論文 J. of The Korean Society for Aeronautical and Space Sciences 45(5), 437-446(2017)
DOI:https://doi.org/10.5139/JKSAS.2017.45.5.437
ISSN 1225-1348(print), 2287-6871(online)

비인과, 객체지향적 언어 모델리카를 이용한 멀티콥터형 드론의 통합 비행 시뮬레이션 프로그램

진재현*

Integrated Flight Simulation Program for Multicopter Drones by Using Acausal and Object-Oriented Language Modelica

Jaehyun Jin*
Department of Aerospace Engineering/Center for Aerospace Engineering Research, Sunchon National University

ABSTRACT

An integrated flight simulation program for multicopter drones is presented. The program includes rigid body dynamics, propeller thrust, battery energy, control, and air. Using this program, users can monitor and analyze the states of drones along flight trajectories. As a programming language, Modelica has been chosen, that specializes in simulation program development. Modelica enables users to develop simulation programs efficiently due to acausal and object oriented properties. For missions including horizontal and vertical maneuvers, many dynamical states of drones have been analyzed with simulation results.

초 록

멀티콥터형 소형 드론의 통합 비행 시뮬레이션 프로그램을 개발한 내용을 소개한다. 드론의 강체 동역학, 프로펠러 추력, 배터리 에너지 변화, 자세 및 경로 제어, 대기 상태 등을 통합적으로 시뮬레이션 하여 비행경로에 따라 드론의 상태를 분석할 수 있다. 프로그램 개발을 위하여 모델리카 언어를 선정하였는데, 모델리카는 비인과적, 객체지향적 특성을 갖추고 있어서, 프로그램 개발의 효율을 높일 수 있다. 수평·수직 이동이 포함된 가상 임무에 대하여, 시뮬레이션 결과를 이용하여 드론의 동적 거동을 분석하였다.

Key Words : Acausal simulation(비인과적 시뮬레이션), Modelica(모델리카), Multicopter (다중콥터), Integrated Flight Simulation(통합 비행 시뮬레이션)

Ⅰ. 서 론

멀티콥터(multicopter) 방식의 소형 드론은 안정성과 조종성이 우수하여, 단·근거리 임무에 가장 적합한 형상으로 평가받고 있다. 최근 한국항공우주연구원에서 공모한 '공공혁신조달 연계 소형무인기 기술개발 지원사업(2016년 공고)'을 살펴보면[1] 6개중 4개는 다중로터 방식을 요구하

† Received : February 5, 2017 Revised : April 15, 2017 Accepted : April 26, 2017
* Corresponding author, E-mail : donworry@scnu.ac.kr

고 있는데, 실종자 수색, 시설물 관리, 기상 관측 등에 적합하다고 판단하는 것이다. 하이브리드 방식도 다중로터에 전진 추력 혹은 틸팅 방식을 접목한 것이 많이 사용되고 있다.

멀티콥터 드론을 개발할 때, 성능해석 및 제어기 설계를 위하여 시뮬레이션 프로그램이 필요한데[2], [3]에서는 모터를 2차함수로 모델링하고, 블레이드 요소 이론으로 프로펠러의 추력 모델을 구했다. [4]에서는 다양한 형상에 대한 동역학 방정식을 제시하였고, [5]에서는 모터와 프로펠러 동역학 모델을 제시하였다.

그러나 멀티콥터 드론은 공기역학, 동역학, 전기공학 등의 다물리(multi physics) 현상이 관련되어 있어서, 이러한 현상을 통합적으로 시뮬레이션 할 수 있는 도구가 필요하다. 예를 들면, 드론이 고도를 변경하면서 기동을 하는 경우에, 고도상승에 따라 밀도가 낮아지기 때문에 필요한 추력을 위해서 프로펠러의 속도가 증가해야 한다. 밀도가 낮아지면 프로펠러의 회전 저항이 줄어들지만, 각속도 증가에 따라 동력이 더 소모되면서 배터리의 에너지는 더 빨리 줄어든다. 또한 주변 대기온도는 배터리의 성능에 영향을 주어, 배터리 사용시간이 더욱 줄어든다.

저자는 멀티콥터 드론의 성능해석을 위한 통합 시뮬레이션 프로그램의 필요성을 파악하였고, 적합한 프로그램을 개발하였다. 특히, 모델리카(Modelica)라는 프로그래밍 언어를 선택하였는데, 모델리카는 시간에 따라 변화하는 물리 현상을 시뮬레이션하기 위한 목적으로 개발한 언어이다[6]. 유럽 산업체는 시스템을 개발할 때부터 모델리카 기반의 시뮬레이션 프로그램을 활용하고 있는데, 항공기[7,9], 헬리콥터[8], 고고도 태양발전 비행기[10], 무인기[11] 설계 사례가 있다. 또한, 모델리카의 비인과적(acausal), 객체지향적 특성은 개발 및 유지보수의 효율성을 높여준다. 그리고 다물체 동역학 기법과 같이 구성 부품을 모델링함으로써 다양한 전공의 개발담당자가 쉽게 이해할 수 있기 때문에, 시스템의 설계 단계에서 개발담당자간의 의사소통과 교차 점검을 위한 통합 시뮬레이션 소프트웨어 개발에 적합하다. 이러한 이유로 모델리카를 개발도구로 선정하였다.

II. 본 론

2.1 모델리카 프로그래밍

Figure 1은 모델링 대상(좌측)과 모델리카 프로그램(우측)을 비교한 것이다. 관성, 스프링 등

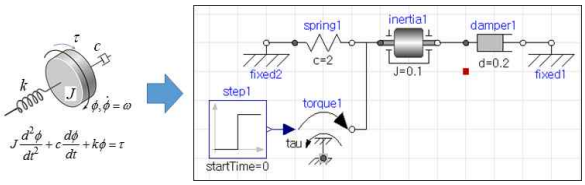

Fig. 1. Modelica program example

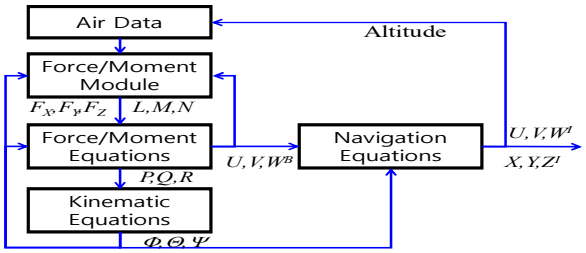

Fig. 2. Architecture of conventional simulation programs

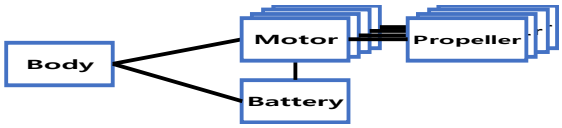

Fig. 3. Multi body programming

의 요소를 연결하여 프로그램을 완성하는데, 직관적으로 파악할 수 있다.

그래픽 프로그램은 텍스트 코드로도 저장이 되는데, Fig. 1의 텍스트 코드는 부록 A에 제시하였다. 요소를 연결하는 것은 두 지점의 회전각도가 동일하다는 구속조건이 추가되는 것이다. 텍스트 코드에서는 다음처럼 나타난다.

connect(a, b) (1)

기존의 비행 시뮬레이션 코드는 Fig. 2와 비슷한 모양으로[12], 입출력을 갖는 함수의 블록이며, 이 함수들은 대상의 형상과 일치하지 않는다.

모델리카를 이용하면, Fig. 3처럼 대상을 실제와 비슷하게 모델링한다(기능적으로 우수하다는 의미는 아니다). 이러한 모델링은 다물체 동역학 기법을 이용하는 것인데[13], 모델리카는 이에 적합한 방법을 제공한다.

이에 더불어 모델리카에서 추구하는 비인과적 프로그래밍 방식은, 추상화가 더해지면서 개발자의 편의를 극대화하고 효율을 높이는 새로운 패러다임을 제시하고 있다[14].

2.2 모듈 분석

멀티콥터 드론의 주요 모듈을 Table 1에 제시하였으며, 외부 환경요소로 air와 gravity를 추가

Table 1. Modules for multi copter drones

Module	Role
body	rigid body dynamics
bodyShell	aerodynamic force/moment
motor	action and reaction torque
propeller	thrust and moment
battery	electric power reservoir
sensor	to sense motions
FCS	flight control system
air	density, temperature, wind
gravity	gravity force

하였다.

2.3 커넥터 설계

커넥터는 모델리카의 독특한 기능으로, 두 지점의 변수에 대한 구속조건인데, Kirchhoff 법칙처럼 동작한다. Fig. 4의 전기회로에서, 전압은 동일하고($V_a = V_b = V_c$), 전류는 $0 = i_a + i_b + i_c$이므로, b점에 대한 전류는 $i_b = -i_a - i_c$이 된다. 전압을 potential 변수, 전류를 flow 변수라고 한다. 이처럼, 힘과 모멘트를 flow 변수로 정의하면 여러 곳에 작용하는 힘과 모멘트를 합할 수 있다.

우측은 그래픽 코딩에서 연결하는 방법을 보여준다. 두 커넥터를 이으면, 텍스트 뷰에는 'connect(A.c, B.c)'라는 코드가 생성된다.

커넥터를 정의하는 예시는 Fig. 5와 같다. 변수 앞에 flow를 붙이면 flow 변수가 된다.

Table 2에는 프로그램에서 사용한 커넥터이다.

2.4 커넥터 ExtFrame의 특징

커넥터 중에서 ExtFrame은 모듈간의 강체 결합 조건을 의미하는데, 가장 중요하다. Fig. 6에

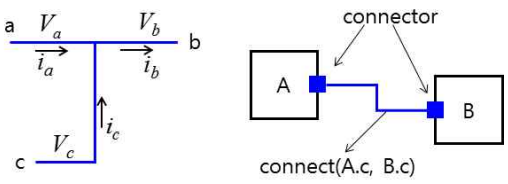

Fig. 4. Modelica connector concept

```
connector PowFlow
  Real Energy;
  flow Real Power;
end PowFlow;
```

Fig. 5. Connector definition code

Table 2. Variables of connectors

Name	potential variable	flow variable
ExtFrame*	position, attitude, angular velocity, velocity, CG position	force, torque, mass, inertia moment, mass×position
PowFlow	energy	power
AirFlow	density, speed of sound, air speed	-
Gravity	position	gravity (g)
Flange**	angle (1d)	torque (1d)

*extended from existing connector, **existing connector

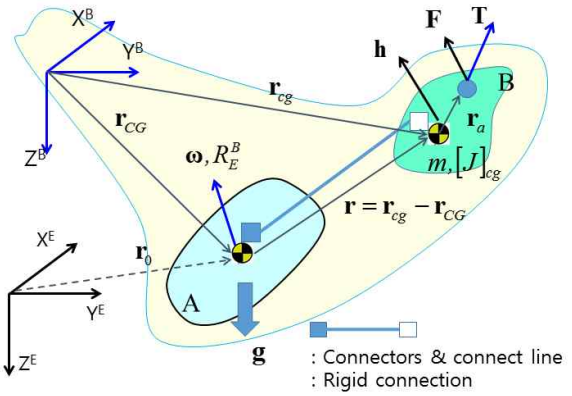

Fig. 6. Rigid body dynamics and multi body

서 $X^EY^EZ^E$는 관성좌표계, $X^BY^BZ^B$는 동체좌표계이고, 두 요소 A, B는 ExtFrame으로 강체 결합된(구속된) 한 몸이다. Table 2의 potential 변수를 보면 알 수 있는데, 위치와 자세가 동일한 것은 두 물체가 강체결합된 것을 의미한다.

힘($\boldsymbol{F}$), 토크($\boldsymbol{T}$), 질량(m), 관성모멘트($[J]_{cg}$), 질량×위치($m\boldsymbol{r}_{cg}$)는 flow 변수이기 때문에 커넥터 ExtFrame를 통해서 B에 작용하는 것은 A로 전달된다. 여러 모듈이 장착될 때, 이런 방식으로 전체 질량, 힘 등을 계산할 수 있다. 즉, 각 로터의 질량과 추력 정보를 주면 모두 합산하여 전체 질량과 전체 추력이 계산된다. 모멘트는 힘에 의한 것과 B의 각운동량($\boldsymbol{h}$)의 자이로스코픽 효과를 추가하는데, Fig. 6에서 전달되는 힘, 모멘트, 관성모멘트 행렬은 다음과 같다.

$$\boldsymbol{F} \tag{2}$$

$$\boldsymbol{T} + (\boldsymbol{r} + \boldsymbol{r}_a) \times \boldsymbol{F} - \boldsymbol{\omega} \times \boldsymbol{h} \tag{3}$$

$$[J]_{cg} + m\left(\boldsymbol{r}^T\boldsymbol{r} I_{3\times 3} - \boldsymbol{r}\boldsymbol{r}^T\right) \tag{4}$$

여러 모듈이 있을 때($i = 1,2,\cdots$), Fig. 4에서처럼, 이들은 커넥터의 기능에 의하여 합해진다. 전

체 질량(m_T)과 질량중심($\boldsymbol{r}_{CG}$)은 다음과 같은데,

$$m_T = \sum m_i, \quad \boldsymbol{r}_{CG} = \left(\sum m_i \boldsymbol{r}_{cg,i}\right)/m_T \tag{5}$$

질량과 '질량×위치'는 flow 변수이기 때문에, 각 모듈의 값을 입력하면 m_T와 $\boldsymbol{r}_{CG}$가 자동으로 계산된다[7]. 모듈을 변경하더라도, 질량과 장착위치(동체좌표 기준)만 입력하면 m_T와 $\boldsymbol{r}_{CG}$가 자동으로 재계산된다. 이러한 특징은 커넥터와 flow 변수라는 독특한 기능 때문에 가능하다.

III. 통합 시뮬레이션 프로그램

3.1 프로그램 구조 설계

프로그램 작성은, 라이브러리에 모듈을 준비해 놓고, 필요한 것을 가져와서 연결하는 방식이다. 필요한 하드웨어를 가져와서 연결한다는 개념으로 모듈을 설계하여, 재사용성을 높게 해야 한다.

3.2 주요 모듈 개발

3.2.1 몸체(강체)

드론은 전체적으로는 강체 운동을 하는 것으로 가정한다. 강체 동역학식은 다음과 같다[12].

$$m\dot{\boldsymbol{v}}^E = \sum \boldsymbol{F}_i^E + m\boldsymbol{g}^E \tag{6}$$

$$[J]_T\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}\times([J]_T\boldsymbol{\omega}) + \sum \boldsymbol{M}_i \tag{7}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}[Q(\boldsymbol{q})]\boldsymbol{\omega} \tag{8}$$

$$\dot{\boldsymbol{r}}_0^E = \boldsymbol{v}^E = [R]_B^E\boldsymbol{v} \tag{9}$$

변수는 속도($\boldsymbol{v}$), 각속도($\boldsymbol{\omega}$), 쿼터니언($\boldsymbol{q}$), 힘($\boldsymbol{F}$), 모멘트($\boldsymbol{M}$), 중력가속도($\boldsymbol{g}$)이고, $[Q(\boldsymbol{q})]$는 각속도와 쿼터니언 변화율과의 변환 행렬이다. 벡터 성분은 동체좌표계(B)에 대한 것이지만, 관성좌표계에 대한 것은 위첨자 E를 표시하였다. 행렬 $[R]_B^E$는 관성좌표계에서 관찰한 벡터를 동체좌표계에서 관찰한 성분으로 변환하는 행렬이다.

이 모듈은, ExtFrame 커넥터를 통하여, 다른 모듈로부터 힘과 모멘트를 전달받아, 식 (6)~(9)의 변수를 계속 갱신한다.

몸체 모듈과 다른 모듈의 관계는 Fig. 6의 관계와 같다. 몸체 모듈은 그림에서 A에 해당하는데, 힘과 모멘트는 커넥터를 통하여 받아들이고 각속도, 자세 정보는 동일한 값으로 공유한다.

3.2.2 프로펠러(로터)

프로펠러의 추력(T_p), 항력 토크(Q_p)는 다음처럼 구해진다[15,16].

$$T_p = C_T\rho A(\Omega R)^2, \quad Q_p = C_Q\rho A(\Omega R)^2 R \tag{10}$$

$$C_T = \frac{1}{2}\frac{N_b c}{\pi R}C_{l_\alpha}\left[\frac{\theta_0}{3} + \frac{\theta_{tw}}{4} - \frac{\lambda}{2}\right] \tag{11}$$

계수 C_T, C_Q는, 이론적(블레이드 요소 이론 등) 혹은 실험적으로 구하는데[15], 시뮬레이션의 정확도를 높일 수 있는 중요한 요소이다. 프로펠러의 항력 토크는 모터로 전달된다.

그리고 프로펠러의 각운동량($J_P\Omega$)과 드론의 회전 운동($\boldsymbol{\omega}$) 때문에, 자이로스코픽 토크가 작용한다. 그래서 프로펠러에 의한 힘과 모멘트는 다음과 같다($\hat{\boldsymbol{u}}_p$는 단위 방향벡터).

$$\boldsymbol{F} = T_p\hat{\boldsymbol{u}}_p \tag{12}$$

$$\boldsymbol{M} = (\boldsymbol{r}_a + \boldsymbol{r})\times\boldsymbol{F} - \boldsymbol{\omega}\times(\pm J_p\Omega\hat{\boldsymbol{u}}_p) \tag{13}$$

±는 프로펠러의 회전 방향에 의해서 결정되는데, $\hat{\boldsymbol{u}}_p$를 기준으로 반시계 방향이 +이다.

고정 피치 프로펠러는 회전속도(Ω)를 조절하여 추력을 제어하며, 가변 피치는 회전속도와 피치각(θ_0)을 조절하여 추력을 제어한다.

3.2.3 모터

모터는 고정자(stator)와 회전자(rotor, J_r)로 나누어져 있으며, 토크를 만들어 회전자에 연결된 프로펠러를 회전시킨다. 필요 토크(T_{req})는 항력 토크, 축 마찰 토크, 관성 토크이다(Fig. 7 참고).

$$T_{req} = Q_p + c_r\Omega + (J_p + J_r)\dot{\Omega} \tag{14}$$

몸체에 전달되는 모멘트는 반작용 토크와 회전자의 자이로스코픽 효과이다.

$$\boldsymbol{M} = -(\pm T_{req}\hat{\boldsymbol{u}}_m) - \boldsymbol{\omega}\times(\pm J_r\Omega\hat{\boldsymbol{u}}_m) \tag{15}$$

마찬가지로 ±는 회전방향에 의하여 결정된다. 모터에서 소모하는 전력은 다음과 같은데, 결국 배터리의 에너지를 소모한다.

$$\dot{E}_{battery} = -P = -T_{req}\Omega \tag{16}$$

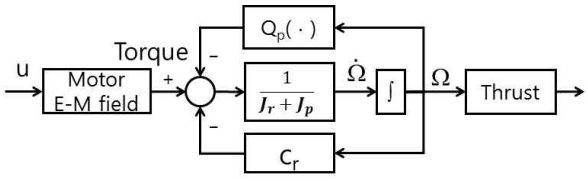

Fig. 7. Required torque for motor rotation

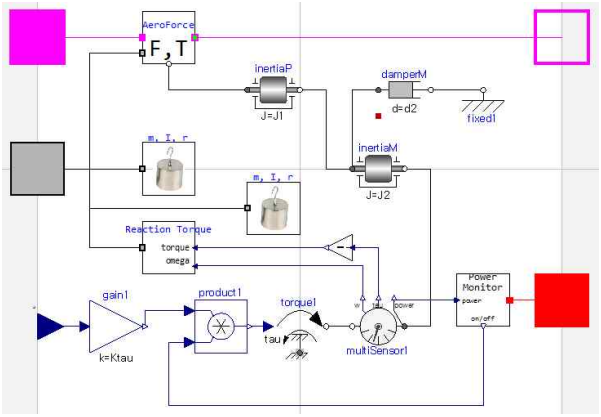

Fig. 8. Propeller and motor module

설명은 따로 했지만, 모터와 프로펠러는 하나의 모듈로 구현하였다(Fig. 8 참고).

3.2.4 배터리

배터리는 커패시터(capacitor)를 기반으로 하는 [17,18] 에너지 저장장치인데, 저장된 에너지는 이론적으로 다음과 같이 구해진다.

$$E = Ah \times V = Wh \tag{17}$$

Ah는 배터리 용량(암페어×시간)인데 주로 mAh 단위로 많이 표시하며, V는 전압이다. 그러나 배터리 전압이 임계값 이하이면 동작이 어렵기 때문에 사용 가능한 에너지는 더 낮다. 에너지 변화율은 소모되는 전력(전압×전류)과 같다.

$$\dot{E} = -P = -VI \tag{18}$$

일반적으로 사용량(방전)에 따라 Fig. 9와 같이 전압이 떨어진다.

참고자료에 의하면[19], 사용가능한 에너지는 방전속도와 온도에 따라 달라진다. Fig. 9는 사례를 보인 것인데, 온도에 더 민감하다.

배터리 유효에너지(E_e)가 줄어든 것인데, 전력 사용에 따른 에너지 감소는 다음과 같다.

$$\dot{E}_e = -P \tag{19}$$

유효 에너지는 저장 에너지와 $E_e = k_{eff}E$

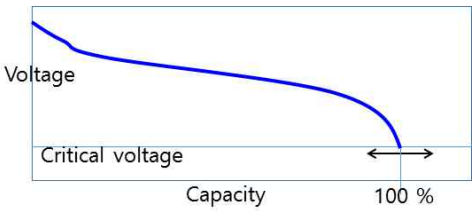

Fig. 9. Relationship between capacity and voltage of Li-Po batteries

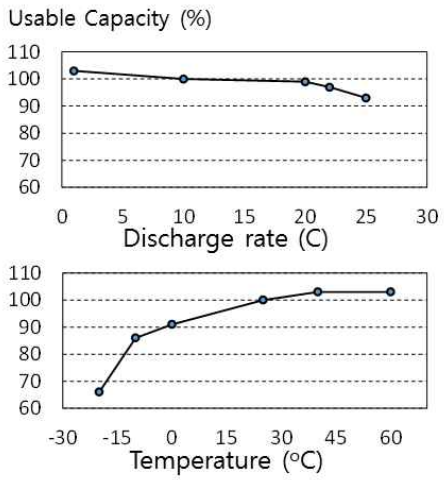

Fig. 10. Effect of discharge rate and temperature

$(0 < k_{eff}(P,T) < 1)$처럼 스케일되는 관계라고 가정하면, 위 식은 다음처럼 된다.

$$k_{eff}\dot{E} = -P \quad \rightarrow \quad \dot{E} = -\frac{1}{k_{eff}}P \tag{20}$$

결국, 온도나 소모율에 따라 실제 소모율은 $1/k_{eff}$ 만큼 배가되는 것으로 생각할 수 있다. 수학적인 처리이지만, 시뮬레이션에서는 유효 에너지 개념보다는 더 쉽게 프로그램 할 수 있다.

배터리를 소모하는 것은 모터 외에 기본 전기부품(FCC, 센서 등)과 카메라 장치 등이 있다.

3.2.5 외형(공기력)

드론이 비행할 때, 발생하는 공기력은 대부분 항력적인 요소이고, 공기력에 의한 힘과 모멘트는 다음처럼 동압에 비례한다.

$$F_a \propto \frac{1}{2}\rho V^2 S_{ref}, \; M_a \propto \frac{1}{2}\rho V^2 S_{ref} l_{ref} \tag{21}$$

비례상수에 해당하는 공기력계수는 풍동시험을 통하여 실험적으로 구해야 하지만, 다양한 조건(자세, 속도 등)에 대한 공기력 데이터를 구축하는 것이 쉽지 않다. 그리고 소형드론을 개발하면서 풍동시험까지 하는 것은 비용 부담이 크다.

본 논문에서는 각 축별로 분리하고 다음과 같이 가정하였다. S_{ref}와 l_{ref}는 축별로 다르게 정의할 수도 있지만[16], 편의상 동일하게 하였다. v_z에서 프로펠러에 의한 후류 속도는 제외하였다.

$$\boldsymbol{F}_a = -\frac{1}{2}\rho S_{ref}\begin{bmatrix} sign(v_x)\,C_{Fx}v_x^2 \\ sign(v_y)\,C_{Fy}v_y^2 \\ sign(v_z)\,C_{Fz}v_z^2 \end{bmatrix} \tag{22}$$

Fig. 11. Drone examples

$$\boldsymbol{M}_a = \frac{1}{2}\rho S_{ref} l_{ref} \begin{bmatrix} C_{Mxy} v_y^2 - C_{Mxz} v_z^2 \\ C_{Myz} v_z^2 - C_{Myx} v_x^2 \\ C_{Mzx} v_x^2 - C_{Mzy} v_y^2 \end{bmatrix} \quad (23)$$

형상이 면 대칭이면 일부 모멘트는 영이 되는데, Fig. 11의 좌측은 모든 면에 대칭이며, 우측은 XZ면 대칭이다($C_{Mzx} = C_{Mxz} = 0$).

3.2.6 대기 모델

공기밀도는 정압이론에 의하여 다음과 같이 계산한다[20].

$$\rho = \frac{p_0}{RT}\left(\frac{T}{T_0}\right)^{g/R\lambda}, \quad T = T_0 - \lambda h \quad (24)$$

여기서 R=288 J/(kg-°K), λ=0.0065 °K/m이고, p_0(Pa)와 T_0(°K)는 드론이 시작하는 지점($h = 0$)에서의 압력 및 온도 값이다.

바람은, 2차원 층류 모델과 고도에 따른 속도 변화만 고려하였다(Fig. 12 참고).

$$v_W = V_{\max}\left(1 - \frac{1}{s \times h + 1}\right) \quad (25)$$

3.2.7 제어기 설계

시뮬레이션 프로그램을 테스트하기 위한 목적으로, 간단한 PID 제어기를 설정하였다. 상태에 대한 정보는 센서를 통하여 모두 측정할 수 있다고 가정하였으며, 측정 노이즈는 제외하였다.

내부 루프(자세 루프)에는 leveling(고도 및 자세 유지)을 기본적으로 수행하며, 외부 루프(유도 루프)는 위치와 헤딩을 제어한다.

자세와 위치 명령(혹은 추력/토크 명령)에 따라 각 모터의 추력을 계산하는 것을 제어력 할당(control allocation)이라고 한다(Fig. 14 참고).

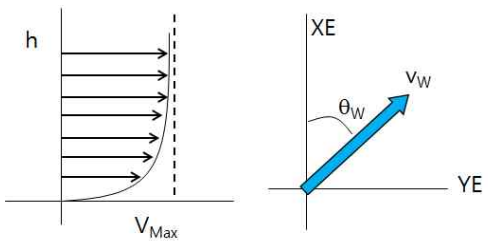

Fig. 12. Wind model

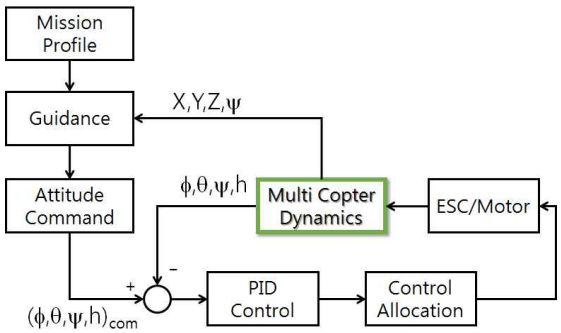

Fig. 13. Control and guidance loop

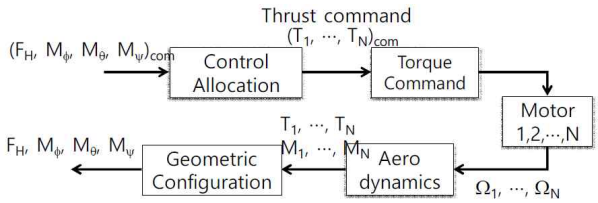

Fig. 14. Motor control and allocation

Figure 19의 +자형 쿼드콥터의 경우에 다음의 관계식이 성립한다.

$$\Delta \boldsymbol{u} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \\ \Delta u_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta F_h \\ \Delta M_\phi \\ \Delta M_\theta \\ \Delta M_\psi \end{bmatrix} \quad (26)$$

만약 ΔM_ϕ가 필요한 경우에는 Δu_i의 조종력 비율을 [0, 1, 0, -1]로 하면 된다는 것이다. 실제 조종력은 적절한 게인이 곱해진다. 그리고 개별 루프에서 구해진 $\Delta \boldsymbol{u}$를 모두 더해서 모터 토크 명령으로 내보낸다. 그리고 과도한 명령을 방지하기 위하여 포화 함수를 적용한다.

$$\Delta \boldsymbol{u} = \Delta \boldsymbol{u}_h + \Delta \boldsymbol{u}_\phi + \Delta \boldsymbol{u}_\theta + \Delta \boldsymbol{u}_\psi \quad (27)$$

3.2.8 경로점 유도

유도 루프는 목표점($(x,y,z)_T$)에 대하여 위치오차가 수렴하도록 한다(Fig. 15 참고).

유도 명령은 수평면 이동(X,Y), 수직 이동(Z), 방향 전환(ψ)으로 분리하며, 수평면에서의 위치오차는 동체좌표계로 나타낸다(Fig. 16 참고)[21].

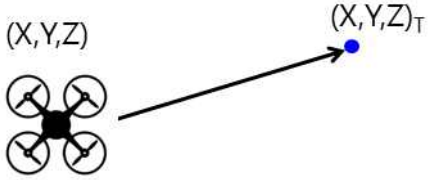

Fig. 15. Waypoint guidance scenario

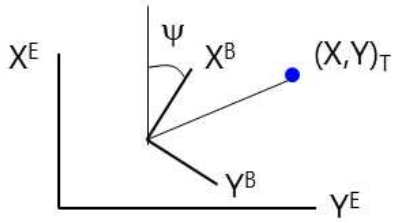

Fig. 16. Body frame and inertial frame

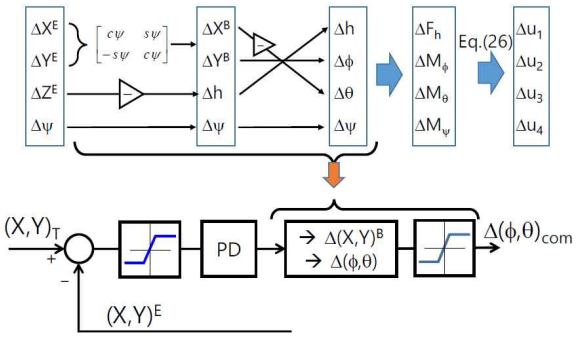

Fig. 17. Guidance loop concept diagram

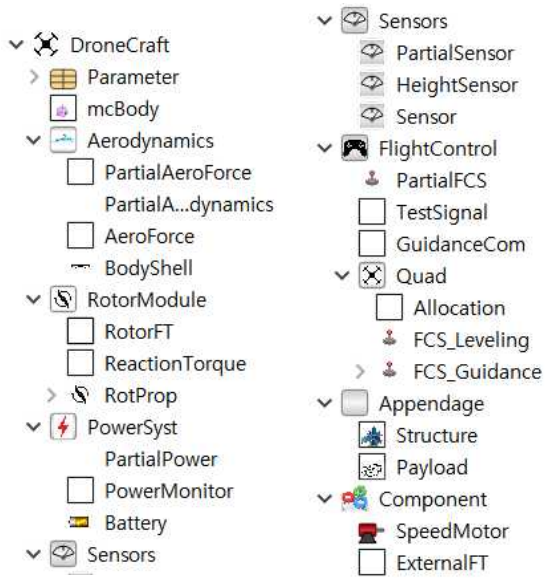

Fig. 18. Libraries for multi copter drones

$$\begin{bmatrix}\Delta X\\ \Delta Y\end{bmatrix}^B = \begin{bmatrix}\cos\psi & \sin\psi\\ -\sin\psi & \cos\psi\end{bmatrix}\begin{bmatrix}\Delta X\\ \Delta Y\end{bmatrix}^E \quad (28)$$

진행할 방향으로 멀티콥터의 자세를 기울여 위치를 제어하는데, 쿼드콥터의 경우에 다음처럼 자세와 변위가 관련된다.

$$X^B \leftrightarrow -\theta,\ Y^B \leftrightarrow \phi,\ Z^E \leftrightarrow -h,\ \psi \leftrightarrow \psi \quad (29)$$

Figure 17의 위 그림은 유도 오차에서 모터 토크 명령에 이르는 과정을, 아래 그림은 수평 이동에 대한 유도 루프를 나타낸 것이다.

과도한 자세 기울기 명령이 발생하면 멀티콥터의 안정성에 영향을 줄 수 있기 때문에, 상하한(포화 함수)을 설정하였다.

3.3 라이브러리 구성

Figure 18은 시뮬레이션 프로그램을 위하여 개발한 라이브러리이다.

Ⅳ. 시뮬레이션 사례

4.1 예제 형상

시뮬레이션 프로그램 테스트를 위하여 Fig. 19과 같은 + 쿼드콥터 형상을 선택하였으며, 설정한 파라미터를 Table 3에 정리하였다.

Figure 20은 개발한 라이브러리를 이용하여 작성한 시뮬레이션 프로그램(그래픽 코드)이다. 그림에서 rotProp이 Fig. 8에 해당하는 것이다.

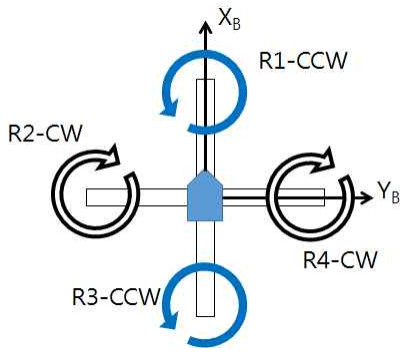

Fig. 19. Quad copter example

Table 3. Example quadcopter parameters

Parameters	Value
m_T	1.34 kg
$[J]_T$	diag[1.11, 1.11, 1.16] (kg-m^2)
S_{ref} , l_{ref}	1 m^2, 1 m
C_F	[0.01, 0.02, 0.04]
C_M	all zeros
arm length	0.5 m
c, R	0.01 m, 0.1 m
N_b	2 (blade number)
blade	$C_{l_\alpha}\theta_0$=2.0, θ_{tw}=0, λ=0, $C_Q = 0.1C_T$
rotor	J_p=J_r=1×10^{-5} kg-m^2, c_r=2×10^{-5} Nm/s
battery	6000 mAh, 15.2 V (initial value)
k_{eff}	100 / [100 - 25+T_0 - 0.0065h]
p_0 , T_0	101.3 kPa, 25℃
wind	v_W=5×[1 - 1/(h+1)], Θ_W=60°
	start at t=50 sec

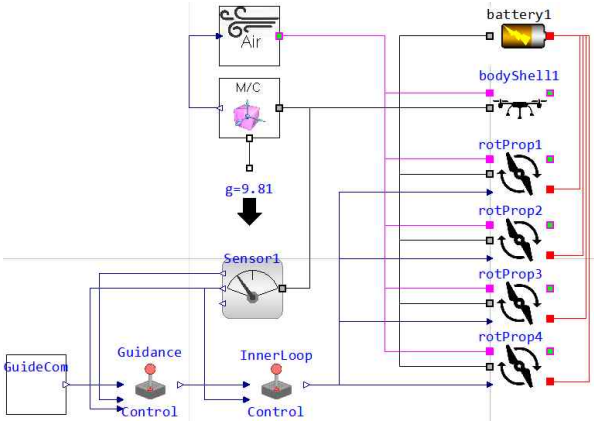

Fig. 20. Simulation program

Table 4. Mission scenario

time (sec)	Target position $(X,Y,h)^E$ (m)
0 ~ 50	0, 0, 100
50 ~ 120	0, 200, 100
120 ~ 190	200, 200, 100
190 ~ 260	200, 200, 500
260 ~ 330	200, 200, 1000
330 ~ 400	200, 200, 1500

4.2 임무 수행 시뮬레이션

설정한 임무는, Table 4에 제시하였는데, 수평 이동 후에 세 단계에 걸쳐 수직 상승하는 것이다. 동작 시간에 근거하여 개루프 방식으로 유도 명령이 출력된다.

Figure 21~24는 시뮬레이션 결과이다. 위치 이동(Fig. 21)을 할 때, 오버슈트가 발생하는데, 특히, 고도에서 오버슈트가 발생하면 전력사용에서 손실이 발생한다. Fig. 22은 배터리 에너지 상태를 보여주는데, 고도 상승할 때, 많은 에너지가 소모된다. 상승 속도를 느리게 하면 에너지 소모율이 줄어들지만, 상승 시간이 늘어나기 때문에, 임무수행에 적합한 전략이 필요할 것이다. Fig. 23는 1번 프로펠러의 회전 속도(Ω_1)를 보여주며, Fig. 24는 고도별 호버링 상태에서 프로펠러의 회전 속도(Ω_1)와 소모 전력을 비교한 것이다. 고도 상승에 따라 전력 소모가 증가하기 때문에, 저고도 비행이 유리하다. 이는 프로펠러 항공기의 성능해석(체공시간) 결과와 비슷하다[19].

100m 상공에서 호버링만 하면, 비행시간은 42분 정도이지만, 임계전압 한계도 있고 다른 부품(FCC 등)이 소모하는 전력도 있기 때문에, 비행시간은 훨씬 줄어든다. 프로펠러의 성능은 드론에 가장 중요한 요소인데, $C_{l_\alpha}\theta_0$값이 클수록 프로펠러의 회전 속도가 줄어들어 유리하지만, 실속(stall)이 발생할 수 있기 때문에 주의해야 한다. 그리고 프로펠러와 모터의 특성(댐핑)은 가정한 것이어서 실제와 차이가 있을 수 있는데, 실물 데이터가 있다면 더 정확한 예측이 가능해진다.

Figure 25는 배터리 에너지가 고갈된 이후에 호버링 상태에서 추력이 급격하게 줄어들면서 추

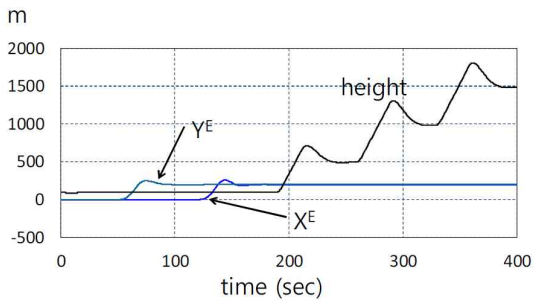

Fig. 21. Simulation results: positions

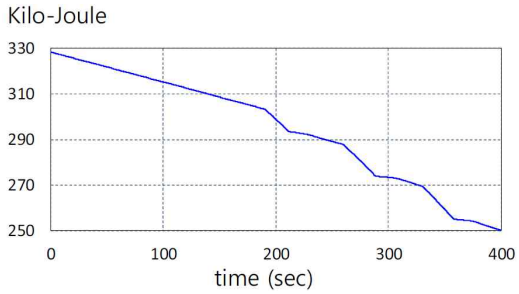

Fig. 22. Simulation results: battery energy

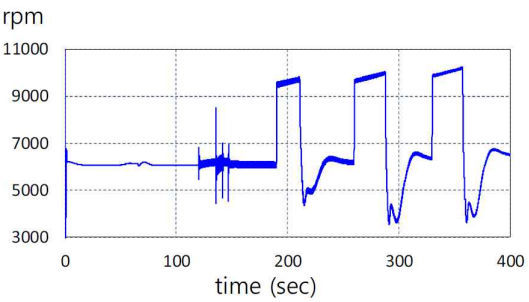

Fig. 23. Simulation results: propeller speed

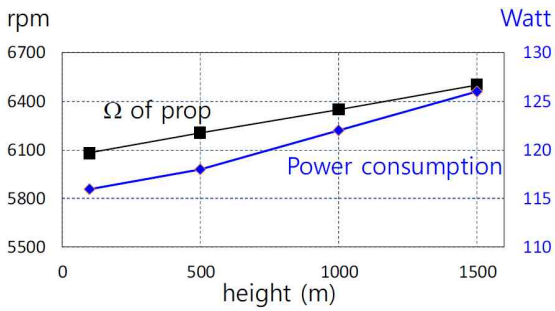

Fig. 24. Simulation results: propeller speed and power consumption

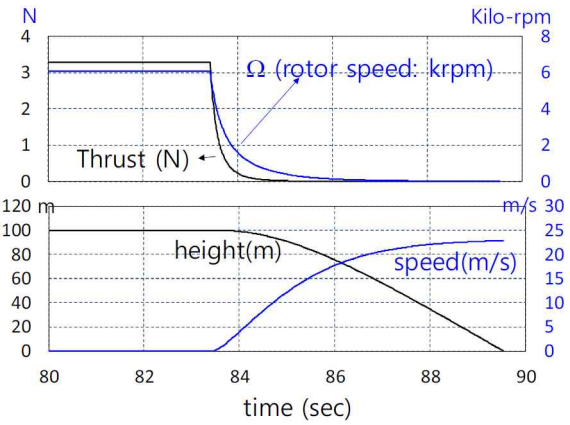

Fig. 25. Free fall results after power loss

락하는 것을 시뮬레이션 한 결과이다. 자세는 그대로 유지된다고 가정했을 때, 지면에 23 m/s의 속력으로 충돌하게 된다. 그러나 뒤집혀 저항이 작은 자세로 떨어지면 훨씬 더 큰 속력으로 충돌할 것이다.

Ⅴ. 결 론

멀티콥터 형상의 소형 드론의 통합 비행 시뮬레이션 프로그램을 개발한 내용을 소개하였다. 소형 드론의 강체 동역학, 공기역학, 프로펠러의 추력, 배터리의 저장 에너지 상태와 소모 등을 통합적으로 시뮬레이션 할 수 있다. 시뮬레이션을 이용한다면, 드론을 설계할 때와 임무계획을 할 때, 다양한 옵션을 정량적인 방법으로 비교를 함으로써, 개발자나 운용자의 평가 및 판단에 필요한 데이터를 제공할 수 있다. 본 논문에서 개발한 통합시뮬레이션 프로그램은, 이러한 측면에서, 매우 유용한 도구라고 사료된다.

그러나 현재 단계에서 여러 가지 부품의 실제 데이터가 부족해서, 본 논문에서는 시뮬레이션의 타당성을 검증할 수 있는 실제 실험결과는 제시하지 못하였다. 그래서 추후 연구계획은, 시뮬레이션 결과의 타당성과 정확도를 높이기 위하여, 실험을 통해서 하드웨어에 대한 데이터(동체 공력계수, 프로펠러 추력/항력계수, 배터리 특성 등)를 확보하고 소형 드론 비행실험을 통하여 검증하는 것이다.

후 기

본 논문은 한국항공우주연구원에서 지원하는 과제 '멀티콥터형 소형무인기 고장·오작동 예측/진단 및 제어기 재구성기술 개발'의 결과입니다.

부 록

모델리카 텍스트 코드 사례(Fig. 1)

```
model test
  Inertia inertia1(J = 0.1);
  Spring spring1(c = 2);
  Damper damper1(d = 0.2);
  Fixed fixed1, fixed2;
  Torque torque1;
  Step step1(startTime = 0);
equation
  connect(step1.y, torque1.tau);
  connect(torque1.flange, inertia1.flange_a);
  connect(damper1.flange_b, fixed1.flange);
  connect(inertia1.flange_b, damper1.flange_a);
  connect(spring1.flange_b, inertia1.flange_a);
  connect(fixed2.flange, spring1.flange_a);
end test;
```

References

1) KARI Homepage(www.kari.re.kr), Notice, 2016. 9. 13.

2) Lee, G., Kim, B., Kim., J., and Kang, T., "Dynamic modeling and design of an attitude controller for quad robot unmanned aerial vehicle," *Proceedings of the 2010 ICROS Conference*, May, 2010, pp.497~499.

3) Lee, S., Wang, J., Lee, S., and Joo, S., "Modelling of the quad-rotor dynamics and controller design," *Proceedings of the 43th KIEE Conference*, July, 2012, pp.1357~1358.

4) Kim, H., Jeong, H., Chong, K., and Lee, D., "Dynamic modeling and control techniques for multi-rotor flying robots," *Transactions of the Korean Society of Mechanical Engineers A (in Korean)*, Vol. 38, No. 2, 2014, pp.137~148.

5) Lee, S. and Kim, Y., "System modeling and waypoint guidance law designing for 6-DOF quadrotor unmanned aerial vehicle," *Journal of The Korean Society for Aeronautical and Space Sciences (in Korean)*, Vol. 42, No. 4, 2014, pp.305~316.

6) Fritzson, P., *Principles of Object Oriented Modeling and Simulation with MODELICA 3.3*, Wiley-IEEE Press, 2015.

7) Iderbrant, A., and Fritzson, P., "Aircraft - A Modelica library for aircraft dynamic simulation," *Proceedings of the 5th EuroSim Congress on Modeling and Simulation*, Paris,

France, Sept. 2004, pp.6~10.

8) Vigano, L., and Magnani, G., "Acausal modelling of helicopter dynamics for automatic flight control application," *Proceedings of the 5th International Modelica Conference*, Vienna, Austria, Sept. 2006, pp.377~384.

9) Looye, G., "The new DLR flight dynamics library," *Proceedings of the 6th International Modelica Conference*, Vol. 1, 2008, pp.193~202.

10) Klöckner, A., Schlabe, D., and Looye, G., "Integrated simulation models for high-altitude solar-powered aircraft," *Proceedings of AIAA Modeling and Simulation Technologies Conference*, Minnesota, US, Aug. 2012, AIAA 2012-4717.

11) Chauvin, F. and Fanmuy, G., "System engineering on 3DEXPERIENCE platform – UAS Use Case," *Proceedings of Workshop of the Complex Systems Design and Management Conference*, Paris, France, Nov., 2014, pp.113~126.

12) Allerton, D., *Principles of Flight Simulation*, Wiley, 2009, p.144.

13) Shin, J. and Choi, K., "Development of a component based helicopter simulation program," *Journal of The Korean Society for Aeronautical and Space Sciences (in Korean)*, Vol. 35, No. 6, 2007, pp.548~555.

14) Kofránek, J., Mateják M., and Privitzer P., "Causal or Acausal Modeling: Labour for Humans or Labour for Machines," *Proceedings of the 16th Annual Conference In Technical Computing*, 2008, pp.1-19.

15) Leishmann, G., *Principles of Helicopter Aerodynamics*, Cambridge University Press, 2006, pp.117-124.

16) Ren, B., Ge, S., Chen, C., Fua, C., Lee, T., *Modeling, Control and Coordinations of Helicopter Systems*, Springer, 2012, pp.10-31.

17) Jang, K. and Chung, G. B., "A SOC estimation using Kalman filter for lithium – polymer battery," *The Transactions of Korean Institute of Power Electronics (in Korean)*, Vol. 17, No. 7, 2012, pp.222~229.

18) Lee, J., Jo, J., Kim, S., and Cha, H., "The state of charge estimation for lithium-polymer battery using a PI observer," *The Transactions of Korean Institute of Power Electronics (in Korean)*, Vol. 20, No. 2, 2015, pp.175~181.

19) http://www.ibt-power.com/Battery_packs/Li_Polymer/Lithium_polymer_tech.html

20) Yoon, Y., *Flight Mechanics (in Korean)*, Kyungmoon Pub., 2011, pp.45, 150~153.

21) Jung, Y., Cho, S., and Shim, H., "Trajectory tracking controller design using L1 adaptive control for multirotor UAVs," *Journal of The Korean Society for Aeronautical and Space Sciences (in Korean)*, Vol. 42, No. 10, 2014, pp.842-850.