[IFAC2023]
MATHWORKS MINIDRONE COMPETITION

서울시립대학교 제어 및 동역학 연구실

(김인겸 변순석 박원영 김준수)

2023년 05월 03일

목차

1 MPC intro

1.1 mpc 설명

다음과 같은 이산 시불변 시스템에 대해 생각해보자

$$\begin{aligned} \overline{x}_{k+1} &= A\overline{x}_k + B\overline{u}_k \\ \overline{y}_k &= C\overline{x}_k \end{aligned} \tag{1}$$ eq:system

여기서 $\overline{x}_k$와 $\overline{u}_k$는 시간이 k일때의 시스템의 상태변수와 입력이다. 시스템 (1)는 가제어성과 가관측성을 만족하고 다음과 같은 상태변수와 입력에 대한 제약조건을 만족한다고 가정하자

$$F\overline{x}_k + G\overline{u}_k \leq 1, \quad \text{for all } i = 1, 2, \cdots \tag{2}$$ eq:costraints_on

F와 G는 constant matrix이다. 시스템 (1)에 대해 다음과 같이 predicted cost function을 정의한다

$$J(x_{0|k}, \{u_{0|k}, u_{1|k}, u_{2|k}, \cdots\}) = \sum_{i=0}^{\infty} \left(\|\overline{x}_{i|k}\|_Q^2 + \|\overline{u}_{i|k}\|_R^2 \right) \tag{3}$$ eq:cost_function

이때 $x_{i|k}$와 $u_{i|k}$는 각각 k 스탭에서 예측한 k+i 스탭의 상태변수와 입력을 의미하고 (A, Q)는 가관측성을 만족한다고 가정한다.

1.2 Unconstrained Optimization

이번 섹터에서는 상태변수와 입력에 대한 제약조건이 존재하지 않는 이산 시불변 선형 시스템 (1)에 대해 생각해보자. 다음의 theorem은 제약조건이 없을 때의 cost function을 최소화하는 입력에 대해 설명한다

⟨thm:LQR⟩

THEOREM 1. *(Discrete time algebraic Riccati equation) 다음의 state feedback control law가 시스템 (1)와 임의의 초기값 x_0에 대해 cost function을 최소화 할 수 있다.*

$$\overline{u} = K\overline{x} \tag{4}$$ eq:optimal_input

여기서 K는 다음과 같다

$$K = -(B^\top PB + R)^{-1}BPA \tag{5}$$ {?}

$$P = A^\top PA + Q - A^\top PB(B^\top PB + R)^{-1}B^\top PA \tag{6}$$ {?}

(4)와 같은 입력을 인가하였을 때 cost function (8)은 다음과 같다

$$J(\overline{x}_0) = \overline{x}_0^\top P\overline{x}_0 \tag{7}$$ {?}

1.3 Dual-mode Framework

MPC를 간략히 설명하면 다음과 같다. 매스탭마다 시스템의 상태변수 $\overline{x}_k$를 피드백 받아 cost function을 최소화하는 입력시퀀스 $\overline{u}_0^*, \overline{u}_1^*, \cdots$을 계산한 후 계산된 입력시퀀스 중 첫번째 입력값를 시스템$\overline{u}_0^*$에 입력한다.

Dual-mode framework에서는 prediction horizon을 Mode1과 Mode2로 나누어서 생각한다. mode1은 0 스탭부터 N 스탭까지의 구간을 의미하며, mode2는 N+1 스탭부터 무한대의 스탭까지를 의미한다. mode1에서의 입력은 QP에 의해 계산되며, mode2에서의 입력은 THEOREM 1에서 명시된 제약조건이 없을때 cost function을 최소화시키는 제어 법칙 (4)에 의해 계산된다. 그러므로 예상되어지는 predicted cost function은 다음과 같다.

$$J(x_{0|k}, \{u_{0|k}, u_{1|k}, u_{2|k}, \cdots\}) = \sum_{i=0}^{N-1} \left(\|\overline{x}_{i|k}\|_Q^2 + \|\overline{u}_{i|k}\|_R^2\right) + \|\overline{x}_{i|k}\|_P^2 \tag{8}$$ eq:cost_function

1.4 Maximal Positively Invariant Set

In last subsection, we introduce the Dual-mode Framework of MPC. But one may wonder if the control law (4) satisfy the constraints (2) on state and input. In this subsection we define a terminal set which is invariant to system (1) and (4), and all components of which satisfy the constraints (2).

defn:MPIset⟩?

DEFINITION 1. *집합 $\mathcal{X}$에 대하여 $\mathcal{X}$의 임의의 원소 x가 다음 명제들을 만족한다면 system* (1) *과 control law* (4) *와 constraints* (2)*에 대해 positively invariant하다고 하자.*

- $\forall x \in \mathcal{X}$, $(A + BK)x \in \mathcal{X}$
- $\forall x \in \mathcal{X}$, $(F + GK)x \leq 1$

DEFINITION 2. *system* (1) *과 control law* (4) *와 constraints* (2)*에 대한 모든 positively invariant set의 합집합을 maximal positively invariant set(MPI set)이라 한다.*

선형시스템과 선형성을 가지는 constraints에 대하여 MPI set은 다음의 theorem을 통해 얻을 수 있다.

THEOREM 2. *system* (1) *과 control law* (4) *와 constraints* (2)*에 대한 MPI set은 다음과 같다.*

$$\mathcal{X}^{MPI} := \{x | (F + GK)(A + BK)^i x \leq 1, \;\; i = 0, \cdots, \nu\} \tag{9}$$ {?}

이때 ν는 $(F+GK)(A+BK)^i x \leq 1$, $i = 0, 1, \cdots, \nu$를 만족하는 x에 대해 $(F+GK)(A+BK)^\nu x \leq 1$를 만족시키는 가장 작은 정수를 의미한다. 만약 A+BK가 strictly stable하고 (A+BK,F+GK)가 observable하다면, ν는 유한한 정수이다.

Proof. **p.22 theorem2.2**를 볼 것. □

1.5 Implementation of MPC

각 스탭마다 시스템으로부터 상태변수를 피드백 받아 다음 최적화 문제를 풀게 된다.

Problem 1. *시간이 t일때 피드백된 상태변수의 값이 x라고 하면, 즉 $\overline{x}_k = x$일때*

$$\min_{\overline{U}^N} \sum_{i=0}^{N-1} \left(\|\overline{x}_{i|k}\|_Q^2 + \|\overline{u}_{i|k}\|_R^2 \right) + \|\overline{x}_{N|k}\|_P^2 \tag{10}$$
$$over\ \ \overline{U}^N = (u_{0|k}, u_{1|k}, \cdots, u_{N|k}) \in \mathbb{U}^N$$

raints_of_qp⟩

$$subject\ to \begin{cases} \overline{x}_{0|k} = \overline{x} & \text{(11a)} \\ F\overline{x}_{i|k} + G\overline{u}_{i|k} \le 1,\ \ for\ i = 0, 1, \cdots & \text{(11b)} \\ \overline{x}_{N|k} \in \mathbb{X}_T & \text{(11c)} \\ \overline{x}_{i+1|k} = A\overline{x}_{i|k} + B\overline{u}_{i|k} & \text{(11d)} \end{cases}$$

여기서 $\mathbb{X}_T$는 terminal set을 의미하는데 terminal set의 정의는 다음 섹터에서 설명하겠다. 상태변수를 피드백받아 Problem 10를 계산하게 되면 N개의 입력 $u_0, u_1, \cdots, u_N$이 계산되는데 이때 u_0를 시스템이 입력하게 된다. 매 스탭마다 상태변수를 피드백 받아 이러한 최적화 문제를 풀어 계산된 입력값을 시스템에 입력하는 것이 MPC의 기본 철학이다.

1.6 Reculsive Feasiblity and Stability

이번 섹션에서 우리는 cost function의 최소값이 closed-loop system의 trajectories을 따라 증가하지 않는다는 사실을 보임으로써 closed loop system의 reculsive feasibility와 stability를 증명할 것이다.

DEFINITION 3. feasible set : *constraints* (15)*를 만족하시키는 입력시퀀스가 존재하는 상태변수를 원소로 하는 집합을 feasible set $\mathbb{X}_f$이라 한다.*

$$\mathbb{X}_f = \{x_0 |\, for\ x_0,\ \ \exists U^N\ satisfying\ constraints\ (15)\} \tag{12}$$

thm:recusive⟩

THEOREM 3. recusive feasibility : *만약 $x_0 \in \mathbb{X}_f$이면, 모든 k에 대해 $x_k \in \mathbb{X}_f$이다.*

Proof. **p.26 theorem 2.4** 볼 것 □

위의 theorem의 의미를 살펴보기 위해 theorem의 statement의 부정을 취해보자.

$x_0 \in \mathbb{X}_f$이지만, 어떤 k에 대해 $x_k \notin \mathbb{X}_f$이다.

비록 초기값이 feasible set에 속해있어서 초기에는 MPC를 수행할 수 있지만 어떤 k에 대해서 feasible set에 속해있지 않으므로 MPC를 수행할 수 없을 수도 있다. 그러므로 theorem (3)를 만족해야 MPC를 끝까지 수행할 수 있다.

THEOREM 4. *임의의 시퀀스* $\{a_0, a_1, \cdots\}$*에 대하여* $\sum_{i=0}^{\infty} a_i$*가 임의의 값으로 수렴한다면* i*가 무한히 커질 때* a_i*는* 0*으로 수렴한다.*

Proof. **p.29 theorem2.7 theorem2.8** 볼 것 □

2 MPC Concept in minidrone

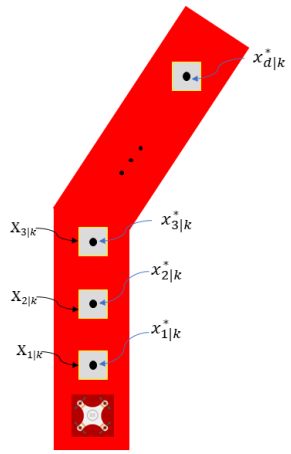

그림 1: 시간 k일때 예측한 d개의 경유 지점과 그에 따른 정사각형

⟨fig:path⟩

먼저 미니드론에 장착되어진 카메라가 0.2초마다 경로에 대한 이미지를 촬영한다. image processing 블럭에서 이미지에 있는 경로 위에 드론이 경유해야 하는 d개의 점의 좌표를 계산한다. 각 점들을 중심으로 하는 변의 길이가 δ인 정사각형들을 만들어 미니드론이 각 스탭마다 해당되는 각각의 정사각형 안에 위치하도록 경로에 대한 constraints를 설정하고 cost function의 최소화하는 미니드론의 경로를 계산한다. 경로에 대한 constraint에 대한 설명을 그림으로 묘사하면 그림 1 같다.

이때 미니드론의 카메라가 이미지를 촬영할 때마다 촬영된 경로의 형태가 달라지고 image processing에서 받은 점들의 좌표들이 달라지므로 경로에 대한 constraints는 time-varying임을 알 수 있다.

먼저 선형화된 미니 드론의 모델은 다음과 같다

$$\begin{aligned} \overline{x}_{t+1} &= A\overline{x}_t + B\overline{u}_t \\ \overline{y} &= C\overline{x}_t \end{aligned} \tag{13} \{?\}$$

다음과 같은 optimization problem을 통해 mpc를 설계하였다

Problem 2. *카메라가 이미지를 촬영할 당시의 미니드론의 상태변수를* $\overline{x}_k$*라고 할때*

$$\min_{\overline{U}^{N-1}} \sum_{k=0}^{N} \left(\|\overline{x}_{i|k} - \overline{x}^*_{d|k}\|^2_Q + \|\overline{u}_{i|k}\|^2_R \right) + \|\overline{x}_{N|k} - \overline{x}^*_{d|k}\|^2_Q \tag{14}$$ {?}

$$\textit{over } \overline{U}^N = (u_{0|k}, u_{1|k}, \cdots, u_{N|k}) \in \mathbb{U}^N$$

$$\textit{subject to} \begin{cases} \overline{x}_t = \overline{x}_{0|k} \\ \overline{x}_{i|k} \in \mathbb{X}_{i|k} \textit{ for } k = 0, 1, \cdots, N \end{cases} \tag{15}$$ eq:constraints_o

이때 $\mathbb{X}_{i|k}$*는 위에서 말한 미니드론이 경유해야 하는 정사각형의 영역을 의미하고* $\overline{X}^*_{i|k}$*는 image processing 블럭에서 시간* k*일때 계산된* i*번째 점의 좌표를 의미한다.*

3 MPC 구현

3.1 사용한 모델링

사용한 드론 모델의 경우는 "Quadrotor control: modeling, nonlinear control design, and simulation" 에서 사용하는 모델링을 따르고 있다.

$$x = [\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad u \quad v \quad w \quad x \quad y \quad z]^\top \tag{16}$$ {?}

$$u = [f_t \quad \tau_x \quad \tau_y \quad \tau_z]^\top \tag{17}$$ {?}

state x와 input u는 위와 같이 구성되어 있다. 각각의 성분에 대한 자세한 내용은 위 PDF를 보면 확인이 가능하다. 우리는 이 방식에 대해서 taylor expansion을 사용한 linearlization을 통해 모델링을 재구성 했다. 이 linearization 방식도 위 pdf 2.7장에서 확인이 가능하다.

다만 이 방식에서 특이한 점이 있다면 $f(\bar{x}, \bar{u}) = 0$을 기준으로 system을 구성하고 있다.$\bar{x}$는 an equilibrium point, $\bar{u}$는 solution of the algebric system을 의미하고 있다. 이렇게 구성할 경우에 x와 u의 형태는 equilibrium point에서 아래와 같이 구성된다.

$$\bar{x} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \bar{x} \quad \bar{y} \quad \bar{z}]^\top \tag{18}$$ {?}

$$\bar{u} = [mg \quad 0 \quad 0 \quad 0]^\top \tag{19}$$ {?}

이렇게 구성된 $\bar{x}, \bar{u}$는 치명적인 우리가 다루는 mpc에서 치명적인 단점을 가지고 있다.

위 $\bar{x}, \bar{u}$를 보면 몇몇 요소들을 제외하고 전부 0이란 값을 가지고 있는 것을 볼 수 있다. 즉 위의 state는 원하는 x,y,z에 도달할 경우 그 자리에서 멈추는 경우를 상정하고 풀어나가는 것을 의미한다. 다만 우리 mpc에서 원하는 것은 매 step마다 멈추고 이동하는 것이 아니라 각 step마다 움직임이 유기적으로 지속되는 것을 원하는 상태라 요구사항이 일치하지 않는다.

이 문제를 해결하기 위해 뒤에서 이야기 하겠지만, mpc를 풀기위해 새로 정의하는 U에 x의 요소들을 넣어서 해결했다. 다만, 이러한 해결 방법은 현재 minidrone에서 사용되는 제어기가 특수한 형태라 이러한 사용이 가능하다는 점을 기억해야한다. 즉, minidrone에서의 mpc는 x,y,z 좌표만 넘겨주고 u의 값에는 전혀 관여를 안 하고 있기 때문에 사용가능하다는 이야기다.

만약 U에 x를 넣지 않고 설계를 하면, 초반 step 3개 정도 동안 멈춘 드론을 움직이기 위해 무의미한 소비 시간이 사용이되고, 그 후 자신의 constraints를 지키기 위해 엄청난 jumping이 존재하게 된다. 따라서, 이러한 제어기를 구성하고도 위와 같은 제어기를 사용하게 됐다. 위 제어기는 u의 값은 이상한 값을 뽑아내지만, x,y,z에 한해서는 유의미한 결과를 만들어주고 있기 때문이다.

3.2 mpc 구현 방법

linearlization을 통해 얻은 space state는 "Quadrotor control: modeling, nonlinear control design, and simulation"여기서 확인이 가능하고 이 pdf에서 사용한 A,B,d,D를 그대로 가져와서 사용하기로 한다. 다만, ideal한 실내에서 구동하는 것을 전제로 설계를 진행하여 외부 바람 요인와 같은 disturbance를 의미하는 $d = [0]$이라 하고 구현을 했다.

3.2.1 Formulation of linear MPC

k=0,1,2,..., k is time instant.

$$\begin{aligned} \min_{u(0|k),u(1|k),\cdots} J_N &= \frac{1}{2}\sum_{i=0}^{N-1}\left(x(i|k)^T Q x(i|k) + u(i|k)^T R u(i|k)\right) \\ \text{subject to} \quad & x(i+1|k) = Ax(i|k) + Bu(i|k), \forall i \\ & u(i|k) \in U, x(i|k) \in X, \forall i = 0, 1, \cdots \\ & `x(0|k) = x(k). \end{aligned} \tag{20}$$ eq:mpc_problem

Q, R 는 cost에서 x,u 각각에 대한 weight를 의미한다.
수식(20)은 0부터 N-1 번째까지의 u를 이용하여 최소값을 가지는 J를 찾는 문제이다. MPC는 위 문제를 한번에 모든 i에 대해 한번에 문제를 푸는 것을 목표로 하고 있어 새로운 $\bar{X}, \bar{U}$를 구성한다.

$$X(k) := \begin{bmatrix} x(1|k) \\ x(2|k) \\ \vdots \\ x(N|k) \end{bmatrix} \in \mathbb{R}^{nN}, \quad U(k) := \begin{bmatrix} u(0|k) \\ u(1|k) \\ \vdots \\ u(N-1|k) \end{bmatrix} \in \mathbb{R}^{mN}, \tag{21}$$ {?}

또한,Γ, Π를 사용하여 $\bar{X}(k) = \Gamma x(k) + \Pi \bar{U}(k)$를 구성하기로 한다. 이렇게 문제를 바꾸면 QP 문제가

$$\begin{aligned} \min_U \frac{1}{2} U^\top H U + q^\top U + \text{const.} \\ \text{subject to} \quad A_{\text{ineq}} U \le b_{\text{ineq}} \\ A_{\text{eq}} U = b_{\text{eq}} \end{aligned} \tag{22}$$ eq:optimal

수식 (22)와 같이 구성되어 U를 구할 수 있다.

$$\Gamma = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \in \mathbb{R}^{nN \times n} \tag{23}$$ {?}

$$\Pi = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ A^2B & AB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \in \mathbb{R}^{mN \times nN} \tag{24}$$ {?}

$$U(k) = \begin{bmatrix} u(1|k) \\ u(2|k) \\ u(3|k) \\ \vdots \\ u(N-1|k) \\ u(N|k) \end{bmatrix} \leq \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ \vdots \\ u_{max} \\ u_{max} \end{bmatrix} = \begin{bmatrix} I_N \otimes u_{max} \end{bmatrix} \tag{25}$$ {?}

$$-U(k) = -\begin{bmatrix} u(1|k) \\ u(2|k) \\ u(3|k) \\ \vdots \\ u(N-1|k) \\ u(N|k) \end{bmatrix} \leq -\begin{bmatrix} u_{min} \\ u_{min} \\ u_{min} \\ \vdots \\ u_{min} \\ u_{min} \end{bmatrix} = -\begin{bmatrix} I_N \otimes u_{min} \end{bmatrix} \tag{26}$$ {?}

수식 20에서의 Σ를 풀어서 쓰면 아래와 같이 표현이 가능하다.

$$J_N = \frac{1}{2}\bar{x}(0|k)^\top Q\bar{x}(0|k) + \frac{1}{2}(\bar{x}(1|k)^\top Q\bar{x}(1|k) + \bar{x}(2|k)^\top Q\bar{x}(2|k) + \cdots + \bar{x}(N-1|k)^\top Q\bar{x}(N-1|k))$$
$$+ \frac{1}{2}(\bar{u}(0|k)^\top R\bar{u}(0|k) + \bar{u}(1|k)^\top R\bar{u}(1|k) + \cdots + \bar{u}(N-1|k)^\top R\bar{u}(N-1|k)$$

여기서 J_n을 보명 x가 i=1 N-1까지 묶인 부분과 y가 i=0 N-1까지 묶인 부분을 볼 수 있다. 우린 이 파트를 묶어서 Part1, Part2라고 하자.

part1:$\bar{U}(k)^\top R_N \bar{U}(k)$

part2:$\bar{X}(k)^\top Q_N \bar{X}(k), (Q_N = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix})$

$$= (\Gamma x(k) + \Pi U(k))^\top Q(\Gamma x(k) + \Pi U(k))$$
$$= U(k)^\top(\Pi^\top Q\Pi)U(k) + 2(\Gamma x(k))^\top Q\Pi U(k) + x(k)^\top \Gamma Q\Gamma x(k)$$

이렇게 part1과 part2가 정리되기 때문에 이 정리된 값을 수식 22에 넣으면 H와 f에 대해서 얻을 수 있다. 그럼 constraints도 자연스럽게 아래와 같이 구성됨을 알 수 있다.

$$\begin{bmatrix} I \\ -I \\ \Pi \\ -\Pi \end{bmatrix} \bar{U} \leq \begin{bmatrix} I_N \otimes u_{\max} \\ -I_N \otimes u_{\min} \\ I_N \otimes x_{\max} - \Gamma x(k) \\ -I_N \otimes x_{\min} + \Gamma x(k) \end{bmatrix}$$

3.2.2 matlab 상 구현

우선 앞에서 언급한 것처럼 U에 대한 설정을 평소와는 좀 다르게 설정했다. $U = \begin{bmatrix} x & u \end{bmatrix}^\top$으로 설정을 하고 $\bar{U}$는 U를 horizon 만큼 쌓은 것을 의미한다. 이렇게 $\bar{U}$를 만들고 나서 3.2.1과 같은 방식으로 mpc에 대해 구성을 했다. 여기서 추가되는 부분은 크게 3가지가 있는데 1개는 destination을 지정하는 것과 추종할 reference를 넣어주는 것, 마지막으로 ϵ을 이용하여 reference에 네모난 constraints를 만들어주는 부분이다.

1. destination 지정

사실 이 부분은 크게 어렵지 않은 부분이다. cost function이 구성되는 부분에서 $x^\top Q x$와 같이 표현된 부분에 destination을 넣어서 $(x - x_{desti} \backslash Jx)^\top Q (x - x_{desti})$와 같이 구성하고 mpc를 만들면 된다. u의 경우는 우리의 입장에서 크게 필요하지 않아 일단 0로 잡아서 그래도 둬도 상관이 없다. 이렇게 구성을 하면 f가 아래와 같이 바뀌게 된다.

$$f = x_0^\top \Gamma * Q * \Pi - desination * Q * \Pi$$

여기서 destination은 목적지를 의미하는 x_{desti}가 horizon 개수만큼 vector로 쌓여 있는 것을 의미한다. 다만, 우리 mpc에서는 각 step마다 reference를 따라가도록 만들었기 때문에 x_{desti}가 동일하지는 않고 각 step마다 위치해 있어야 할 reference를 의미하고 있다.

2. reference 만들기

reference는 아래 section에서 설명하도록 하겠다.

3. epsilon을 이용하여 constraints를 만들기

epsilon이 의미하는 바는 각 step에서의 reference를 기준으로 상하좌우에 epsilon 만큼의 여유 공간을 주겠다는 것이다. 그럼 mpc에서 문제를 풀 때, 그 공간 내에서 step을 다음 움직임을 정하게 된다. epsilon은 12 by 1 matrix가 horizon만큼 쌓여있는 형태이다. 10, 11 번째 요소가 x,y를 의미하기 때문에 이 2가지 epsilon parameter 선정에 유의하면 된다.

추가로 코드 마지막에 y=Gamma*x+PI*U; 다음과 같이 있는데 이건 u의 값들을 빼고 x의 값들만 보기 위해 설정해둔 부분이다. 우리는 U에 x,y가 다 들어있기 때문에 U자체를 16주기로 10,11에 대해 확인하면 동일하게 x,y를 확인할 수 있다.

4 reference 생성

4.1 생성 방법

old x,old y, the old x, the old y 라는 parameter를 사용하여 드론의 이전 위치, 이전 이전 위치에 대한 값을 저장했다. 그리고 현재 드론 위치에서 visaul쪽에서 부터 받은 위치 2개를 close x, close y, far x, far y로 받아오고 있다. over x, over y라는 것은 close와 far 두 점을 직선으로 이어서 x,y 각각 (far-close)*overstep만큼 더 뒤로 보낸 것을 의미한다.
mpc에서 reference는 2가지 case로 나눠서 구성된다. the old와 현재 드론의 위치가 0.06보다 작은 경우와 큰 경우로 나눠진다.
먼저 0.06보다 작은 경우에 대해 다뤄보겠다.
우리 mpc의 horizon은 10이다. 이 10개의 step 나눠서 첫번째 2개 step 동안 the old로 이동한다. 그리고 다음 2 step 동안 old로 이동하고 다음 3개로 close, 나머지로 far로 이동한다.
다음으로는 0.06보다 큰 상황에 대해 다루겠다.
이 경우에는 step 1번에 old로 2 tunning11까지 close로 나머지로 over로 이동을 한다. 다만 tunning1과 tunning2라는 것을 사용하여 third dx=(close x-old x)/tuning1, forth dx=(over x-close x)/tuning2; 임의로 좀 더 늘리거나 줄이는 방식으로 reference를 조절할 수 있다.
이렇게 0.06을 기준으로 나눈 이유는 0.06보다 작은 경우는 주로 90도보다 작은 코너인 경우가 많기 때문에, 이 경우에 속도를 줄이기 위해 사용했다. 0.06보다 큰 경우는 over까지 사용하며 속도를 더 높이기 위한 구성을 갖추고 있다.

5 landing

5.1 landing flag 뜬 경우

mpc를 사용하여 이동하고 있다가 landing flag가 visual 쪽에서 부터 들어오면 landing이 시작된다고 보면 된다. 이 때, control part에서는 switch를 사용하여 mpc의 사용을 중단하고 dot follower block을 사용하여 reference만 추종하는 움직임을 보여준다.

5.2 착륙

dot follower를 따라 이동하고 있으면 그 다음 단에 있는 block을 통해 count가 세지기 시작한다. 0.05초 단위로 cnt 변수가 1씩 커지며 cnt가 400이 넘어가면 그 때 딱 들어온 reference 점으로 landing 점이 고정이 된다. 그 뒤로는 어떤 신호가 들어와서 이미 고정된 점까지만 이동한다. 그러다 그 점과 의 거리가 x,y 각각 0.001보다 작아지만 z의 값이 0으로 내리면서 착륙을 진행한다.