A Model Predictive Control Approach for Trajectory Tracking of Quadrotors

Andrei-Carlo Papuc(4772385), Jingwen Tang(5281962)

***Abstract*—We study a model predictive control (MPC) approach for trajectory tracking of quadrotors. With our design, reference tracking and path following achieved impressive performance. The state based disturbance rejection is also performed. Two methods are used to construct the terminal set, and it has been proven analytically and numerically that the system yields asymptotic stability around the origin. In addition, different design parameters such as prediction horizon and terminal cost weight are investigated and the tracking performance is compared with the unconstrained LQR method.**

I. INTRODUCTION

Quadrotors have a broad spectrum of applications, including aerial photography, surveillance, precision agriculture, and search and rescue operations. PID and LQR control methods are commonly used for quadrotors. In comparison, the key advantages of Model Predictive Control (MPC) are its ability to handle complex and uncertain systems, and account for control and state constraints. In this paper, we design a model predictive control method to control a quadrotor with strict state and control constraints.

Our contributions[1] are as follows:

1) We desigend a MPC controller which can perform reference tracking and path following with short prediction horizon (less than 10 simulation steps).
2) By designing the terminal cost and terminal set appropriately [1], asymptotic stability of the origin for the quadrotor system is guaranteed.
3) The tracking task can be performed with state based disturbance rejection. So it is offset-free MPC.

II. DYNAMICS MODEL

A quadcopter features four distinct rotors. It has six degrees of freedom (DoF) that describe its motion (x, y, z) and orientation (ϕ, θ, ψ) in three-dimensional space. The quadrotor model employed in this study is a minor modification of the model introduced by [2] and [3]. Here the aerodynamic thrust drag and aerodynamic moment drag are not considered.

As shown in Figure 1, there are four independent rotors with different rotational speeds $\Omega_{1,2,3,4}$. Each of them creates upwards thrusts and also rotational moments. The four thrusts $K_f\Omega^2_{1,2,3,4}$ offer translational acceleration as u_1. The thrusts created by $K_f\Omega_2^2$ and $K_f\Omega_4^2$ provide the torque u_2 for roll rotation θ. The thrusts created by $K_f\Omega_1^2$ and $K_f\Omega_3^2$ provide the torque u_3 for pitch rotation ϕ. And the four moments $K_M\Omega^2_{1,2,3,4}$ along z direction offer the torque u_4 for yaw rotation ψ. K_f and K_M are the thrust coefficient and moment coefficient respectively. More details refer to [3]. In our model, the input vector is $[u_1, u_2, u_3, u_4]^T$. Equation (1) shows the relations mentioned above, where k_f and k_M are the thrust coefficient and moment coefficient respectively.

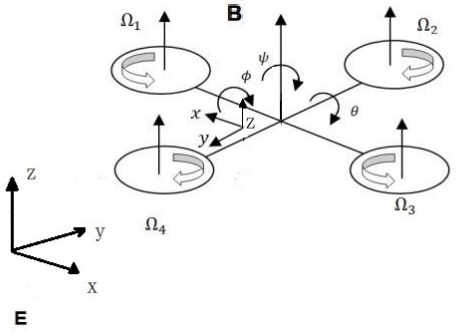

Fig. 1. Configuration of quadrotor [3]. B and E denote the body frame and the earth-fixed frame respectively.

$$\underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}}_{u} = \underbrace{\begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f \\ K_f & 0 & -K_f & 0 \\ K_m & -K_m & K_m & -K_m \end{bmatrix}}_{=\mathrm{K}} \underbrace{\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}}_{\Omega} \tag{1}$$

The state vector is selected as
$x = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^T$,
where $\dot{\phi}$,$\dot{\theta}$ and $\dot{\psi}$ are the rotational velocities in the earth-fixed frame, which can be calculated by the rotational velocities in the body frame p, q, and r as below.

[4]

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2}$$

The translational equations of motion given by $m\ddot{r} = \begin{bmatrix} 0 & 0 & -mg \end{bmatrix}^T + \mathbf{R}(\phi, \theta, \psi) \begin{bmatrix} 0 & 0 & u_1 \end{bmatrix}^T$ with details below [3].

Andrei-Carlo Papuc and Jingwen Tang are master students at TU Delft, The Netherlands. E-mail addresses: {a.c.papuc, j.tang-7}@student.tudelft.nl.

[1]Code is available at https://github.com/andrejcarlo/mpc_drone

$$\begin{aligned}\ddot{x} &= \frac{1}{m}\left[u_1(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)\right]\\ \ddot{y} &= \frac{1}{m}\left[u_1(\sin\phi\cos\psi - \cos\phi\sin\psi\sin\theta)\right]\\ \ddot{z} &= \frac{1}{m}\left[-mg + u_1\cos\phi\cos\theta\right]\end{aligned} \tag{3}$$

The rotational equation of motion is $J\dot{\omega} + \omega \times J\omega + \omega \times \begin{bmatrix} 0 & 0 & J_r\omega_r \end{bmatrix}^T = \begin{bmatrix} -lu_2 & -lu_3 & -lu_4 \end{bmatrix}^T$ [2]. ω_r is the rotors's relative speed: $\omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$. In details, the rotational acceleration can be calculated as below, where $I_{x,y,z}$ is from quadrotor's diagonal inertia Matrix and I_r is the inertia for Gyroscopic moments, and l is the arm length.

$$\begin{aligned}\dot{p} &= \frac{-1}{I_x}\left[-lu_2 - I_y qr + I_z qr + I_r q\omega_r\right]\\ \dot{q} &= \frac{-1}{I_y}\left[lu_3 - I_x pr + I_z pr + I_r p\omega_r\right]\\ \dot{r} &= \frac{-1}{I_z}\left[u_4 + I_x pq - I_y pq\right]\end{aligned} \tag{4}$$

The nonlinear dynamics of the system $\dot{x} = f(x,u)$ are modelled as equations (2) to (4). Then the system is linearized and time discretized before MPC design is applied to the system. The operating point must satisfy the hover condition [3] where $\ddot{z} = 0$, so the operating point is selected as $x_{op} = \mathbf{0}$ and $u_{op} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T$. The system dynamics after linearization and time discretization with time step t_s are shown below. The numerical values of A and B are shown in the **??**.

$$\begin{aligned}\mathbf{A} &= \mathbf{I} + t_s \frac{\partial f(x,u)}{\partial x}\bigg|_{x=x_{op},u=u_{op}}\\ \mathbf{B} &= t_s \frac{\partial f(x,u)}{\partial u}\bigg|_{x=x_{op},u=u_{op}}\\ x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k\end{aligned} \tag{5}$$

III. MPC Problem

In this section, we formulate the MPC problem, which is designed to apply the optimal control input to the system which minimizes the tracking error and control input over a finite prediction horizon, subject to system constraints.

The quadrotor's angular limitations impose constraints on its states at every time step. Since we linearized our system during the hover condition, the quadrotor's orientation is restricted to small angles.

$$\begin{bmatrix} -\frac{1}{9}\pi \\ -\frac{1}{9}\pi \end{bmatrix} \leq \begin{bmatrix} \phi \\ \theta \end{bmatrix} \leq \begin{bmatrix} \frac{1}{9}\pi \\ \frac{1}{9}\pi \end{bmatrix}$$

The angular velocities of the four motors are limited to a range between minimum and maximum values. As a result, the input values $u_{1,2,3,4}$ are constrained by these limits, leading to the following constraints[5], :

$$\underbrace{\begin{bmatrix} u_{1\,(\text{lb})} \\ -11.2680 \\ -11.2680 \\ -0.54 \end{bmatrix}}_{u_{lb}} \leqslant \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \leqslant \underbrace{\begin{bmatrix} 45.0720 \\ 11.2680 \\ 11.2680 \\ 0.54 \end{bmatrix}}_{u_{ub}}$$

where the lower bound of u_1 is given by $\mathbf{K}^{-1}u_k \geq -\mathbf{K}^{-1}u_{op}$ as we linearize the system at operating point.

We form the control input space and state space as $\mathbb{U} = \left\{u \in \mathbb{R}^4 \mid u_{lb} \leq [u_1,u_2,u_3,u_4]^T \leq u_{ub}\right\}$ and $\mathbb{X} = \left\{x \in \mathbb{R}^{12} \mid -\frac{1}{9}\pi \leq \phi,\theta \leq \frac{1}{9}\pi\right\}$.

State feedback offset-free MPC for reference tracking is designed. At each time step, we solve the optimal control input of such problem:

$$\begin{aligned}&\min_u V\left(x_0,u,y_{\text{ref}}\right)\\ &\quad= \sum_{k=0}^{N-1}\ell(x(k)-x_{\text{ref}},u(k)-u_{\text{ref}}) + V_f(x_N - x_{\text{ref}})\\ &\text{s.t. } u(k)\in\mathbb{U},\quad x(k)\in\mathbb{X}\\ &\quad x(k+1) = Ax(k) + Bu(k)\\ &\quad \ell(x(k),u(k)) = \frac{1}{2}x(k)^T Q x(k) + \frac{1}{2}u(k)^T R u(k)\\ &\quad V_f(x_N) = \frac{1}{2}x_N^T P x_N\\ &\quad x(0) = x_0,\quad x(N)\in\mathbb{X}_f(y_{\text{ref}})\end{aligned} \tag{6}$$

Here Q = $0.1I_{12}$ and $R = 0.01I_4$. The P matrix in the terminal cost is designed as the solution of DARE for this optimization problem.

We need our system to track a given output reference y_{ref}, however, the measurement is influenced by an unknown constant disturbance d with noise v as $y = Cx + d + v$. Thus we used the Luenberger observer to estimate the unknown disturbance $\hat{d}^+ = \hat{d} + L(y - Cx - \hat{d})$, which updates itself with the difference of the current estimate and the measured value of $y - Cx$. Hence the system can be augmented as below. Output matrix C extracts the position states from x shown in Appendix A.

$$\begin{aligned}\begin{bmatrix} x^+ \\ d^+ \end{bmatrix} &= \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix}u\\ y &= \begin{bmatrix} C & I \end{bmatrix}\begin{bmatrix} x \\ d \end{bmatrix} + v\end{aligned} \tag{7}$$

With the disturbance estimate, we need the system to stay at steady state x_{ref} with optimal control u_{ref}, hence the Optimal Target Selection problem is built to obtain x_{ref} and u_{ref} in our MPC problem above when y_{ref} is given:

$$(x_{\text{ref}}, u_{\text{ref}}) \left(\hat{d}, y_{\text{ref}}\right) \in \begin{cases} \arg\min_{x_r, u_r} J(x_r, u_r) \\ \text{s.t.} \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_r \\ u_r \end{bmatrix} = \begin{bmatrix} 0 \\ y_{\text{ref}} - \mathbf{x}_{t+k} \end{bmatrix} \\ (x_r, u_r) \in \mathbb{Z} \\ C x_r + \hat{d} \in \mathbb{Y} \end{cases} \tag{8}$$

They are solved online, which means x_{ref} and u_{ref} in our MPC formulation equation (5) are actually dependent on disturbance estimate as $x_{ref}(\hat{d})$ and $u_{ref}(\hat{d})$. The gain matrix L is chosen to make sure the eigenvalues of $(I - L)$ are inside the unit disk so that the error of disturbance estimate $(d - \hat{d})$ converges to zero. The above OTS process is incorporated as computeOTS function shown in algorithm 1.

Algorithm 1: Integration of solving OTS in simulation

```
// Compute OTS and Tset offline;
x_ref, u_ref ← computeOTS(y_ref, d̂ = 0);
X_f ← computeTSet(x_ref);
d̂ ← 0;
for t ∈ T do
    // Recompute OTS and Tset online;
    if d ≠ 0 then
        x_ref, u_ref ← computeOTS(y_ref, d̂);
        X_f ← computeTSet(x_ref);
    end
    u* ← MPCsolver(x_ref, u_ref, x_t);
    x_{t+1}, y_{t+1} ← system_dynamics(x_t, u*);
    d̂ ← d̂ + L × (y_{t+1} − C × x_{t+1} − d̂);
end
```

IV. STABILITY ANALYSIS

A. Linearized System Stability Analysis

For the linearized system dynamics shown in equations (5), the controllability matrix $\left[B, AB, A^2B, \ldots, A^{11}B\right]$ is full row rank, which means the system is controllable. For linear systems with control and state constraints, if assumptions 2.2, 2.3, and 2.14 are satisfied, and the terminal set contains the origin in its interior, by Theorems 2.19 and 2.21, the origin is exponentially stable in $\mathbb{X}_N$. It is described in section 2.5.4 of book [6].

Assumption 2.2 Continuity of system and cost
As shown by equations (2) to (4), $f(x, u)$ is continuous, with continuous state space $\mathbb{X}$ and control input space $\mathbb{U}$. That we approximated the system with a discrete-time representation does not influence its contiuous-time dynamics.
The linearized system satisfies $f(0, 0) = 0$. For the reference tracking task, the origin is at (x_{ref}, u_{ref}), so stage cost $l(x(k), u(k))$ shown in equation (6) satisfies that $l(x_{ref}, u_{ref}) = 0$. The terminal cost V_f shown in equation (6) is a quadratic function, it satisfies $V_f(x_{ref}, u_{ref}) = 0$.

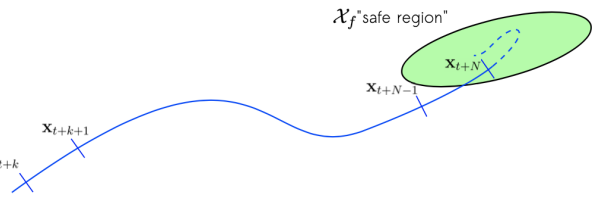

Fig. 2. Illustration of the terminal set $\mathcal{X}_f$ [7], where x_{t+N} is shown inside the control invariant set.

Assumption 2.3 Properties of constraint sets
The set $\mathbb{Z} = \{(x, u) \mid x \in \mathbb{X}, u \in \mathbb{U}\}$ is closed and contains the origin since both $\mathbb{X}$ and $\mathbb{U}$ is closed and contain the origin. Because our reference state and reference control input are chosen from $\mathbb{Z}$ in our OTS process shown in the last session.

Assumption 2.14 Properties of V_f, $\mathbb{X}_f$ and l
2.14(a) We construct our terminal cost as a quadratic function with matrix P as the solution of discrete-time algebraic Riccati equations, which is

$$P = A_k^T P A_k + Q_k$$

where $A_k = A + BK, Q_k = Q + K^T R K$. The terminal costs for the next state are given by:

$$V_f(x^+) = \frac{1}{2}(A_k x)^T P (A_k x) = \frac{1}{2} x^T (P - Q_k) x$$

while the stage cost is given by $l(x, u) = \frac{1}{2} x^T Q_k x$. Thus

$$V_f(x^+) = V_f(x) - l(x, u)$$

When we construct the terminal cost in this way, we approximate the optimal cost-to-go for the constrained system with the infinite-horizon optimal cost-to-go $V_\infty^{uc}(x)$ of the corresponding unconstrained system. The above equations showed that there exists at least one control input $u = Kx$ we can have Lyapunov decrease of the terminal cost. To verify this input is feasible, it is shown in session B that we construct our terminal set in the way that for any state x in the terminal set $Kx \in \mathbb{U}$ satisfies. Thus it is verified that for all $x \in \mathbb{X}_f$, there exists a feasible u satisfying $V_f(f(x, u)) - V_f(x) \leq -\ell(x, u)$. In addition, we construct the terminal set as a sublevel set of V_f, so it satisfies $f(x, u) \in \mathbb{X}_f$.
2.14(b) Our Q and R in stage cost are both positive definite, so a quadratic function constructed with the minimal eigenvalue of Q is a $\mathcal{K}_\infty$ function satisfies assumption (b). Matrix P in the terminal cost is also positive definite, so a quadratic function with the maximum eigenvalue of P as the leading coefficient satisfies.

$$\begin{aligned} \ell(x, u) &= \frac{1}{2}\left(x^T Q x + u^T R u\right) \geq \frac{1}{2} x^T Q x \\ &\geq \lambda_{\min}(Q)|x|^2 = \alpha_1(|x|) \\ V_f(x) &= \frac{1}{2} x^T P x \leq \frac{1}{2} \lambda_{\max}(P)|x|^2 = \alpha_2(|x|) \end{aligned} \tag{9}$$

B. Constructing terminal set X_f

The terminal set constraint guarantees recursive feasibility by enforcing the terminal states in the receding horizon problem to belong to this positive invariant set as shown in Figure

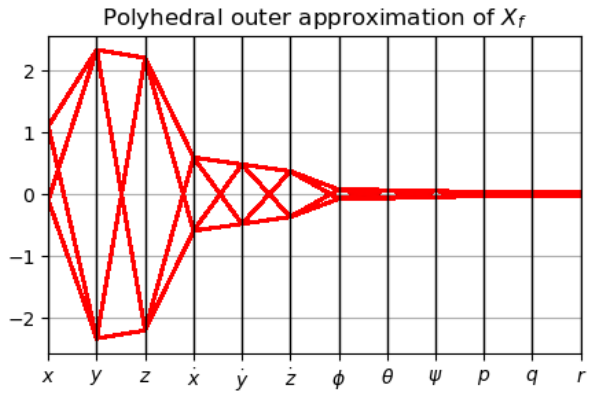

Fig. 3. Vertices of the 12 dimensional polyhedral approximation displayed using parallel lines. State dimensions are represented on the horizontal axis, while positional values of the vertices are given on the vertical axis.

2. The design of the terminal set follows a quadratic level set approach. By constructing a sublevel set of the existing terminal cost $V_f(x)$ as presented in Equation 10, the hyper-parameter constant c is introduced.

$$\begin{aligned} \mathrm{X}_f &= \{x \in \mathrm{R}^n | V_f(x) \leq c\} \\ c > 0 \quad &\text{s.t.} \quad \mathrm{X}_f \subseteq \mathrm{X}, K\mathrm{X}_f \subseteq \mathrm{U} \end{aligned} \tag{10}$$

This design choice of the ellipsoidal terminal set transforms the problem into a quadratically constrained one. Therefore, by switching to a QCQP (quadratically constrained quadratic program) from a QP, computational expense of the solver becomes higher. Since the terminal set now is an approximation of the terminal cost by a level set, the set is now considered positive invariant for $x^+ = A_K x$.

Computation of the hyper-parameter c is not trivial and therefore, an optimisation method of choosing it has been developed. To minimise the computational effort needed to check whether X_f satisfies the set of controllable states and actions, a polyhedral outer approximation of the ellipsoid is used. In the case of the system's 12 dimensional state vector x, the approximation of the ellipsoid is going to have 2^{12} vertices that need to be checked for state and input constraints satisfaction. That is $x \in \mathrm{X}$, $Kx \in \mathrm{U}$ for all $x \in$ **vertices of** $\mathcal{P}_{out}$. To compute the vertices of $\mathcal{P}_{out}$ the semi-axis of the ellipsoid, which are equal to the eigenvectors of the discrete algebraic Riccati solution P can be summed and checked for constraint satisfaction. Algorithm 2 presents the method of calculating c, and Figure 3 shows the computed 12 dimensional polyhedral approximation displayed using parallel lines representing each dimension.

Another method of constructing the terminal set consists of approximating the terminal set by scaling $V_f(x)$ appropriately. The shape of the terminal set $\mathbb{X}_f$ exactly matches the contour lines of V_f because our design choice of terminal set is a sublevel set of the terminal cost. This is also the reason why a scaling factor can take the place of the hard terminal set requirement. By setting a high penalty on the final state x_N, the terminal set constraint can be eliminated by choosing a large enough value of $\beta \geq 1.0$, and adding it as a factor for the terminal cost in the optimal cost minimisation problem. The optimal control problem becomes:

Algorithm 2: Solving for the level set curve c

// first guess of c;
$c \leftarrow 1.0$;
while *not*(*verts* $\in$ X *and* *Kverts* $\in$ U) **do**
  $c \leftarrow c/1.01$;
  $\lambda, v \leftarrow$ *eigenvectors*(P);
  $\lambda \leftarrow \lambda \cdot c$;
  // Combinations of vecs (-i,-j), (i,j), (-i,j), (i,-j);
  verts $\leftarrow$ *polyhedra*(λ, v);
  // Translate Tset by x_{ref};
  verts $\leftarrow$ verts $+ x_{\text{ref}}$;
end

$$\begin{aligned} &\min_u V^\beta(x_0, u, y_{\text{ref}}) \\ &\quad = \sum_{k=0}^{N-1} \ell(x(k) - x_{\text{ref}}, u(k) - u_{\text{ref}}) + \boldsymbol{\beta} V_f(x_N - x_{\text{ref}}) \end{aligned} \tag{11}$$

Where the rest of the constraints remain the same, with the exception of the terminal set. The constraint $x(N) \in \mathbb{X}_f(y_{\text{ref}})$ can now be omitted. The idea behind this is that for a large β the state x_N is highly penalised by βV_f. The following lemma is proposed:

$$\exists \beta \geq 1.0 \text{ s.t. } x_N^\beta(N; x) = \phi(N; x; \boldsymbol{u}_N^\beta) \in \mathbb{X}_f$$

V. NUMERICAL SIMULATIONS

In this section, we show several numerical simulations of the designed model predictive controller. We perform comparisons by altering the MPC parameters and proving asymptotic stability. Firstly, the stability of the controller will be proved numerically, in a reference tracking scenario, where the MPC is tasked with reaching a single target y_{ref}, from a given starting state. Next, a path following simulation is performed, where the controller is tasked with following a given trajectory, using bang-bang velocity profiles [8].

In the following subsections, the constant disturbance d introduced in the OTS is set to be zero in order to better visualise the performance of MPC without disturbance rejection. The starting position of the quadrotor state is assumed to be at the origin, with zero velocity and under the hover assumption, while and the target y_{ref} is set to be $[1.0, 1.0, 1.0]$.

A. Prediction Horizon N

First, two simulations are run for different values of N, first without making use of the terminal cost and terminal set and vice-versa. As the goal of the controller is to bring the quadrotor from the origin towards the goal of $[1.0, 1.0, 1.0]$, the absolute tracking error in the first states x, y, z is computed using the euclidean distance from current position to goal.

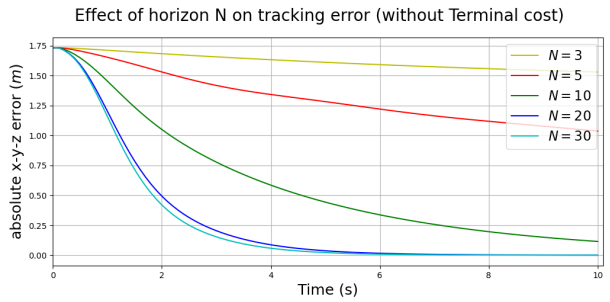

Fig. 4. Absolute tracking error for a range of prediction horizons N. MPC does not use terminal cost nor terminal set conditions here.

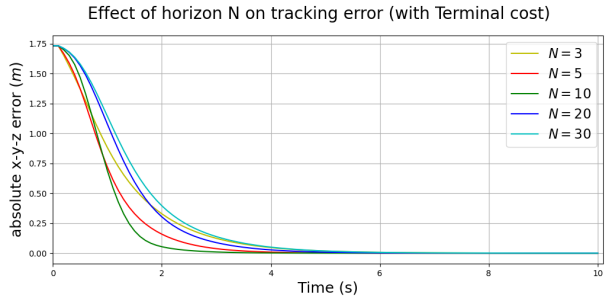

Fig. 5. Absolute tracking error for a range of prediction horizons N. Terminal cost and terminal set constraints have been enabled in the optimisation problem.

The results are presented in Figure 4 and Figure 5, respectively. Comparing the two figures, major differences can be seen. As expected, MPC without the addition of terminal set and cost does not manage to converge for low horizons, as it does not approximate the infinite horizon cost-to-go.

As can be observed from Figure 5 the prediction horizon has a minor effect on the tracking error, even with the lowest horizon, $N = 5$, the controller converges to the target. On the other hand, Figure 4, increasing N has a much more significant effect on the tracking performance, only around $N = 20$, MPC without terminal cost starts to gets close to the terminal constrained MPC. Therefore, by including an approximation of the best cost-to-go with an infinite horizon as the terminal cost, the optimal solution discovered by MPC is less short-sighted, achieving the goal more quickly and with less control effort.

B. MPC vs LQR

In this subsection the performance of the proposed MPC is used in comparison with that of the LQR controller. It's worth noting that the LQR is unconstrained, therefore the limits on control actions and states discussed in the previous chapters are not taken into account when computing the next state using the LQR control gain K.

Looking at the tracking performance in Figure 6, it can be seen that MPC approaches LQR with increasing horizon N. It appears that the LQR approaches the spatial goal more smoothly than an MPC with a lower horizon. Moreover, the smoothness of the LQR controller can also be seen in Figure 7, where the four inputs of the quadrotor are presented. The first input u_0 represents the total upward force of the drone, which

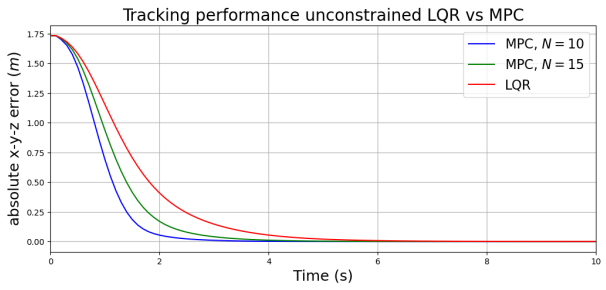

Fig. 6. Tracking performance between unconstrained LQR and MPC for two N configurations.

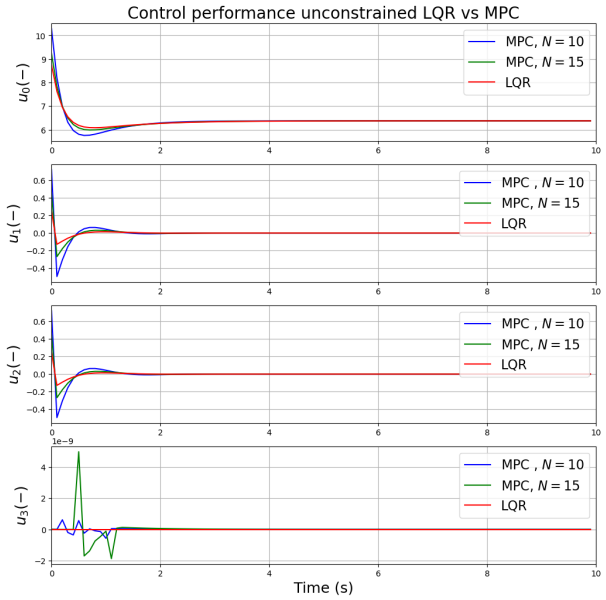

Fig. 7. Control actions difference between unconstrained LQR and MPC for two N configurations.

controls the movement of going up and down. From the plot, it can be depicted that LQR starts with a smaller control input than MPC and therefore needs to counteract less in order to stabilise around the hover point. Similar behaviour happens with the other control inputs.

Finally, it can be seen that after 2-4 seconds, the results of both controllers coincide and therefore confirming asymptotic stability when reaching the target. Tracking errors converge to the origin and actions of MPC converge to those of LQR.

C. Terminal cost scaling β

As discussed previously, because the terminal set follows a level set curve of the terminal cost, it can be approximated by scaling V_f using $\beta \geq 1.0$. The effect of this scaling in comparison to a non-existent V_f is simulated . To better illustrate the effect of increasing β, a simulation using increasing values along with one resembling non-existent terminal cost is performed.

In Figure 8, an MPC simulation with a low horizon of $N = 3$ is unstable when eliminating terminal cost by choosing $\beta = 0.01$, while for $\beta \geq 1.0$ systems are stable and converge to the origin, meaning the target has been achieved success-

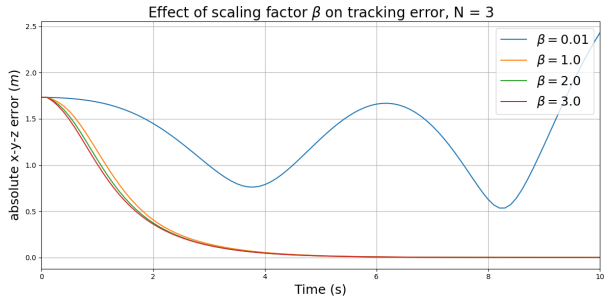

Fig. 8. Effect of changing the scale factor β for terminal cost $V_f(x)$. Tracking error decreases with increasing β, as it better steers the states towards the goal, and approximating X_f

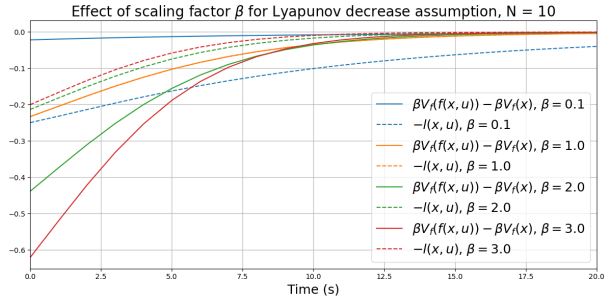

Fig. 9. Effect of changing the scale factor β for terminal cost $V_f(x)$. Experimentally showing the Lyapunov decrease of the terminal cost.

fully. Moreover, a different simulation in Figure 9 shows the effect on the lyapunov decrease when β is increased. For lyapunov decrease to be met, $\beta V_f(x^+) - \beta V_f(x) \leq -\ell(x,u)$. In the case of $\beta = 1.0$, the two curves are on top of each other, while for increasing factors, the separation is larger. Therefore confirming *Assumption 2.14* while the solution converges to the ideal terminal set. The curve showcasing $\beta = 0.1$, does not satisfy the lyapunov inequality and therefore does not guarantee stability.

To further display the improvement of using a scaling factor β on terminal set approximation, Figure 10 shows the comparison of using scaled terminal cost versus using a terminal set calculation. The large time difference is attributed to the number of iterations that need to be computed for a polyhedra consisting of 2^{12} vertices.

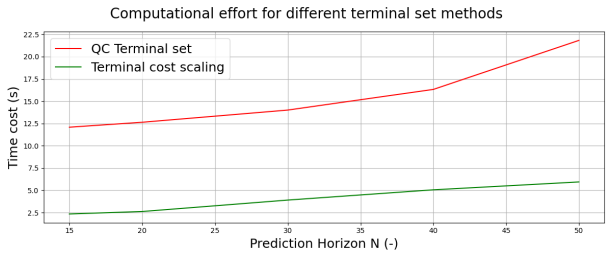

Fig. 10. Computational effort of two terminal set construction methods. Red curve represents terminal set based on level curve set c, while the green curve approximates the terminal set scaling the terminal cost using β.

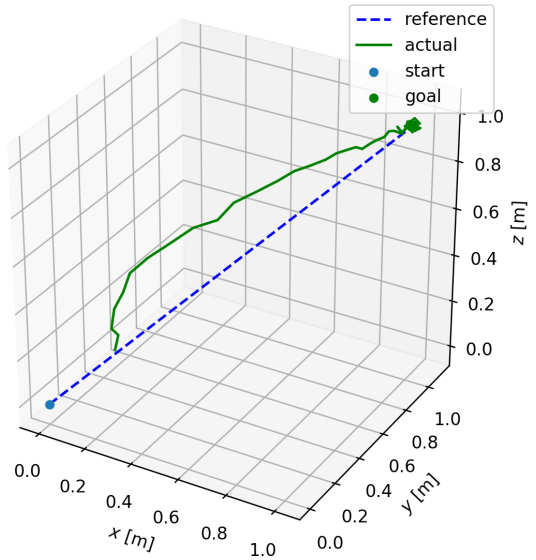

Fig. 11. Offset-free MPC tracking with a constant disturbance in the y-direction. Quadrotor manages to reject disturbance and approach the goal target.

D. Disturbance rejection

In the previous subsections, disturbance d was assumed to be zero in order to qualitatively analyse the performance of MPC with respect to parameter changes. Next, a simulation with non zero disturbance can be considered. The output of the controller is dictated by Equation 7, where y is 3 dimensional and represents the quadrotor's position in x, y, z. In this section, d is set to be a constant lateral disturbance in the y direction equal to 0.5. The vector d is then going to be equal to $[0.0, 0.5, 0.0]$, and the task of the offset-free MPC is to achieve the final goal whilst counteracting this disturbance.

According to algorithm 1, the OTS and terminal set construction will now be solved online, meaning they will be recomputed at every time step. This is because both of these, depend on disturbance estimate $\hat{d}$ which is iteratively updated through the observer proposed in the MPC formulation. When estimating $\hat{d}$ a measurement error v is also introduced which is considered as gaussian white noise. It follows a normal distribution and it's expectance $E[v] = 0$.

Figure 11 show's the reference tracking in real time, the quadrotor starts with an offset of 0.5 in the y-direction from the starting point, the origin. Then, it slowly counteracts the disturbance due to the augmented system and observer. An a-posteriori confirmation that disturbance is rejected is shown in Figure 12, where the norm of the error between real and estimated disturbance converges to zero, therefore validating asymptotic stability conditions. The jitter present towards the bottom is due to measurement noise Finally, this is further confirmed by Figure 13, which shows lyapunov decrease still holds under the presence of disturbance.

E. Path following

This final subsection presents the case of trajectory tracking of the quadrotor. Three separate parametric trajectories have

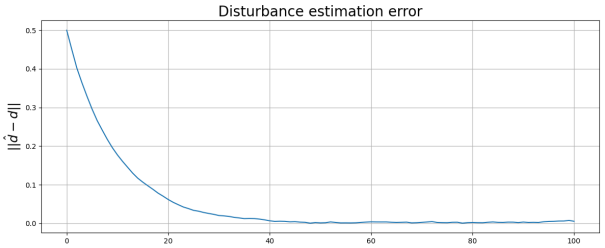

Fig. 12. Norm of error between estimated disturbance and real constant disturbance. A posteriori confirmation of asymptotic stability as disturbance is rejected through observer.

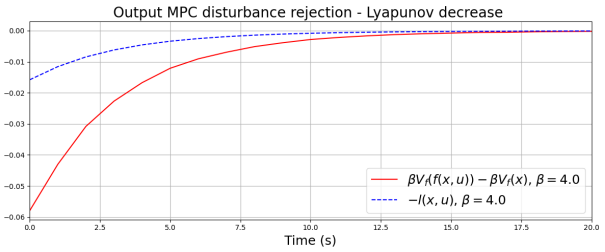

Fig. 13. Lyapunov decrease ensured during offset-free MPC simulation.

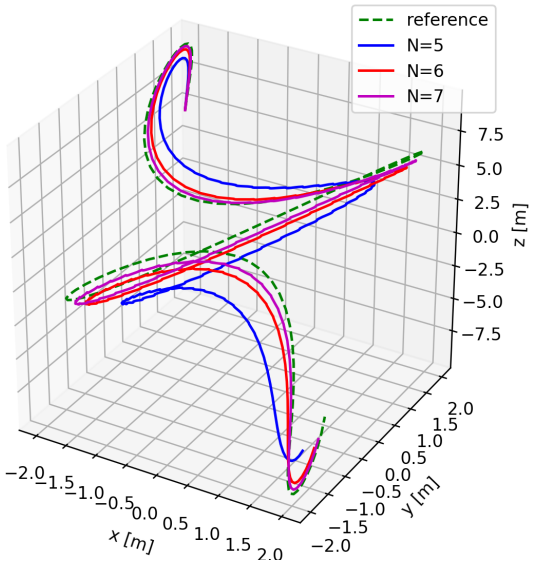

Fig. 14. Path following a Lissajous reference in green dotted curve. Simulation shows effect of increasing the horizon N.

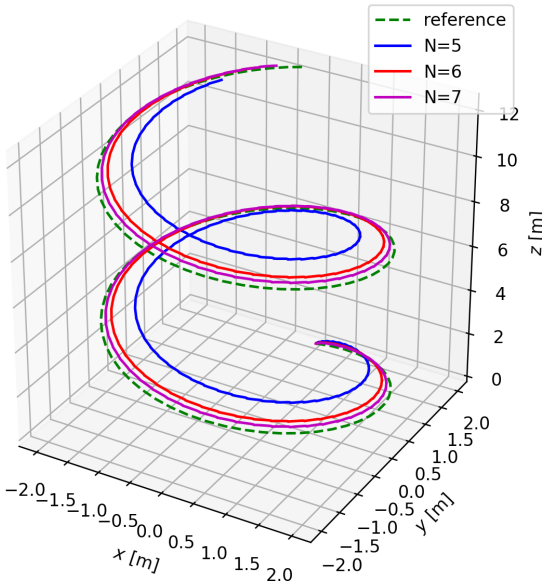

Fig. 15. Path following an upward helix reference in green dotted curve. Simulation shows effect of increasing the horizon N.

been proposed, a lissajous, a helix and a zigzag pattern respectively. Each of them representing real-life use case scenarios of quadrotor control for lighting shows or area monitoring control. An important aspect of the following simulations is the use of bang-bang velocity profiles, meaning the drone accelerates to the maximum acceleration and then decelerates to a minimum, to achieve a hovering state between each point of the trajectory.

Figure 14 and Figure 15 show two different trajectories for different N prediction horizons, without the use of terminal cost. As discussed earlier, the lower the prediction horizon, the more short-sighted the controller is and the bigger the tracking error. This is shown by the different curves, the larger the N the real paths close on the green dotted reference.

On the other hand, when introducing a penalty for x_N with the addition of the terminal cost, Figure 16 and Figure 17 presents the major differences. With only a $N = 5$ and terminal cost added, in addition to terminal cost approximation through β scaling, the trajectories are close to perfectly followed. While the non-terminal-constrained MPC has very poor performance on the zigzag pattern due to being too short-sighted.

VI. CONCLUSION & FUTURE WORK

Our paper presents a model predictive control (MPC) approach for trajectory tracking of quadrotors, where a mathematical model of the quadrotor is developed with consideration of aerodynamic moment and drag to improve the simulation's real-world accuracy. The simulation results demonstrate the quadrotor's ability to track trajectories and avoid disturbances while satisfying state and control input constraints with satisfactory tracking error. The implementation of MPC with terminal constraints and penalties is successful, and we have proven the recursive local asymptotic stability of our MPC controller for trajectory tracking. Additionally, we have validated all assumptions and demonstrated stability through a confirmation of Lyapunov decrease and disturbance rejection a-posteriori.

For future work, we'd like to study non-linear adaptive MPC for tracking trajectories as presented in [9], to be able to perform highly agile maneuvers, which are not possible under the linearisation assumptions proposed in this paper. Moreover, another interesting study is the use of quaternion-based quadrotor [10] over euler based tracking. This can result in better tracking errors and less computational effort.

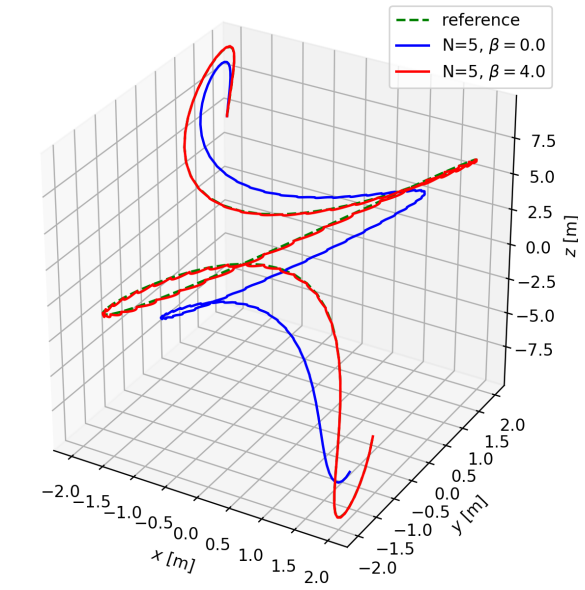

Fig. 16. Path following a Lissajous reference in green dotted curve. Simulation of $N = 5$ shows effect of introducing the scaling factor β for terminal set approximation.

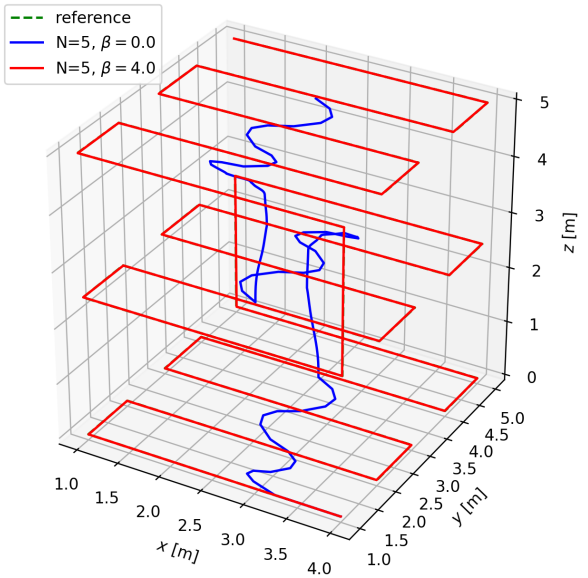

Fig. 17. Path following a typical zigzag drone coverage reference in green dotted curve. Simulation of $N = 5$ shows effect of introducing the scaling factor β for terminal set approximation.

References

[1] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109899002149

[2] H. Elkholy and A. U. in Cairo. School of Engineering Interdisciplinary Program, *Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches*, ser. Thesis (American University in Cairo. School of Engineering Interdisciplinary Program). American University in Cairo, 2014. [Online]. Available: https://books.google.nl/books?id=nZr0rQEACAAJ

[3] M. Islam, M. Okasha, and M. M. Idres, "Dynamics and control of quadcopter using linear model predictive control approach," *IOP Conference Series: Materials Science and Engineering*, vol. 270, no. 1, p. 012007, dec 2017. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/270/1/012007

[4] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[5] C. Kanellakis, S. S. Mansouri, and G. Nikolakopoulos, "Dynamic visual sensing based on mpc controlled uavs," in *2017 25th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2017, pp. 1201–1206.

[6] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2017.

[7] M. S. Kamel, T. Stastny, K. Alexis, and R. Siegwart, *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System*, 05 2017.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[9] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors," *CoRR*, vol. abs/2109.04210, 2021. [Online]. Available: https://arxiv.org/abs/2109.04210

[10] M. Islam, M. Okasha, M. Idres, and H. Mansor, "Trajectory tracking of quaternion based quadrotor using model predictive control," *International Journal of Engineering and Technology(UAE)*, vol. 7, pp. 125–136, 10 2018.

Appendix

The numerical values of state matrix, control matrix and output matrix are as below.

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

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -0.98 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -0.98 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.1538 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -3.0667 & 0 & 0 \\ 0 & 0 & -3.0667 & 0 \\ 0 & 0 & 0 & -1.7692 \end{bmatrix}$$