49.05$ N.
- Limits total thrust based on pitch angle (θ), reducing it when $|\theta|$ is large for cautious speed, e.g., a limit of 24.525 N when $|\theta| = 0.1$.
- Restricts torque by limiting the difference between thrusts ($|u_k(1) - u_k(2)| \leqslant 2$), reducing energy-intensive angular acceleration.

States

- Constrains horizontal and vertical positions (x, y) to $[-10, 10]$ m, defining the operational workspace.
- Limits speeds ($\dot{x}$, $\dot{y}$) to $[-5, 5]$ m/s, reducing drag and maintaining valid linearization.
- Restricts pitch angle (θ) to $[-0.1, 0.1]$ rad, ensuring linearization holds (e.g., $\sin(\theta) \approx \theta$).
- Caps angular velocity ($\dot{\theta}$) to $[-2, 2]$ rad/s, stabilizing orientation.

Objective Function Weights for Drones

- **Drone 1:**
  - **Tracking Weights (Q1):** Prioritizes accurate tracking of position with a matrix $Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, giving equal importance to horizontal and vertical position errors.
  - **Control Weights (R1):** Applies a penalty on control effort with $R_1 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$, moderately penalizing thrust usage (u_1, u_2) to balance control effort and tracking performance.
- **Drone 2:**
  - **Tracking Weights (Q2):** Emphasizes tracking with $Q_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, doubling the penalty on position errors compared to Drone 1.
  - **Control Weights (R2):** Uses $R_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$, applying a lighter penalty on control effort to allow more aggressive thrust adjustments.
- **Drone 3:**
  - **Tracking Weights (Q3):** Reduces tracking priority with $Q_3 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, halving the penalty on position errors compared to Drone 1.
  - **Control Weights (R3):** Imposes a heavier penalty with $R_3 = \begin{bmatrix} 7 & 0 \\ 0 & 7 \end{bmatrix}$, discouraging large thrust inputs and prioritizing energy efficiency.

Additional Weights for Drone 2 and Drone 3 Objective Functions

- **Velocity Penalty (Qv):** Applies a penalty on velocities ($\dot{x}, \dot{y}$) with $Q_v = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, moderately discouraging high speeds to reduce energy consumption.
- **Pitch Penalty (q_θ):** Sets a scalar weight $q_\theta = 2$ to penalize deviations of pitch angle (θ), emphasizing stable orientation.
- **Desired Pitch (θ_{ref}):** Defines a target pitch angle $\theta_{\text{ref}} = 0.1$ radians, corresponding to a slightly tilted flight posture (not level flight, which typically implies $\theta = 0$).
- **Terminal Weight Scaling ($Q_{\text{N factor}}$):** Scales the terminal state penalty by a factor of 5, ensuring the drone's final state at the end of the prediction horizon is heavily weighted to encourage convergence.

3) Implementation: The implementation is carried out using the MATLAB parser CasADi. A solver capable of handling nonlinear systems and non-convex optimization problems is required, and for this implementation, IPOPT is selected.

4) Prediction Horizon: The prediction horizon is a critical parameter to evaluate, as it governs the controller's predictive capabilities. It is closely tied to the sampling time, as together they define the time horizon. Considering the system's dynamic response and after thorough analysis, a sampling time of $T_s = 0.1$ s is chosen. Given that the maximum speed ranges from 1 to 5 m/s, the desired time horizon is set to 1–5 seconds to achieve a prediction distance of 1–10 m from the robot. Consequently, the prediction horizon is set to $H_p = 10$–50. To balance prediction capability and performance while minimizing computational complexity, the prediction horizon should be as small as possible, as larger values significantly increase the number of optimization variables and, consequently, the computation time.

IV. Results

To assess the results, the Key Performance Indicator (KPI) employed will be the Mean Squared Error (MSE) between the target path and the path achieved by the controller. The reference path is defined as a vertically sinusoidal wave progressing in the direction of increasing x.

A. LMPC performance

As anticipated, the controller's performance is strongly influenced by the selected parameters. The Linear Model Predictive Control (LMPC) lacks the ability to incorporate disturbance forecasting into the model due to the highly nonlinear nature of the drag forces. Consequently, the performance evaluation of the LMPC will be conducted without considering disturbances.

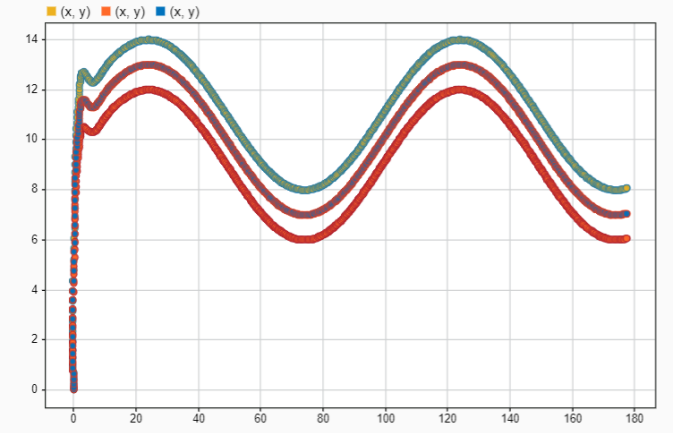

Fig. 1. Performance LMPC. Q = 1,R = 1,Hp = 20

The MSE obtained in is the following,

$$MSE_x = 0.5255; \quad MSE_y = 0.4211 \tag{14}$$

The minimum prediction horizon is set to $H_p = 15$, as values below this threshold cause system instability. figure below illustrates the evolution of the control inputs, which actuate each degree of freedom to track the desired trajectory.

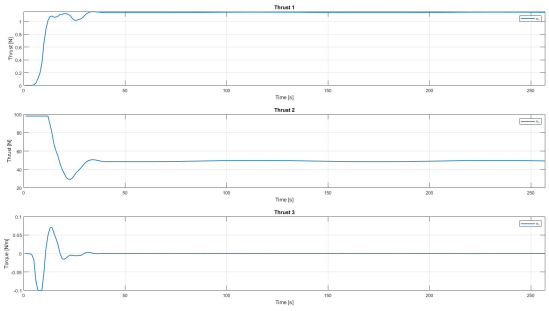

Fig. 2. Controls evolution for 3 drone.

B. NLMPC performance

In the case of the NLMPC implementation, the controller can take into account the disturbance produced by the wind. The obtained MSE is the following:

$$MSE_x = 2.5661; \quad MSE_y = 0.2624 \tag{15}$$

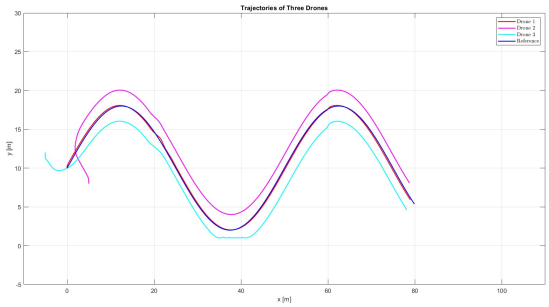

Fig. 3. Performance NLMPC. $w = 2.5$ m/s, $-45°$

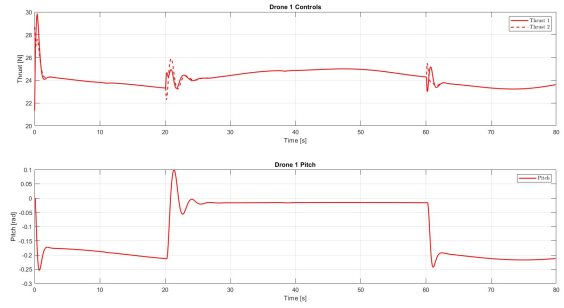

Fig. 4. Controls and pitch evolution for drone 1

For Drone 1, thrust starts high (around 28 N) and stabilizes around 24 N with minor fluctuations, while pitch oscillates between -0.25 and 0.1 radians, indicating tilt adjustments for trajectory tracking. Drone 3 exhibits similar thrust behavior but with more pronounced oscillations (around 30-40 N), and

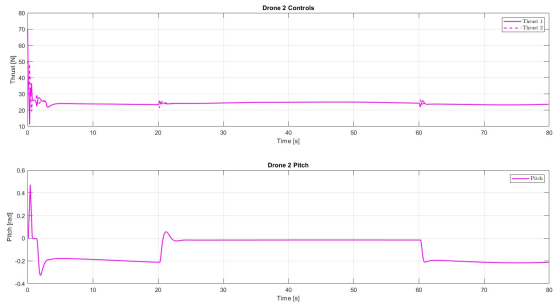

Fig. 5. Controls and pitch evolution for drone 2

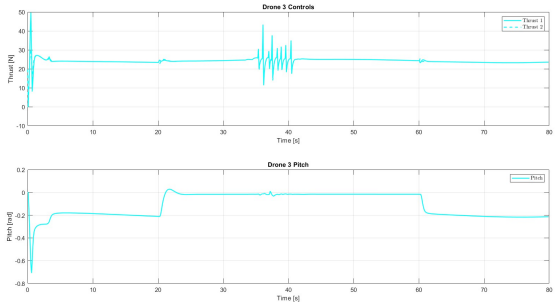

Fig. 6. Controls and pitch evolution for drone 3

pitch varies from -0.6 to 0.2 radians, suggesting greater tilt variations. Drone 2 shows higher initial thrust (up to 60 N) stabilizing around 30 N, with pitch ranging from -0.4 to 0.4 radians, reflecting significant tilt for acceleration. The transient spikes at the start and periodic adjustments align with the need to follow a sinusoidal trajectory, with pitch adjustments compensating for upward thrust forces.

V. Conclusion

This work draws key insights from the implementation of Model Predictive Control (MPC) strategies, focusing on the rationale for adopting this control approach. MPC proves advantageous when its predictive capabilities can be leveraged for control purposes. In this study, the effectiveness of prediction is evident in the disturbance handling and performance outcomes, where prior knowledge of disturbance impacts on dynamics enables effective compensation. Additionally, the MPC framework supports the integration of constraints into the optimization problem, which is critical for real-world robotic applications where actuators have limitations. For instance, a y-coordinate limit is imposed to ensure the robot remains airborne.

Performance and computational efficiency favor Linear MPC (LMPC), as expected, due to the simplification achieved by fully actuating the robot—a scenario less common in rotorcrafts, which lack thrusters for each coordinate. However, this simplification necessitates a supplementary follow-up controller to adjust the robot's pitch to the desired inputs for real-system control. In contrast, the Non-Linear MPC (NLMPC) implementation offers a more practical solution. It delivers comparable performance while autonomously managing disturbance compensation, making it better suited for controlling the actual robotic system.

References

1 Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, M., and Diehl, M., "Casadi – a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

2 Löfberg, J., "Yalmip : A toolbox for modeling and optimization in matlab," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.