ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MODEL PREDICTIVE CONTROL REPORT

Developing MPC controllers for the quadcopter

JIANHAO ZHENG (SCIPER: 323146)
SHUHAN HE (SCIPER: 322317)
YUJIE HE (SCIPER: 321657)

1st January 2021

Part2 Linearization and Diagonalization

Deliverable 2.1

Explain why we can break the system into four independent/non-interacting systems, and whether this would occur at other steady-state conditions.

The state, input, and output of the quadrotor system can be defined as follows:

$$\begin{aligned}
&\mathbf{x} = [\dot{\alpha}, \dot{\beta}, \dot{\gamma}, \alpha, \beta, \gamma, \dot{x}, \dot{y}, \dot{z}, x, y, z]^{\mathsf{T}} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}]^{\mathsf{T}} \\
&\mathbf{u} = [u_1, u_2, u_3, u_4]^{\mathsf{T}} \\
&\mathbf{v} = T\mathbf{u} = [F, M_\alpha, M_\beta, M_\gamma]^{\mathsf{T}} = [v_1, v_2, v_3, v_4]^{\mathsf{T}} \\
&\mathbf{y} = \mathbf{x}
\end{aligned} \quad , \tag{1}$$

where $\mathbf{u}$ and $\mathbf{v}$ indicate the input in the original and transformed system, respectively.

As defined in `Quad.m`, the inertia matrix of the quadrotor is $I = \text{diag}(10, 10, 15)$. Thus, the state-space model of the quadrotor can be computed as:

$$\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \frac{-5x_2x_3+v_2}{10} \\ \frac{5x_1x_3+v_3}{10} \\ \frac{v_4}{15} \\ x_1 \\ x_2 \\ x_3 \\ \frac{v_1 \sin x_5}{m} \\ -\frac{v_1 \sin x_4 \cos x_5}{m} \\ \frac{v_1 \cos x_4 \cos x_5}{m} - g \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} . \tag{2}$$

Let $\bar{\mathbf{x}} = \mathbf{x}_s$ and $\bar{\mathbf{v}} = \mathbf{v}_s$, the linearized model obtained by small signal linearization is:

$$\mathbf{x} = \bar{\mathbf{x}} + \tilde{\mathbf{x}}, \quad \mathbf{v} = \bar{\mathbf{v}} + \tilde{\mathbf{v}}, \quad \mathbf{y} = \bar{\mathbf{y}} + \tilde{\mathbf{y}} . \tag{3}$$

As a result, the space model can be reformulated as follows:

$$\begin{aligned}
\dot{\tilde{\mathbf{x}}} =& A\tilde{\mathbf{x}} + B\tilde{\mathbf{u}} \\
=& A\tilde{\mathbf{x}} + BT^{-1}\tilde{\mathbf{v}} \\
=& \begin{pmatrix}
0 & -\frac{\bar{x}_3}{2} & -\frac{\bar{x}_2}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\bar{x}_3}{2} & 0 & \frac{\bar{x}_1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\bar{v}_1 \cos \bar{x}_5}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{\bar{v}_1 \cos \bar{x}_4 \cos \bar{x}_5}{m} & \frac{\bar{v}_1 \sin \bar{x}_4 \sin \bar{x}_5}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{\bar{v}_1 \sin \bar{x}_4 \cos \bar{x}_5}{m} & -\frac{\bar{v}_1 \cos \bar{x}_4 \sin \bar{x}_5}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix} \tilde{\mathbf{x}}
\end{aligned}$$

$$+\begin{pmatrix} 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & \frac{1}{10} & 0 \\ 0 & 0 & 0 & \frac{1}{15} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\sin\bar{x}_5}{m} & 0 & 0 & 0 \\ \frac{\sin\bar{x}_4\cos\bar{x}_5}{m} & 0 & 0 & 0 \\ \frac{\cos\bar{x}_4\cos\bar{x}_5}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\tilde{\mathbf{v}}\,. \tag{4}$$

All the states $\bar{\mathbf{x}}$ and inputs $\bar{\mathbf{v}}$ obtained by `quad.trim()` is 0 except that $\bar{v}_1 = mg$. Therefore, the linearized model can be simplified as:

$$\dot{\tilde{\mathbf{x}}} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&g&0&0&0&0&0&0&0 \\ 0&0&0&-g&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \end{pmatrix}\tilde{\mathbf{x}} + \begin{pmatrix} 0&0.1&0&0 \\ 0&0&0.1&0 \\ 0&0&0&0.0667 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 1/m&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}\tilde{\mathbf{v}}\,. \tag{5}$$

Initially, matrix B implies that one input could have influence on several different states. After transformation, the new matrix $B' = BT^{-1}$ allows every single new input v_i can only effect one state. In this case, the matrix A also ensures that there is no interacting between the states of the four sub-systems (sys_x, sys_y, sys_z and sys_yaw). These two conditions allow the whole system to be separated into four independent sub-systems. Taking sys_x (x_2, x_5, x_7, x_{10}) as an example, $\dot{\tilde{x}}_2 = 0.1\tilde{v}_2, \dot{\tilde{x}}_5 = \tilde{x}_2, \dot{\tilde{x}}_7 = g\tilde{x}_5, \dot{\tilde{x}}_{10} = \tilde{x}_7$. No other state nor other input except v_2 occurs in the state equation.

This could happen because we take nice value on $\mathbf{x_s}$. As we can see in equation (4), $\dot{\tilde{x}}_1$ is related to $\tilde{x}_2$ and $\tilde{x}_3$. In this special case, the value of $\bar{x}_2$ and $\bar{x}_3$ are all set to be 0. Thus, the relationship decouples. Similar thing happen on the other states.

This would occur at any other steady-state conditions. According to equation (4), the whole system can be separated as long as $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5$ are set to be 0. Derived by equation (2), only $\bar{x}_6, \bar{x}_{10}, \bar{x}_{11}, \bar{x}_{12}, \bar{v}_1$, i.e. yaw, x, y, z and F, can take non-zero value for all steady-state. Thus, the requirement mentioned before are satisfied. Therefore, we can safely break the system whatever the steady-state is.

Part3 Design MPC Controllers for Each Sub-System

In this part, we present the design procedure of linear model predictive control (LMPC)-based regulators and tracking controllers for the quadcopter, respectively. Then, the choice of tuning parameters is discussed according to the quantitative and qualitative experiments. Finally, the time-varying state and input variables as well as the comparison between the quadcopter trajectories and reference path are visualized to demonstrate the overall performance.

Deliverable 3.1 Design MPC Regulators

Explanation of design procedure that ensures recursive constraint satisfaction

In the Part 2, the dynamics of the quadcopter are linearized and discretized with a sampling period $T_s = 0.2s$. Then, we can implement the MPC with quadratic performance measure to constrain the state x_i and the input u_i from step 0 to $N-1$ as well as the terminal state x_N after increasing to step N.

We take the subsystem x is taken as an example, and the design for subsystem y, z and yaw are similar. The MPC regulation cost can be formulated as:

$$\begin{aligned} \min \ & \sum_{i=0}^{N-1} x_i^\top Q x_i + u_i^\top R u_i + x_N^\top Q_f x_N \\ \text{s.t. } & x_{i+1} = A x_i + B u_i \\ & H_x x_i \le h_x \\ & H_u u_i \le h_u \\ & H_f x_N \le h_f \\ & x_i \in \mathbb{X} \\ & u_i \in \mathbb{U} \end{aligned} , \tag{6}$$

where matrix A and B denote the linearized dynamics of each dimensions; matrix H_x and h_x are state constraints while matrix H_u and h_u are input constraints, which are categorized according to four subsystems in Table 1. Specially, we drop the H_x and h_x in subsystem z and yaw due to the absence of specified constraints on state values.

TABLE 1: State and input constraints of the quadcopter

Subsystem	State	Input
MPC_x	$\lvert\alpha\rvert \le 2^\circ = 0.035\text{rad}$	$-0.3 \le M_\alpha \le 0.3$
MPC_y	$\lvert\beta\rvert \le 2^\circ = 0.035\text{rad}$	$-0.3 \le M_\beta \le 0.3$
MPC_z	–	$-0.2 \le F \le 0.3$
MPC_yaw	–	$-0.2 \le M_\gamma \le 0.2$

Taking MPC_x as an example, based on the input and state constraints, the corresponding can be computed as shown in Table 2.

To ensure recursive feasibility and stability, the terminal maximal invariant set using the LQR control law $u = Kx$ while terminal cost in form of $x_N^\top Q_f x_N$ is integrated into the total cost, resulting in the Lyapunov function. The corresponding variable K and Q_f can be obtained obtained by using `LTISystem` function in mpt3 library.

Table 2: State and input matrix for the LMPC

Subsystem	H_x	h_x	H_u	h_u
MPC_x	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0.035 \\ 0.035 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.3 \\ 0.3 \end{pmatrix}$
MPC_y	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0.035 \\ 0.035 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.3 \\ 0.3 \end{pmatrix}$
MPC_z	–	–	$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.3 \\ 0.2 \end{pmatrix}$
MPC_yaw	–	–	$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix}$

A terminal LQR controller is employed in each subsystem and the discrete-time algebraic Riccati equation for the LQR controller can be computed as follows:

$$\begin{aligned} &P = Q + A^\top PA - A^\top PB\left(R + B^\top PB\right)^{-1} B^\top PA \\ &K = -\left(R + B^\top PB\right)^{-1} B^\top PA \\ &Q_f = P \end{aligned} \quad . \tag{7}$$

According to the implementation of the equations above, the solver finally returns the first value in the the control optimal sequence $\mathbf{u}^*$, i.e., u_0^*, as an input to the system.

Explanation of choice of tuning parameters. (e.g., Q, R, N, terminal components)

Table 3: Main parameters of the LMPC

Parameter	Subsystem x and y	Subsystem yaw and z
Horizon N	20	20
Cost matrix Q	I_4	$\begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}$
Cost matrix R	1	1

Table 3 shows the key parameters utilized in the proposed LMPC.

N will influence stability, feasibility, and invariance. With a smaller N, the solution is infeasible. A feasible solution can be obtained when increasing N to 12 but the settling time exceeds eight seconds in subsystem x and y. To ensure the feasibility of the proposed MPC from the initial states, N is set to 20 for all four subsystems, which is big enough to guarantee the infinite horizon approximation.

Q and R are relevant to the overall cost computation and the size of the terminal set as well. An incorrect set of both matrices may lead to infeasible MPC. Through iteratively tuning the parameters for each subsystem to meet the performance criteria that settling time should be limited within eight seconds, we found that penalizing more on states x_{10} and x_{11}, i.e. the x position and the y position, in subsystem x and subsystem y do not show a large performance difference. Therefore, we set both matrices Q and R to be identity matrices in subsystem x and subsystem y. On the other hand, penalizing more on states x_6 and x_{12}, i.e. the yaw angle and the z position, in subsystem yaw and subsystem z can improve the performance of the controller and the quadcopter will reach the reference faster. Considering larger penalizing coefficients may lead to smaller terminal set, we set the matrix Q as diag(1, 5) and matrix R as an identity matrix in both subsystem yaw and subsystem z.

The terminal Q_f in each subsystem can be obtained via the optimal LQR solution from `LTISystem` in the mpt3 library.

Plot of terminal invariant set for each of the dimensions, and explanation of how they were designed and tuning parameters used

The maximum terminal invariant set is achieved by intersecting the current set X_f with its pre-set $\mathrm{pre}(X_f)$ until convergence. The iteration ends when $X_f = \mathrm{pre}(X_f)$, and the obtained X_f is the terminal invariant set. This visualization of the terminal invariant set for x, y, z, and yaw can be shown in Figure 1, 2, 3, and 4. Specially, the projection of the subsystem x and y are presented due to higher dimensions.

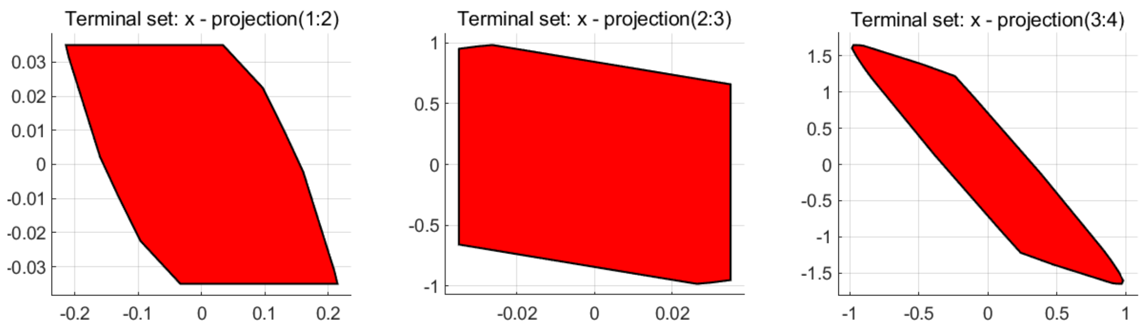


FIGURE 1: Projection of terminal invariant set for x

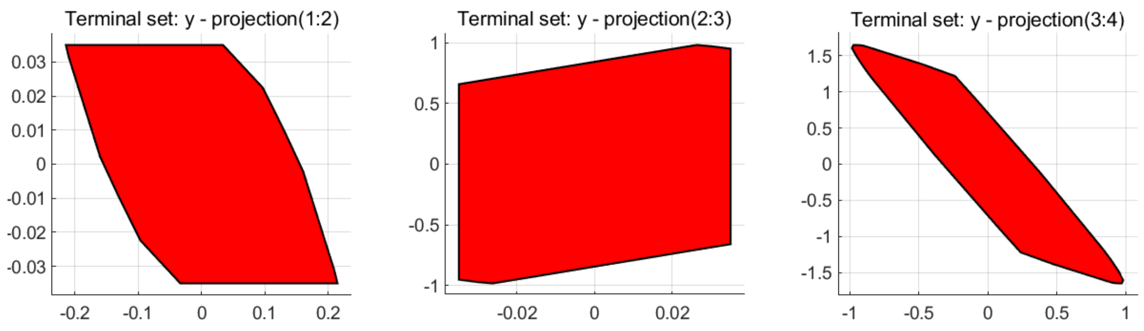


FIGURE 2: Projection of terminal invariant set for y

Plot for each dimension starting stationary at two meters from the origin (for x, y and z) or stationary at 45 degrees for yaw

Figure 5, 6, 7, and 8 present the regulation performance of each subsystem under LMPC. As expected, each subsystem converges towards the origin for every dimension.

Related m-codes for the controllers are attached in Deliverable_3_1.zip. The main script for producing the results is `Deliverable_3_1.m`.

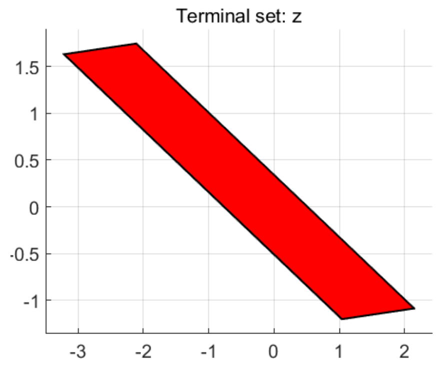


FIGURE 3: Terminal invariant set for z

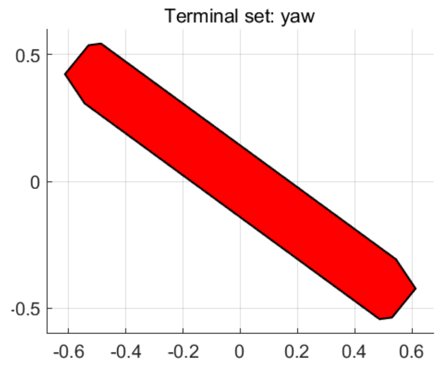


FIGURE 4: Terminal invariant set for yaw

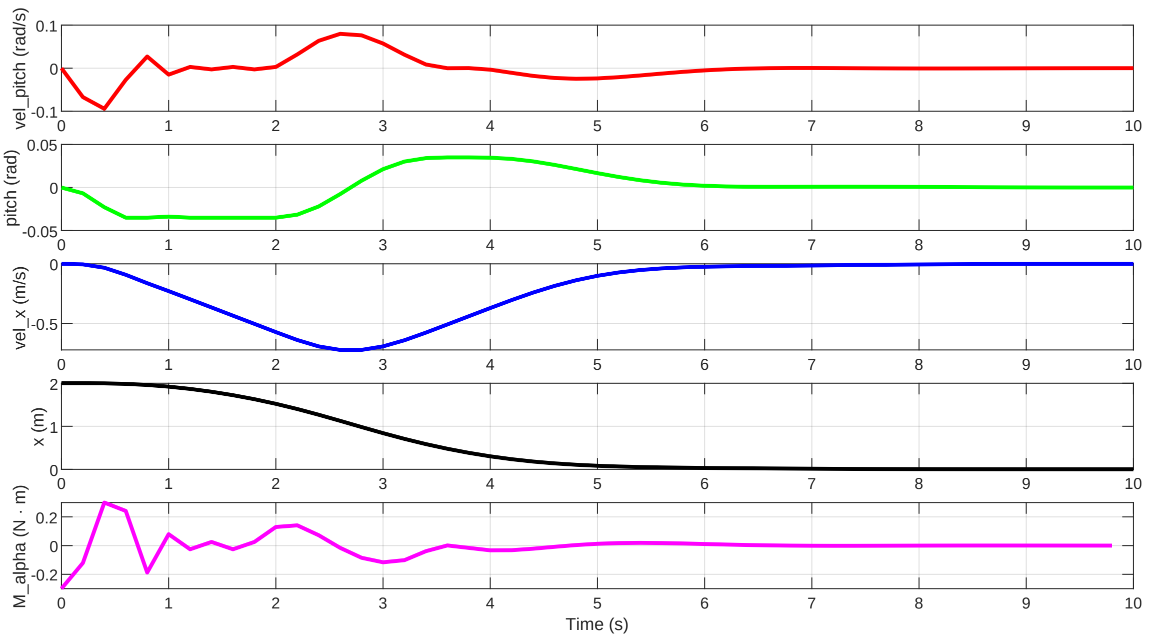


FIGURE 5: Quadcopter state and input variations of subsystem x under regulating controller

Deliverable 3.2 Design MPC Tracking Controllers

Explanation of your design procedure and choice of tuning parameters

In this design procedure, the first step is to compute the expected steady-state (x_s, u_s) given the tracking reference r. Similar to the design procedure of the regulating controller under LMPC, the subsystem x is taken as an example. Also, the design for subsystem y, z, and yaw are similar. Specially, we drop the H_x and h_x in subsystem z and yaw due to the absence of specified constraints on state values.

Assuming that the target problem is feasible, the steady-state target problem for each

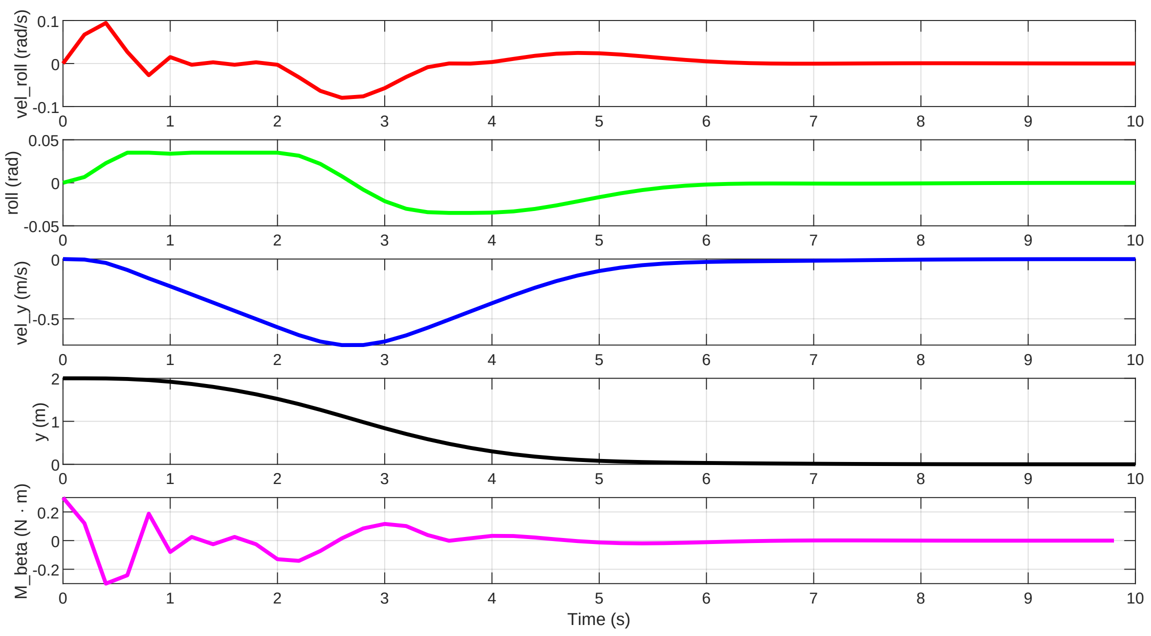
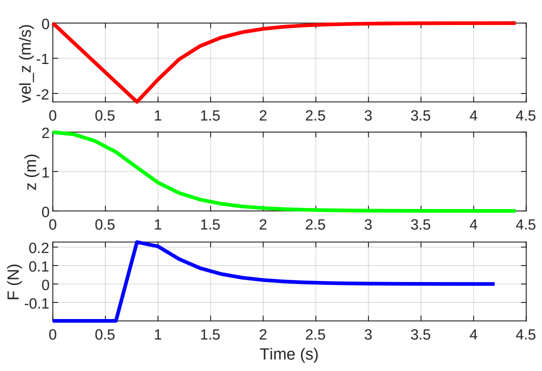


FIGURE 6: Quadcopter state and input variations of subsystem y under regulating controller



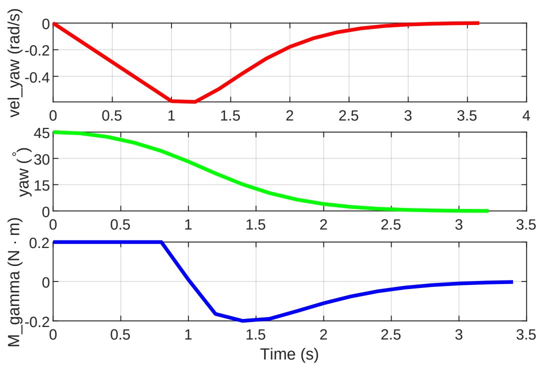


FIGURE 7: Quadcopter state and input variations of subsystem z under regulating controller

FIGURE 8: Quadcopter state and input variations of subsystem yaw under regulating controller

subsystem can be formulated as follows:

$$\begin{aligned} \min\ & u_s^{\mathsf{T}} R_s u_s \\ \text{s.t. } & x_s = A x_s + B u_s \\ & C x_s = r \\ & H_x x_s \le h_x \\ & H_u u_s \le h_u \end{aligned}, \tag{8}$$

where A, B, and C correspond to discrete time state-space model matrices. Further, the state and input constraint parameters (H_x, h_x, H_u, h_u) for each controller are defined in Table 2 according to the given problem description.

We can obtain the expected steady-state (x_s, u_s) by solving equation (8) with gurobi solver. If there exists a steady state (x_s, u_s) that keeps system at target, the main idea of second step is to treat set point tracking as regulation problem with a coordinate transformation compared to the origin.

By applying the original regulation problem in delta-formulation, the racking problem can be formulated as follows:

$$\begin{aligned}
\min \sum_{i=0}^{N-1} & \Delta x_i^{\mathsf{T}} Q \Delta x_i + \Delta u_i^{\mathsf{T}} R \Delta u_i + \Delta x_N^{\mathsf{T}} Q_f \Delta x_N \\
\text{s.t. } & \Delta x_0 = \Delta x \\
& \Delta x_{i+1} = A\Delta x_i + B\Delta u_i \\
& H_x \Delta x_i \leq h_x - H_x x_s \\
& H_u \Delta u_i \leq h_u - H_u u_s \\
& H_f (x_N - x_s) \leq h_f \\
& \Delta x_i \in \mathbb{X} \\
& \Delta u_i \in \mathbb{U}
\end{aligned} \quad , \tag{9}$$

where Δx, $\Delta x_i = x_i - x_s$, and $\Delta u_i = u_i - u_s$ are initial state, deviation variables of the evolved state and input, which satisfy the same model equations under LMPC. The terminal cost and constraint set can be obtained using LQR following the same methodology in equation (6).

After obtaining the optimal sequence $\Delta \mathbf{u}^*$, we can apply the input to the system by adding the offset as $u_0^* = \Delta u_0^* + u_s$.

In terms of the choice of the tuning parameters, the horizon N, the control matrix Q and R are the same as shown in Table 3, which work well for part 4.

Plot for each dimension of the system starting at the origin and tracking a reference to -2 meters from the origin (for x, y and z) and to 45 degrees for yaw

This section presents the tracking performance of each sub-system starting from the origin under LMPC. Figure 9, 10, 11, and 12 show that each controller can stabilize as the reference successfully. Compared to subsystem x and y, the other two subsystems are able to reach the given state faster.

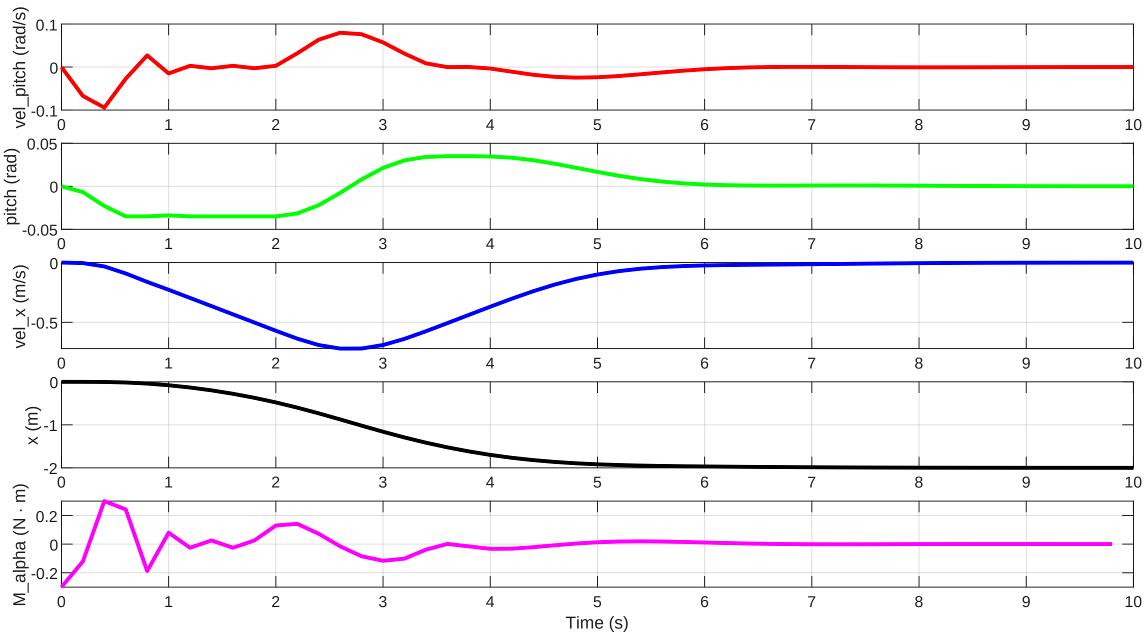


FIGURE 9: Quadcopter state and input variations of subsystem x under tracking controller

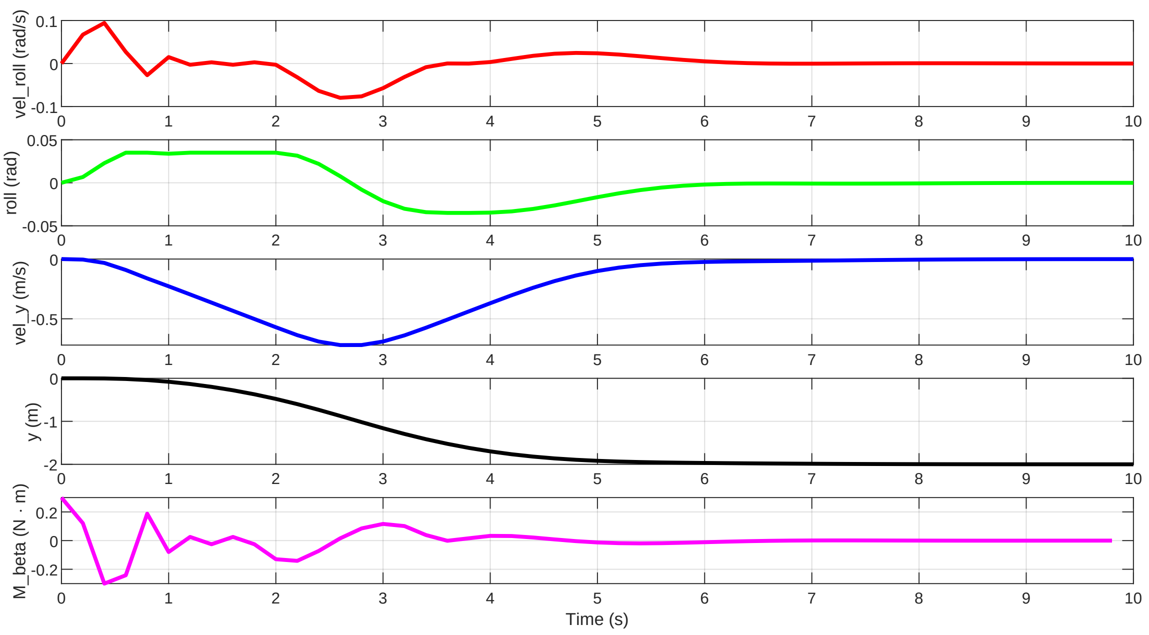


FIGURE 10: Quadcopter state and input variations of subsystem y under tracking controllery

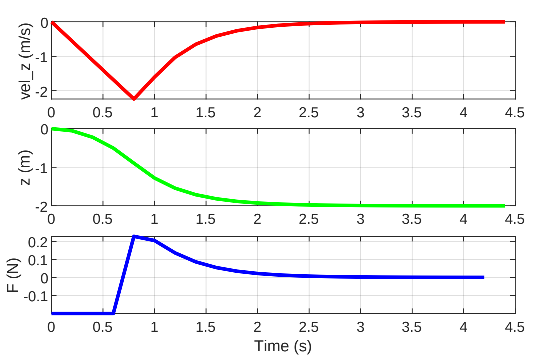


FIGURE 11: Quadcopter state and input variations of subsystem z under tracking controller

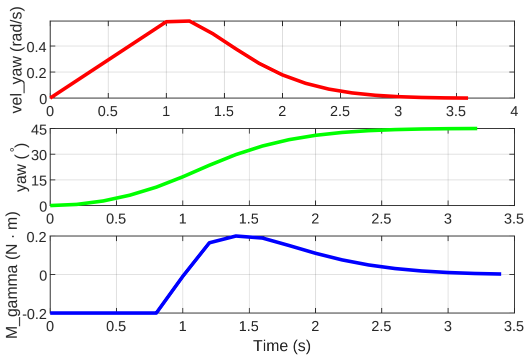


FIGURE 12: Quadcopter state and input variations of subsystem yaw under tracking controller

Related m-codes for the controllers are attached in Deliverable_3_2.zip. The main script for producing the results is `Deliverable_3_2.m`.

Part4 Simulation with Nonlinear Quadcopter

Deliverable 4.1

A plot of your controllers successfully tracking the path

This section presents the qualitative experimental results that all the controllers simultaneously working towards tracking a reference trajectory.

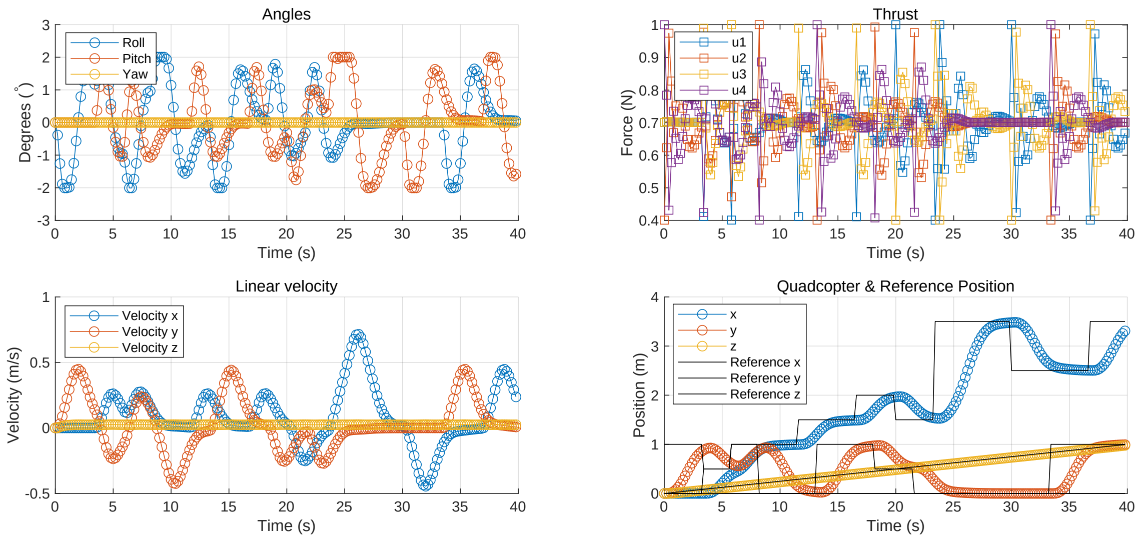


FIGURE 13: States and inputs with the proposed linear MPC

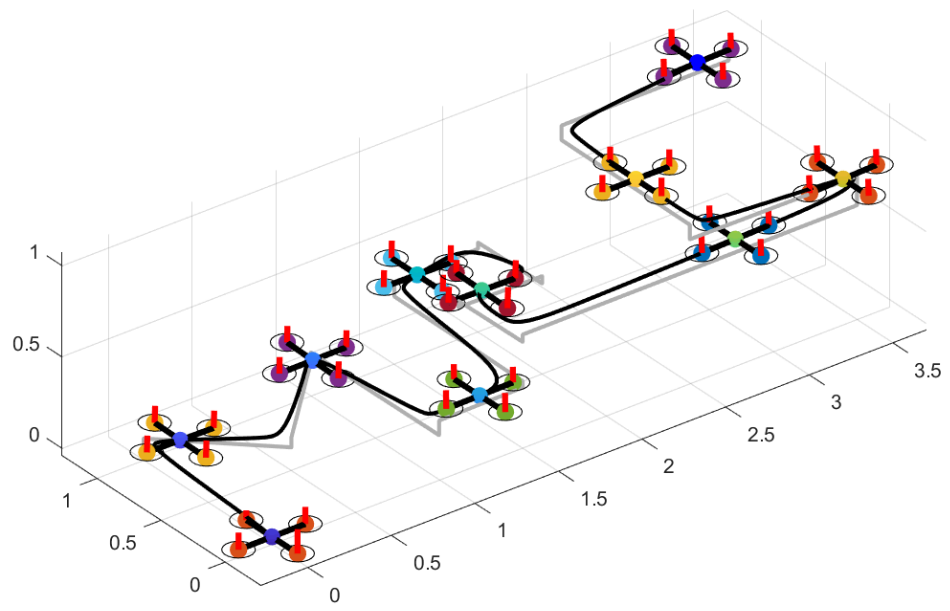


FIGURE 14: Orthographic view of the reference and quadcopter trajectories with linear MPC

Figure 13 shows the state and input variables of the quadcopter, i.e., angles, the linear velocities, the position, and the motor thrust. Especially, the quadcopter yaws slighltly along the track. The quadcopter can reach the designated location under MPC with latency. Figure 14 illustrates the quadcopter tracking references over the pre-defined path with good precision.

Related script for producing the results is `Deliverable_4_1.m`.

Part5 Offset-Free Tracking

Deliverable 5.1

Explanation of your design procedure and choice of tuning parameters

The augmented model of the system in the z-direction is

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Bd_k \\ d_{k+1} &= d_k \\ y_k &= Cx_k \end{aligned} \quad . \tag{10}$$

Then an observer (specifically, a Luenberger observer) is designed to estimate both the system state and the disturbance

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (C\hat{x}_k - y_k) \ . \tag{11}$$

The derived error dynamics are demonstrated as

$$\begin{aligned} \begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} &= \left(\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & 0 \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \\ &= \left(\hat{A} + \hat{L}\hat{C} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \end{aligned} \ . \tag{12}$$

Given that the errors should converge to zero, $\hat{L} = \begin{bmatrix} L_x \\ L_d \end{bmatrix}$ is tuned to ensure the matrix $\left(\hat{A} + \hat{L}\hat{C} \right)$ is Schur.

For the details of tuning parameters, $\hat{L}$ is computed as follows

```
F = rand(n+m, 1)./5 + 0.1;
L = -place(A_bar', C_bar', F)';
```

With respect to the eigenvalues of the observer, obviously small norms will induce fast convergence. On the other hand, small norms could increase the initial overshoot of disturbance estimation. Hence, the norms are constrained between 0.2 and 0.3 (The time of convergence hardly decreases with smaller norms).

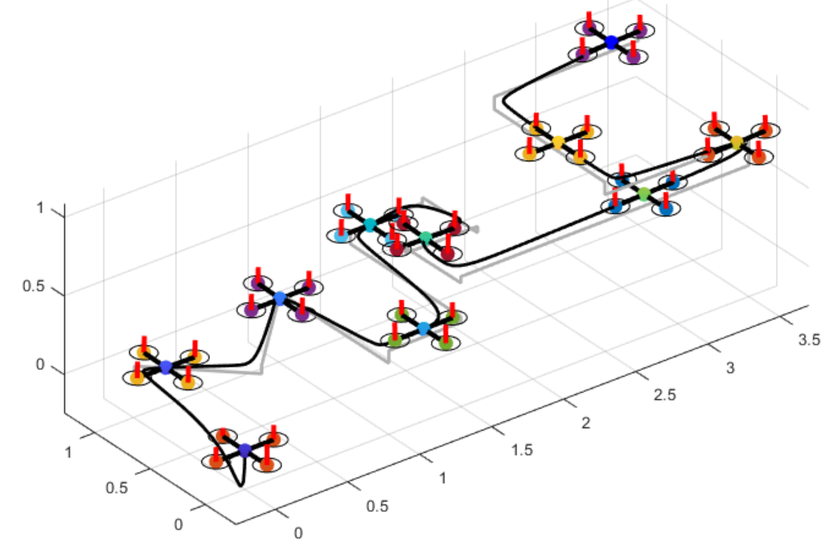
The next step is to compute steady state target based on the estimated state and disturbance. The process is accomplished by solving the following convex optimization problem

$$\begin{aligned} \min_{u_s} \quad & u_s^{\mathsf{T}} R_s u_s \\ \text{s.t.} \quad & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B\hat{d} \\ r \end{bmatrix} \\ & H_u u_s \leq h_u \end{aligned} \tag{13}$$

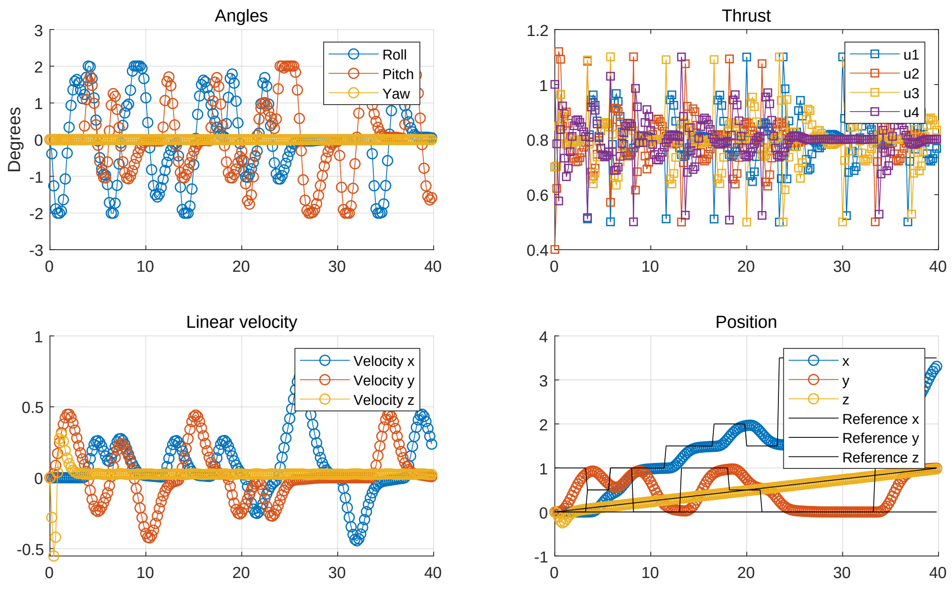
where $H_u = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $h_u = \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$ in subsystem z. The rest of the problem is quite straightforward and could be solved through similar procedures as stated in Deliverable 3.2 – a MPC problem for tracking in delta-formulation.

Plot showing that the z-controller achieves offset-free tracking

Figure 15 illustrates the performance of offset-free z-controller while Figure 16 serves as a baseline, i.e. non-offset-free controller. Related m-codes for the controllers are attached in Deliverable_5_1.zip. The main script for producing the results is `Deliverable_5_1.m`. One can obtain Figure 16 by uncommenting line 193 in `MPC_Control_z.m`.

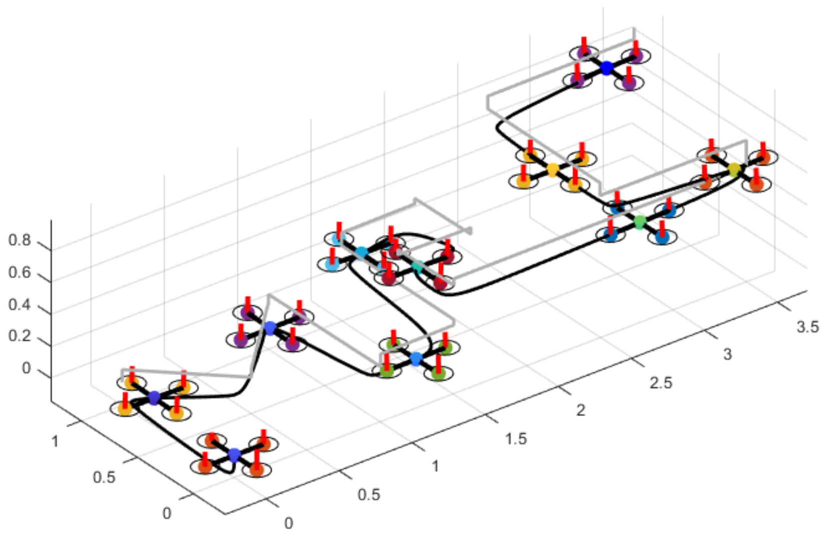


(a) Orthographic view

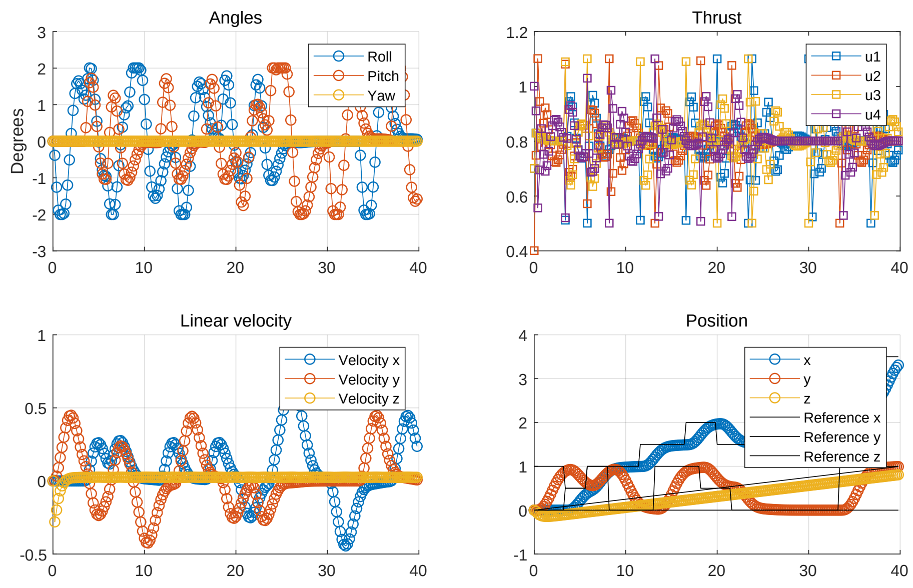


(b) States and inputs

FIGURE 15: Performance of offset-free controller



(a) Orthographic view



(b) States and inputs

FIGURE 16: Performance of non-offset-free controller

Part6 Nonlinear MPC

Deliverable 6.1

Explanation of your design procedure and choice of tuning parameters

Compared to LMPC, Nonlinear MPC (NMPC) takes the full state $\mathbf{x}$ as input and takes control input as $\mathbf{u}$ instead of $\mathbf{v}$. In our NMPC controller, the RK4 is employed as the integration approximation with the sample period of $h = 0.2s$.

$$\mathbf{x_{k+1}} = f_{RK4}(\mathbf{x_k}, \mathbf{u_k}) = \mathbf{x_k} + h(\frac{\mathbf{k_1}}{6} + \frac{\mathbf{k_2}}{3} + \frac{\mathbf{k_3}}{3} + \frac{\mathbf{k_4}}{6}) \, , \tag{14}$$

where

$$\begin{aligned} \mathbf{k_1} &= f(\mathbf{x_k}, \mathbf{u_k}) \\ \mathbf{k_2} &= f(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{u_k}) \\ \mathbf{k_3} &= f(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{u_k}) \\ \mathbf{k_4} &= f(\mathbf{x_k} + h\mathbf{k_4}, \mathbf{u_k}) \end{aligned} \, . \tag{15}$$

The state constraints listed in Table 1 can be modified in NMPC controller as the former constraint is just to ensure the validity of our linear approximation. But, we still need constraints on the roll angle and pitch angle to keep the quadcopter from turning over. Therefore, we let the two angles not exceed $\frac{\pi}{6}$ and the constraints are satisfied recursively.

$$-0.5236 < x_4 < 0.5236, -0.5236 < x_5 < 0.5236 \, . \tag{16}$$

The recursive constraint for all control input u_i $(i = 1, 2, 3, 4)$ is:

$$0 \leq u_i \leq 1.5 \tag{17}$$

In order to track, the steady state $\mathbf{x_s}$ and input $\mathbf{u_s}$ are defined and satisfy the constraints stated in equation (16) and equation (17). Besides, the values of $\mathbf{x_s}$ and input $\mathbf{u_s}$ should match the steady state property, which is:

$$\mathbf{x_s} = f_{RK4}(\mathbf{x_s}, \mathbf{u_s}) \, . \tag{18}$$

For tracking, some value in $\mathbf{x_s}$ should be equal to the reference $\mathbf{r}$:

$$x_{s,10} = r_1, x_{s,11} = r_2, x_{s,12} = r_3, x_{s,6} = r_4 \, . \tag{19}$$

The objective cost function to be minimized is defined as:

$$J = \sum_{i=0}^{N-1} (\mathbf{x_i} - \mathbf{x_s})^\intercal Q(\mathbf{x_i} - \mathbf{x_s}) + (\mathbf{u_i} - \mathbf{u_s})^\intercal R(\mathbf{u_i} - \mathbf{u_s}) + (\mathbf{x_N} - \mathbf{x_s})^\intercal Q_f(\mathbf{x_N} - \mathbf{x_s}) \, . \tag{20}$$

The horizon length N is set to be 20. With several experiments, we found that setting larger value on Q_f has a slight improvement on the controller. Therefore, Q_f is tuned to be equal to Q for convenience.

Initially, Q is intuitively tuned as diag(1,1,1,1,1,10,1,1,1,10,10,10) and R is tuned as I since we want to penalize more on the deviation of the position. As we can see from Figure 19, the x and y position of the quadcopter track the reference very well. However, Figure 17 indicates a vibration on the z position when t is near 25s. According to Figure 18, there's a huge deviation on x position during that time so that the controller implements inputs that focus on fixing the deviation on x. Due to the limitation of the thrusts of the rotors, the other states (z) can not be guaranteed to follow the reference as the all the thrusts are implemented to fix x, which leads to the oscillation during that time. Generally, this set of parameters perform well, but has some flaws, which need some modification and improvement.

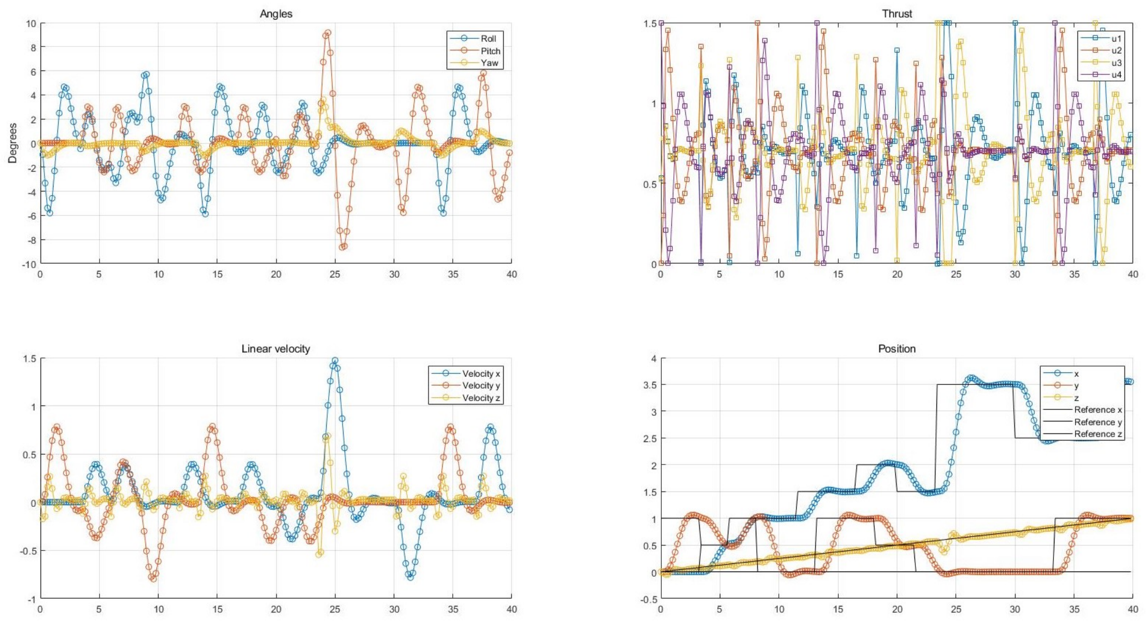


FIGURE 17: States and inputs under the proposed NMPC1

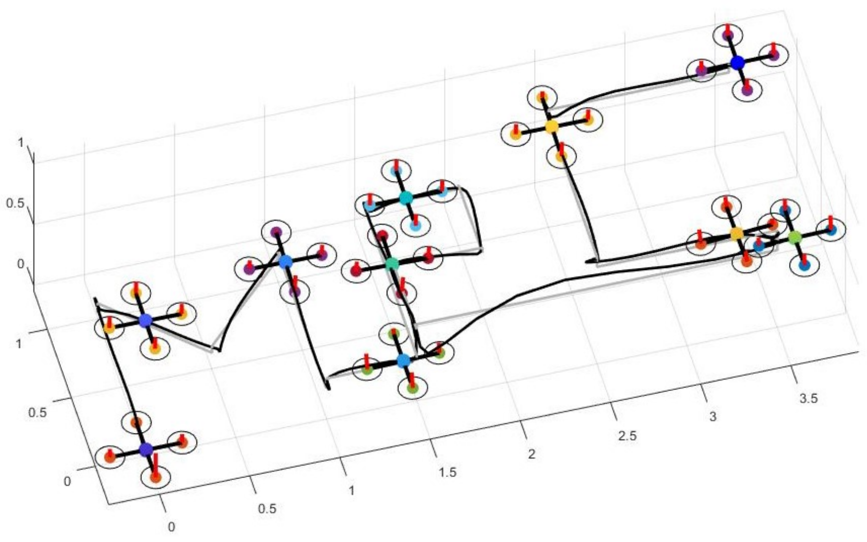


FIGURE 18: Orthographic view of the reference and quadcopter trajectories with NMPC1

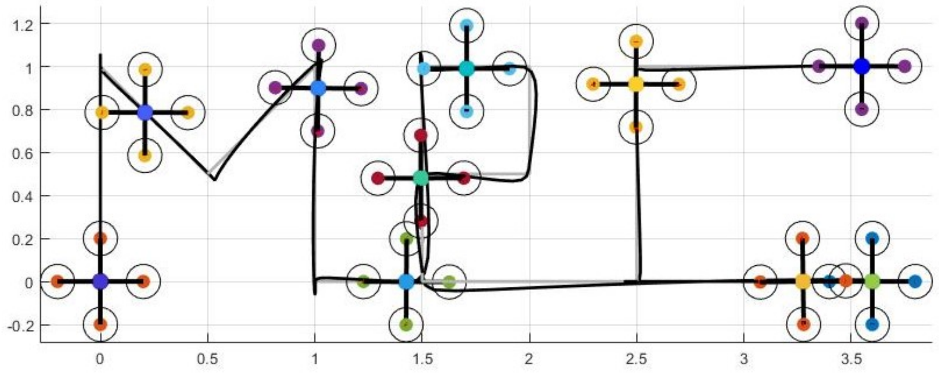


FIGURE 19: Top view of the reference and quadcopter trajectories with NMPC1

In this specified case, the z position changes linearly with the time, while x,y position behave as step changes. Meanwhile, there's a huge step difference on the reference of x position. Therefore, the deviation of z should be penalized more in order to solve the oscillation problem is stated before. Thus, our final Q is modified to be $\mathrm{diag}(1,1,1,1,1,5,1,1,1,5,5,50)$ and R remains to be I. As shown in Figure 20 and 21, the quadcopter follows the reference quite well and we can clearly see the characters "MPC".

Another possible way to solve the oscillation problem is penalizing more on the velocity of x, y and z or setting constraints on the velocity. This could prevent x from overusing the thrusts so that z can have enough inputs to track the reference. Nevertheless, it could also let the quadcopter approach the reference slower. In our experiments, it performs worse in this specified trajectory than the controller described in the last paragraph. So, we decide not to take this controller as our final one. But, if we don't know specifically what's our reference is, this could be a better option.

Explanation of why your nonlinear controller is working better than your linear one.

Compared to the linear one, the nonlinear controller has several advantages as follows:

First of all, the constraints on the row angle and pitch angle can be loosen. The constraints in the linear controller are so tight and are used to ensure the linear approximation. They're not physical constraints. As we can see in Figure 13, the angle of row and pitch reach the constraint in the simulation controlled by LMPC. This limits the speed of the quadcopter to approach the reference. In NMPC, this is not exactly an issue, we just need to ensure the quadcopter won't turn over. According to Figure 20, the maximum angle is 10 degree, which is quite safe.

Moreover, the NMPC controller takes the full state of the quadcopter as input. Without decomposition, the controller can exploit the full system dynamics. In some motions required several subsystems to couple, taking full system dynamics can lead to faster response compared to LMPC controller. This may also lead to some overshoot. In our case, the overshoot is trivial due to proper selection of the parameters.

As shown in Figure 21, the quadcopter under NMPC controller follows the reference quite well and performs much better in agile scenarios.

Plots showing the performance of your controller.

The performance of our controller is showed in Figure 20, 21 and 22.

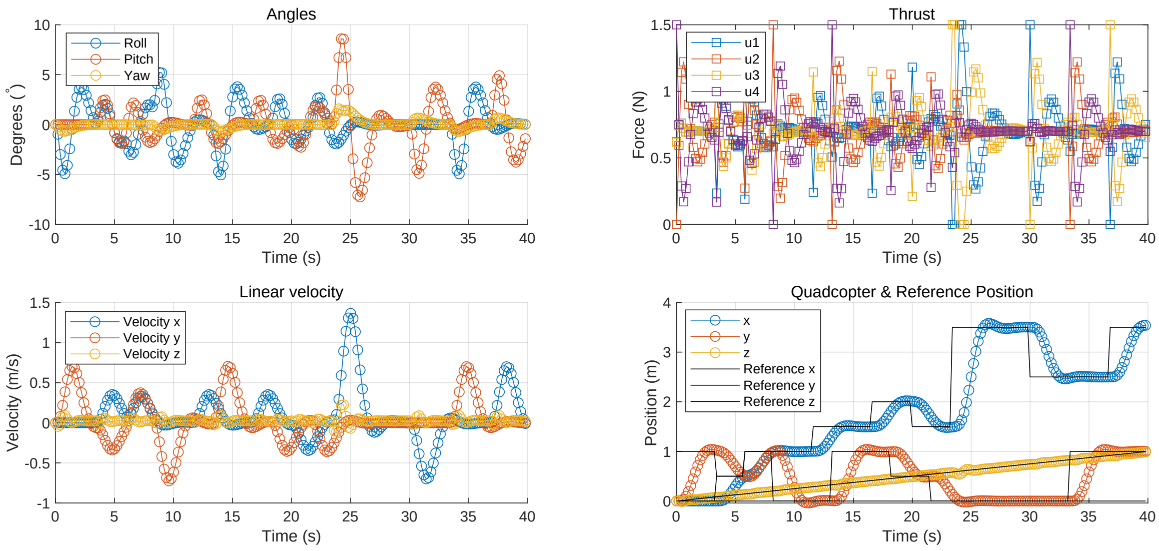


FIGURE 20: States and inputs under the proposed NMPC2

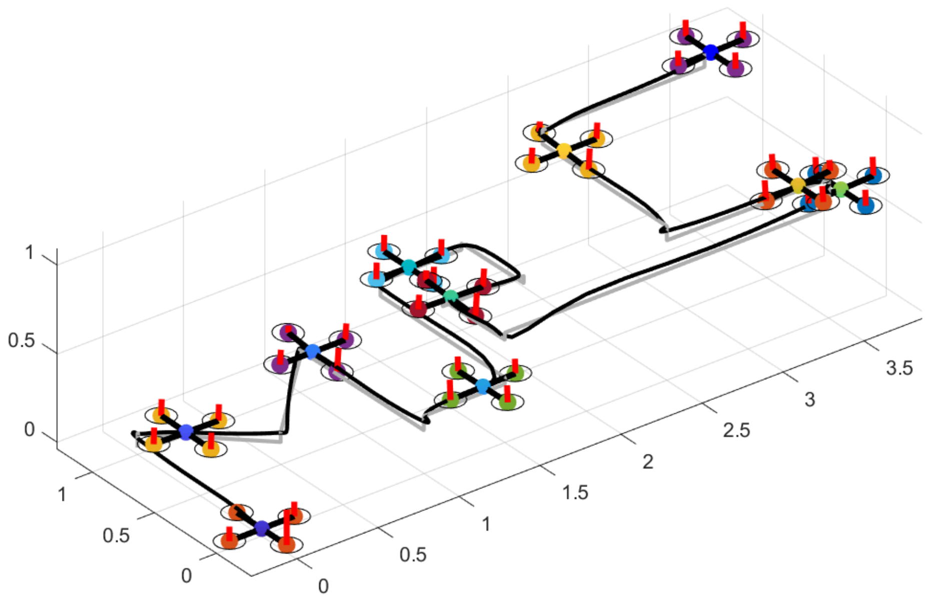


FIGURE 21: Orthographic view of the reference and quadcopter trajectories with NMPC2

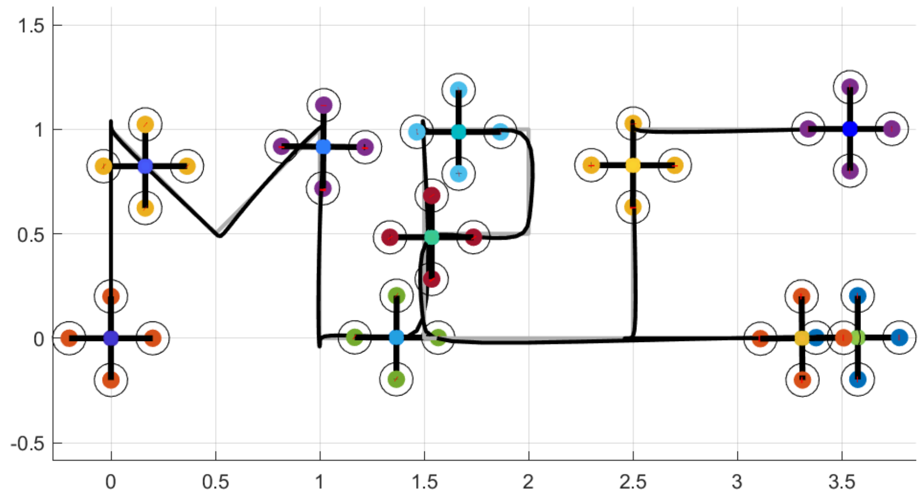


FIGURE 22: Top view of the reference and quadcopter trajectories with NMPC2

Related m-codes for the controllers are attached in Deliverable_6_1.zip. The main script for producing the results is `Deliverable_6_1.m`.