Paper Review

Lee, Daewon, H. Jin Kim, and Shankar Sastry. "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter." International Journal of control, Automation and systems 7 (2009): 419-428.

서울대학교 항공우주공학과 공영경

Contents

Nomenclature

x	x-axis position of UAV with respect to inertial frame
y	y-axis position of UAV with respect to inertial frame
z	z-axis position of UAV with respect to inertial frame
ϕ	roll Euler angle
θ	pitch Euler angle
ψ	yaw Euler angle
m	mass of UAV
ρ	force-to-moment scaling factor
F_i	lift force created by i-th rotor
l	length between center of UAV and a rotor
J_i	moment of inertia with respect to i-th axis
$\overrightarrow{e_3}$	$[001]^T$
g	gravitational acceleration

1. Quadrotor helicopter model

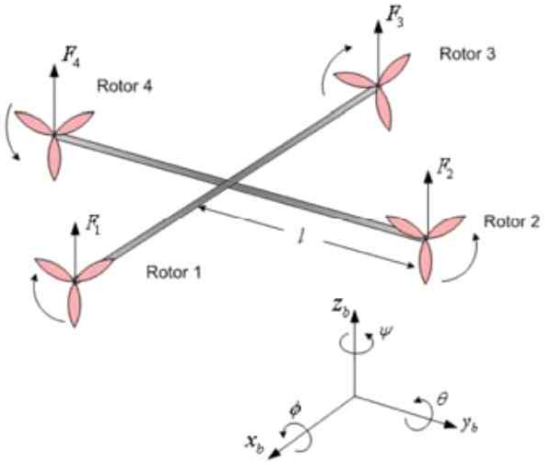


Fig 1. Quadrotor configuration

쿼드로터는 각 로터에서 발생시키는 양력과 관련된 4개의 입력, 쿼드로터의 선변위와 각변위를 표현하는 6개의 출력으로 이루어진 MIMO 시스템이다. 쿼드로터가 x축 방향으로 진행하기 위해서는 y축을 중심으로 시계방향으로 각변위를 만들어 pitch를 기울여야 한다. y축 방향 진행도 마찬가지로 x축을 중심으로 각변위를 만들어 roll을 생성해내야 한다. z축 방향으로 진행하기 위해선 각 로터의 출력을 균일하게 증가시키면 된다. 반시계 방향의 roll motion을 만들어내기 위해서는 2번 로터의 양력을 증가시키고, 4번 로터의 양력을 감소시켜야 한다. 같은 원리로, 반시계 방향의 pitch motion을 만들어내기 위해서는 3번 로터의 양력을 증가시키고, 1번 로터의 양력을 감소시켜야 한다. 반시계 방향의 yaw motion을 만들어내기 위해서는 1번과 3번 로터의 양력을 증가시키고, 2번과 4번 로터의 양력을 감소시켜야 한다. 따라서 쿼드로터는 각 상태변수가 highly coupled 되어있는 시스템이다.

쿼드로터 운동방정식 유도에는 지상에 고정된 관성좌표계와 드론의 몸체에 붙어

있는 동체좌표계가 사용된다. 동체좌표계에서 관성좌표계로 변환하기 위한 회전행렬을 구해보자. roll-pitch-yaw가 (ϕ,θ,ψ)이므로 회전행렬은 ZYX Euler angle로 구할 수 있다. 따라서 회전행렬은 다음과 같다.

$$R = {}^{I}_{B}R = Rot(z,\psi)\,Rot(y,\theta)\,Rot(x,\phi)$$

$$= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$= \begin{bmatrix} c\theta c\phi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

여기서 sin(*), cos(*)는 $s(*)$, $c(*)$로 표기하였다. 회전행렬의 좌상단 첨자 I는 관성좌표계를, 좌하단 첨자 B는 동체좌표계를 나타낸다. 각 로터가 생성하는 양력 벡터 $F_i\vec{e_3}$는 쿼드로터의 동체좌표계 기준이므로 이를 관성좌표계 기준으로 바꿔주기 위해 앞서 구한 회전행렬을 양력 벡터에 left multiplication 해야 한다. 양력과 더불어 쿼드로터에는 관성좌표계 기준으로 $-g\vec{e_3}$의 중력이 작용하고 있다. 또한 z 방향으로 지면효과(ground effect)가 작용한다.

헬리콥터나 드론이 비행 중 고도를 낮춰 지면 가까이에 도달할 때 로터의 회전에 의한 공기의 하향 흐름이 지면에 부딪히게 된다. 그리고 이러한 공기의 흐름은 지면에 반사되어 상승하게 되고 이것이 기체에 작용하여 연직 위 방향으로 작용하는 힘을 만들어내는데 이를 지면효과라고 한다. 지면효과는 지면 가까이에서 기체가 호버링할 때 추력을 절감할 수 있다는 장점도 있지만 기체가 지면에 착륙할 때 방해하는 힘으로 작용하기도 한다. 제어기 설계 및 시뮬레이션 부분에서 다시 다루겠지만 본 논문에서의 최종 제어 목적은 쿼드로터를 원하는 위치에 착륙시키는 것이므로 지면효과에 관련된 항이 uncertain parameter로서 작용하게 된다. 지면효과는 기체가 지면과 가까워질수록 증가되므로 다음과 같이 모델링할 수 있다.

$$g_r(z) = \begin{cases} \dfrac{A}{(z+z_{cg})^2} - \dfrac{A}{(z_0+z_{cg})^2} & (0 < z \le z_0) \\ 0 & (else) \end{cases}$$

g_r은 지면효과에 의한 힘에 의한 가속도를 나타내고 고도 z_0 이하에서 작용한다. A는 지면효과 상수, z_{cg}는 쿼드로터 무게중심의 고도이다. 이때 쿼드로터를 대칭 구조로 가정하면 쿼드로터의 무게중심은 쿼드로터 프레임이 교차하는 중심지점이므로 시뮬레이션상에서 z 좌표의 값과 같다.

이를 종합하여 선가속도를 $\vec{F} = m\vec{a}$를 통해 계산하면 다음과 같다.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m}(\sum_{i=1}^{4} F_i)R\vec{e_3} + (g_r(z) - g)\vec{e_3}$$

각 로터가 발생시키는 토크는 중심에서 로터까지의 거리 l에 양력을 곱해 얻을 수 있다. 단, yaw 방향 각가속도의 경우 힘-모멘트 변환 상수(force-to-moment scaling factor)를 곱해 얻는다. 각가속도를 $\vec{\tau} = I\vec{\alpha}$를 통해 계산하면 다음과 같다.(단순한 모델을 위해 gyro effect와 Coriolis effect 무시)

$$\ddot{\phi} = \frac{l(F_2 - F_4)}{J_1}$$

$$\ddot{\theta} = \frac{l(-F_1 + F_3)}{J_2}$$

$$\ddot{\psi} = \frac{\rho(F_1 - F_2 + F_3 - F_4)}{J_3}$$

단순화를 위해 입력을 다음과 같이 설정한다.

$$u_1 = \frac{F_1 + F_2 + F_3 + F_4}{m}$$

$$u_2 = \frac{F_2 - F_4}{J_1}$$

$$u_3 = \frac{-F_1 + F_3}{J_2}$$

$$u_4 = \frac{\rho(F_1 - F_2 + F_3 - F_4)}{J_3}$$

u_1은 질량으로 normalized된 전체 양력이고 u_2, u_3, u_4는 각각 roll, pitch, yaw 모멘트의 제어 입력을 나타낸다. 이를 이용하면 쿼드로터의 동역학은 다음과 같은 식으로 표현된다.

$$\begin{cases} \ddot{x} = u_1(c\phi s\theta c\psi + s\phi s\psi) \\ \ddot{y} = u_1(c\phi s\theta s\psi - s\phi c\psi) \\ \ddot{z} = u_1(c\phi c\theta) - g + g_r(z) \\ \ddot{\phi} = u_2 l \\ \ddot{\theta} = u_3 l \\ \ddot{\psi} = u_4 \end{cases}$$

$\boldsymbol{x} = [x, y, z, \phi, \theta, \psi]^T$와 $\boldsymbol{u} = [u_1, u_2, u_3, u_4]^T$를 이용하여 위 식을 state-space form으로 표현하면 다음과 같다.

$$\ddot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} + f_r(\boldsymbol{x})$$

$$f(\boldsymbol{x}) = \begin{bmatrix} 0 \\ 0 \\ -g \\ 0 \\ 0 \\ 0 \end{bmatrix},\ g(\boldsymbol{x}) = \begin{bmatrix} c\phi s\theta c\psi + s\phi s\psi & 0 & 0 & 0 \\ c\phi s\theta s\psi - s\phi c\psi & 0 & 0 & 0 \\ c\phi c\theta & 0 & 0 & 0 \\ 0 & l & 0 & 0 \\ 0 & 0 & l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ f_r(\boldsymbol{x}) = \begin{bmatrix} 0 \\ 0 \\ g_r(z) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

최종적으로 쿼드로터 플랜트는 $\ddot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}$ 꼴로 표현되는 비선형 MIMO 시스템임을 확인할 수 있다.

2. Feedback linearization control

- Theoretical backgrounds

Relative degree

상대차수(relative degree)란 시스템을 전달함수로 표현했을 때 분모 다항식의 차수에서 분자 다항식의 차수를 뺀 것으로 정의된다. s-domain에서 다음과 같은 식을 생각해보자.

$$Y(s) = P(s)U(s) \tag{1}$$

$Y(s)$는 시스템의 출력 $y(t)$의 라플라스 변환이고 $U(s)$는 시스템에 가해지는 입력 $u(t)$의 라플라스 변환, $P(s)$는 전달함수이다. 시스템의 전달함수 $P(s)$를 다음과 같이 정의하자.

$$P(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0} \quad (\text{단, } n > m,\ a_i, b_i \text{는 상수})$$

그렇다면 이 시스템의 상대차수는 $n-m$이 된다.

상대차수는 입력 $u(t)$가 출력 $y(t)$에 영향을 미치기까지 적분기를 몇 개 통과해야 하는가를 나타내는 값이다. 즉, 출력 $y(t)$를 상대차수 횟수만큼 미분하면 입력항 $u(t)$가 명시적으로 보이게 된다. s-domain에서 미분은 s를 곱하는 것으로 나타나므로 (1)식을 $n-m$번 미분하기 위해 (1)식의 양변에 s^{n-m}을 곱해보자.

$$s^{n-m}Y(s) = s^{n-m}P(s)U(s) \tag{2}$$

(2)식을 라플라스 역변환하면 다음과 같다.

$$y^{(n-m)} = \mathcal{L}^{-1}\{s^{n-m}P(s)U(s)\} \tag{3}$$

이때 $s^{n-m}P(s)U(s)$를 정리하면 다음과 같다.

$$s^{n-m}P(s)U(s) = s^{n-m}\frac{b_m s^m + b_{m-1}s^{m-1} + ... + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + ... + a_1 s + a_0}$$

$$= (b_m + \frac{As^{n-1} + ...}{s^n + a_{n-1}s^{n-1} + ... + a_1 s + a_0})U(s) \quad (A\text{는 상수})$$

$$= b_m U(s) + \frac{As^{n-1} + ...}{s^n + a_{n-1}s^{n-1} + ... + a_1 s + a_0}U(s) \tag{4}$$

최종적으로, (4)식을 라플라스 역변환하여 (3)식과 연립하면 다음과 같다.

$$y^{(n-m)}(t) = b_m u(t) + ...$$

위 과정을 살펴보면 출력 $y(t)$를 $n-m$번 보다 적게 미분했을 때는 입력항 $u(t)$가 명시적으로 드러나지 않다가 $n-m$번 미분하는 순간 $u(t)$가 명시적으로 드러나는 것을 확인할 수 있다.

Byrnes-Isidori normal form

임의의 전달함수 $P(s)$를 식(5)와 같이 서로소인 분자다항식과 분모다항식의 꼴로 나타내보자.

$$P(s)=\frac{N(s)}{D(s)} \tag{5}$$

이때 $N(s)$는 m차, $D(s)$는 n차, $n-m\geq 1$로 설정하자. 분모다항식의 차수가 분자다항식보다 더 크므로 분모다항식을 분자다항식으로 나눌 수 있고, 그때의 몫을 $Q(s)$, 나머지를 $R(s)$라고 하면 $Q(s)$는 $n-m$차, $R(s)$는 m차 미만의 다항식이다.

$$D(s)=Q(s)N(s)+R(s) \tag{6}$$

(6)을 (5)에 대입하여 정리하면 다음과 같다.

$$P(s)=\frac{N(s)}{Q(s)N(s)+R(s)}=\frac{\frac{1}{Q(s)}}{1+\frac{1}{Q(s)}\frac{R(s)}{N(s)}} \tag{7}$$

식(7)을 살펴보면 식의 형태가 (forward gain)/(loop gain)의 꼴인 것을 확인할 수 있다. 따라서 전달함수 $P(s)$를 Figure 2와 같이 closed loop 꼴로 변환할 수 있다.

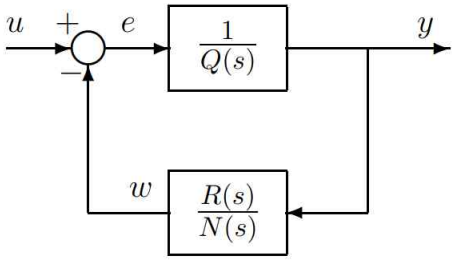


Fig 2. Closed loop form of transfer function[1]

Forward gain과 feedback gain을 각각 state-space form으로 realization하여 합쳐 보자. 먼저 $\frac{1}{Q(s)}$를 $\boldsymbol{x}$로 realization 한다. $r := n-m$이라 하면 $\frac{1}{Q(s)}$의 상대차수는 r이다. 이는 출력 y를 r번 미분하면 입력항 e가 명시적으로 드러난다는 뜻이다. 이를 state-space form으로 나타내어 보면 다음과 같다.

$$\begin{aligned} y &= x_1 \\ \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{r-1} &= x_r \\ \dot{x}_r &= \phi^T \boldsymbol{x} + ge \end{aligned} \tag{8}$$

이때 $\phi \in \mathbb{R}^r$, $g \in \mathbb{R}$이다. 다음으로는 $\frac{R(s)}{N(s)}$를 $\boldsymbol{z}$로 realization 해보자. 일반적인 state-space model인

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{aligned}$$

를 생각하자. 이때 $N(s)$의 차수는 m이고 $R(s)$의 차수는 m보다 작으므로($\because$ 나머지 정리) 상대차수는 1 이상이다. 따라서 $D=0$이다.($y = C\boldsymbol{x} + Du$에서는 y를 미분하지 않아도 u가 드러나므로 $D \neq 0$이면 상대차수가 0이다.) 이를 고려하면 $\frac{R(s)}{N(s)}$는 다음과 같이 realization 가능하다.

$$\begin{aligned} \dot{\boldsymbol{z}} &= S\boldsymbol{z} + Gy \\ w &= \psi^T \boldsymbol{z} \end{aligned}$$

이때 $S \in \mathbb{R}^{m \times m}$, $G \in \mathbb{R}^m$, $\psi \in \mathbb{R}^m$ 이다. 최종적으로 두 모델을 합쳐보자. Figure 2를

보면 $e = u - w$의 관계가 성립하는 것을 확인할 수 있다. 따라서 식(8)에서 e를 $u - w$로 치환하면 다음과 같다.

$$\dot{x_r} = \phi^T \boldsymbol{x} + g(u - w) = \phi^T \boldsymbol{x} + g(u - \psi^T \boldsymbol{z}) = \phi^T \boldsymbol{x} - g\psi^T \boldsymbol{z} + gu$$

이때 $-g\psi^T$는 차원이 같은 임의의 벡터로 치환할 수 있으므로 식을 간단히 하기 위해 ψ^T로 놓으면 다음과 같다.

$$\dot{x_r} = \phi^T \boldsymbol{x} + \psi^T \boldsymbol{z} + gu$$

따라서 최종적으로 전달함수 $P(s)$는 다음과 같은 state-space representation으로 표현될 수 있다.

$$\begin{aligned} y &= x_1 \\ \dot{x_1} &= x_2 \\ \dot{x_2} &= x_3 \\ &\vdots \\ \dot{x_r} &= \phi^T \boldsymbol{x} + \psi^T \boldsymbol{z} + gu \\ \dot{\boldsymbol{z}} &= S\boldsymbol{z} + Gy \end{aligned} \tag{9}$$

어떤 시스템이 식(9)와 같은 형태로 표현될 때 이 시스템은 Byrnes-Isidori normal form으로 표현되었다고 한다.

Zero dynamics

Zero dynamics란 어떤 시스템에 적절한 초기 조건과 입력을 가했을 때 그 시스템의 출력이 0이 되는 조건 하에서 시스템의 상태변수를 지배하는 미분방정식을 말한다. 어떤 시스템이 Byrnes-Isidori normal form으로 표현되어 있다면 zero dynamics를 쉽게 판별 가능하다는 장점이 있다. 식(9)로 표현되는 시스템에서 출력이 0이라면 상태변수 $\boldsymbol{x}$에 관한 식들은 모두 0이 되고 가장 마지막 식인 $\dot{\boldsymbol{z}} = S\boldsymbol{z} + Gy$에서 $\dot{\boldsymbol{z}} = S\boldsymbol{z}$만 남게 된다. 따라서 시스템을 normal form으로 표현한다면 zero dynamics는 subsystem $\dot{\boldsymbol{z}} = S\boldsymbol{z}$ 이고 이때의 입력은 $u(t) = \frac{1}{g}(-\psi^T \boldsymbol{z})$이다. 출력은 표면적으로 0이 유지되는 상황이라도 plant 안에서는 $\dot{\boldsymbol{z}} = S\boldsymbol{z}$에 따라 상태변수 $\boldsymbol{z}$가 움직이고 있다. 따라서 안정적인 플랜트를 유지하기 위해서는 zero dynamics 또한 안정적이어야 하므로 선형시스템의 경우, S를 Hurwitz하게 설계하여야 한다.

또한 zero dynamics의 pole은 original plant의 zero라는 특징이 있다. 이는 zero dynamics가 그렇게 이름 붙여진 이유이기도 하다. 앞서 살펴본 바와 같이 $\dot{\boldsymbol{z}} = S\boldsymbol{z} + Gy$는 전달함수가 $\frac{R(s)}{N(s)}$인 시스템의 state-space form이다. 따라서 $\frac{R(s)}{N(s)}$의 pole은 S의 eigenvalue이다. 한편, $\frac{R(s)}{N(s)}$의 pole은 $N(s)$의 root이고 $P(s) = \frac{N(s)}{D(s)}$이므로 S의 eigenvalue는 original plant $P(s)$의 zero가 된다.

Feedback linearization in SISO system

궤환 선형화(Feedback linearization)는 비선형 시스템을 제어할 수 있는 제어 기법으로서, 다음과 같은 형태를 가진 비선형 시스템에 적용할 수 있다.

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + g(\boldsymbol{x}(t))u(t)$$
$$\boldsymbol{x}(t)\in\mathbb{R}^n,\ u(t)\in\mathbb{R}$$

궤환 선형화의 목적은 비선형 좌표변환을 통해 비선형 시스템을 equivalent한 선형 시스템으로 만들고 상태변수 $\boldsymbol{x}$를 0으로 보내는 입력 $u=\alpha(\boldsymbol{x})$를 찾는 것이다. ($\alpha:\mathbb{R}^n\to\mathbb{R}$)

비선형 좌표변환 $z=\boldsymbol{\Phi}(\boldsymbol{x})$를 생각하자.

$$z=\boldsymbol{\Phi}(\boldsymbol{x})$$
$$\boldsymbol{\Phi}:\mathbb{R}^n\to\mathbb{R}^n$$

이때 $z=\boldsymbol{\Phi}(\boldsymbol{x})$는 미분동형사상(diffeomorphism)이라 불리고 미분가능하다. 또한 bijective하며 그 역도 미분가능하다는 특징을 가지고 있다. 이 미분동형사상 좌표변환을 통해 시스템을

$$\dot{\boldsymbol{z}} = A\boldsymbol{z}+Bv$$

꼴의 선형 시스템으로 변환했다고 가정하자. 그렇다면 입력 v를 $v=-K\boldsymbol{z}$로 선택하고 $A-BK$가 Hurwitz가 되게 만들어 이 시스템을 안정적으로 제어할 수 있게 된다. 이때, 상대차수가 n인 normal form을 생각해보자.

$$\dot{z_1} = z_2$$
$$\dot{z_2} = z_3$$
$$\vdots$$
$$\dot{z_n} = \phi(\boldsymbol{x}) + \pi(\boldsymbol{x})u$$

입력을 $u = \frac{1}{\pi(\boldsymbol{x})}(-\phi(\boldsymbol{x}) + v)$ (단, $\pi(\boldsymbol{x}) \neq 0$)으로 설정하면 위 식은 다음과 같이 바뀌고,

$$\dot{z_1} = z_2$$
$$\dot{z_2} = z_3$$
$$\vdots$$
$$\dot{z_n} = v$$

$\dot{\boldsymbol{z}} = A\boldsymbol{z} + Bv$에서 행렬 A와 B는

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

와 같이 표현된다. 따라서 미분동형사상 좌표변환을 통해 시스템을 상대차수가 n인 normal form으로 표현 가능하다면 비선형시스템을 선형시스템으로 변환할 수 있다.

시스템을 위와 같은 형태로 표현해주는 미분동형사상을 구해보자. 시스템의 상대차수가 n이 되도록 하려면 출력을 정의하는 것이 필요하다. 상대차수는 전달함수를 통해 정의되고 전달함수가 존재하려면 출력이 존재해야 하기 때문이다. 따라서 시스템의 상대차수가 n이 되도록 하는 가상의 출력 $y = h(\boldsymbol{x})$를 잡고 이를 z_1으로 놓자.

$$y = h(\boldsymbol{x}) = z_1$$

위 식을 시간에 대해 미분하면,

$$\dot{z_1} = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}}(t) = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x}) + \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} g(\boldsymbol{x}) u$$

이다.

이때 시스템의 상대차수가 n이므로 $z_i (0 < i < n$인 정수)를 미분했을 때 입력항이 나타나지 않는다. 따라서 $\frac{\partial h(x)}{\partial x} g(x) u = 0$이다. 식을 간편하게 나타내기 위해 Lie derivative라는 개념을 도입하자. Lie derivative는 $f(\boldsymbol{x})$ 벡터 방향으로 $h(\boldsymbol{x})$를 방향 미분한 것으로 정의하고 다음과 같이 표기한다.

$$L_f h(\boldsymbol{x}) := \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x})$$

$$L_f^2 h(\boldsymbol{x}) := L_f (L_f h(\boldsymbol{x}))$$

$$\vdots$$

Lie derivative를 도입하면

$$\dot{z_1} = L_f h(\boldsymbol{x})$$
$$\dot{z_2} = L_f^2 h(\boldsymbol{x})$$

$$\vdots$$

이므로 최종적으로 구하고자 하는 미분동형사상 좌표변환은 다음과 같다.

$$\therefore z = \Phi(\boldsymbol{x}) = \begin{bmatrix} h(\boldsymbol{x}) \\ L_f h(\boldsymbol{x}) \\ L_f^2 h(\boldsymbol{x}) \\ \vdots \\ L_f^{n-1} h(\boldsymbol{x}) \end{bmatrix}$$

Feedback linearization in MIMO system

궤환 선형화를 적용할 수 있는 MIMO 시스템은 다음과 같은 형태로 나타낼 수 있다.

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \sum_{i=1}^{m} g_i(\boldsymbol{x}(t))u_i(t),\ (i = 1, \ldots, m)$$

입력의 개수가 m개이므로 출력의 개수를 m개로 잡는다.

$$y_i = h_i(\boldsymbol{x}),\ (i = 1, \ldots, m)$$

SISO 시스템에서는 출력을 상대차수가 n이 되도록 설정하였다. MIMO 시스템에서는 출력이 여러 개이므로 상대차수가 스칼라가 아니라 각 출력에 해당하는 값이 있는 벡터 형태로 나타나게 된다. 이를 벡터 상대 차수라고 한다. $y_i = h_i(\boldsymbol{x})$를 r_i번 미분했을 때 m개의 입력항 중 하나라도 명시적으로 드러난다면 r_i는 출력 $y_i = h_i(\boldsymbol{x})$에 대한 상대 차수이고 MIMO 시스템 전체의 벡터 상대 차수는 $[r_1, r_2, \ldots, r_m]$이 된다.

입력항이 나타날 때까지 각 출력을 미분하면 다음과 같이 나타난다.

$$y_i^{(r_i)} = L_f^{r_i} h_i + \sum_{j=1}^{m} L_{g_j} L_f^{r_i - 1} h_i u_j \ \ (L_{g_j} L_f^{r_i - 1} h_i(\boldsymbol{x}) \neq 0)$$

위 식을 모든 출력에 대해 표시하면,

$$\begin{bmatrix} y_1^{(r_1)} \\ \vdots \\ y_m^{(r_m)} \end{bmatrix} = \begin{bmatrix} L_f^{r_1} h_1(\boldsymbol{x}) \\ \vdots \\ L_f^{r_m} h_m(\boldsymbol{x}) \end{bmatrix} + \Lambda(\boldsymbol{x})\boldsymbol{u}$$

$$\text{where } \Lambda(\boldsymbol{x}) = \begin{bmatrix} L_{g_1} L_f^{r_1-1} h_1(\boldsymbol{x}) & \cdots & L_{g_m} L_f^{r_1-1} h_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_m-1} h_m(\boldsymbol{x}) & \cdots & L_{g_m} L_f^{r_m-1} h_m(\boldsymbol{x}) \end{bmatrix}$$

이 된다. 이때 $\Lambda(\boldsymbol{x})$를 decoupling matrix라고 한다. 만약 decoupling matrix가 nonsingular 하다면 다음과 같이 입력을 정의하는 것이 가능하다.

$$\boldsymbol{u} = \Lambda^{-1} \begin{bmatrix} v_1 - L_f^{r_1} h_1 \\ \vdots \\ v_m - L_f^{r_m} h_m \end{bmatrix}$$

그리고 이렇게 구한 입력을 대입하여 출력에 대한 식으로 나타내면 다음과 같이 간단한 식으로 표현할 수 있다.

$$y_i^{(r_i)} = v_i$$

따라서 제어 목적에 따라 원하는 변수를 안정화시키는 v_i를 설계하여 입력을 만든다면 비선형 MIMO 시스템을 궤환 선형화로 제어할 수 있다.

- Controller design

쿼드로터의 동역학에서 단순한 서보 모델의 정립을 위해 $|\phi|,|\theta|,|\psi| \ll 1$로 가정할 수 있다.(small angle assumption) 그리고 지면효과를 무시하면, 쿼드로터의 동역학을 다음과 같은 형태로 간단히 나타낼 수 있다.

$$\begin{aligned}
\ddot{x} &= u_1 \sin\theta \\
\ddot{y} &= -u_1 \sin\phi \\
\ddot{z} &= u_1 \cos\phi\cos\theta - g \\
\ddot{\phi} &= u_2 l \\
\ddot{\theta} &= u_3 l \\
\ddot{\psi} &= u_4
\end{aligned}$$

제어 입력의 개수가 4개이므로 4개의 출력을 잡는다.

Feedback linearization을 적용하기 위해서는 먼저 적절한 출력을 잡고 이렇게 잡은 출력들을 입력항들이 명시적으로 드러날 때까지 미분하는 동적 확장(dynamic extension)을 수행하여 decoupling matrix를 invertible하게 설계해야 한다. 가장 먼저 z, ϕ, θ, ψ를 출력으로 잡아보자. 왜냐하면 u_2, u_3, u_4가 $\ddot{\phi}, \ddot{\theta}, \ddot{\psi}$항에 포함되어 있기 때문이다. z, ϕ, θ, ψ를 출력으로 잡았을 때 시스템을 zero dynamics 관점에서 분석해 보자. Zero dynamics를 구하기 위해 $\ddot{z} = 0$로 잡으면,

$$0 = u_1(\cos\phi\cos\theta) - g$$

$$u_1 = \frac{g}{\cos\phi\cos\theta} \tag{10}$$

식(10)를 대입하여 $\ddot{x}, \ddot{y}$를 구해보자.

$$\ddot{x} = g\left(\frac{\cos\phi\sin\theta\cos\psi}{\cos\phi\cos\theta} + \frac{\sin\phi\sin\psi}{\cos\phi\cos\theta}\right) = g\left(\tan\theta\cos\psi + \frac{\sin\psi}{\cos\theta}\tan\phi\right) \approx g\tan\theta$$

$$\ddot{y} = g\left(\frac{\cos\phi\sin\theta\sin\psi}{\cos\phi\cos\theta} - \frac{\sin\phi\cos\psi}{\cos\phi\cos\theta}\right) = g\left(\tan\theta\sin\psi - \frac{\cos\psi}{\cos\theta}\tan\phi\right) \approx -g\frac{\tan\phi}{\cos\theta}$$

위 식을 보면 zero dynamics가 unstable하다는 것을 확인할 수 있다. 따라서 z, ϕ, θ, ψ를 출력으로 잡는 것은 적절하지 않다.

다음으로, 이번엔 출력으로 x, y, z, ψ를 잡아보자. 이렇게 잡은 출력들을 미분해 보면,

$$x^{(3)} = \dot{u}_1 \sin\theta + u_1 \dot{\theta}\cos\theta$$

$$x^{(4)} = \ddot{u}_1 \sin\theta + 2\dot{u}_1 \dot{\theta}\cos\theta - u_1 \dot{\theta}^2 \sin\theta + u_1 \ddot{\theta}\cos\theta$$

$$y^{(3)} = -\dot{u}_1 \sin\phi - u_1 \dot{\phi}\cos\phi$$

$$y^{(4)} = -\ddot{u}_1 \sin\phi - 2\dot{u}_1 \dot{\phi}\cos\phi + u_1 \dot{\phi}^2 \sin\phi - u_1 \ddot{\phi}\cos\phi$$

$$z^{(3)} = \dot{u}_1 \cos\theta\cos\phi - u_1 \dot{\theta}\sin\theta\cos\phi - u_1 \dot{\phi}\cos\theta\sin\phi$$

$$\begin{aligned} z^{(4)} = & -2\dot{u}_1 \dot{\theta}\sin\theta\cos\phi - 2\dot{u}_1 \dot{\phi}\cos\theta\sin\phi + 2u_1 \dot{\theta}\dot{\phi}\sin\theta\sin\phi - u_1(\dot{\theta}^2 + \dot{\phi}^2)\cos\theta\cos\phi \\ & + \ddot{u}_1 \cos\theta\cos\phi - u_1 \ddot{\theta}\sin\theta\cos\phi - u_1 \ddot{\phi}\cos\theta\sin\phi \end{aligned}$$

$$\ddot{\psi} = u_4$$

이다. 이때 $u_2 = \ddot{\phi}/l$, $u_3 = \ddot{\theta}/l$임을 이용하면 위 식에서 모든 입력항 u_1, u_2, u_3, u_4가 명시적으로 드러남을 확인할 수 있다. $x-y-z$를 따로 떼어내서 출력의 4계 미분을 행렬 형태로 나타내보면 다음과 같다.

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \\ z^{(4)} \end{bmatrix} = \begin{bmatrix} 2\dot{u}_1 \dot{\theta}\cos\theta - u_1 \dot{\theta}^2 \sin\theta \\ -2\dot{u}_1 \dot{\phi}\cos\phi + u_1 \dot{\phi}^2 \sin\phi \\ -2\dot{u}_1 \dot{\theta}\sin\theta\cos\phi - 2\dot{u}_1 \dot{\phi}\cos\theta\sin\phi + 2u_1 \dot{\theta}\dot{\phi}\sin\theta\sin\phi - u_1(\dot{\theta}^2 + \dot{\phi}^2)\cos\theta\cos\phi \end{bmatrix} + \begin{bmatrix} \sin\theta & 0 & u_1\cos\theta l \\ -\sin\phi & -u_1\cos\phi l & 0 \\ \cos\theta\cos\phi & -u_1\cos\theta\sin\phi l & -u_1\sin\theta\cos\phi l \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{11}$$

앞서 살펴본 바와 같이 decoupling matrix의 역행렬을 통해 입력을 계산해보자.

$$\begin{bmatrix}\ddot{u}_1\\ u_2\\ u_3\end{bmatrix}=\begin{bmatrix}\sin\theta & 0 & u_1\cos\theta l\\ -\sin\phi & -u_1\cos\phi l & 0\\ \cos\theta\cos\phi & -u_1\cos\theta\sin\phi l & -u_1\sin\theta\cos\phi l\end{bmatrix}^{-1}$$
$$\cdot\begin{bmatrix}v_1-2\dot{u}_1\dot{\theta}\cos\theta+u_1\dot{\theta}^2\sin\theta\\ v_2+2\dot{u}_1\dot{\phi}\cos\phi-u_1\dot{\phi}^2\sin\phi\\ v_3+2\dot{u}_1\dot{\theta}\sin\theta\cos\phi+2\dot{u}_1\dot{\phi}\cos\theta\sin\phi-2u_1\dot{\theta}\dot{\phi}\sin\theta\sin\phi+u_1(\dot{\theta}^2+\dot{\phi}^2)\cos\theta\cos\phi\end{bmatrix}$$

위의 입력을 대입하면,

$$\begin{bmatrix}x^{(4)}\\ y^{(4)}\\ z^{(4)}\end{bmatrix}=\begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix}.$$

원하는 제어 목적은 원하는 $x-y-z$ 좌표로 쿼드콥터를 이동시키는 것이다. 따라서 $[v_1, v_2, v_3]^T$를 다음과 같이 설계하면 $x\to x_d$, $y\to y_d$, $z\to z_d$를 달성할 수 있다.

$$v_1=x_d^{(4)}-k_{x1}e_x^{(3)}-k_{x2}\ddot{e}_x-k_{x3}\dot{e}_x-k_{x4}e_x$$
$$v_2=y_d^{(4)}-k_{y1}e_y^{(3)}-k_{y2}\ddot{e}_y-k_{y3}\dot{e}_y-k_{y4}e_y$$
$$v_3=z_d^{(4)}-k_{z1}e_z^{(3)}-k_{z2}\ddot{e}_z-k_{z3}\dot{e}_z-k_{z4}e_z$$
$$e_x:=x-x_d,\ e_y:=y-y_d,\ e_z:=z-z_d$$

error dynamics를 살펴보면,

$$e_x^{(4)}+k_{x1}e_x^{(3)}+k_{x2}\ddot{e}_x+k_{x3}\dot{e}_x+k_{x4}e_x=0$$
$$e_y^{(4)}+k_{y1}e_y^{(3)}+k_{y2}\ddot{e}_y+k_{y3}\dot{e}_y+k_{y4}e_y=0$$
$$e_z^{(4)}+k_{z1}e_z^{(3)}+k_{z2}\ddot{e}_z+k_{z3}\dot{e}_z+k_{z4}e_z=0.$$

따라서 위 error dynamics의 각 error가 0으로 수렴할 수 있도록 위 시스템을 안정하게 만드는 gain 값을 정해주면 제어 목적을 달성할 수 있다.

위 과정을 통해 $[\ddot{u}_1, u_2, u_3]^T$가 도출된다. 그렇다면 왜 u_1 대신 $\ddot{u}_1$을 구해서 2번 적분하는 것일까?

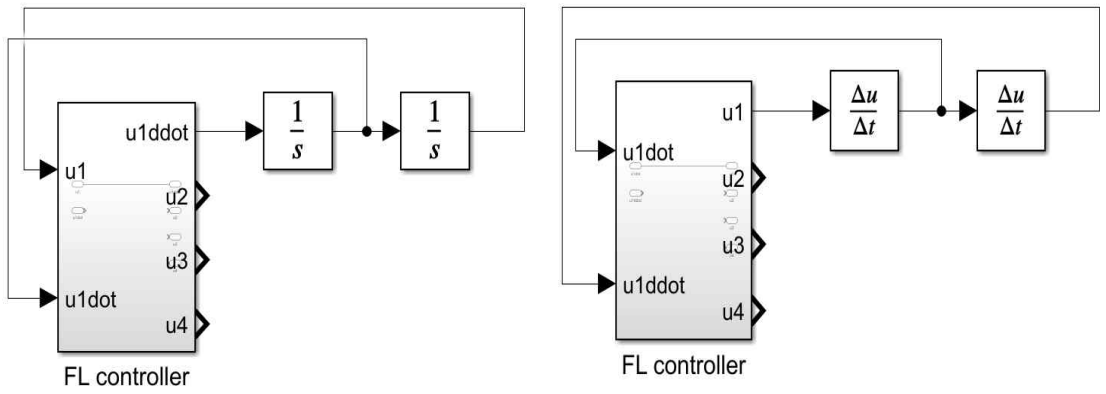


Fig 3. Simulink model of feedback linearization controller

Figure 3.에서 $\ddot{u}_1$을 구했을 때와 u_1을 구했을 때의 모식도를 나타내었다. 만약 u_1이 제어기의 출력으로 나온다면, 식(11)에 의해 제어기의 입력단에는 $\ddot{u}_1$이 들어가야 한다. 이때 시뮬레이션에서 사용되는 미분은 수치미분이므로 출력 신호에서의 작은 noise나 error로 인해 신호가 불안정해지게 된다. 따라서 제어기의 출력으로 u_1대신 $\ddot{u}_1$를 사용한다. 이렇게 feedback linearization의 결과로 나온 제어기의 출력을 적분하여 플랜트의 입력으로 사용하는 방식을 dynamic feedback linearization이라고 한다.

한편, ψ-제어기의 경우는 u_1, u_2, u_3와 분리되어 따로 작동한다. ψ-제어기는 PD 제어기를 사용하여 입력을

$$u_4 = \ddot{\psi}_d + k_{\psi 1}(\dot{\psi}_d - \dot{\psi}) + k_{\psi 2}(\psi_d - \psi)$$

로 설계하면 $\psi \rightarrow \psi_d$를 달성할 수 있다.

3. Adaptive sliding mode control

- Theoretical Backgrounds

Lyapunov stability

Nonlinear autonomous system $\dot{x}=f(x)$를 생각하자. x_e가 시스템의 equilibrium point라고 하면 $f(x_e)=0$이 성립한다. 이때

$$\forall \epsilon > 0,\ \exists \delta(\epsilon) > 0 \text{ s.t. } \| x(0)-x_e \| < \delta \Rightarrow \forall t \geq 0,\ \| x(t)-x_e \| < \epsilon \quad (12)$$

가 성립하면 평형점 x_e는 Lyapunov stable이라고 한다. 식(12)를 해석하여 보면 아무리 작은 양수 ϵ을 잡더라도 그 ϵ보다 작은 양수 δ를 선택하고 그 δ 영역(ball) 내부의 임의의 점을 초기값으로 잡았을 때 상태변수가 ϵ ball 밖을 벗어나지 않는다면 그 시스템은 Lyapunov stable이라 부른다는 것이다. 즉, Lyapunov stable은 평형점 부근의 초기값이 있다면 상태변수가 그 주변에서만 움직이고 발산하지 않는다는 것을 의미한다.

다른 개념으로, $\lim_{t\to\infty} x(t)=x_e$이면 평형점 x_e는 attractive라고 한다. 그리고 평형점 x_e가 Lyapunov stable하고 attractive하다면 asymtotically stable하다고 한다.

Stability를 판단하기 어려운 비선형 시스템의 경우, Lyapunov's direct method를 사용하여 판단하는 것이 보편적이다. Lyapunov's direct method는 비선형 시스템의 동적 특성을 잘 모르더라도 Lyapunov function을 이용하여 간접적으로 stability를 해석하는 방법이다.

어떤 scalar function $V(x(t))$에 대하여 다음을 만족하는 $V(x(t))$를 positive

definite function이라고 한다.

$$\begin{cases} V(x)=0 \ (x=x_e) \\ V(x)>0 \ (x\neq x_e) \end{cases}$$

그리고 다음을 만족하는 $V(x(t))$를 negative definite function이라고 한다.

$$\begin{cases} V(x)=0 \ (x=x_e) \\ V(x)<0 \ (x\neq x_e) \end{cases}$$

각 조건의 부등호에 등호(=)가 포함되면 각각 positive semi-definite, negative semi-definite function이라고 부른다.

Nonlinear autonomous system $\dot{x}=f(x)$를 생각하자. x_e가 시스템의 equilibrium point라고 하면 $f(x_e)=0$이 성립한다. 이때 positive definite function $V(x)$에 대해서 $\dot{V}$가 negative semi-definite function이면 $V(x)$를 Lyapunov function이라 부르고 평형점 x_e는 Lyapunov stable하다. 만약 $\dot{V}$가 negative definite function이면 평형점 x_e는 asymtotically stable하다. 이렇게 하여 stability를 판별하는 방법을 Lyapunov's direct method라고 한다.

평형점 x_e는 asymtotically stable한 경우에 대해서 Lyapunov's direct method의 원리를 파악해보자. $x\neq x_e$인 점에서 $V=V(x_1, x_2, \dots, x_n)>0$이고,

$$\dot{V}=\frac{\partial V}{\partial x_1}\frac{dx_1}{dt}+\dots+\frac{\partial V}{\partial x_n}\frac{dx_n}{dt}=\nabla V \cdot \frac{d\boldsymbol{x}}{dt}=|\nabla V|\left|\frac{d\boldsymbol{x}}{dt}\right|\cos\phi<0 \qquad (13)$$

이다. 이때 ϕ는 ∇V 벡터와 $\frac{d\boldsymbol{x}}{dt}$ 벡터 사이의 각도이다. 식(13)에 따라, ϕ의 범위는 $90^\circ<\phi<270^\circ$가 된다.

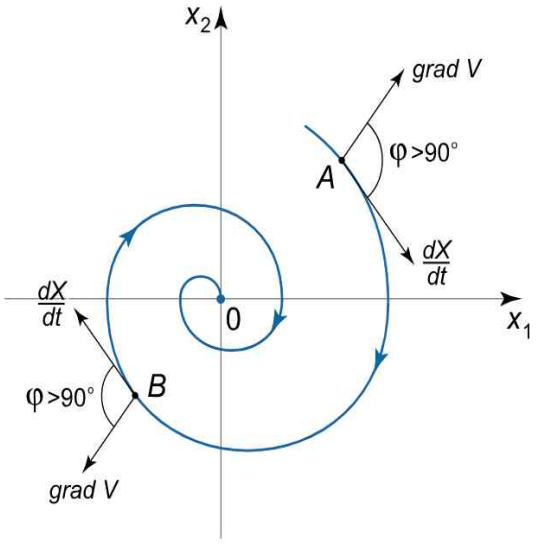


Fig 4. Geometric analysis of Lyapunov function

∇V는 V의 등위면에서 수직으로 나오는 방향이고 $\frac{d\boldsymbol{x}}{dt}$는 상태변수 궤적의 접선 방향이므로 $90^\circ < \phi < 270^\circ$를 적용하면 Figure 4.와 같이 상태변수의 궤적은 원점(평형점)을 향하여 점근적으로 수렴해간다.

Lyapunov function은 실제 역학계에서 에너지 개념과 연관이 있다. 에너지는 Lyapunov function과 같이 positive definite function이고 시간이 지날수록 점점 0에 수렴한다는 특성이 있다($\dot{V} < 0$). Mass-spring-damper system의 예시를 살펴보자. 외력이 없을 때의 운동방정식은 다음과 같다.

$$m\ddot{x} + c\dot{x} + kx = 0$$

이를 state-space form으로 나타내면,

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

이고, 평형점은 $x = 0, \dot{x} = 0$인 점이다. 이 시스템의 역학적 에너지는 다음과 같이 표

현된다.

$$V = \frac{1}{2}kx^2 + \frac{1}{2}m\dot{x}^2$$

이를 미분하면,

$$\dot{V} = kx\dot{x} + m\dot{x}\ddot{x} = kx\dot{x} + m\dot{x}(-\frac{c}{m}\dot{x} - \frac{k}{m}x) = kx\dot{x} - c\dot{x}^2 - kx\dot{x} = -c\dot{x}^2$$

이다. 따라서 V는 positive definite, $\dot{V}$은 negative definite이 되어 역학적 에너지 V는 Lyapunov function이다. Damping constant $c = 0$이라면 $\dot{V} = 0$이고 $V = const$가 되어 감쇄하지 않는 상황에서 역학적 에너지가 보존됨을 확인할 수 있고 $c > 0$이면 $\dot{V} < 0$이고 $V \to 0$이 되어 감쇄가 있는 상황에서 역학적 에너지가 감소됨을 확인할 수 있다.

Sliding mode control

슬라이딩 모드 제어(sliding mode control, SMC)는 nonlinear system에 대하여 discontinous한 control signal u를 적용하여 상태변수 공간 $x \in \mathbb{R}^n$을 두 부분으로 나누는 sliding surface S와 Lyapunov function을 적절히 설계하고, 상태변수 x가 S에 도달한 후 sliding surface를 타고 원하는 평형점에 도달하도록 하는 제어 기법이다.

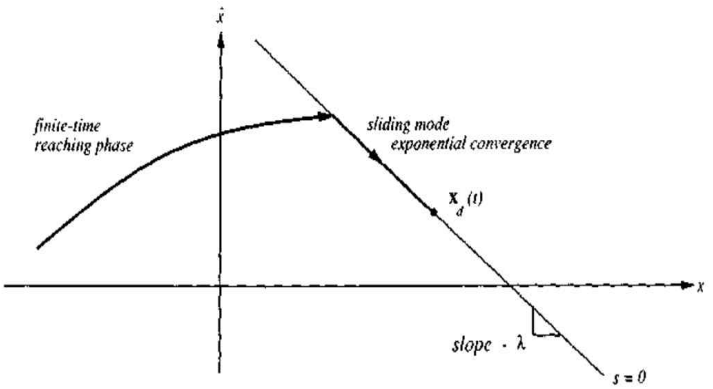


Fig 5. Reaching phase and sliding phase of SMC[2]

Figure 5.에서 볼 수 있듯이 SMC의 작동은 상태변수가 slding surface에 접근하는 reaching phase와 sliding phase를 타고 원하는 지점(원점)으로 수렴하는 sliding phase의 두 부분으로 이루어져 있다.

· Sliding phase

시스템을 normal form으로 표현하여 상태변수들이 다음과 같은 관계를 가지도록 한다.

$$\boldsymbol{x} = [x_1, x_2, x_3, \dots, x_n]^T = [x_1, \dot{x_1}, \ddot{x_1}, \dots, x_1^{(n-1)}]^T$$

그리고 이때 sliding surface S를 다음과 같이 설계한다.

$$S = x_n + k_1 x_1 + k_2 x_2 + ... + k_{n-1} x_{n-1}$$

그러면 $S=0$은 sliding surface가 나타내는 hyperplane(manifold) 위의 점들의 집합을 나타낸다.

$S=0$이 유지되면 상태변수는 계속 sliding surface에 있게 된다. 따라서 $S=0$을 유지하면서 상태변수를 원점으로 수렴시켜야 한다. 이는 적절하게 k_i $(i=1,2,...,n-1)$를 설계함으로써 달성될 수 있다. $S=0$은 다음과 같은 식으로 고쳐 쓸 수 있다.

$$x_n = -k_1 x_1 - k_2 x_2 - ... - k_{n-1} x_{n-1}$$
$$\dot{x}_{n-1} = -k_1 x_1 - k_2 x_2 - ... - k_{n-1} x_{n-1}$$

따라서,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ -k_1 & -k_2 & -k_3 & \cdots & -k_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

$$\dot{\boldsymbol{z}} = A\boldsymbol{z} \text{ where } \boldsymbol{z} = [x_1, x_2, ..., x_{n-1}]^T$$

$\boldsymbol{z} \to 0$이면 $\boldsymbol{x} \to 0$이 되므로 상태변수를 원점으로 수렴시킬 수 있다. 이때 $\boldsymbol{z} \to 0$를 달성하기 위해선 A를 Hurwitz하게 만들면 된다. A는 companion form matrix이므로 $k_i > 0$이면 A를 Hurwitz하게 만들 수 있다.

만약 수렴점이 원점이 아니라 $\boldsymbol{x_d}$라면 state vector를 error vector $\boldsymbol{e} := \boldsymbol{x} - \boldsymbol{x_d}$로 정의한다.

· Reaching phase

Reaching phase는 상태변수가 유한시간 내에 initial condition에서 S로 도달하게 하는 부분이다. 임의의 state $\boldsymbol{x}(0)$에서 $S=0$인 상태로 수렴해야 하므로 S를 변수로 하는 Lyapunov function $V(s)$의 존재를 증명하면 state가 유한시간 내에 $S=0$에 도달함을 보일 수 있다. 따라서 $V(s)$가 Lyapunov function이 되도록 $u(t)$를 설계해야 한다.

Lyapunov function candidate $V(s)=\frac{1}{2}S^2$를 생각하자. $V(s)$는 이미 positive definite function 이므로 $\frac{d}{dt}V(s)<0$를 만족하면 $V(s)$는 Lyapunov function이다. 따라서

$$\frac{d}{dt}V(S(\boldsymbol{x}(t)))=\frac{\partial V}{\partial S}\frac{dS}{dt}=\frac{\partial V}{\partial S}\frac{\partial S}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}=S\dot{S}<0$$

이어야 한다.

이때 보통 $S\dot{S}<0$ 조건은 $S\dot{S}\leq-\eta|S|$(η-reachability condition)으로 대체될 수 있다($\eta>0$). $S\dot{S}=\frac{1}{2}\frac{d}{dt}(S^2)\leq-\eta|S|$에서 S^2은 임의의 좌표 $S(\boldsymbol{x}(t))$에서 $S=0$까지 거리의 제곱을 나타내므로 위 식은 상태변수와 $S=0$까지의 거리가 계속 줄어들다가 $S=0$에 도달 후에는 계속 $S=0$에 머무른다고 해석할 수 있다.

한편, 상태변수가 유한시간 내에 $S=0$에 도달함을 증명해보자. $S\dot{S}\leq-\eta|S|$의 양변을 $t=0$부터 $t=t_{reach}$까지 적분해보자. t_{reach}는 상태변수가 $S=0$에 도달하는 시각이다.

$$\int_0^{t_{reach}}S\dot{S}dt\leq\int_0^{t_{reach}}-\eta|S|dt$$

$$S(t_{reach})-S(0)\leq-\eta(t_{reach}-0)sgn(S)$$

이때 $t=t_{reach}$에서 상태변수가 $S=0$에 도달하므로 $S(t_{reach})=0$이다.

$$-S(0) \le -\eta t_{reach} sgn(S)$$

$$t_{reach} \le \frac{S(0)}{\eta sgn(S)}$$

이때 $t=0$부터 $t=t_{reach}$까지 $sgn(S)=sgn(S(0))$이므로,

$$t_{reach} \le \frac{|S(0)|}{\eta}$$

이다. 따라서 상태변수가 유한시간 내에 $S=0$에 도달함을 확인할 수 있다.

$\dot{\boldsymbol{x}}=f(\boldsymbol{x})+g(\boldsymbol{x})u$ 꼴의 플랜트를 SMC를 통해 제어해보자. $\dot{V}=S\dot{S}\le-\eta|S|$를 만족하기 위한 u를 찾아야한다. $S\dot{S}=S(\frac{\partial S}{\partial \boldsymbol{x}}(f(\boldsymbol{x})+g(\boldsymbol{x})u))$에서

$$u=-(\frac{\partial S}{\partial \boldsymbol{x}}g(\boldsymbol{x}))^{-1}(\frac{\partial S}{\partial \boldsymbol{x}}f(\boldsymbol{x})+\rho sgn(S))$$

로 잡아보자. 그러면

$$S\dot{S}=S(\frac{\partial S}{\partial \boldsymbol{x}}f(\boldsymbol{x})+\frac{\partial S}{\partial \boldsymbol{x}}g(\boldsymbol{x})u)=S(-\rho sgn(S))=-\rho|S| \quad (\because |S|=Ssgn(S))$$

이 되어 $\rho \ge \eta$이면 $S\dot{S}\le-\eta|S|$를 만족한다. η은 임의의 양수이므로 ρ가 양수이기만 하면 $u=-(\frac{\partial S}{\partial \boldsymbol{x}}g(\boldsymbol{x}))^{-1}(\frac{\partial S}{\partial \boldsymbol{x}}f(\boldsymbol{x})+\rho sgn(S))$를 통해 유한시간 내에 $S=0$에 도달할 수 있고 앞서 살펴본 바와 같이 S를 적절히 설계하면 상태변수를 원점에 수렴시킬 수 있다.

SMC는 대표적인 강건제어 기법 중 하나이다. 그 이유는 parameter uncertainty에 대해서도 부등식을 상황에 맞게 잘 세운다면 앞서 살펴본 바와 같이 η-reachability condition을 만족시킬 수 있기 때문이다.

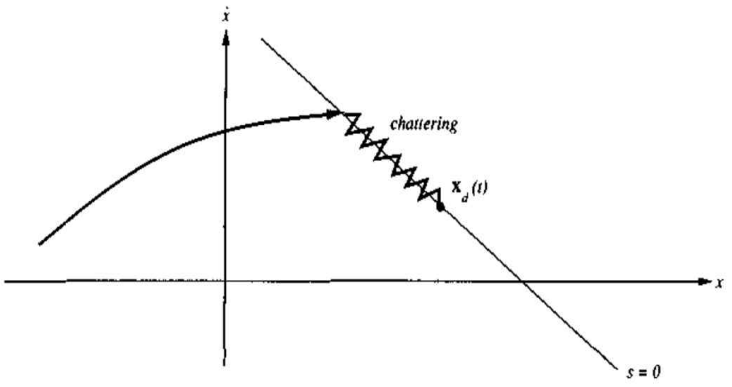


Fig 6. Chattering in SMC

SMC를 실제 구현하면서 생기는 문제점 중에 대표적인 것으로는 chattering problem이 있다. Chattering이란 Figure 6.에서와 같이 상태변수가 sliding surface에 도달한 뒤 surface에 수렴하지 못하고 진동하는 현상이다. 실제 digital 구현에서는 sampling period 동안에 제어값을 switching하지 못할뿐더러 actuator와 sensor의 dynamics로 인한 delay 효과 때문에 chattering이 발생한다.

또한 SMC는 matched uncertainty 조건에서만 작동 가능하다는 제약이 있다. Matched uncertainty란 system의 불확실성이나 외란에 관한 항이 입력항과 같은 위치(채널)에 존재해야 한다는 것이다.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_k &= f(\boldsymbol{x}) + g(\boldsymbol{x})u + d \end{aligned} \tag{14}$$

예를 들어, 식(14)과 같은 시스템은 외란 d가 입력 u와 같은 위치에 있어 matched uncertainty 조건을 만족하고 SMC로 제어 가능하다.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 + d \\ &\vdots \\ \dot{x}_k &= f(\boldsymbol{x}) + g(\boldsymbol{x})u \end{aligned} \tag{15}$$

그러나 식(15)와 같은 시스템은 외란 d가 입력 u로 제어할 수 없는 위치에 있으므로 unmatched uncertainty이고 SMC로 제어할 수 없다.

Filippov solution

SMC의 control law는 continuous하지 않기 때문에 closed-loop system 해의 존재성과 유일성이 보장되지 않는다(Lipschitz continuous하지 않으므로 Picard - Lindelöf theorem에 의해). 따라서 이러한 시스템의 해는 sense of Filippov의 관점에서 이해될 수 있다.

$\dot{x} \in F(x)$, $F(x) : \mathbb{R}^n \rightarrow$subset of $\mathbb{R}^n$(이러한 함수를 set-valued map이라고 한다.)인 경우, 이를 만족하는 $x(t)$를 diffential inclusion이라고 한다. 만약 $n=1$이라면 $\dot{x} \in F(x) = \{z \in \mathbb{R} : z \leq 1\}$을 만족하게 되고 최종적인 해는 $\dot{x}(t) \leq 1$이 된다.

$\dot{x} = f(x)$의 'solution in the sense of Filippov'란 아래와 같이 정의된 set-valued map $F(x)$에 대하여 differential inclusion $\dot{x} \in F(x)$의 해를 말한다.

$$F(x) = \begin{cases} \{f_+(x)\} & \text{if } x \in \Omega_+ \\ \{f_-(x)\} & \text{if } x \in \Omega_- \\ co\{\overline{f}_+(x), \overline{f}_-(x)\} & \text{if } x \in S \end{cases}$$

이때,

$$\overline{f}_+(x) := \lim_{z \rightarrow x, z \rightarrow \Omega_+} f_+(z)$$

$$\overline{f}_-(x) := \lim_{z \rightarrow x, z \rightarrow \Omega_-} f_-(z)$$

$$co\{\overline{f}_+(x), \overline{f}_-(x)\} = \{\alpha \overline{f}_+(x) + (1-\alpha)\overline{f}_- : \alpha \in [0.1]\}$$

를 의미한다.

Fillippov solution이 적용되는 경우는 위와 같이 상태공간 $\mathbb{R}^n$이 경계선 S에 의해 집합 Ω_+와 집합 Ω_-로 나누어져 있는 상황이다. 만약 시스템을 표현하는 식이 다음과 같다면,

$$\dot{x} = f(x) = \begin{cases} f_+(x), & at\ \Omega_+ \\ f_-(x), & at\ \Omega_- \\ don't\ care, & at\ S \end{cases}$$

단, $f_+(x)$와 $f_-(x)$는 C^1 vector fields.

수학적으로는 경계선 S에서 해가 존재하지 않는 상황이므로 고전적인 해의 개념으로는 해가 존재하지 않을 수 있다. 하지만 Fillippov solution으로 해를 정의하게 되면 위 시스템은 항상 해를 가질 수 있다.

Adaptive control

적응제어(adpative control)는 system parameter가 변하거나 uncertain한 경우 적용 가능한 제어 기법이다. 적응제어는 system의 신호를 이용하여 플랜트의 parameter를 추정하고 추정된 parameter들을 제어 입력에 반영하는 방식으로 작동한다. 이때 추정되는 parameter들은 adaptation rule에 따라 각 스텝마다 on-lline으로 계산된다.

적응제어와 비슷한 개념으로 강건제어(robust control)가 있다. 강건제어의 경우, system에 대한 사전 정보가 주어졌을 때(e.g. system parameter의 upper bound) 혹은 외란이 있을 때 이에 대한 영향을 억누르고 stability와 performance를 유지하는 것이 목적이다. 반면 적응제어는 system에 대한 사전 정보 없이 parameter estimation을 통해 control law를 바꿔가며 제어 목적을 달성한다. 적응제어는 adaptation이 진행됨에 따라 제어기의 performance가 향상되지만 강건제어는 단순히 consistent한 performance를 유지시킨다. 따라서 적응제어는 상수 혹은 천천히 변하는 unknown parameter를 포함하는 플랜트에 대해서 강건제어보다 더 유리하다.

적응제어 기법 중 하나인 model-reference adaptive control(MRAC)를 예시를 통해 이해해보자. 제어하고자 하는 플랜트는 $\dot{x}=ax+bu$ 꼴이고 a는 unknown constant로 설정하자. 이때 상태변수 x를 안정화시키기 위해서는 입력을 다음과 같이 설계해야 한다.

$$u=\frac{1}{b}(-ax-Kx)\ (K:\text{Hurwitz})$$

하지만 a가 unknown이므로 입력에 a에 관한 항이 포함될 수 없고 따라서 a 대신 a를 estimation한 $\hat{a}$을 대입한다. 그러면 입력은 다음과 같다.

$$u=\frac{1}{b}(-\hat{a}x-Kx)\ (K:\text{Hurwitz})$$

입력을 대입하면 전체 시스템은 다음과 같다.

$$\dot{x} = (a - \hat{a})x - Kx$$

$\tilde{a} := a - \hat{a}$로 정의하면,

$$\dot{x} = \tilde{a}x - Kx$$

이때 a와 $\hat{a}$이 충분히 가까워서 $\tilde{a}$이 0으로 간다면 상태변수 x도 0으로 간다. Stability condition을 판단하기 위해 Lyapunov method를 적용시켜보자. x와 $\tilde{a}$를 독립변수로 하는 좌표평면 상에서 (0,0)을 equilibrium point로 잡고 Lyapunov function candidate를 다음과 같이 설정한다.

$$V = \frac{1}{2}x^2 + \frac{1}{2}\tilde{a}^2$$

이렇게 잡은 V는 equilibrium point에서 0이고 이외의 점에서는 양수이므로 positive definite function이다. V를 미분하면 다음과 같다.

$$\begin{aligned}\dot{V} &= x\dot{x} + \tilde{a}\dot{\tilde{a}} \\ &= x(\tilde{a}x - Kx) + \tilde{a}(-\dot{\hat{a}}) \\ &= -Kx^2 + \tilde{a}(x^2 - \dot{\hat{a}})\end{aligned}$$

a가 constant이므로 $\dot{\tilde{a}}$은 $-\dot{\hat{a}}$이다. 이때 $\dot{\hat{a}} = x^2$이면 $\dot{V} = -Kx^2$이 되는데, $\tilde{a}$값에 상관없이 $x = 0$이면 $\dot{V} = 0$이 되므로 $\dot{V}$은 negative semi-definite function이다. 그리고 $\dot{\hat{a}} = x^2$를 update law라고 한다. Lyapunov theory에 따르면 $\dot{V}$이 negative definite function이어야 asymtotically stable한 시스템이다. 그렇지만 $\dot{V}$가 negative semi-definite일 때도 특정 조건을 만족시킨다면 asymtotically stable하다는 것을 아래의 정리를 통해 보일 수 있다.

Def) Invariant set

Dynamic system의 trajectory가 어떤 집합 G에서 출발하여 모든 시간 t에 대하여 G 안에 머물러 있을 때 G를 invariant set으로 정의한다.

Thm) Local invariant set theorem

$\dot{x}=f(x)$꼴의 autonomous system을 생각하자. f는 연속이고 scalar function $V(x)\in C^1$이다. 이때

· Ω_l : $l>0$에 대해서 $V(x)<l$인 영역

· $\dot{V}(x)\le 0$ in Ω_l

· R : $\dot{V}(x)=0$을 만족하는 Ω_l 내부의 영역

· M : R 내부의 가장 큰 invariant set

라고 하자. 그러면 $t\to\infty$일 때, Ω_l에서 출발한 모든 solution $x(t)$는 M에 점근적으로 가까워진다.

이를 global한 영역으로 확장한 정리는 다음과 같다.

Thm) Global invariant set theorem

$\dot{x}=f(x)$꼴의 autonomous system을 생각하자. f는 연속이고 scalar function $V(x)\in C^1$이다.

· $V(x)\to\infty$ as $\|x(t)\|\to\infty$

· $\dot{V}(x)\le 0$

· R : $\dot{V}(x)=0$을 만족하는 점들의 집합

· M : R 내부의 가장 큰 invariant set

라고 하자. 그러면 $t\to\infty$일 때, 모든 solution $x(t)$는 M에 점근적으로 가까워진다.

위 시스템을 살펴보면 , $M=\{(0,0)\}$이다. 따라서 global invariant set theorem에 의해 모든 solution $x(t)$는 원점에 점근적으로 가까워진다. 즉, $\dot{\hat{a}}=x^2$일 때 전체 시스템은 asymtotically stable하고, $\dot{\hat{a}}=x^2$이 올바른 estimation임을 확인할 수 있다.

unknown parameter의 수렴 속도를 조절하기 위해 Lyapunov function candidate를

$$V=\frac{1}{2}x^2+\frac{1}{2\gamma}\tilde{a}^2$$

와 같이 잡기도 하는데, 이때 $\hat{a}$은 $\dot{\hat{a}}=\gamma x^2$으로 추정할 수 있다. 그리고 추정값의 수렴 속도를 조절하는 hyper parameter γ를 adaptation gain이라고 한다.

- Controller design

앞서 살펴본 바와 같이, sliding mode control을 할 때, SMC의 Lyapunov function candidate $V(s)$에 대하여 $\frac{d}{dt}V(s) < 0$를 만족시키도록 control law u를 설계하는 과정이 존재한다. 이때 $\dot{V} = S\dot{S} \leq -\eta|S|$ 꼴을 만들기 위해선 u에 플랜트의 parameter들을 삭제시킬 수 있는 항들이 존재해야 하고, 따라서 플랜트에서 u 앞에 곱해져 있는 $g(\boldsymbol{x})$가 invertible하여 u에 $g(\boldsymbol{x})^{-1}$항이 포함되어야 한다. 하지만 $g(\boldsymbol{x})$는 6x4 행렬이므로 invertible하지 않다. 정방행렬을 가지는 시스템을 만들기 위해 부가변수 $\boldsymbol{g_s}$와 $\boldsymbol{u_s}$를 각각 $g(\boldsymbol{x})$, $\boldsymbol{u}$에 추가한다. 이를 통해 쿼드로터의 동역학을 정리하면,

$$\ddot{\boldsymbol{x}} = f(\boldsymbol{x}) + G(\boldsymbol{x})\,U - \nu + f_r(\boldsymbol{x}) \tag{16}$$
$$G = [g(\boldsymbol{x}), \boldsymbol{g_s}],\ U = [\boldsymbol{u}^{\boldsymbol{T}}, \boldsymbol{u_s^T}]^T,\ \nu = \boldsymbol{g_s u_s}$$

만약 $\boldsymbol{g_s}$를

$$\boldsymbol{g_s} = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \end{bmatrix}^T$$

로 잡는다면 $G(\boldsymbol{x})$를 invertible하게 만들수 있고 이때,

$$G^{-1}(\boldsymbol{x}) \approx \begin{bmatrix} 0\,0\,1 & 0 & 0 & 0 \\ 0\,0\,0 & 1/l & 0 & 0 \\ 0\,0\,0 & 0 & 1/l & 0 \\ 0\,0\,0 & 0 & 0 & 1 \\ 1\,0\,0 & 0 & 0 & 0 \\ 0\,1\,0 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

이 된다. 한편, $\boldsymbol{u_s} = [u_5, u_6]^T$로 놓으면

$$\nu = \boldsymbol{g_s u_s} = [u_5, u_6, 0, 0, 0, 0]^T$$

가 된다.

다음으로 sliding phase를 설계해보자. error vector를 다음과 같이 정의하자.

$$\boldsymbol{e} := \boldsymbol{x} - \boldsymbol{x}_d \quad (\boldsymbol{x}_d = [x_d, y_d, z_d, \phi_d, \theta_d, \psi_d]^T)$$

Sliding surface를 잡고 $S=0$을 유지하면서 이 error vector를 0으로 수렴시켜야 한다. 이때,

$$S = [s_1, s_2, s_3, s_4, s_5, s_6]^T = \dot{\boldsymbol{e}} + K\boldsymbol{e}$$

로 sliding surface를 잡아보자. 각 s_i가 sliding surface고 S는 6개 sliding surface를 모아놓은 벡터이다. 각 sliding surface를 구성하는 state variable은 e_i와 $\dot{e_i}$이다. K의 eigenvalue가 모두 양수가 되게 하면 $S=0$일 때 error dynamics를 stable하게 만들 수 있고 상태변수가 desired value를 추종하게 된다. 이를 만족하도록 K는 다음과 같이 잡는다.

$$K = diag[k_1, \ldots, k_6] \quad (k_i > 0 \ \text{for}\ ^{\forall} i)$$

다음으로, 적응제어를 위해 Lyapunov method를 적용해보자. 수정된 플랜트 $\ddot{\boldsymbol{x}} = f(\boldsymbol{x}) + G(\boldsymbol{x})U - \nu + f_r(\boldsymbol{x})$에서 추정해야 할 parameter는 ν와 $f_r(\boldsymbol{x})$이다. $\tilde{\nu} := \nu - \hat{\nu}$, $\tilde{f_r} := f_r - \hat{f_r}$이라 하면 원하는 제어 목적이 $S=0$, $\tilde{\nu}=0$, $\tilde{f_r}=0$이므로 equilibrium point를 $(S, \tilde{\nu}, \tilde{f_r}) = (0,0,0)$으로 하는 Lyapunov function을 다음과 같이 놓자.

$$L = \frac{1}{2}S^T S + \frac{1}{2}\tilde{\nu}^T \Gamma \tilde{\nu} + \frac{1}{2}\tilde{f_r}^T \Omega \tilde{f_r}$$

SMC의 reaching phase에서는 $\frac{1}{2}S^TS>0$이므로 이미 $\frac{1}{2}S^TS$항 만으로도 L이 positive definite하다는 조건이 만족된다. 따라서 나머지 항은 $\tilde{\nu}^T\Gamma\tilde{\nu}\geq 0,\ \tilde{f_r}^T\Omega\tilde{f_r}\geq 0$ 만을 만족하면 되므로 Γ, Ω는 positive semi-definite matrix이면 된다. Γ, Ω는 adaptation gain의 역할을 한다. 적응제어를 적용하기 위해 ν와 f_r이 매우 천천히 변하는 parameter라고 가정하면 $\dot{\tilde{\nu}}\approx-\dot{\hat{\nu}},\ \dot{\tilde{f}}_r\approx-\dot{\hat{f}}_r$라고 할 수 있다. L은 이미 positive definite으로 설계했고, $\dot{L}$을 negative definite하게 설계하여 stability를 만족시키도록 해보자. $\dot{L}$을 구해보면,

$$\begin{aligned}\dot{L}&=S^T\dot{S}+\tilde{\nu}\Gamma\dot{\tilde{\nu}}+\tilde{f}_r^T\Omega\tilde{f}_r\\&=S^T(\ddot{e}+K\dot{e})+\tilde{\nu}\Gamma\dot{\tilde{\nu}}+\tilde{f}_r^T\Omega\tilde{f}_r\\&=S^T[f(x)+G(x)U-\nu+f_r-\ddot{x}_d+K\dot{e}]+\tilde{\nu}^T\Gamma(-\dot{\hat{\nu}})+\tilde{f}_r^T\Omega(-\dot{\hat{f}}_r)\end{aligned}$$

이 된다. $\dot{L}\leq-\eta|S|$꼴을 만들기 위해서 U를 다음과 같이 설계하자.

$$U=G^{-1}(x)[-f(x)+\hat{\nu}-\hat{f}_r+\ddot{x}_d-K\dot{e}-diag[c_1,\ldots,c_6]sgn(S)] \tag{18}$$

U를 대입하면 다음과 같다.

$$\dot{L}=\tilde{\nu}^T(-S-\Gamma\dot{\hat{\nu}})+\tilde{f}_r^T(S-\Omega\dot{\hat{f}}_r)-C^T|S|$$

$$C=[c_1,\ldots,c_6]^T$$

이때 만약 $-S-\Gamma\dot{\hat{\nu}}=0,\ S-\Omega\dot{\hat{f}}_r=0$이고 $c_i>0$라면

$$\dot{L}=-C^T|S| \tag{19}$$

가 된다. 위의 식에는 $\tilde{\nu}$와 $\tilde{f}_r$ 항이 없으므로 $\dot{L}$은 negative semi-definite이고 global invariant set theorem에 의해 상태변수가 유한시간내에 $S=0$에 도달함과 더불어 $\tilde{\nu}$

와 $\tilde{f}_r$이 점근적으로 0에 접근한다. update law는 $-S-\Gamma\dot{\hat{\nu}}=0$와 $S-\Omega\dot{\hat{f}}_r=0$이다. $\Gamma=diag[110000]$, $\Omega=diag[001000]$로 잡으면 update law는 다음과 같다.

$$\dot{\hat{\nu}}=-[\dot{e_x}+k_1e_x, \dot{e_y}+k_2e_y, 0, 0, 0, 0]^T$$

$$\dot{\hat{f}}_r=[0, 0, \dot{e_z}+k_3e_z, 0, 0, 0]^T$$

플랜트가 steady-state에 도달했을 때 estimates $\hat{\nu}$, $\hat{f}_r$이 각각 unknown parameter ν, f_r를 추종하는지 검증해보자. 식(16)에 식(18)을 대입하면 다음과 같다.

$$\ddot{\boldsymbol{x}}=\hat{\nu}-\nu+f_r-\hat{f}_r+\ddot{\boldsymbol{x}}_d-K\dot{\boldsymbol{e}}-diag[c_1, \ldots, c_6]sgn(S)$$

$$\ddot{\boldsymbol{e}}+K\dot{\boldsymbol{e}}=\hat{\nu}-\nu+f_r-\hat{f}_r-diag[c_1, \ldots, c_6]sgn(S)$$

$$\dot{S}=\hat{\nu}-\nu+f_r-\hat{f}_r-diag[c_1, \ldots, c_6]sgn(S)$$

한편, steady-state는 sliding phase에 속하므로 이론적으로 $S=0$이고 따라서 $\dot{S}=0$, $sgn(S)=0$이다. 최종적으로, $\nu-\hat{\nu}=f_r-\hat{f}_r$이 성립하고 이를 풀어쓰면 다음과 같다.

$$\begin{bmatrix} u_5-\hat{u_5} \\ u_6-\hat{u_6} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ g_r-\hat{g_r} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

따라서 estimates가 unknown parameter를 잘 추종한다고 말할 수 있다.

· Translational position controller 설계

Input U의 각 항을 살펴보자.

$$U = G^{-1}\left(\begin{bmatrix}0\\0\\g\\0\\0\\0\end{bmatrix} - \begin{bmatrix}\int \dot{e}_x + k_1 e_x dt\\ \int \dot{e}_y + k_2 e_y dt\\ \int \dot{e}_z + k_3 e_z dt\\ 0\\0\\0\end{bmatrix} + \begin{bmatrix}\ddot{x}_d\\ \ddot{y}_d\\ \ddot{z}_d\\ \ddot{\phi}_d\\ \ddot{\theta}_d\\ \ddot{\psi}_d\end{bmatrix} - K\begin{bmatrix}\dot{e}_x\\ \dot{e}_y\\ \dot{e}_z\\ \dot{e}_\phi\\ \dot{e}_\theta\\ \dot{e}_\psi\end{bmatrix} - diag[c_1,\ldots,c_6]sgn\left(\begin{bmatrix}\dot{e}_x + k_1 e_x\\ \dot{e}_y + k_2 e_y\\ \dot{e}_z + k_3 e_z\\ \dot{e}_\phi + k_4 e_\phi\\ \dot{e}_\theta + k_5 e_\theta\\ \dot{e}_\psi + k_6 e_\psi\end{bmatrix}\right)\right)$$

이때 G^{-1}가 식(17)과 같으므로 u_1, u_2, u_3, u_4는 각각 $e_z, e_\phi, e_\theta, e_\psi$, 그에 대한 적분과 1계 미분에 관한 식으로 표현된다. 따라서 위의 제어기만을 가지고 x,y좌표를 제어할 수 없다. 그러므로 translational position controller가 필요하다.

Small angle assumption에 따라 $\ddot{x} = u_1 \sin\theta$, $\ddot{y} = -u_1 \sin\phi$이다. 이를 시스템으로 보면 θ를 입력으로 하여 출력 x가 나오고 ϕ를 입력으로 하여 출력 y가 나온다고 해석할 수 있다. 따라서 x,y좌표를 제어하기 위해서는 θ와 ϕ를 적절한 값으로 설정해야 한다. 일반적으로 θ_d와 ϕ_d를 각각 x와 y에 관한 식으로 만들어 position을 제어하는데 이를 translational position controller라고 한다. Translational position control에 가장 쉬우면서도 흔히 사용되는 제어 기법은 PID 제어이다. 원하는 좌표인 x_d와 현재 x좌표 차이가 크다면 큰 값의 θ가 필요하고 차이가 줄어들수록 θ값도 작아져야 한다. 이는 y좌표 제어에서도 마찬가지이다. 본 논문에서는 proportional gain이 1인 PD 제어기를 사용하였다. 앞서 error variable을 (current value)-(desired value)로 정의하였으므로 PD controller를 식으로 나타내면 다음과 같다.

$$\phi_d = -\dot{e}_y - k_\phi e_y$$
$$\theta_d = -\dot{e}_x - k_\theta e_x$$

· Advantage of adaptive sliding mode controller

앞서 살펴본 adaptive sliding mode controller(ASMC)에서 adaptive control이 non-adaptive case에 비해 갖는 장점을 살펴보자. Adaptive control일 때 $\dot{L}$는 식(19)와 같다. 이를 풀어 쓰면 다음과 같다.

$$\dot{L} = S^T \begin{bmatrix} -c_1 sgn|s_1| \\ -c_2 sgn|s_2| \\ -c_3 sgn|s_3| \\ -c_4 sgn|s_4| \\ -c_5 sgn|s_5| \\ -c_6 sgn|s_6| \end{bmatrix} < 0$$

다음으로 non-adaptive case를 살펴보자. Non-adaptive case에서는 ν와 f_r을 추정하지 않는다. 따라서 Lyapunov function은 $L_{non-adaptive} = \frac{1}{2} S^T S$이다. $\dot{L}_{non-adaptive}$를 구해보면

$$\begin{aligned} \dot{L}_{non-adaptive} &= S^T \dot{S} = S^T(\ddot{\boldsymbol{e}} + K\dot{\boldsymbol{e}}) \\ &= S^T(f(\boldsymbol{x}) + G(\boldsymbol{x})U - \nu + f_r(\boldsymbol{x}) - \ddot{\boldsymbol{x}}_d + K\dot{\boldsymbol{e}}) \end{aligned}$$

이다. 따라서 $U_{non-adaptive}$는 다음과 같이 설계된다.

$$U_{non-adaptive} = G^{-1}(-f(\boldsymbol{x}) + \ddot{\boldsymbol{x}}_d - K\dot{\boldsymbol{e}} - diag[c_1, \ldots, c_6] sgn(S))$$

$U_{non-adaptive}$를 식()에 대입하면 다음과 같다.

$$\dot{L}_{non-adaptive} = S^T(-\nu + f_r(\boldsymbol{x}) - diag[c_1, \ldots, c_6] sgn(S))$$

이를 풀어쓰면 다음과 같다.

$$\dot{L}_{non-adaptive} = S^T \begin{bmatrix} -u_5 - c_1 sgn|s_1| \\ -u_6 - c_2 sgn|s_2| \\ g_r - c_3 sgn|s_3| \\ -c_4 sgn|s_4| \\ -c_5 sgn|s_5| \\ -c_6 sgn|s_6| \end{bmatrix}$$

Adaptive case에서는 $c_i < 0$이기만 하면 $\dot{L} < 0$이 만족된다. 하지만 non-adaptive case에서는 $\dot{L}_{non-adaptive}$이 negative definite이 되려면 c_1, c_2, c_3가 각각 u_5, u_6, g_r항을 compensate해야 한다. 따라서 $U_{non-adaptive}$의 첫 번째, 두 번째, 세 번째 element의 magnitude가 U보다 커지게 되고 이로 인해 chattering 문제가 야기되거나 전력 효율 면에서 불리하다는 단점이 있다.

· Sensor noise analysis

Calibration error를 다음과 같이 정의하자.

$$E = [\epsilon_x, \epsilon_y, \epsilon_z, \epsilon_\phi, \epsilon_\theta, \epsilon_\psi]^T$$

상태변수를 X로 놓으면 상태변수의 측정값은 다음과 같다.

$$\hat{X} = X + E$$

따라서 측정값의 각 성분에 calibration error가 포함된다. 또한 ϕ_d, θ_d는 error variable로 이루어져 있으므로 역시 calibration error가 포함된다.

(*)항에 포함되어 있는 error term을 η_*라고 하자. 예를 들어, $\hat{X}_d$는 다음과 같으므로

$$\hat{X}_d = \begin{bmatrix} x_d \\ y_d \\ z_d \\ \dot{y} + \hat{\epsilon_y} - \dot{y_d} + k_\phi (y + \epsilon_y - y_d) \\ \dot{x} + \hat{\epsilon_x} - \dot{x_d} + k_\theta (x + \epsilon_x - x_d) \\ \psi_d \end{bmatrix}$$

η_{X_d}는 다음과 같다.

$$\eta_{X_d} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\epsilon}_y + k_\phi \epsilon_y \\ \dot{\epsilon}_x + k_\theta \epsilon_x \\ 0 \end{bmatrix}$$

같은 방법으로, $\eta_{\ddot{X}_d}$, $\eta_{\dot{e}}$를 구해보면 다음과 같다.

$$\eta_{\ddot{X}_d} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \epsilon_y^{(3)} + k_\phi \ddot{\epsilon}_y \\ \epsilon_x^{(3)} + k_\theta \ddot{\epsilon}_x \\ 0 \end{bmatrix}, \quad \eta_{\dot{e}} = \begin{bmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_z \\ \ddot{\epsilon}_\phi - \ddot{\epsilon}_y - k_\phi \dot{\epsilon}_y \\ \ddot{\epsilon}_\theta - \ddot{\epsilon}_x - k_\theta \dot{\epsilon}_x \\ \dot{\epsilon}_\psi \end{bmatrix}$$

최종적으로 η_U를 구해보자. U에서 error term이 등장하는 부분은 $\ddot{X}_d - K\dot{e}$이므로 앞서 구한 결과를 활용하면 다음과 같다.

$$\eta_U = G^{-1} \begin{bmatrix} -k_1 \dot{\epsilon}_x \\ -k_2 \dot{\epsilon}_y \\ -k_3 \dot{\epsilon}_z \\ \epsilon_y^{(3)} + k_\phi \ddot{\epsilon}_y - k_4 (\ddot{\epsilon}_\phi - \ddot{\epsilon}_y - k_\phi \dot{\epsilon}_y) \\ \epsilon_x^{(3)} + k_\theta \ddot{\epsilon}_x - k_5 (\ddot{\epsilon}_\theta - \ddot{\epsilon}_x - k_\theta \dot{\epsilon}_x) \\ -k_6 \dot{\epsilon}_\psi \end{bmatrix}$$

4. Simulation results

시뮬레이션은 MATLAB과 SIMULINK를 이용하여 수행하였다. 제어 목표는 쿼드로터가 roll, pitch, yaw 모션을 최소화하며 주어진 목표 지점에 착륙하는 것이다. 쿼드로터는 위치 좌표 (10, 10, 20)에서 출발하여 waypoint 지점 (20, -10, 10)을 거친 뒤 원점 (0, 0, 0)에 착륙하여야 한다. 시뮬레이션은 총 30초 동안 수행하였으며, 처음 15초 동안은 초기 지점에서 waypoint까지 비행하여 도달하고 남은 15초 동안은 waypoint에서 원점까지 비행하여 착륙하도록 하였다.

시뮬레이션에 사용한 모델 파라미터 및 초깃값은 다음과 같다.

l = 1 m

$x(0)$ = 10 m, $y(0)$ = 10 m, $z(0)$ = 20 m

$\phi(0)$ = 30 deg, $\theta(0)$ = 30 deg, $\psi(0)$ = 30 deg

· FL

FL controller에 사용되는 gain $[k_{x1}, \dots, k_{x4}]^T$, $[k_{y1}, \dots, k_{y4}]^T$, $[k_{z1}, \dots, k_{z4}]^T$는 LQR(Linear Quadratic Regulator)을 이용하여 설정하였다. Controller spec을 만족시키는 gain $K = [k_4 k_3 k_2 k_1]$는 다음과 같은 characteristic polynomial을 안정화시키는 gain이다.

$$s^4 - k_1 s^3 - k_2 s^2 - k_3 s - k_4$$

따라서 gain K는 companion form을 활용하면

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu$$

$$y = C\boldsymbol{x}$$

$$A = \begin{bmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},\ C = [1000],\ u = -K\boldsymbol{x}$$

로 표현되는 시스템을 안정화시키는 gain이 된다. Positive definite matrix $Q = diag[1, 10, 5, 1]$, $R = 0.01$로 놓고 cost function

$$J(u) = \int_0^\infty (\boldsymbol{x}^{\boldsymbol{T}} Q\boldsymbol{x} + u^T Ru)dt$$

를 최소화하는 gain K를 구하면 다음과 같다.

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix} = \begin{bmatrix} k_{x1} \\ k_{x2} \\ k_{x3} \\ k_{x4} \end{bmatrix} = \begin{bmatrix} k_{y1} \\ k_{y2} \\ k_{y3} \\ k_{y4} \end{bmatrix} = \begin{bmatrix} k_{z1} \\ k_{z2} \\ k_{z3} \\ k_{z4} \end{bmatrix} = \begin{bmatrix} 10.00 \\ 42.49 \\ 40.27 \\ 13.43 \end{bmatrix}$$

또한 ψ control에 쓰이는 gain $k_{\psi 1}$, $k_{\psi 2}$는 PD gain tunning을 이용하여 각각 30, 50으로 설정하였다.

위 파라미터들을 대입하여 시뮬레이션을 수행하면 그 결과는 다음과 같다.

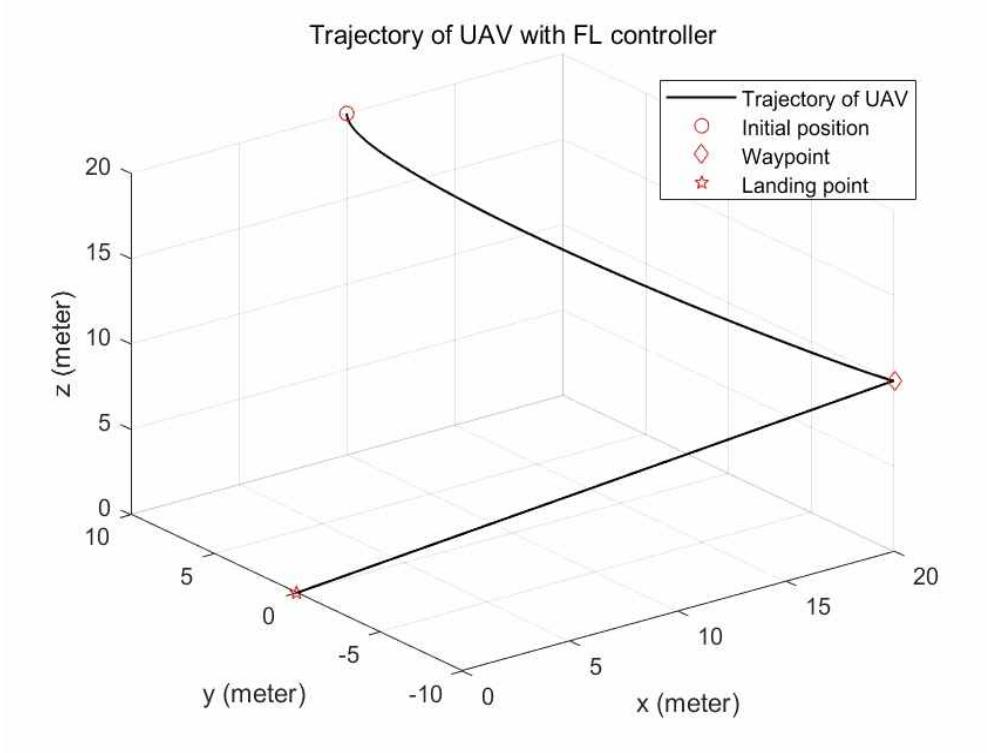
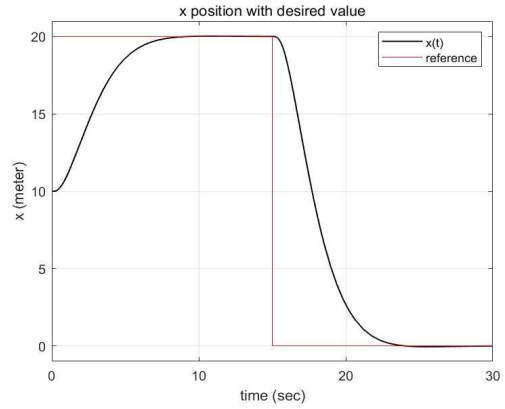
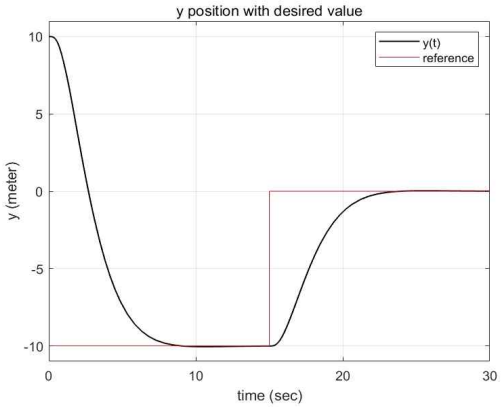


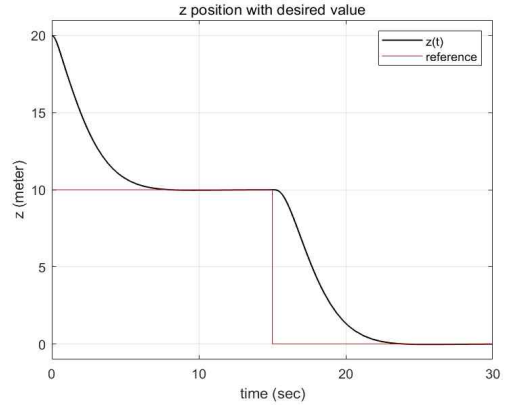
Fig 7. Trajectory of UAV in 3-D axes with FL controller without uncertainty and sensor noise



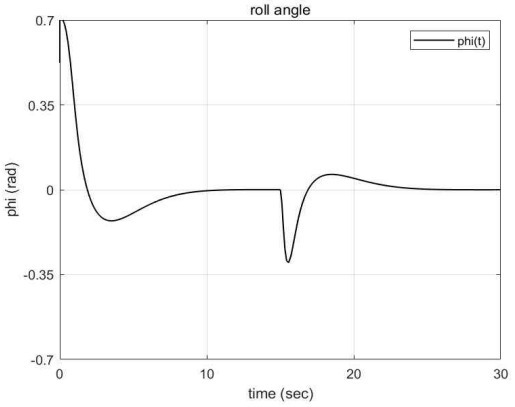
(a)



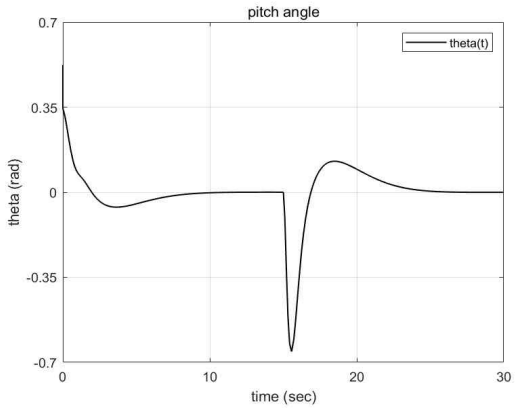
(b)



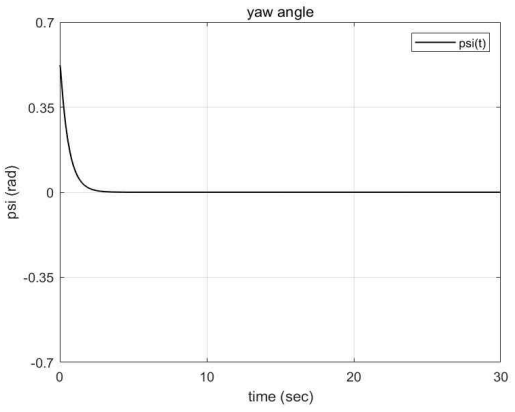
(c)



(d)



(e)

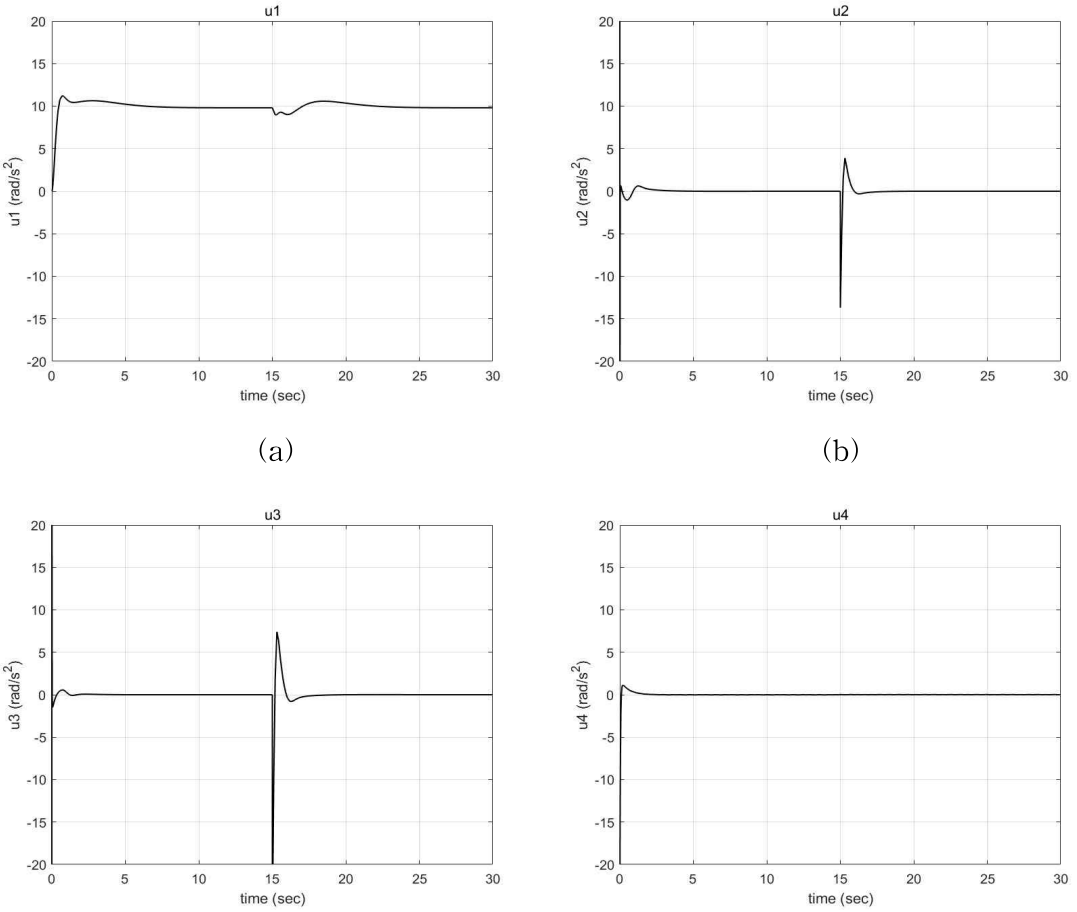


(f)

Fig 8. FL controller results without uncertainty and sensor noise

(a), (b), (c) : x, y, z positions, (d), (e), (f) : roll, pitch, yaw angles

black line : state variables of UAV, red line : desired values



(a) (b)

(c) (d)

Fig 9. FL controller inputs without uncertainty and sensor noise

· ASMC without sensor noise

ASMC 시뮬레이션에 사용된 파라미터 값은 다음과 같다.

$C = [5, 5, 5, 1, 1, 1]^T$

$K = diag[1, 1, 0.7, 5, 5, 10]$

$A = 0.4668$

$z_0 = 2\text{m}$

또한 후술할 문제점으로 인해 ϕ_d와 θ_d를 다음과 같은 PID controller를 통해 설계하였다.

$$\phi_d = -k_P e_y - k_I \int_0^t e_y dt - k_D \dot{e_y}$$

$$\theta_d = -k_P e_x - k_I \int_0^t e_x dt - k_D \dot{e_x}$$

각 gain 값은 $k_P = 5$, $k_D = 3$, $k_I = 3$이다.

그리고 ASMC에 포함되는 $sgn(S)$에 의한 chattering problem을 없애기 위해 $sgn(S)$ 대신 S를 사용하였다.

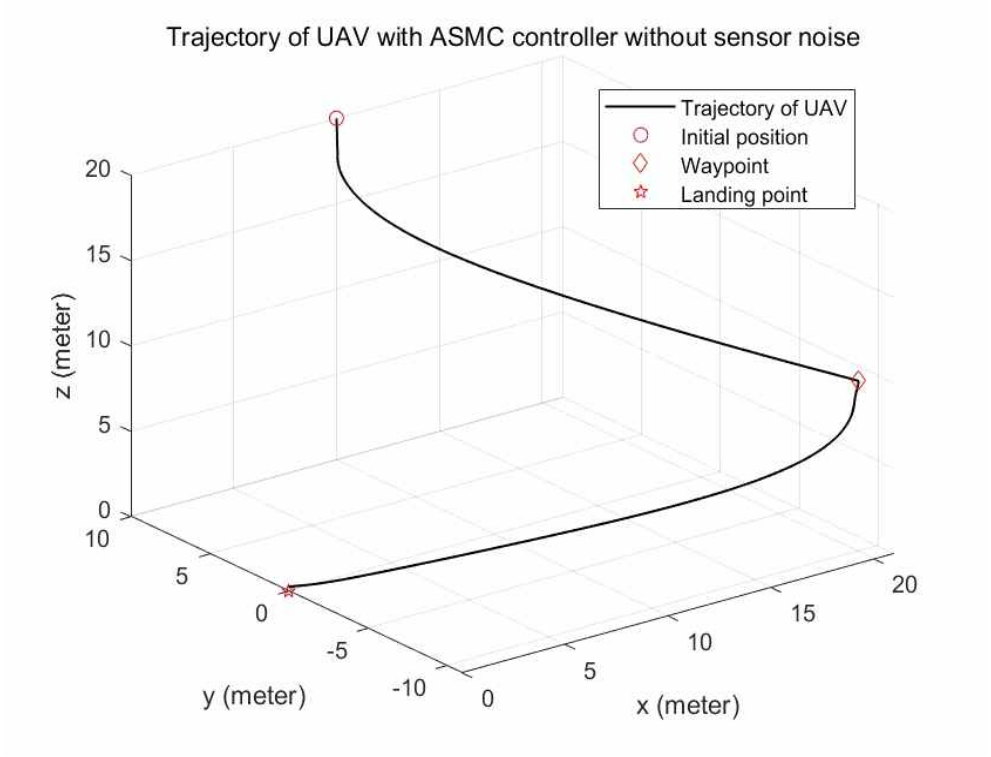


Fig 10. Trajectory of UAV in 3-D axes with ASMC controller with uncertainty but without sensor noise

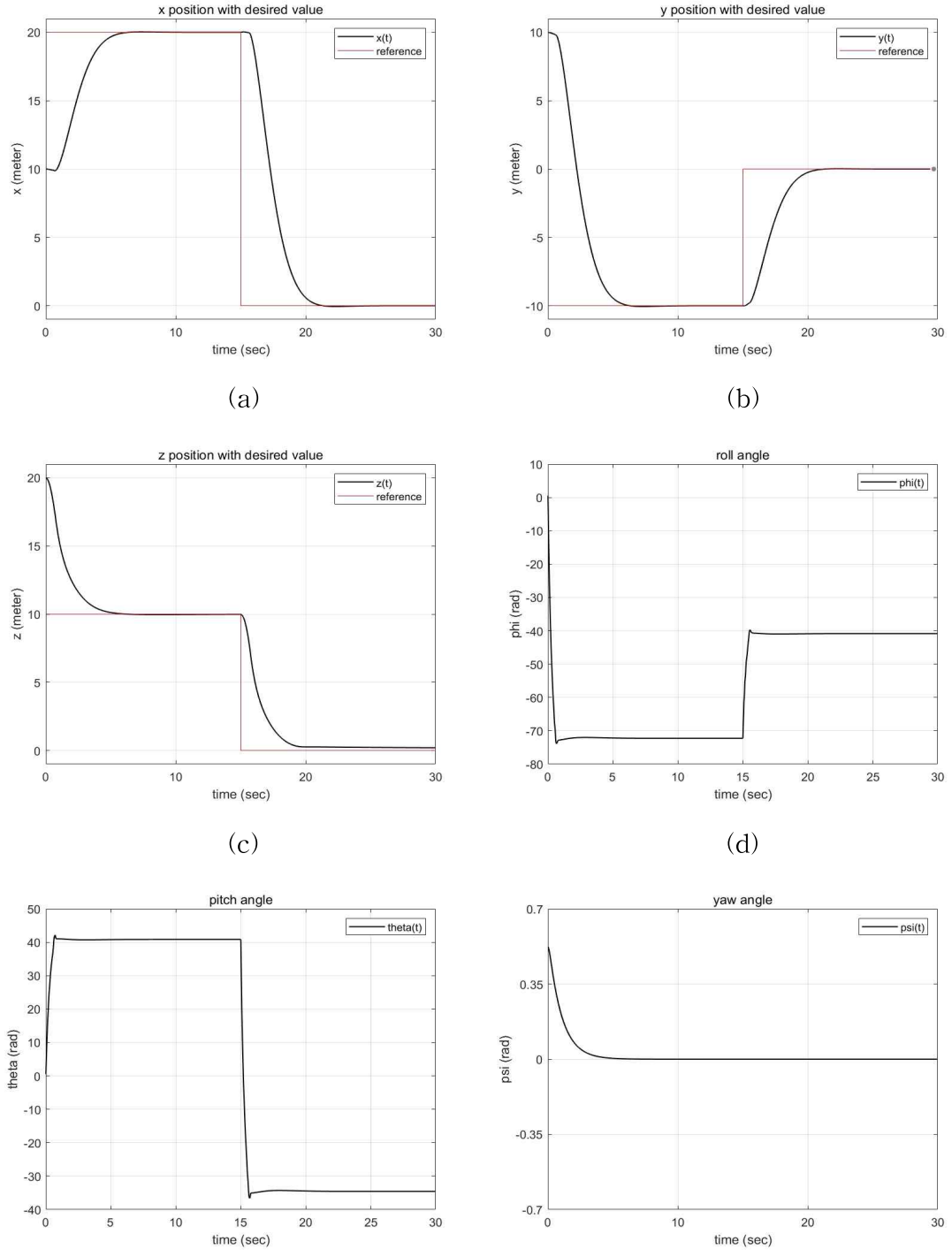


Fig 11. ASMC controller results with uncertainty but without sensor noise
(a), (b), (c) : x, y, z positions, (d), (e), (f) : roll, pitch, yaw angles
black line : state variables of UAV, red line : desired values

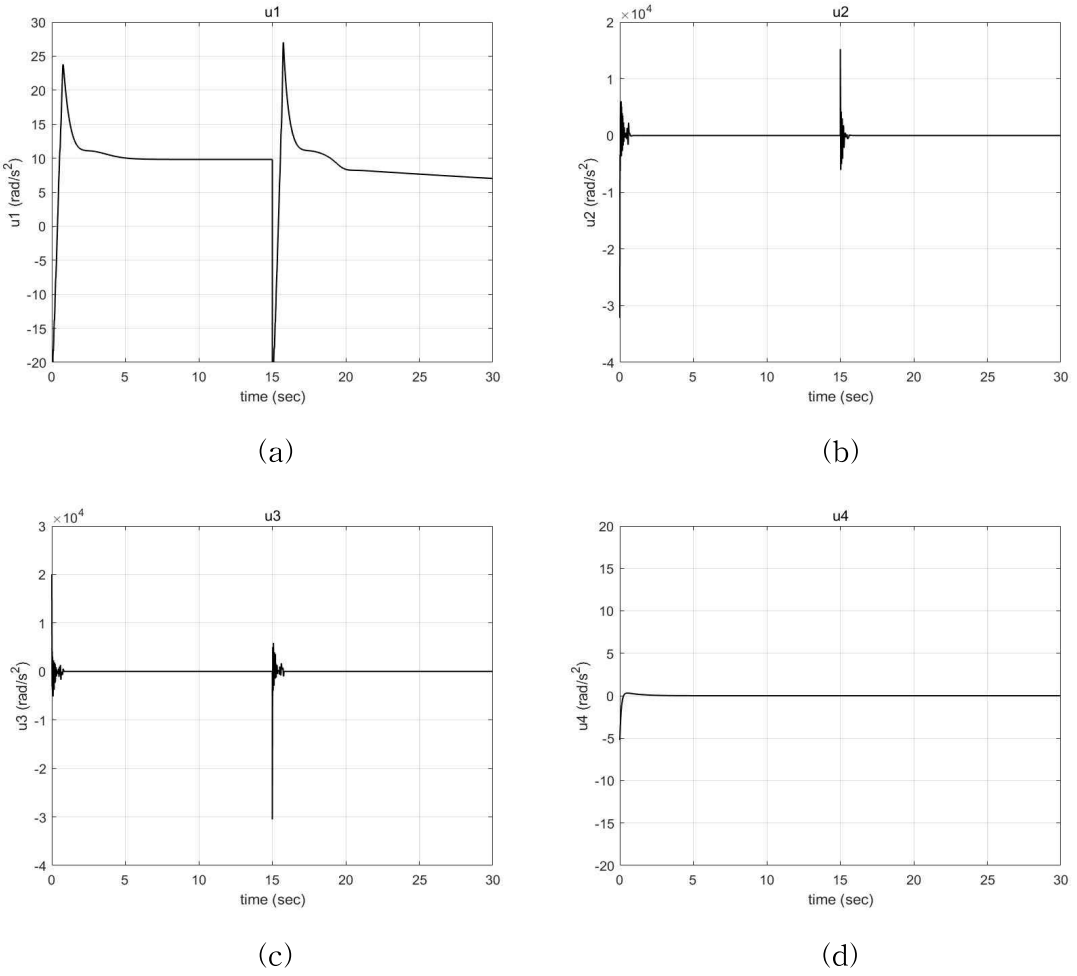
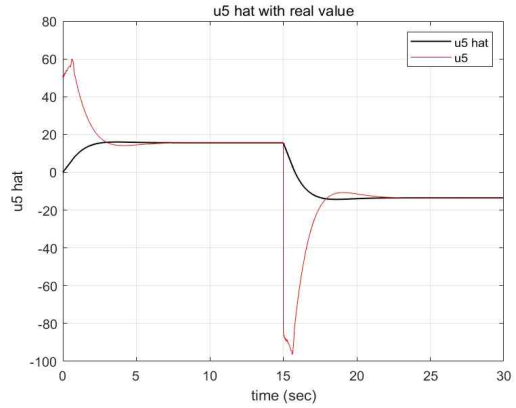
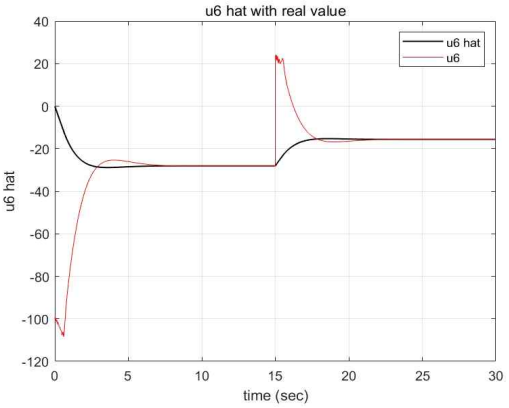


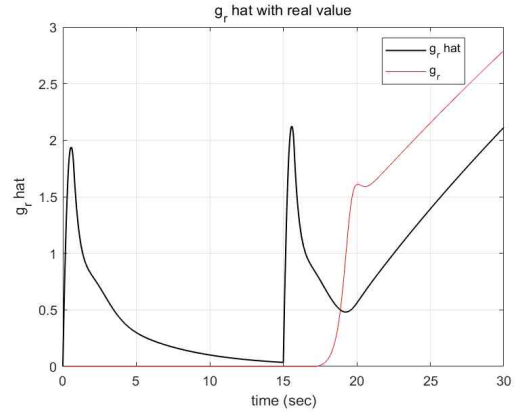
Fig 12. ASMC controller inputs with uncertainty but without sensor noise



(a)



(b)



(c)

Fig 13. ASMC controller estimates with uncertainty but without sensor noise
black line : estimates, red line : real value

· ASMC without sensor noise

Position에 대한 sensor noise는 평균 0m, 표준편차 0.05m이고 roll, pitch, yaw angle에 대한 sensor noise는 평균 0rad, 표준편차 0.01rad으로 설정하였다.

입력 U에서 sensor noise의 영향을 줄이기 위해 다음과 같은 first order low pass filter를 달았다.

$$F(s) = \frac{1}{0.02s + 1}$$

이 filter의 Bode plot은 다음과 같다.

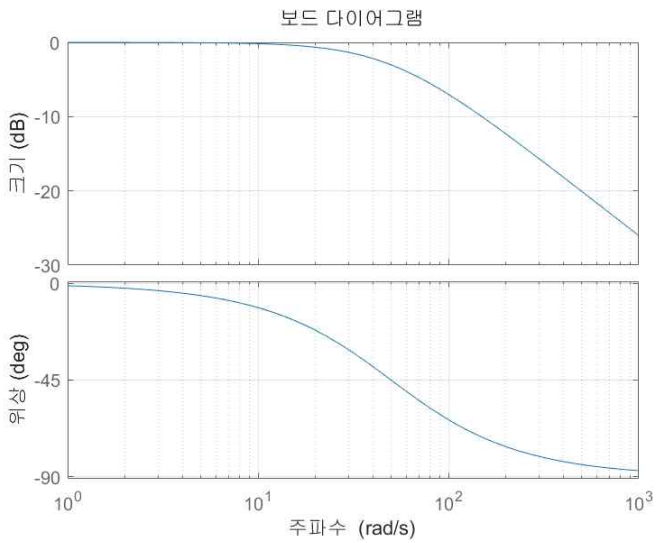


Fig 14. Bode plot of the LPF filter

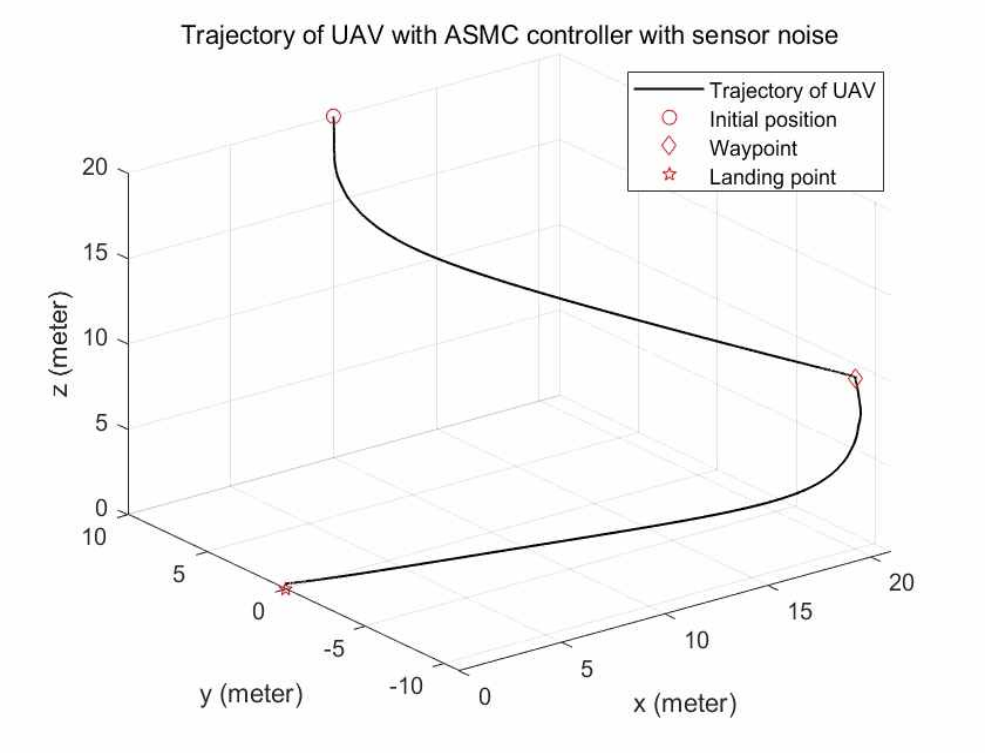


Fig 15. Trajectory of UAV in 3-D axes with ASMC controller with uncertainty and sensor noise

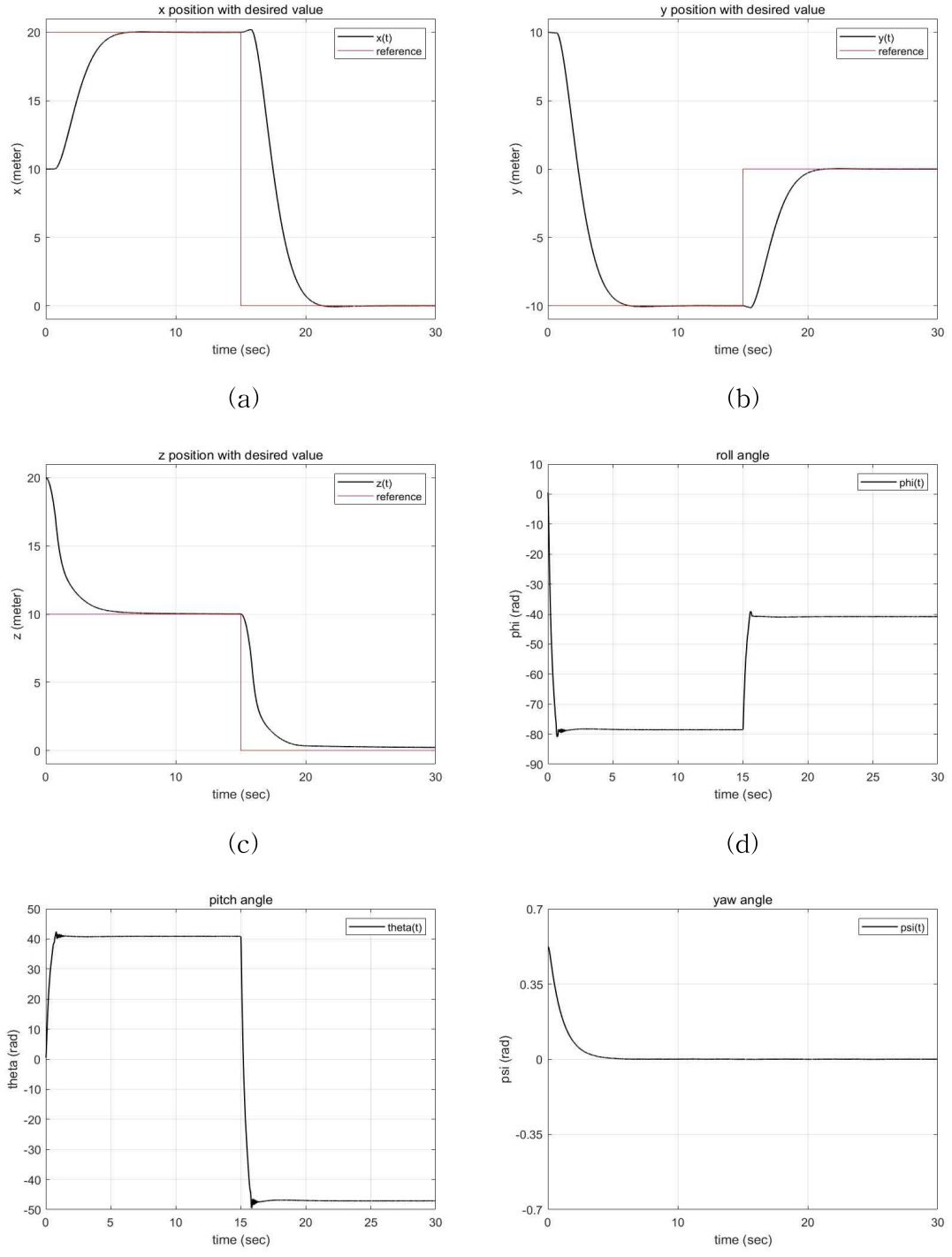
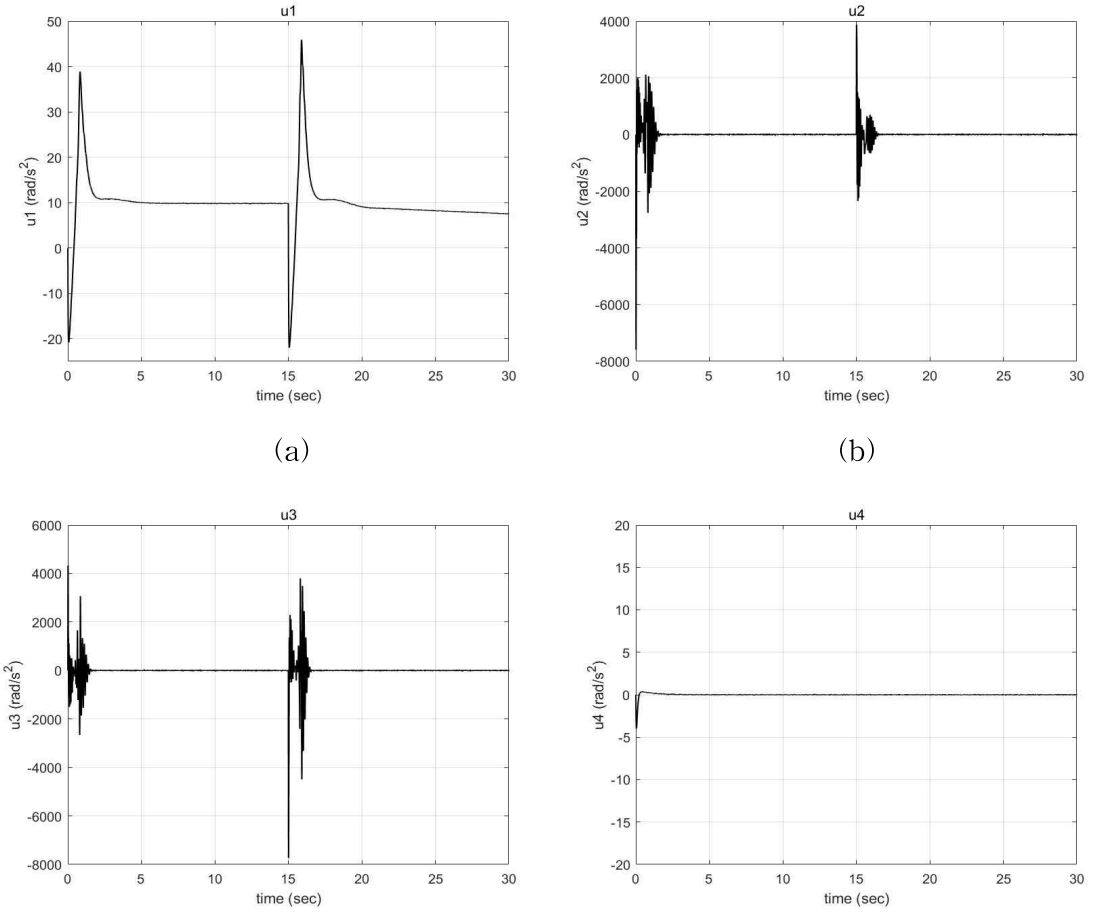


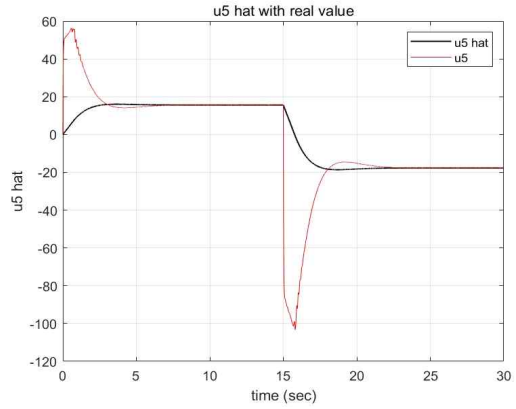
Fig 16. ASMC controller results with uncertainty and sensor noise (a), (b), (c) : x, y, z positions, (d), (e), (f) : roll, pitch, yaw angles black line : state variables of UAV, red line : desired values



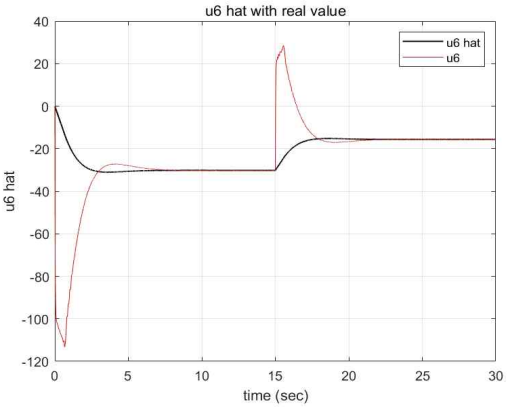
(a) (b)

(c) (d)

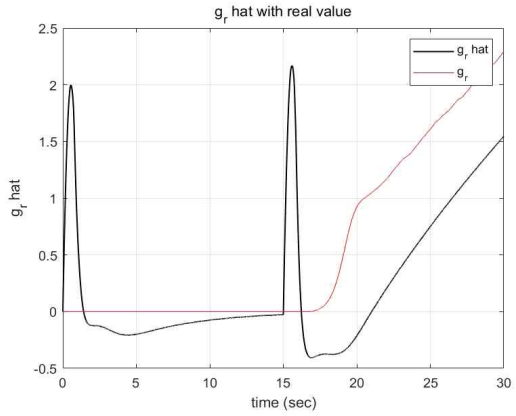
Fig 17. ASMC controller inputs with uncertainty and sensor noise



(a)



(b)



(c)

Fig 18. ASMC controller estimates with uncertainty but without sensor noise
black line : estimates, red line : real value

· Result

Noise가 없는 상황에서 FL controller는 desired value를 잘 추종하는 것을 확인할 수 있었다. SMC 또한 만족스러운 성능을 보였지만 입력 신호의 관점에서 본다면, chattering problem으로 인해 ASMC보다는 FL controller의 입력이 시스템의 안정성 면에서 더 장점을 가진다.

하지만 ground effect로 인한 uncertainty와 sensor noise에 대해서 FL controller는 stability를 보장해주지 않는다. FL controller design에는 고차 미분항이 포함되어 noise에 취약하고 FL method는 system의 정확한 모델 정보를 필요로 하기 때문이다. 반면에 ASMC는 uncertainty와 sensor noise에 대해서 강건한 거동을 보여줄 수 있음을 이론적으로 증명하였고 이를 시뮬레이션 결과로도 확인하였다.

또한 ASMC의 경우, update law에 의해서 추정한 estimate들이 대체로 실제값을 잘 추종하는 것을 확인할 수 있었다. Ground effect로 야기되는 uncertain parameter에 대한 estimate를 통해 쿼드로터의 정밀한 착륙을 유도할 수 있다.

5. 논문을 읽으면서 확인한 사소한 오류들

1. 논문 기준 p.422 식(22)

(수정 전)

$$z^{(3)} = \dot{u_1}\cos\theta\cos\phi - u_1\dot{\theta}\sin\theta\cos\phi - u_1\dot{\phi}\cos\theta\cos\phi$$

(수정 후)

$$z^{(3)} = \dot{u_1}\cos\theta\cos\phi - u_1\dot{\theta}\sin\theta\cos\phi - u_1\dot{\phi}\cos\theta\sin\phi$$

2. 논문 기준 p.422 식(23)

$u_2 = \ddot{\phi}/l$, $u_3 = \ddot{\theta}/l$인데 l항을 빼고 $u_2 = \ddot{\phi}$, $u_3 = \ddot{\theta}$로만 대입함.

3x3 행렬 부분의 2열과 3열이 뒤바뀜.

(수정 전)

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \\ z^{(4)} \end{bmatrix} = \begin{bmatrix} 2\dot{u_1}\dot{\theta}\cos\theta - u_1\dot{\theta}^2\sin\theta \\ -2\dot{u_1}\dot{\phi}\cos\phi + u_1\dot{\phi}^2\sin\phi \\ -2\dot{u_1}\dot{\theta}\sin\theta\cos\phi - 2\dot{u_1}\dot{\phi}\cos\theta\sin\phi + 2u_1\dot{\theta}\dot{\phi}\sin\theta\sin\phi - u_1(\dot{\theta}^2 + \dot{\phi}^2)\cos\theta\cos\phi \end{bmatrix} + \begin{bmatrix} \sin\theta & u_1\cos\theta & 0 \\ -\sin\phi & 0 & -u_1\cos\phi \\ \cos\theta\cos\phi & -u_1\sin\theta\cos\phi & -u_1\cos\theta\sin\phi \end{bmatrix} \begin{bmatrix} \ddot{u_1} \\ u_2 \\ u_3 \end{bmatrix}$$

(수정 후)

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \\ z^{(4)} \end{bmatrix} = \begin{bmatrix} 2\dot{u_1}\dot{\theta}\cos\theta - u_1\dot{\theta}^2\sin\theta \\ -2\dot{u_1}\dot{\phi}\cos\phi + u_1\dot{\phi}^2\sin\phi \\ -2\dot{u_1}\dot{\theta}\sin\theta\cos\phi - 2\dot{u_1}\dot{\phi}\cos\theta\sin\phi + 2u_1\dot{\theta}\dot{\phi}\sin\theta\sin\phi - u_1(\dot{\theta}^2 + \dot{\phi}^2)\cos\theta\cos\phi \end{bmatrix} + \begin{bmatrix} \sin\theta & 0 & u_1\cos\theta l \\ -\sin\phi & -u_1\cos\phi l & 0 \\ \cos\theta\cos\phi & -u_1\cos\theta\sin\phi l & -u_1\sin\theta\cos\phi l \end{bmatrix} \begin{bmatrix} \ddot{u_1} \\ u_2 \\ u_3 \end{bmatrix}$$

3. 논문 기준 p.423 식(31)

S^T 대신 S를 써야 행렬 연산의 dimension이 맞음.

(수정 전)

$$U = G^{-1}(\boldsymbol{x})[-f(x) + \hat{\nu} - \hat{f}_r + \ddot{\boldsymbol{x}}_d - K\dot{\boldsymbol{e}} - diag[c_1, \ldots, c_6]sgn(S^T)]$$

(수정 후)

$$U = G^{-1}(\boldsymbol{x})[-f(x) + \hat{\nu} - \hat{f}_r + \ddot{\boldsymbol{x}}_d - K\dot{\boldsymbol{e}} - diag[c_1, \ldots, c_6]sgn(S)]$$

4. 논문 기준 p.424 식(36),(37)

앞서 error variable을 (current value)-(desired value)로 정의하였으므로 PD controller에서 각 항에 (-)를 붙여주어야 함.

(수정 전)

$$\phi_d = \dot{e_y} + k_\phi e_y$$

$$\theta_d = \dot{e_x} + k_\theta e_x$$

(수정 후)

$$\phi_d = -\dot{e_y} - k_\phi e_y$$

$$\theta_d = -\dot{e_x} - k_\theta e_x$$

5. 논문 기준 p.424 식(44)

$\dot{\boldsymbol{e}} = \boldsymbol{x} - \boldsymbol{x_d}$이고 ϕ_d와 θ_d에는 각각 ϵ_y, ϵ_x항이 포함되어 있으므로 $\eta_{\dot{e}} \neq \dot{E}$. 마찬가지로 그 아래의 η_U도 수정하여야 함.

(수정 전)

$$\eta_{\dot{e}} = \dot{E}$$

(수정 후)

$$\eta_{\dot{e}} = \begin{bmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_z \\ \ddot{\epsilon}_\phi - \ddot{\epsilon}_y - k_\phi \dot{\epsilon}_y \\ \ddot{\epsilon}_\theta - \ddot{\epsilon}_x - k_\theta \dot{\epsilon}_x \\ \dot{\epsilon}_\psi \end{bmatrix}$$

6. 논문 기준 p.425 식(45)

관성모멘트의 단위는 Nms^2/rad.

(수정 전)

Ns^2/rad

(수정 후)

Nms^2/rad

7. 논문 기준 p.425 Fig 4.

ϕ와 θ의 그래프가 서로 바뀜.

8. 논문 기준 p.425 Fig 5.

u_2와 u_3의 그래프가 서로 바뀜.

9. 논문 기준 p.426 Fig 7.

ψ의 그래프가 θ의 그래프로 대체됨.

6. 보완해야 할 점

논문을 공부하고 시뮬레이션을 하면서 가장 고민했었던 부분은 ASMC의 translational position controller를 구현하는 부분이다. 앞서 '사소한 오류들'의 4번에서 설명한 것과 같이 부호가 반대로 되어있기도 했고 논문에 나온 PD gain을 사용하면 x, y 좌표가 제대로 제어가 되지 않았다.

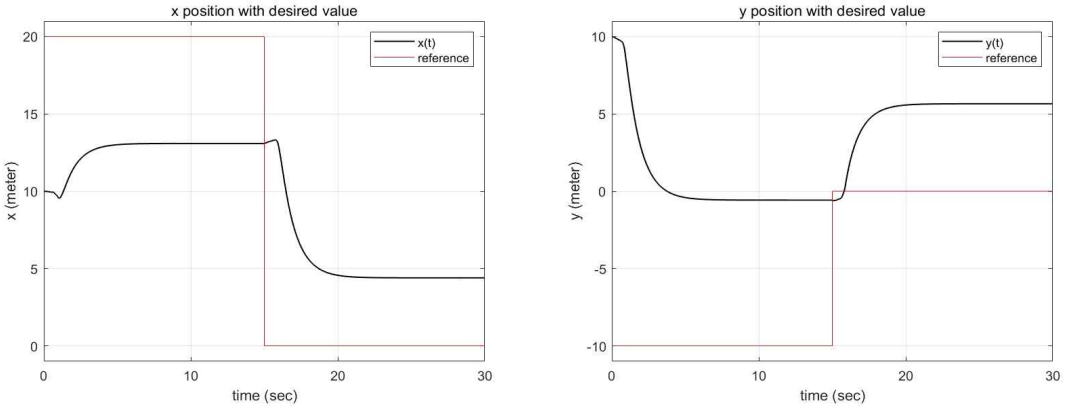


Fig 19. Using PD controller to control x, y position

위의 Figure 19.는 논문에 나온 gain 값을 사용하였을 때의 결과이다. 결과값이 reference와 큰 차이가 나는 것을 확인할 수 있다. 따라서 PD controller 대신 아래와 같은 PID controller를 사용하고 gain tunning을 시도하였다.

$$\phi_d = -k_P e_y - k_I \int_0^t e_y dt - k_D \dot{e_y}$$

$$\theta_d = -k_P e_x - k_I \int_0^t e_x dt - k_D \dot{e_x}$$

그 결과 x, y 좌표는 원하는 값으로 제어할 수 있었지만 roll과 pitch 값이 논문에 나온 그래프와는 다른 양상을 보였다. 그로 인해서 입력 신호와 estimate 값들이 논문과는 차이가 있었다. 이러한 문제점을 해결하고자 이 논문의 제1저자이신 이대원 박사님께 메일을 보내 조언을 구하였다. 바쁘신 와중에 정말 감사하게도 메일로 좋은 말씀과 조언을 해주셨다. 그렇지만 문제 해결에 큰 도움이 되진 않았다. Translational position controller의 제어 방식을 PID가 아닌 다른 방식으로 시도해 볼 계획이다. 이 부분은 앞으로 고민이 좀 더 필요해 보인다.

Reference

[1] Khalil, Hassan K. "Nonlinear control." (No Title) (2015).

[2] Slotine, Jean-Jacques E., and Weiping Li. Applied nonlinear control. Vol. 199. No. 1. Englewood Cliffs, NJ: Prentice hall, 1991.