공학석사학위논문

저궤도 큐브위성 자세 결정 제어 시스템의 센서 보정 및 단일 축 HIL 검증 실험

Sensor Calibration and Single-axis HIL Verification of ADCS for Low Earth Orbit Cube-satellite

2017년 8월

서울대학교 대학원

기계항공공학부

최 민 규

초 록

서울대학교 GNSS 연구실에서 개발한 2U 크기의 큐브위성인 SNUGLITE는 이중주파수 GPS수신기를 탑재하여 실제 운용을 주요 임무로 하고 있으며, 과학적 임무로는 우주환경 관측 및 데이터 수집을 목표로 하고 있다. 통신은 UHF와 지향성이 있는 S-band 안테나를 통해 지상국과 임무데이터를 송수신한다. 이때, 안정적이고 성공적인 데이터 송수신을 위하여 큐브위성의 지구지향자세 제어가 필수적이다. 이를 위해서는 자세 추정 및 제어 알고리즘인 ADCS(Attitude Determination and Control System)이 On Board Computer(OBC) Processor에서 real-time으로 구현이 되어야 한다. 자세 추정알고리즘으로는 EKF(Extended Kalman Filter)를 사용하며. 자세제어 기법으로는 LQG(Linear Quadratic Gaussian) 알고리즘을 이용하고 있다.

큐브위성 ADCS 알고리즘 개발단계로, 첫째, 알고리즘을 S/W Simulation 측면에서 검증하는 SILS(Software In the Loop Simulation) [이하 SILS]를 통하여 알고리즘에 대한 설계를 한 후, 다음 단계로 PILS(Processor In the Loop Simulation) [이하 PILS]를 통하여 실제 OBC(On Board Computer)에 알고리즘을 탑재하여 검증하는 단계를 거치며, 최종단계로 ADCS알고리즘을 실험적으로 검증하고자, 우주환경을 모사하여 자세 추정 및 제어 알고리즘을 검증하는 단계인 HILS(Hardware In the Loop

Simulation) [이하 HILS]검증과정이 있다. 이러한 일련의 검증과정을 통하여 큐브위성의 ADCS 알고리즘을 설계하게 된다. 이전 연구로서는 ADCS에 대한 알고리즘에 대한 검증내용으로 SILS 연구가 수행된 바 있다. [1], [2]

본 논문에서는 실제 OBC에 구현하기 위한 PILS 에 관한내용과 실제 하드웨어에 적용하여 실험적으로 검증한 내용인 단일 축 HIL 실험 검증에 대한 내용을 다루었다. 특히, 실제 탑재센서를 사용해야 하므로 이러한 센서 오차 모델링과 보정에 관한 연구를 함께 수행하였다.

PILS 수행 내용으로서는 MATLAB 기반 SILS 의 알고리즘을 모두 C언어로 변환하였으며, 이때, OBC의 계산수행 능력에 맞도록, EKF의 Time Update부분에서 Van-Loan 알고리즘을 이용하여, State-transition matrix를 구할 때, 1차로 간략화 하였다. 또한, 큐브위성이 탑재하고 있는 Actuator 특성상 지구 자기장을 이용한 제어를 수행하기 때문에, 제어를 위한 LQR-gain을 실시간으로 구해주어야 하므로, 시스템의 Eigenvalue/Eigenvector 문제 구하는 알고리즘을 탑재하여 Potter’s method를 이용하여 LQR-gain을 구해주고 있다. 또한, 자세 결정센서로 5면에 부착된 Coarse sun sensor(태양센서)와 OBC에 탑재된 Gyroscope (각속도계)와 Magnetometer(지자계)를 이용하여 자세결정을 수행하며, 이 센서들의 성능을 올바르게 사용하기 위해서는 센서 오차 모델링 및 보정이 필수적이다. Coarse sun sensor의 경우, Photodiode를 이용한 저가형 센서로서 태양벡터가 3축으로 들어온다면,

Trigonometric method를 이용하고, 2축으로 들어올 경우, Conical shell model을 이용하여, 각 센서의 상대적인 빛의 크기(Intensity)를 normalized하여, Sun 벡터를 구하게 되며, 이때, 센서 오차 모델링을 수행하여 Sine 함수로 output을 갖도록 보정을 해 주어야 한다. 다음으로, Magnetometer는 지구 절대자기장 대비 실제 측정되는 자기장의 크기를 scale factor를 이용하여 보정하며, 3축의 경우 Hard iron과 Soft iron compensation을 통하여 보정을 수행하였다. Gyroscope의 경우에는 Scale factor를 확인하였다. 또한 노이즈 분석을 위한 Allan Variance 분석을 통하여 bias 모델을 1st order Gauss Markov Process로 적용하였다.

ADCS 알고리즘을 지상에서 검증을 위하여 HILS 검증은 보통 3축 자세 결정, 제어를 위한 Air-bearing 기반의 HILS 시뮬레이터를 이용하여 검증하는 실험과정을 거친다 하지만, 본 논문에서는 단일 축 HIL 실험검증에 대한 내용으로 검증을 위한 기구 설계가 복잡하고 비용측면에서 불리한 3축 HILS 검증대신에 간단하고 비용적인 측면에서 유리한 단일 축 HIL 실험검증을 채택하여 연구를 수행하였으며 실험적으로 자세 추정 및 제어결과를 확인하였으며, 이를 자세 추정 및 제어 요구조건과 확인하였다.

주요어 : Cube Satellite, ADCS, PILS(Processor in the Loop Simulation), HILS(Hardware in the Loop Simulation), EKF, LQG 제어, 자세 결정 및 제어, 센서 보정, 단일 축 HIL 실험
학 번 : 2015-22738

목 차

그림 목차

표 목차

제 1 장 서 론

제 1 절 연구 동기 및 목적

큐브위성의 성공적인 임무수행을 위하여 자세 결정 및 제어 알고리즘 인 ADCS을 개발 및 검증하고자 하며, 기존에 수행되었던 SILS 알고리즘을 바탕을 실제 우주용 OBC(On Board Computer)에 알고리즘을 탑재하여 PILS 검증과정을 수행하며, 이를 바탕으로 단일 축 HIL 검증실험을 통하여 실제 큐브위성의 우주환경에서의 자세 추정 및 제어 성능을 검증하고자 하는데 목표가 있다. 큐브위성의 ADCS 개발단계를 도식화 하면 Figure 1-1과 같다.

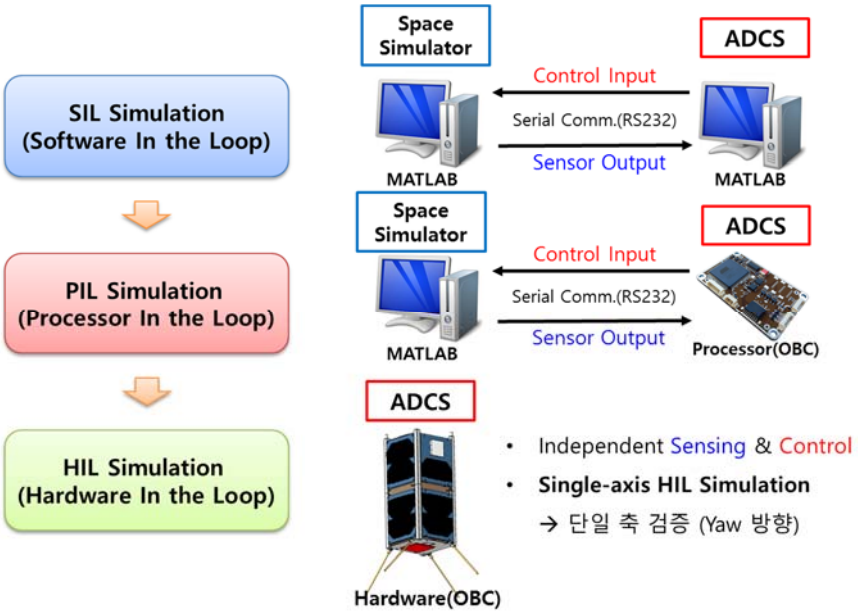

Figure 1-1 Cubesat ADCS algorithm Development Step

먼저, PILS 단계는 SILS에서 개발한 알고리즘을 직접 OBC(On Board Computer)에 탑재하기 위하여, MATLAB 기반의 SILS 알고리즘을 모두 C기반의 알고리즘으로 변경을 해야 Processor 상에서 구동 가능하게 된다.

GNSS 연구실에서 개발하고 있는 저궤도 큐브위성 SNUGLITE의 임무 및 운용시나리오에 대해서 도식화 하여 설명하면 Figure 1-2로 나타낼 수 있다.

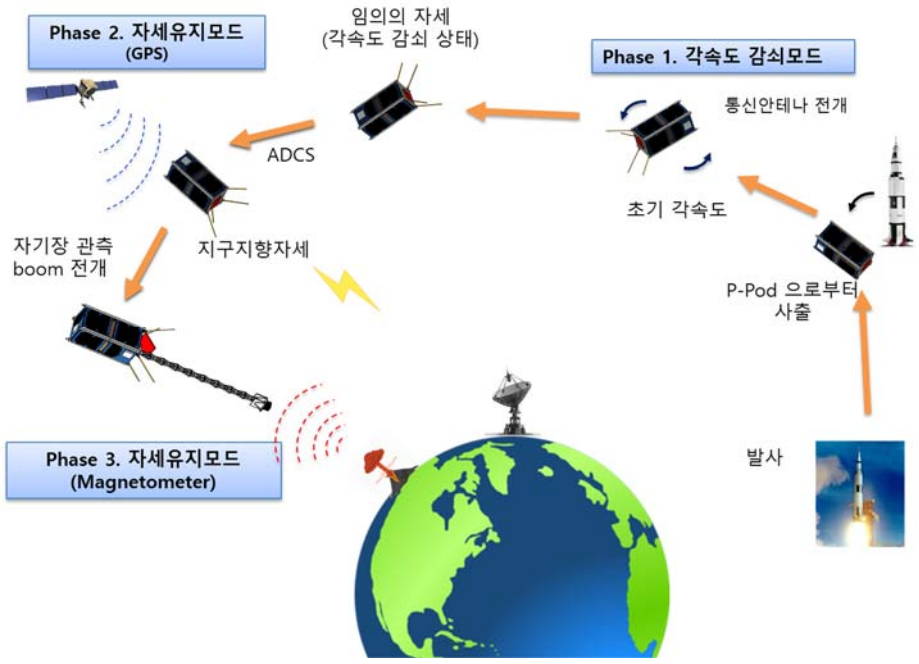

Figure 1-2 Low Earth Orbit SNUGLITE Operation Scenario

큐브위성은 2018년 초(예상) 민간발사체인 Falcon 9에 탑재되어, 발사 예정 중이며, 발사가 되어 일정 궤도에 진입하게 되면, P-Pod으로부터 큐브위성을 사출하게 된다. 이때, 큐브위성은 비교적 큰 초기 각속도를 가지기 때문에, 각속도 감쇄모드를

수행하여 각속도를 감쇄시킨다. 이때 큐브위성은 임의의 자세로 각속도가 감쇄된 상태이며, 지상국과 원활한 통신을 위하여 ADCS 알고리즘을 이용한 지구지향 자세를 유지하게 된다. 이때 주요 미션인 TELACE와 공동개발하고 있는 이중주파수 우주용 GPS 수신기를 이용하여 전리층 모델의 정밀화 연구와 정밀 Magnetometer를 이용하여 지구 자기장 관측 임무를 수행하는데 목표가 있다.

GNSS 연구실에서 이미 수행된 연구 내용으로 SILS에 관한 연구로써, ADCS알고리즘 개발 및 연구에 대한 내용이 주를 이루고 있다. [1],[2] MATALB 기반의 SILS 알고리즘을 바탕으로 PILS에 적용하기 위한 Discretize 및 단순화 과정이 필수적이며 많은 경우 개발단계를 PILS까지만 검증하는 경우가 많지만 센서기반의 Hardware에 적용하여 알고리즘을 검증하기 위해서는 HILS 검증단계가 필수적이다. 이를 위해서는 먼저, 실제 큐브위성에 탑재되게 될 센서들의 스펙을 확인하고 이를 Filter에 반영하는 부분이 선행되어야 하며, 센서의 측정치를 정확하게 얻기 위해서는 센서 오차 모델링 및 보정이 필수적이다.[3] 이후에 이를 검증하는 내용 및 실험으로서, HILS 과정을 수행하게 된다. 본 논문에서는 이러한 PILS 과정부터 HILS 과정까지의 내용을 담고 있으며, 특히 HILS는 실용적이고 비용측면에서 유리한 단일 축 HILS 실험검증을 통하여 알고리즘 검증을 한 내용을 담고 있다.[5]

제 2 절 연구 동향

큐브위성의 탑재체는 임무의 목적에 따라 달라진다. 임무용 탑재체를 제외한 자세 추정 및 결정(ADCS)을 위한 센서로는, 태양센서(Sun sensor), 지자계(Magnetometer), 각속도계(Gyroscope)가 사용된다. 또한, 제어를 위해서는 자기 모멘트를 이용한 자기장 토커(Magnetic Torquer)와 반작용 휠 (Reaction Wheel)또는 CMG(Control Moment Gyro)를 엑추에이터(Actuator) 로 사용한다. 본 논문에서는 태양센서로서는 저가형 비정밀 태양센서(Coarse sun sensor)를 사용하며, OBC에 탑재된 자세결정용 지자계(Magnetometer), 각속도계(Gyroscope)를 사용하며 엑추에이터(Actuator)로서는 자기장 토커(Magnetic Torquer)를 사용하고 있다.

비정밀 태양센서의 경우에는 보통 5~10도 이내의 정확도를 보이며, 고가의 정밀 태양센서(Fine sun sensor)의 경우에는 0.01도 이하의 정확한 정확도를 보인다. (타입에 따라 analog, digital 선센서로 나뉘기도 한다.) 임무 목적에 따라 자세 결정 및 제어 정확도가 달라지게 되며, 이에 따라 센서의 정확도에 대한 선택이 이루어 진다. (GNSS연구실의 SNUGLITE의 경우에는 3축 자세 추정 정확도는 Point angle 기준으로 5도 이내이며, 자세 제어 정확도는 10도 이내이다.)

자세 추정 및 제어 알고리즘(ADCS)으로 구현 방법에 따라 여러가지 알고리즘이 있지만, GNSS연구실의 SNUGLITE는 EKF(Extended Kalman Filter)와 LQG(Linear Quadratic Gaussian)방법으로 자세 추정 및 제어 알고리즘으로 선정을 하여 구현하였다.

제 3 절 연구 내용 및 방법

ADCS의 EKF(Extended Kalman Filter)를 Processor에 적용하기 위해서는 Processor인 OBC의 계산처리 능력을 고려하여, Time update시, Van-Loan 알고리즘을 적용하였으며, State-Transition Matrix를 1차로 단순화 하여 적용하였다. 또한, 제어 알고리즘의 MATLAB의 lqr 함수를 C기반의 함수로 변경하기 위해 Potter's method를 이용한 steady-state Algebraic Riccati equation의 해인 eigenvalue/eigenvector를 이용하여 실시간으로 LQR gain 을 구하게 된다. 이를 구현하기 위한 참고문헌은 [3] 과 같다. 또한, 검증시 중요한 부분이 Processor 계산 속도를 확인하는 과정이다. Processor 특성상, Real-time OS를 지원하는 OBC이기 때문에 EKF와 LQG알고리즘의 처리속도가 다른 임무들을 수행하기 위하여 가능한 적은 시간을 소요 해야한다. EKF는 약 0.2초가 LQR gain을 구하는 알고리즘은 약 0.25초가 수행된다. 따라서 총 약 0.5초가 소요가 되며, 추정만 하는 경우, 제어를 하는

경우보다 Processor의 계산 여유가 더 많다고 볼 수 있다.

다음으로 PILS에서 구현한 알고리즘이 동일하게 잘 구현되었는지 확인하기 위하여 컴퓨터-OBC간의 PILS 검증을 수행하였다. 이를 SILS(컴퓨터-컴퓨터)간의 검증결과와 비교하여, 추정 알고리즘만 검증시 오차가 OBC의 matrix double 형 연산으로 인한 오차인 truncation 수준의 오차 정도임을 확인하였다. 따라서 이로써, Processor에 검증된 알고리즘이 탑재가 되었다고 할 수 있다.

다음 알고리즘 검증과정인 HILS 과정으로 지상에서 우주환경을 모사하는 시뮬레이터나 실험환경을 구성하여 실제 센서로부터 측정치를 사용하여 Hardware 측면에서 ADCS를 검증하고자 하는 단계이다. 이때 보통 3축 자세결정 및 제어 알고리즘을 검증하기 위한 Air-bearing이나 고가의 시뮬레이터를 이용하게 되는데 Figure 1-3과 같은 구성의 장비를 이용하여 우주 환경을 모사하게 된다.

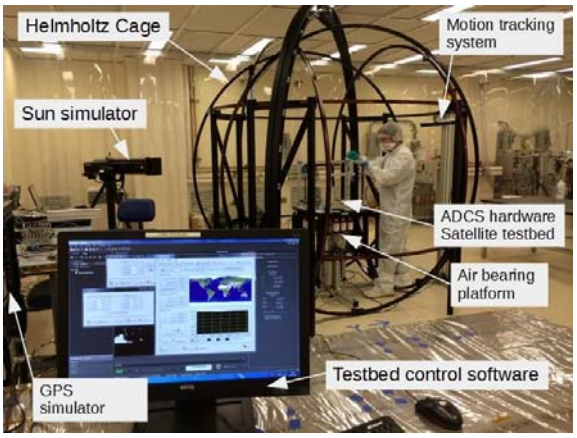

Figure 1-3 Cubesat ADCS Facility (HILS)

이러한 장비를 이용한 경우, Cubesat 뿐만 아니라, 소형의 Satellite의 ADCS 검증을 위한 시뮬레이터로도 사용 가능하다. 이러한 HILS 구성을 위해서는 마찰이 없는 Air-bearing 기반의 실험환경이 필요하다. 하지만 본 논문에서는 실용적이고 비교적 간단하게 낚싯줄을 이용한 단일 축 HIL 실험 검증을 위한 테스트를 구성하였으며 Figure 1-4 와 같은 구성으로 단일 축 HILS 검증 환경을 구축하여 알고리즘 검증을 수행하였다.

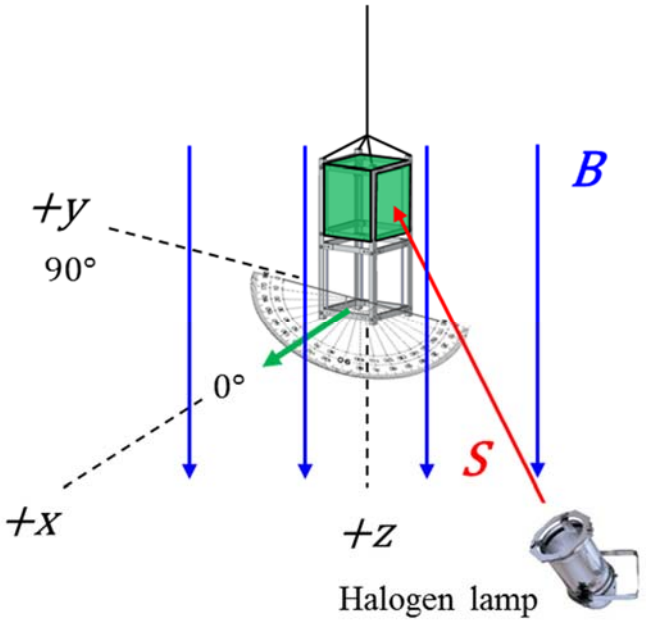

Figure 1-4 Single-axis HIL experiment for ADCS verification

제 4 절 연구의 기여도

PILS와 HILS 측면에서 정리한 연구 기여부분은 다음과 같다. PILS의 통합개발환경(IDE, Integrated Development Environment)를 간단히 소개하면, LINUX 기반의 eclipse 프로그램을 이용하여 개발하였으며, Gomspace의 A3200 OBC를 기반의 real-time OS를 이용하였다. OBC 기반 자세 결정 및 제어 알고리즘(EKF, LQG)은 대부분 double 형의 Matrix연산으로 되어 있으며 C기반의 코드로 작성하였다. 또한, Potter's method 이용한 LQR 함수의 eigenvalue / eigenvector 의 해를 구하는 함수는 Fortran 기반의 함수이며, 참조 [3]의 책의 내용에서 그 알고리즘을 참고하였다. PILS에서의 자세추정결과를 SILS의 결과와 비교하여, 오차가 Truncation error 수준정도가 됨을 확인하였으며, 이후 HIL 실험 검증을 위하여 자세 결정 센서를 보정을 하는 내용으로 coarse sun sensor, Magnetometer에 대한 보정을 수행하였다. Gyroscope의 bias 모델로 1st order Gauss Markov Process로 모델링하였으며, 이는 MATLAB의 실험 측정치를 이용하여, fitting을 통하여 parameter를 도출하였다.

HILS를 위해서 FM(Flight Model)과 동일한 스펙의 센서를 사용한 EM(Engineering Model)을 제작하여 PC-EM 큐브위성간 통신을 수행하여 추정 및 제어 검증실험을 진행하였다. 단일 축 실험 검증의 경우에는, 낚싯줄을 이용한 Yaw축 검증 실험이며, 추정된 state를 바탕으로 LQG 알고리즘을 이용한 제어실험을 통해

3축 추정 알고리즘 과 1축 제어 알고리즘을 검증하였다. 이로써, 실제 software 알고리즘을, 실제 탑재되게 될 OBC에 맞도록 단순화하여 구현하였으며, 실제 단일 축 HIL 검증 실험을 통하여 실제 우주에서 운용되게 될 알고리즘을 지상에서 검증을 하여 제어 가능성을 확인하였다.

제 2 장 **ADCS 알고리즘**

SNUGLITE는 의 주요임무로, GPS데이터 수집 및 지상국과의 성공적인 통신을 위해서는 지구지향 자세를 만족해야 한다. 이를 위해서는, 3축 기준(point angle) 추정 5도, 제어 10도 이내를 목표로 하고 있다.

제 1 절 **좌표계의 정의**

1) 지구중심 관성 좌표계

(ECI, Earth Center Inertial Coordinate System)

Figure 2-1을 착모하면 지구 질량의 중심을 좌표계의 원점으로 하여 천체의 위치를 나타내는 좌표계이다. 지구의 적도면을 XY축으로 하며, X축은 춘분점, Z축은 XY평면에서 북극쪽으로 수직인 선, Y축은 XZ축에 직각인 축으로 한다. 지구 중심 지구 고정 좌표계(ECEF)와 함께 우주 공간상의 한점의 위치를 나타내는 좌표계로 사용된다. 본 논문에서는 태양 모델인 JPL의 DE405가 ECI좌표로 정의 되어 있다. 또한, RIC좌표를 구할 때, ECI 기준성분을 이용하여 좌표변환을 수행한다.

2) 지구중심 지구 고정 좌표계

(ECEF, Earth Centered Earth Fixed Coordinate System)

지구와 같이 회전하는 좌표계로서 지구 상의 위치를 결정하는 데 용이한 좌표계이다. 지구 중심을 좌표계 원점으로 하며, 적도면을 XY축으로 삼아, X축은 그리니치 자오선(경도 0°), Y축은 경도 90°, Z축은 지구의 평균 회전축으로 한다. GPS의 방송 궤도력은 지구 사용자를 위하여 ECEF기준으로 항법해를 도출하므로, ECI Frame으로 변환할 때에는 ecef2eci 함수를 사용하여 사용한다.

[REF] [네이버 지식백과] 지구 중심 관성 좌표계 [Earth Center Inertial Coordinate System, 地球中心慣性座標系] (국방과학기술용어사전, 2011., 국방기술품질원)

3) 지역좌표계 (Local-level frame)

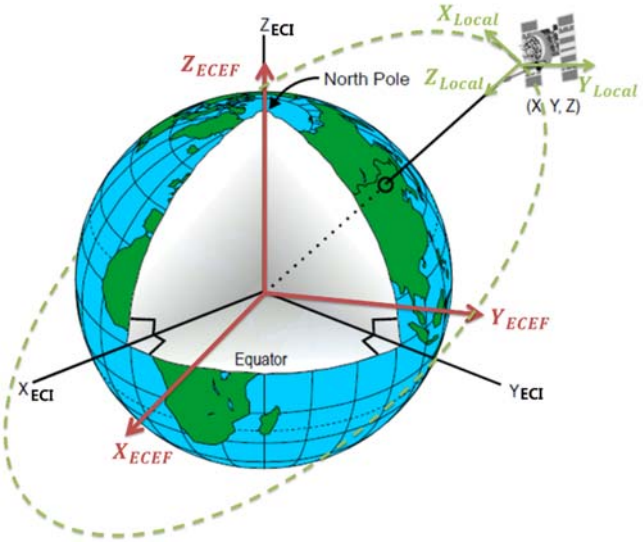

Figure 2-1 Definition of coordinate frame(ECI, ECEF, Local)

사용자가 위치하는 좌표계로, 위성의 궤도를 기준으로 나타낸 좌표계를 지역 좌표계(Local Frame)라고 부르기도 하며, 항법 좌표계 (Navigation Frame)라고 부르기도 한다. 지역좌표계는 위성의 질량중심으로부터의 진행방향(속도)를 x축, 지구 중심방향을 z축, y축은 x와 z의 외적으로 나타낸 좌표로 정의한다. 특히, Figure 2-2와 같이 오일러 각과 쿼터니안은 Local 좌표계로부터 Body 좌표계까지의 각으로 정의된다.

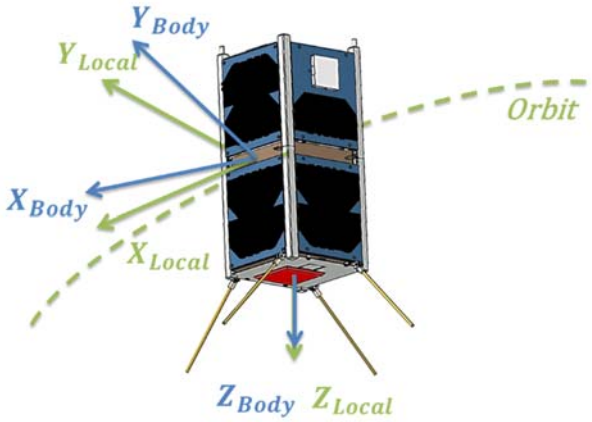

Figure 2-2 Definition of Coordinate frame

4) 몸체 좌표계 (Body frame)

Figure 2-2 를 보면 몸체 좌표계는 위성의 자세 방향을 나타내고, Local 좌표계로부터 정의된 오일러 각 또는 쿼터니안으로 위성의 자세를 나타낸다. 위성의 각 면에 수직한 방향을 축으로 정의한다.

제 2 절 자세추정 알고리즘

SNUGLITE의 자세 추정 알고리즘은, 초기에 태양벡터와 자기장벡터의 기준 좌표와 측정치를 이용하여 초기자세를 구하는 TRIAD Method와 이후 태양센서, 각속도계, 지자기 센서를 이용하여 EKF(Extended Kalman Filter)를 이용하여 자세를 결정하게 된다. 또한, reference 벡터인 model vector를 Local Frame이나 Body Frame으로 변환하여 사용하기 위해서는 GPS의 위치, 속도, 측정치를 이용한 Transformation Matrix를 구하는 알고리즘이 필수적이다.

1. 항법해를 이용한 Local Frame 좌표변환 알고리즘

좌표변환 R_{ECI}^{Local} 을 구하기 위해서는 GPS 위치 속도 항법해 또는 TLE(Tow-Line Elements)를 이용한 위치, 속도 벡터를 이용하여 구하여 준다. 이때, GPS는 ECEF를 기준으로 하고(비교적 정확), TLE는 ECI를 기준으로 하게 된다. (GPS에 비해 정확도가 낮음)

(1) GPS 항법해 이용한 변환 (ECEF 기준)

GPS 항법해는 ECEF frame기준으로 속도, 위치 벡터를 출력하므로, ECI기준으로 사용하는 태양모델을 이용할 때에는 ecef2eci 함수를 이용하고, Local frame으로 변환을 할 때에도

ECI기준으로부터 좌표 변환 해 준다. 이는 수식(2.1)에 표시하였다.

$$\begin{aligned}\mathbf{r}^{ECI} &= R_{ECEF}^{ECI}\mathbf{r}^{ECEF} \\ \mathbf{v}^{ECI} &= R_{ECEF}^{ECI}\mathbf{v}^{ECEF}\end{aligned} \tag{2.1}$$

또한, ECI frame에서의 GPS의 속도와 위치를 이용하여 Local Frame으로 변환하게 되는데, 이때, 원 궤도 일 경우에는 다음수식(2.2)의 알고리즘으로 R_{ECI}^{Local} 를 구할 수 있다.

$$\begin{aligned}\mathbf{e}_1 &= \frac{\mathbf{v}^{ECI}}{\left|\mathbf{v}^{ECI}\right|} \\ \mathbf{e}_3 &= -\frac{\mathbf{r}^{ECI}}{\left|\mathbf{r}^{ECI}\right|} \\ \mathbf{e}_2 &= \mathbf{e}_3 \times \mathbf{e}_1 \\ R_{ECI}^{Local} &= [\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3]^T\end{aligned} \tag{2.2}$$

하지만, 실제 위성의 궤도는 타원 궤도이기 때문에, $e_1 \times e_3 \neq 0$ 이 되므로, 기저들의 직교성을 만족할 수 없다. 따라서 다음의 RIC(Radial, In-Track, Cross-Track coordinate system) frame을 이용하여, 직교성을 만족해주도록 한다. Figure 2-3과 수식(2.3)으로 이를 나타내면 다음과 같다.

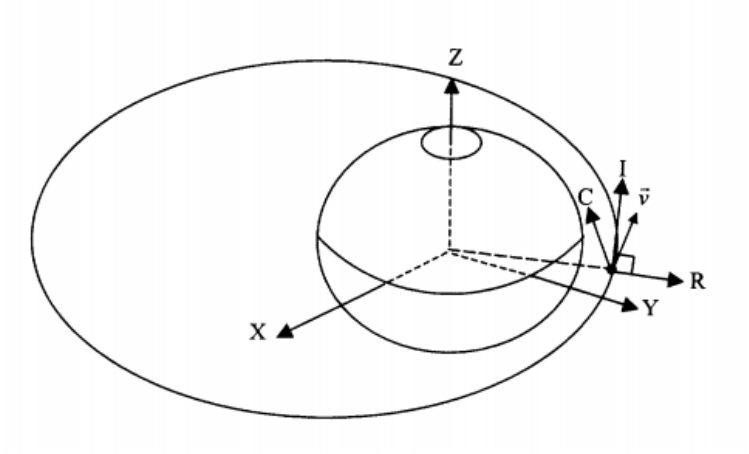

Figure 2-3 Radial, In-Track, Cross-Track Coordinate System

$$\begin{aligned}
\bar{R} &= \frac{\mathbf{r}^{ECI}}{\left|\mathbf{r}^{ECI}\right|} \\
\bar{C} &= \frac{\mathbf{r}^{ECI} \times \mathbf{v}^{ECI}}{\left|\mathbf{r}^{ECI} \times \mathbf{v}^{ECI}\right|} \\
\bar{I} &= \bar{C} \times \bar{R} \\
R_{ECI}^{Local} &= [\bar{I} \quad -\bar{C} \quad -\bar{R}]^T
\end{aligned} \tag{2.3}$$

(2) TLE (Two-line element)를 이용한 변환 (ECI 기준)

우주 케플러 궤도 6요소와 mean motion의 도함수, 이계도함수 그리고 항력계수 값 등의 정보를 포함하고 있는 80열의 ASCII 텍스트로 되어있는 2줄의 궤도정보를 나타낸다.

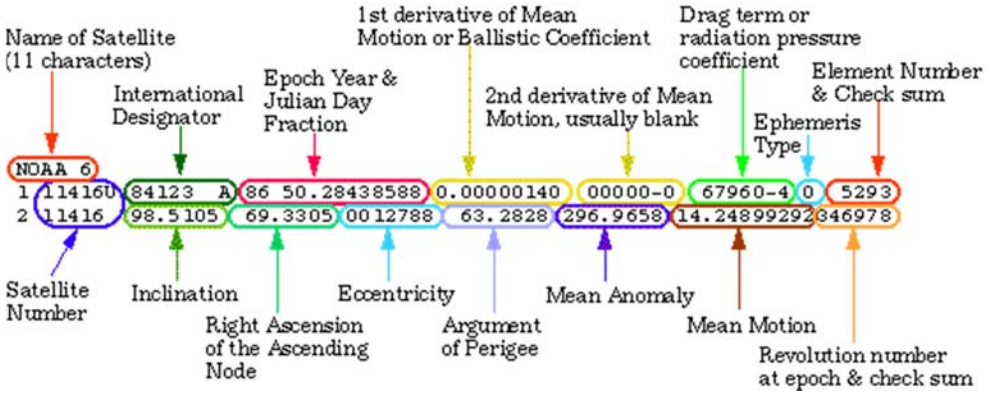

Figure 2-4 TLE Information

TLE를 이용하여 구한 사용자의 위치, 속도는 ECI Frame이므로, 변환과정 없이, 바로 Local 로 변환하여 준다.

$$\begin{aligned} \bar{R} &= \frac{\mathbf{r}^{ECI}}{\left|\mathbf{r}^{ECI}\right|} \\ \bar{C} &= \frac{\mathbf{r}^{ECI} \times \mathbf{v}^{ECI}}{\left|\mathbf{r}^{ECI} \times \mathbf{v}^{ECI}\right|} \\ \bar{I} &= \bar{C} \times \bar{R} \\ R_{ECI}^{Local} &= [\bar{I} \quad -\bar{C} \quad -\bar{R}]^T \end{aligned} \tag{2.4}$$

2. GPS항법해를 이용한 자기장, 태양 모델벡터 생성

(1) 자기장 모델 벡터

자기장벡터는 IGRF12 모델을 이용하고, 이를 구하는 알고리즘은 참조[2] 논문에 상세히 설명되어 있다. 간단히 input, output 관계를 살펴보면 수식(2.5)와 같다.

$$\mathbf{B}^{NED}[\mathrm{nT}] = igrf\,(t_{num}, \phi, \lambda, h)$$

[Input]
t_{num} : serial date numbers of Julian time (from GPS)
ϕ[deg] : Latitude of user (from GPS)
λ[deg] : Longitude of user (from GPS)
h[km] : Height of user (from GPS)

(2.5)

[Output]
$\mathbf{B}^{NED}[\mathrm{nT}]$: 3-axis magnetic field in NED frame (ECEF)

(2) 태양 모델 벡터

태양벡터는 JPL의 DE405모델을 이용하여 태양의 위치를 ECI frame으로 구한 후, GPS를 통해 얻은 위치를 이용하여. 태양벡터를 계산한다. Input, output의 입력을 수식(2.6)으로 표시할 수 있다.

$$\mathbf{r}_{sun}^{ECI}[km] = \mathrm{sun\_jpl}(t_{num})$$

[Input]
t_{num} : serial date numbers of Julian time (from GPS)

(2.6)

[Output]
$\mathbf{r}_{sun}^{ECI}[km]$: 3-axis sun position in ECI frame
$\mathbf{S}^{ECI} = \mathbf{r}_{GPS}^{ECI}[km] - \mathbf{r}_{sun}^{ECI}[km]$: 3-axis sun vector in ECI frame
$\mathbf{s}^{ECI} = \mathbf{S}^{ECI} / norm(\mathbf{S}^{ECI})$

3. TRIAD Method

TRIAD 는 자세 결정 방법 중 계산 알고리즘이 가장 단순하고 하지만 정확도는 낮은 방법으로 추정 초기자세를 구하기 위해 사용한다. 알고리즘은 Local Frame에서 정의된, 두 개의 reference vector와 Body frame 에서 측정된 두개의 측정치 벡터를 이용하여 임의의 기저로부터의 정규직교기를 이용하여 초기 자세를 결정한다.

1) Get orbit axis vectors using GPS output and Sun(JPL DE405)&IGRF model. Get body axis measurement vectors using magnetometer & sun sensor

$$\begin{aligned} \mathbf{t}_1^B &= \mathbf{s}_{meas}^B & \mathbf{t}_1^L &= \mathbf{s}_{\text{model}}^L \\ \mathbf{t}_2^B &= \frac{\mathbf{s}_{meas}^B \times \mathbf{b}_{meas}^B}{|\mathbf{s}_{meas}^B \times \mathbf{b}_{meas}^B|} & \mathbf{t}_2^L &= \frac{\mathbf{s}_{\text{model}}^L \times \mathbf{b}_{\text{model}}^L}{|\mathbf{s}_{\text{model}}^L \times \mathbf{b}_{\text{model}}^L|} \\ \mathbf{t}_3^B &= \frac{\mathbf{t}_1^B \times \mathbf{t}_2^B}{|\mathbf{t}_1^B \times \mathbf{t}_2^B|} & \mathbf{t}_3^L &= \frac{\mathbf{t}_1^L \times \mathbf{t}_2^L}{|\mathbf{t}_1^L \times \mathbf{t}_2^L|} \end{aligned} \tag{2.7}$$

2) Augment vectors and get DCM

$$\begin{aligned} &\mathbf{R}_x^B = \left[\mathbf{t}_1^B \quad \mathbf{t}_2^B \quad \mathbf{t}_3^B\right] \text{ and } \mathbf{R}_x^L = \left[\mathbf{t}_1^L \quad \mathbf{t}_2^L \quad \mathbf{t}_3^L\right] \\ &\rightarrow \mathbf{R}_L^B = \mathbf{R}_x^B \left(\mathbf{R}_x^L\right)^T \end{aligned} \tag{2.8}$$

3) Get attitude in quaternion from DCM

$$\mathbf{R}_L^b \rightarrow \hat{\mathbf{q}}_0 = \left[\hat{q}_0 \quad \hat{q}_1 \quad \hat{q}_2 \quad \hat{q}_3\right]^T \tag{2.9}$$

4. EKF(Extended Kalman Filter)

확장 칼만필터는 비선형 시스템을 nominal point에서 선형화하여 선형 칼만필터를 사용할 수 있는 장점이 있다. 본 논문에서의 칼만필터 알고리즘은 논문[2]에 있는 알고리즘과 동일하며, 이후 Processor에 적용하기 위하여 단순화 한 과정을 이후에 소개하도록 하겠다.

수식(2.10)을 보면, 위성의 자세 정보인 쿼터니언과 각속도 그리고 자이로 바이어스의 10 가지 벡터를 상태변수로 사용한다.

$$\hat{\mathbf{x}}(t_k)=\begin{bmatrix}\hat{\mathbf{q}}(t_k)\\ \hat{\boldsymbol{\omega}}(t_k)\\ \hat{\mathbf{b}}(t_k)\end{bmatrix},\ \hat{\mathbf{x}}(t_k):\text{state vector} \tag{2.10}$$

$$\hat{\mathbf{q}}=\begin{bmatrix}q_0 & q_1 & q_2 & q_3\end{bmatrix}^T,\ \hat{\boldsymbol{\omega}}=\begin{bmatrix}\omega_1 & \omega_2 & \omega_3\end{bmatrix}^T,\ \hat{\mathbf{b}}=\begin{bmatrix}b_1 & b_2 & b_3\end{bmatrix}^T$$

먼저, 쿼터니언의 kinematics 방정식은 다음과 같다.

$$\dot{\mathbf{q}}=\frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q}=\frac{1}{2}\boldsymbol{\Xi}(\mathbf{q})\boldsymbol{\omega} \tag{2.11}$$

$$\boldsymbol{\Omega}(\boldsymbol{\omega})=\begin{bmatrix}0 & -\omega_1 & -\omega_2 & -\omega_3\\ \omega_1 & 0 & \omega_3 & -\omega_2\\ \omega_2 & -\omega_3 & 0 & \omega_1\\ \omega_3 & \omega_2 & -\omega_1 & 0\end{bmatrix}_{4\times4},\quad \boldsymbol{\Xi}(\mathbf{q})=\begin{bmatrix}-q_1 & -q_2 & -q_3\\ q_0 & -q_3 & q_2\\ q_3 & q_0 & -q_1\\ -q_2 & q_1 & q_0\end{bmatrix}_{4\times3}$$

비선형 운동방정식은 수식(2.12)와 같다.

$$\dot{\hat{\mathbf{x}}}(t_k) = f\left(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k)\right) + \mathbf{w}(t_k),\ \mathbf{w}(t) \sim N(0, \mathbf{Q}) \qquad (2.12)$$

$$f\left(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k)\right) + \mathbf{w}(t_k) = \begin{bmatrix} \dot{\hat{\mathbf{q}}}(t_k) \\ \dot{\hat{\boldsymbol{\omega}}}(t_k) \\ \dot{\hat{\mathbf{b}}}(t_k) \end{bmatrix} + \mathbf{w}(t_k)$$

$$= \begin{bmatrix} \frac{1}{2}\boldsymbol{\Omega}\left(\hat{\boldsymbol{\omega}}_{LB}^{B}(t_k)\right)\hat{\mathbf{q}}(t_k) \\ \bar{\bar{\mathbf{I}}}^{-1}\left(\mathbf{u}(t_k) \times \mathbf{B}_{meas}^{B} - \hat{\boldsymbol{\omega}}(t_k) \times (\bar{\bar{\mathbf{I}}}\hat{\boldsymbol{\omega}}(t_k))\right) \\ -\frac{1}{\tau}\hat{\mathbf{b}}(t_k) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\eta}_D(t_k) \\ \boldsymbol{\eta}_{bias}(t_k) \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{0}_{4\times4} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \left(\sigma_{GG}^2 + \sigma_{SRP}^2 + \sigma_{AD}^2\right)\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \left(\sigma_{RRW}\right)^2 \cdot \mathbf{I}_{3\times3} \end{bmatrix}$$

σ_{RRW} : Gyroscope rate random walk noise

σ_{GG} : Worst case gravity gradient torque process noise

σ_{SRP} : Worst case solar radiation pressure torque process noise

σ_{AD} : Worst case aerodynamic torque process noise

Q-matrix는 외란모델을 적용하여 외란의 불확실성을 고려하여 주었다. 이에 대한 수식은 논문[2]를 참고.

(1) Linearization

확장칼만필터를 사용하기 위해서 선형화 과정을 수식(2.13)과 같이 진행한다. 시스템 matrix인 F는 수식 (2.14)와 같이 표현할 수 있다.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)) \rightarrow \dot{\mathbf{x}} = F\mathbf{x} + G\mathbf{u} + \Gamma\mathbf{w} \qquad (2.13)$$

$$F = \frac{\partial \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u})}{\partial \hat{\mathbf{x}}} = \begin{bmatrix} \frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \hat{\mathbf{q}}} & \frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \hat{\boldsymbol{\omega}}} & \frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \hat{\mathbf{b}}} \\ \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \hat{\mathbf{q}}} & \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \hat{\boldsymbol{\omega}}} & \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \hat{\mathbf{b}}} \\ \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \hat{\mathbf{q}}} & \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \hat{\boldsymbol{\omega}}} & \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \hat{\mathbf{b}}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_{LB}^{B})_{4\times4} & \frac{1}{2}\boldsymbol{\Xi}(\hat{\mathbf{q}})_{4\times3} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \begin{matrix} 0 & a\hat{\omega}_3 & a\hat{\omega}_2 \\ b\hat{\omega}_3 & 0 & b\hat{\omega}_1 \\ c\hat{\omega}_2 & c\hat{\omega}_1 & 0 \end{matrix} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & -\frac{1}{\tau}\mathbf{I}_{3\times3} \end{bmatrix} \qquad (2.14)$$

$$\frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_{LB}^{B}) = \frac{1}{2}\boldsymbol{\Omega}\left(\hat{\boldsymbol{\omega}} - \hat{R}_{L}^{B}\boldsymbol{\omega}_{EL}^{L}\right) - \frac{1}{2}\boldsymbol{\Omega}\left(\frac{\partial \hat{R}_{L}^{B}}{\partial \hat{\mathbf{q}}}\boldsymbol{\omega}_{EL}^{L}\right)\hat{\mathbf{q}} \qquad (2.15)$$

수식 (2.15) 에서 $\frac{\partial \mathbf{R}_{Orbit}^{Body}}{\partial \mathbf{q}}$ 가 나타내는 것은 Direct Cosine Matrix (DCM) $\mathbf{R}_{Orbit}^{Body}$ 을 쿼터니언에 대하여 편미분한 항이 된다.

$$\frac{1}{2}\boldsymbol{\Omega}\left(\frac{\partial \hat{R}_{L}^{B}}{\partial \mathbf{q}}\boldsymbol{\omega}_{EL}^{L}\right)\mathbf{q} = \left[\frac{1}{2}\boldsymbol{\Omega}\left(\frac{\partial \hat{R}_{L}^{B}}{\partial q_0}\boldsymbol{\omega}_{EL}^{L}\right)\mathbf{q} \quad \cdots \quad \frac{1}{2}\boldsymbol{\Omega}\left(\frac{\partial \hat{R}_{L}^{B}}{\partial q_3}\boldsymbol{\omega}_{EL}^{L}\right)\mathbf{q}\right] \qquad (2.16)$$

$$\frac{\partial R_L^B}{\partial q_0} = 2\begin{bmatrix} q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} \qquad \frac{\partial R_L^B}{\partial q_1} = 2\begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & q_0 \\ q_3 & -q_0 & -q_1 \end{bmatrix}$$

$$\frac{\partial R_L^B}{\partial q_2} = 2\begin{bmatrix} -q_2 & q_1 & -q_0 \\ q_1 & q_2 & q_3 \\ q_0 & q_3 & -q_2 \end{bmatrix} \qquad \frac{\partial R_L^B}{\partial q_3} = 2\begin{bmatrix} -q_3 & q_0 & q_1 \\ -q_0 & -q_3 & q_2 \\ q_1 & q_2 & q_3 \end{bmatrix} \tag{2.17}$$

$$a = \frac{I_y - I_z}{I_x}, b = \frac{I_z - I_x}{I_y}, c = \frac{I_x - I_y}{I_z}$$

식 (2.3)을 제어입력으로 미분한 input matrix(G) 를 유도하면 수식(2.8)과 같다. 또한, 노이즈에 대하 Gamma matrix는 (2.19)와 같다.

$$G(t) = \frac{\partial f(\hat{\mathbf{x}}, \mathbf{u})}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \mathbf{u}} & \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \mathbf{u}} & \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \mathbf{u}} \end{bmatrix}^T \tag{2.18}$$

$$\frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \mathbf{u}} = \mathbf{0}_{4\times 3}, \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \mathbf{u}} = \begin{bmatrix} 0 & \frac{B_z(t)}{I_x} & -\frac{B_y(t)}{I_x} \\ -\frac{B_z(t)}{I_y} & 0 & \frac{B_x(t)}{I_y} \\ \frac{B_y(t)}{I_z} & -\frac{B_x(t)}{I_z} & 0 \end{bmatrix}, \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \mathbf{u}} = \mathbf{0}_{3\times 3}$$

$$\Gamma = \frac{\partial \mathbf{f}(\hat{\mathbf{x}}, \mathbf{w})}{\partial \boldsymbol{\eta}} = \begin{bmatrix} \frac{\partial \dot{\hat{\mathbf{q}}}}{\partial \boldsymbol{\eta}} & \frac{\partial \dot{\hat{\boldsymbol{\omega}}}}{\partial \boldsymbol{\eta}} & \frac{\partial \dot{\hat{\mathbf{b}}}}{\partial \boldsymbol{\eta}} \end{bmatrix}^T = \begin{bmatrix} \mathbf{0}_{4\times 4} & \mathbf{0}_{4\times 3} & \mathbf{0}_{4\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} \end{bmatrix}_{10\times 9} \tag{2.19}$$

$$\mathbf{y}(t) = \mathbf{z}(t) - h(\mathbf{x}(t)) \rightarrow \mathbf{y} = \mathbf{z} - H\mathbf{x}$$
$$\mathbf{z}(t_k) = \mathbf{h}(\mathbf{x}(t_k)) + \mathbf{v}(t_k) \quad (2.20)$$

수식(2.19)를 보면, $\mathbf{y}(t)$ 는 innovation 벡터로, Body 측정치와 모델 측정치 사이의 오차를 나타낸다.

측정치는 다음과 같이 일기간/식기간 일 때와 추정 또는 추정 및 제어 일 때의 4가지 경우로 나뉜다. 따라서, 그에 따라 알고리즘도 4가지로 아래와 같이 분기 되어있다.

Table 2-1 Measurement Vector for Eclipse & Day

	Day	Eclipse
Estimation	$\mathbf{z}(t_k) = \begin{bmatrix} \mathbf{b}_{meas}(t_k) \\ \mathbf{s}_{meas}(t_k) \\ \boldsymbol{\omega}_{meas}(t_k) \end{bmatrix}$	$\mathbf{z}(t_k) = \begin{bmatrix} \mathbf{b}_{meas}(t_k) \\ \boldsymbol{\omega}_{meas}(t_k) \end{bmatrix}$
Estimation & Control	$\mathbf{z}(t_k) = \begin{bmatrix} \mathbf{s}_{meas}(t_k) \\ \boldsymbol{\omega}_{meas}(t_k) \end{bmatrix}$	$\mathbf{z}(t_k) = \boldsymbol{\omega}_{meas}(t_k)$

(1) GPS 측정치 사용시 (ECEF기반)

$$\mathbf{y}(t_k) = \mathbf{z}(t_k) - \mathbf{h}(\hat{\mathbf{x}}(t_k))$$
$$= \begin{bmatrix} \mathbf{b}^B_{meas}(t_k) \\ \mathbf{s}^B_{meas}(t_k) \\ \boldsymbol{\omega}^B_{meas}(t_k) \end{bmatrix} - \begin{bmatrix} \mathbf{b}^B_{model}(t_k) \\ \mathbf{s}^B_{model}(t_k) \\ \hat{\boldsymbol{\omega}}(t_k) + \hat{\mathbf{b}}(t_k) \end{bmatrix} = \begin{bmatrix} \mathbf{b}^B_{meas}(t_k) \\ \mathbf{s}^B_{meas}(t_k) \\ \boldsymbol{\omega}^B_{meas}(t_k) \end{bmatrix} - \begin{bmatrix} \hat{R}^B_L \tilde{R}^{Local}_{ECI} \tilde{R}^{ECI}_{ECEF} R^{ECEF}_{NED} \mathbf{b}^{NED}_{IGRF} \\ \hat{R}^B_L \tilde{R}^{Local}_{ECI} \mathbf{s}^{ECI}_{JPL} \\ \hat{\boldsymbol{\omega}}(t_k) + \hat{\mathbf{b}}(t_k) \end{bmatrix} \quad (2.21)$$

$$\mathbf{b}_{IGRF}^{NED} = \frac{\mathbf{B}_{IGRF}^{NED}}{|\mathbf{B}_{IGRF}^{NED}|}$$

$\hat{R}_L^B$: by $\hat{x}(t)$, estimated attitude

$\tilde{R}_{ECI}^{Local}$: by GPS meas. $(\overline{\mathbf{v}}_{ECEF}, \overline{\mathbf{r}}_{ECEF} \rightarrow \tilde{R}_{ECEF}^{ECI} \rightarrow \tilde{\mathbf{v}}_{ECI}, \tilde{\mathbf{r}}_{ECI})$

$$\tilde{R}_{ECI}^{Local} = [I \quad -C \quad -R]^T, \quad R = \frac{\tilde{\mathbf{r}}^{ECI}}{\left|\tilde{\mathbf{r}}^{ECI}\right|}, \quad C = \frac{\tilde{\mathbf{r}}^{ECI} \times \tilde{\mathbf{v}}^{ECI}}{\left|\tilde{\mathbf{r}}^{ECI} \times \tilde{\mathbf{v}}^{ECI}\right|}, \quad I = C \times R$$

$\tilde{R}_{ECEF}^{ECI}$: reduced model

모델 벡터를 구할 때, GPS측정치를 사용할 때와 TLE를 사용할때로 다음의 2절 1번과 2번에서 설명했으며, 수식 (2.21)에서는 GPS 측정치 사용시, 수식 (2.22)에서는 TLE측정치 사용시에 따라서 달라지는 것을 알 수 있다.

(2) TLE 측정치 사용시 (ECI기반)

$$\mathbf{y}(t_k) = \mathbf{z}(t_k) - \mathbf{h}(\hat{\mathbf{x}}(t_k))$$

$$= \begin{bmatrix} \mathbf{b}_{meas}^B(t_k) \\ \mathbf{s}_{meas}^B(t_k) \\ \boldsymbol{\omega}_{meas}^B(t_k) \end{bmatrix} - \begin{bmatrix} \mathbf{b}_{model}^B(t_k) \\ \mathbf{s}_{model}^B(t_k) \\ \hat{\boldsymbol{\omega}}(t_k) + \hat{\mathbf{b}}(t_k) \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{meas}^B(t_k) \\ \mathbf{s}_{meas}^B(t_k) \\ \boldsymbol{\omega}_{meas}^B(t_k) \end{bmatrix} - \begin{bmatrix} \hat{R}_L^B \overline{R}_{ECI}^{Local} \tilde{R}_{ECEF}^{ECI} R_{NED}^{ECEF} \mathbf{b}_{IGRF}^{NED} \\ \hat{R}_L^B \overline{R}_{ECI}^{Local} \mathbf{s}_{model}^{ECI} \\ \hat{\boldsymbol{\omega}}(t_k) + \hat{\mathbf{b}}(t_k) \end{bmatrix} \quad (2.22)$$

$$\mathbf{b}_{IGRF}^{NED} = \frac{\mathbf{B}_{IGRF}^{NED}}{|\mathbf{B}_{IGRF}^{NED}|}$$

$\hat{R}_L^B$: by $\hat{x}(t)$, estimated attitude

$\overline{R}_{ECI}^{Local}$: by TLE meas. $(\overline{v}_{ECI}, \overline{r}_{ECI})$

$$\overline{R}_{ECI}^{Local} = [I \quad -C \quad -R]^T, \; R = \frac{\overline{\mathbf{r}}^{ECI}}{\left|\overline{\mathbf{r}}^{ECI}\right|}, \quad C = \frac{\overline{\mathbf{r}}^{ECI} \times \overline{\mathbf{v}}^{ECI}}{\left|\overline{\mathbf{r}}^{ECI} \times \overline{\mathbf{v}}^{ECI}\right|}, \quad I = C \times R$$

$\tilde{R}_{ECEF}^{ECI}$: reduced model

Kalman Filter의 measurement update와 time update를 나타낸 수식은 (2.23)과 (2,24)와 같다.

$$\begin{aligned}\hat{\mathbf{x}}^{+}(t_k) &= \hat{\mathbf{x}}^{-}(t_k) + \mathbf{K}(t_k)\left[\mathbf{z}(t_k) - \mathbf{h}\left(\hat{\mathbf{x}}^{-}(t_k)\right)\right]^{-1} \\ \mathbf{P}^{+}(t_k) &= \left(\mathbf{I} - \mathbf{K}(t_k)\mathbf{H}(t_k)\right)\mathbf{P}^{-}(t_k)\left(\mathbf{I} - \mathbf{K}(t_k)\mathbf{H}(t_k)\right)^{T} + \mathbf{K}(t_k)\mathbf{R}(t_k)\mathbf{K}(t_k)^{T}\end{aligned} \tag{2.23}$$

$$\begin{aligned}\dot{\hat{\mathbf{x}}}^{-}(t_{k+1}) &= \mathbf{f}\left(\hat{\mathbf{x}}^{+}(t_k), \mathbf{u}(t_k)\right) \\ \dot{\mathbf{P}}^{-}(t_{k+1}) &= \mathbf{F}(t_k)\mathbf{P}^{+}(t_k) + \mathbf{P}^{+}(t_k)\mathbf{F}(t_k)^{T} + \mathbf{\Gamma}(t_k)\mathbf{Q}(t_k)\mathbf{\Gamma}(t_k)^{T}\end{aligned} \tag{2.24}$$

(3) Discretization 과정

OBC processor에 적용하기 위해서는 time update부분에서 MATLAB의 ODE45 적분과정을 사용할 수 없다. 따라서, time update시에 Van-Loan 알고리즘을 이용한 Discretize 및 단순화 과정를 반영항 수식은 (2.25)와 같다.

$$\begin{aligned}
&\dot{\hat{\mathbf{x}}}(t) = F\hat{\mathbf{x}}(t) + \Gamma\mathbf{w}(t),\ \ \mathbf{w}(t) \sim W(0,W) \\
&\rightarrow \hat{\mathbf{x}}(t_{k+1}) = \Phi_d \hat{\mathbf{x}}(t_k) + \mathbf{w}_d(t_k),\ \mathbf{w}_d \sim N(0,W_d) \\
&C = \begin{bmatrix} -F & \Gamma W \Gamma^T \\ 0 & F^T \end{bmatrix} \\
&\mathrm{expm}(CT_s) = \begin{bmatrix} F_2 & G_2 \\ 0 & F_3 \end{bmatrix} \\
&\Phi_d = F_3^T \\
&W_d = F_3^T G_2
\end{aligned} \tag{2.25}$$

위의 exponential matrix를 1차로 고려하여 단순화 하면,

$\mathrm{expm}(CT_s) \cong I + CT_s$

$$\begin{aligned}
&C = \begin{bmatrix} -F & Q^T \\ 0 & F^T \end{bmatrix} \\
&\mathrm{expm}(CT_s) = I + CT_s = \begin{bmatrix} (I-F)T_s & QT_s \\ 0 & (I+F^T)T_s \end{bmatrix} \\
&\Phi_d = (I+F)T_s \\
&W_d = \Phi_d Q T_s^2
\end{aligned} \tag{2.26}$$

Van-Loan 알고리즘과 State transition matirx를 1차로 고려한 최종적인 EKF measurement update와 time update의 수식은 (2,27), (2.28)의 수식과 같다.

$$
\begin{aligned}
&\dot{\hat{\mathbf{x}}} = F\hat{\mathbf{x}} + \Gamma\mathbf{w}, \;\; \mathbf{w} \sim W(0,W) \\
&\hat{\mathbf{x}}^-(t_{k+1}) = f\left(\hat{\mathbf{x}}^+(t_{k-1}), \mathbf{u}(t_{k-1})\right) \\
&Q = \Gamma W \Gamma^T \\
&\Phi_d = I + F\Delta t, \;\; W_d = \Phi Q \;\; \text{(Van-Loan, 1st order)} \\
&P^- = \Phi_d P^+ {\Phi_d}^T + W_d \\
&\hat{\mathbf{x}}^-(t_{k+1}) = \dot{\hat{\mathbf{x}}}^-(t_k)\Delta t + \hat{\mathbf{x}}^+(t_k) = f(\hat{\mathbf{x}}^-(t_k))\Delta t + \hat{\mathbf{x}}^+(t_k)
\end{aligned}
\tag{2.27}
$$

$$
\begin{aligned}
&P^+ = (I - KH)P^-(I - KH)^T + KRK^T \\
&\hat{\mathbf{x}}^+(t_k) = \hat{\mathbf{x}}^-(t_k) + K\left[\mathbf{z}(t_k) - \mathbf{h}\left(\hat{\mathbf{x}}^-(t_k)\right)\right]^{-1}
\end{aligned}
\tag{2.28}
$$

제 3 절 자세제어 알고리즘

큐브위성의 자세 제어는 크게 두 단계로 나누어진다. 첫 번째 단계는 각속도 감쇠 모드이며 두 번째 단계는 임무 모드이다. 앞서 운영 시나리오 절에서 잠깐 언급하였지만 큐브위성은 초기에 궤도에 진입하면 약 최대 매초 15도의 각속도로 회전을 하기 때문에 안정적으로 임무를 수행하기 위해서는 이 각속도를 감쇄시켜야 한다. 따라서 각속도 감쇠 모드를 통하여 각속도가 특정 기준 이하로 수렴하면 임무 모드로 전환하여 임무 수행에 필요한 지구지향 자세 제어를 수행한다. 이러한 과정을 설명하는 그림은 아래와 같다.

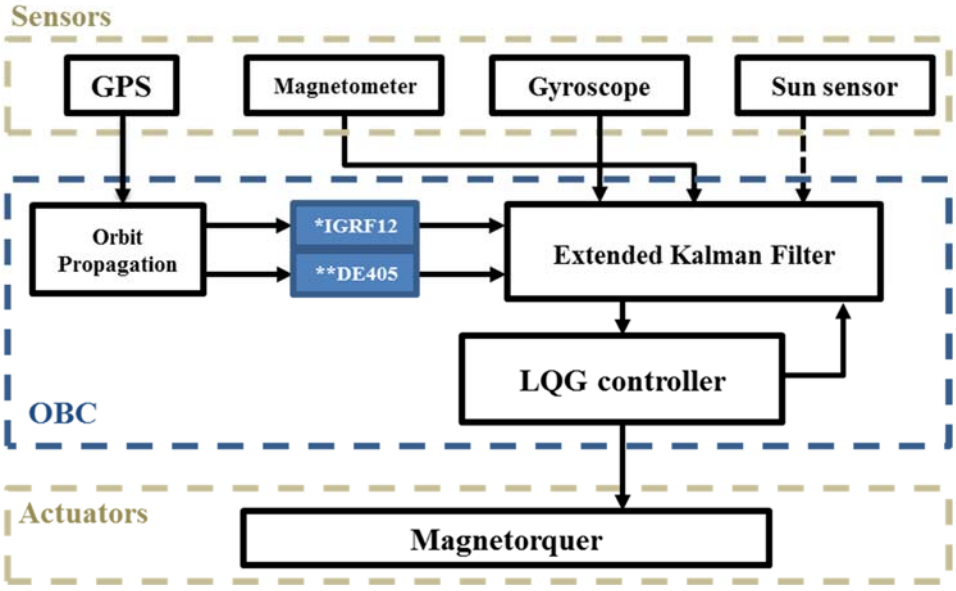

Figure 2-5 Block diagram of Attitude Determination and Control

1. LQG(Linear Quadratic Gaussian)

추정알고리즘을 통하여 추정된 state를 이용하여, 수식(2.2)쿼터니안을 오일러 각으로 변환한 후, 추정된 오일러 각과 각속도를 이용하여 제어 알고리즘을 수행한다.

$$\hat{\mathbf{x}}_c(t_k) = \left[\hat{\phi}(t_k) \quad \hat{p}(t_k) \quad \hat{\theta}(t_k) \quad \delta\hat{q}(t_k) \quad \hat{\psi}(t_k) \quad \hat{r}(t_k) \right]^T \tag{3.1}$$

알고리즘은 이전 reference [2]와 동일한 알고리즘을 사용하였다. 하지만, 이를 C에 직접 구현하기 위해서는 LQR을 구하는 알고리즘이 필요하게 된다. 이는 알고리즘 설명 후에 기술 하도록 하겠다.

LQG 제어기는 수식(의 cost function, J 를 최소화하는 제어입력 u를 생성한다.

$$J = \frac{1}{2}\mathbf{x}(t_k)^T S\mathbf{x}(t_k) + \frac{1}{2}\int_{t_0}^{t_k} \left(\mathbf{x}^T A\mathbf{x} + \mathbf{u}^T B\mathbf{u} \right) dt \tag{3.2}$$

위성의 nonlinear dynamic equation은 수식(2.31)과 같이 오일러 각에 대하여 단순화 할 수 있다.

$$\begin{aligned} I_x\dot{p} + (I_z - I_y)qr &= M_x \\ I_y\dot{q} + (I_x - I_z)rp &= M_y \\ I_z\dot{r} + (I_y - I_x)pq &= M_z \end{aligned} \tag{3.3}$$

위성의 nonlinear kinematic equation 은 다음과 같다. LQG제어기를 설계하기 위해 다음의 nonlinear equation을 선형화 하여 사용한다. 아래 각속도는 오일러 각의 변화와 수식(2.32)와 같은 관계를 갖고 있으며, 위성의 궤도 운동에 대한 항도 함께 고려를 하여 준다. 이때, 오일러 각을 이용한 회전변환은 수식(2.33)과 같이 나타낼 수 있다.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}(\phi,1)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}(\phi,1)\mathbf{R}(\theta,2)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}(\phi,1)\mathbf{R}(\theta,2)\mathbf{R}(\psi,3)\begin{bmatrix} 0 \\ -n \\ 0 \end{bmatrix} \tag{3.4}$$

$$\begin{aligned} \mathbf{R}(\phi,1) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \\ \mathbf{R}(\theta,2) &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \\ \mathbf{R}(\psi,3) &= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.5}$$

최종적으로 오일러 변화율을 각속도, 궤도공전속도 대한 kinematics 수식은 (2.34) 와 같다.

$$\begin{aligned} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sec\theta\sin\phi & \sec\theta\cos\phi \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ &+ \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sec\theta\sin\phi & \sec\theta\cos\phi \end{bmatrix}\begin{bmatrix} \sin\psi\cos\theta \\ \sin\psi\sin\theta\sin\phi+\cos\psi\cos\phi \\ \sin\psi\sin\theta\cos\phi-\cos\psi\sin\phi \end{bmatrix} n \end{aligned} \tag{3.6}$$

우주에서의 태양풍과 공기마찰의 외력이 크기가 중력구배토크 보다 작고 자세에 따라 외란의 방향과 크기가 달라지기 때문에, 시스템 모델로 고려하지 않고 외란으로 간주한다. 하지만 중력구배토크는 수식(2.35)와 같이 시스템과 같이 고려를 해줄 수 있다.

$$3n^2\hat{r}_c \times (\mathbf{I}\hat{r}_c) = 3n^2 \begin{pmatrix} I_z \sin\phi\cos\phi(\cos\theta)^2 - I_y \sin\phi\cos\phi(\cos\theta)^2 \\ -I_x \sin\theta\cos\theta\cos\phi + I_z \sin\theta\cos\theta\cos\phi \\ -I_y \sin\phi\sin\theta\cos\theta + I_x \sin\phi\sin\theta\cos\theta \end{pmatrix} \quad (3.7)$$

위 식에서 오일러 각은 작다고 가정 $(\phi,\ \theta,\ \psi \ll 1)$ 하고 비대각 행렬의 관성모멘트를 0이라고 가정하면 $(I_{xy} = I_{yz} = I_{zx} = 0)$ 수식(2.36)과 같이 단순화 할 수 있다.

$$3n^2\hat{r}_c \times (\mathbf{I}\hat{r}_c) \approx 3n^2 \begin{bmatrix} (I_z - I_y)\phi \\ (I_z - I_x)\theta \\ 0 \end{bmatrix} \quad (3.8)$$

다음으로 Magnetic Torquer에 의하여 생성되는 입력(Dipole moment)과 지구 자기장(Magnetic field)와 외적하여 수식(2.37) 과 같이 구할 수 있다.

$$\mathbf{M}(t_k) = \boldsymbol{\mu}(t_k) \times \mathbf{B}^{Body} = \begin{bmatrix} \mu_y B_z - \mu_z B_y \\ \mu_z B_x - \mu_x B_z \\ \mu_x B_y - \mu_y B_x \end{bmatrix} \quad (3.9)$$

입력인 자기모멘트(Magnetic moment 또는 Dipole moment는 각 축의 PWM의 Duty ratio의 실효치 전압에 의해 생성된 전류 (I[A]) 코일의 면적 $Area[m^2]$를 곱하여 수식(2.38)과 같이 생성이 된다.

$$\boldsymbol{\mu}(t_k) = \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} = \begin{bmatrix} I_x \cdot A \\ I_y \cdot A \\ I_z \cdot A \end{bmatrix} \tag{3.10}$$

앞에서 유도한 중력구배토크와 입력토크 식을 external moment M에 대입하여 정리하면 다음과 같이 angular velocity의 미분항에 대하여 수식 (2.39)와 같이 정리할 수 있다.

$$\begin{aligned} \dot{p} &= -\frac{I_z - I_y}{I_x} qr + \frac{3n^2(I_z - I_y)}{I_x} \sin\phi\cos\phi(\cos\theta)^2 + \frac{1}{I_x}\left(\mu_y B_z - \mu_z B_y\right) \\ \dot{q} &= -\frac{I_x - I_z}{I_y} rp + \frac{3n^2(I_z - I_x)}{I_y} \sin\theta\cos\theta\cos\phi + \frac{1}{I_y}\left(\mu_z B_x - \mu_x B_z\right) \\ \dot{r} &= -\frac{I_y - I_x}{I_z} pq + \frac{3n^2(I_x - I_y)}{I_z} \sin\phi\sin\theta\cos\theta + \frac{1}{I_z}\left(\mu_x B_y - \mu_y B_x\right) \end{aligned} \tag{3.11}$$

오일러 각이 작다면 수식(2.40)과 같이 선형화하여 정리할 수 있다.

$$\begin{aligned} \dot{p} &\approx \frac{3n^2(I_z - I_y)}{I_x}\phi + \frac{n(I_z - I_y)}{I_x} r + \frac{1}{I_x}\left(\mu_y B_z - \mu_z B_y\right) \\ \dot{q} &\approx \frac{3n^2(I_z - I_x)}{I_y}\theta + \frac{1}{I_y}\left(\mu_z B_x - \mu_x B_z\right) \\ \dot{r} &\approx \frac{n(I_y - I_x)}{I_z} p + \frac{1}{I_z}\left(\mu_x B_y - \mu_y B_x\right) \end{aligned} \tag{3.12}$$

이를 control state에 대하여 정리하면 선형화 된 system equation을 사용할 수 있다.

$$\begin{bmatrix} \dot{\phi} \\ \dot{p} \\ \dot{\theta} \\ \delta\dot{q} \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & n & 0 \\ \frac{3n^2(I_z - I_y)}{I_x} & 0 & 0 & 0 & 0 & \frac{n(I_z - I_y)}{I_x} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{3n^2(I_z - I_x)}{I_y} & 0 & 0 & 0 \\ -n & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{n(I_y - I_x)}{I_z} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ p \\ \theta \\ \delta q \\ \psi \\ r \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{B_z}{I_x} & -\frac{B_y}{I_x} \\ 0 & 0 & 0 \\ -\frac{B_z}{I_y} & 0 & \frac{B_x}{I_y} \\ 0 & 0 & 0 \\ \frac{B_y}{I_z} & -\frac{B_x}{I_z} & 0 \end{bmatrix} \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} \triangleq \dot{\hat{\mathbf{x}}}(t_k) = F\hat{\mathbf{x}}(t_k) + G\mathbf{u}(t_k) \qquad (3.13)$$

Estimation state와는 다르게, control state에서 $\delta q = q + n$ 이라는 상태변수를 정의하여, q 를 제어하는 대신에 δq 를 0으로 제어함으로써 $q = -n$ 으로 제어하여 큐브위성이 궤도를 도는 속도로 제어할 수 있도록 궤도 각속도를 반영하여 준다.

2. MATLAB 의 LQR알고리즘 C변환과정

OBC Processor에서 LQR알고리즘을 이용하기 위해서는 c코드 기반의 칼만 Gain을 구하는 알고리즘이 필요하다. LQR을 구하는 알고리즘에는 여러가지 방법이 있지만, 본 논문에서는 Potter's method를 이용한 steady-state Algebraic Riccati Equation(ARE)의 의 해를 이용하여 칼만 Gain을 구하는 알고리즘을 적용하였다. [7]

Cost function은 수식(2.42)와 같을 때,

$$J = \frac{1}{2}\mathbf{x}(t_f)^T S\mathbf{x}(t_f) + \frac{1}{2}\int_0^{t_f}(\mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u})dt \tag{3.14}$$

Continuous Algebraic Riccati equation은 수식(2,43)와 같다.

$$\begin{aligned} &\dot{P} + A^T P + PA + Q - PBR^{-1}B^T P = 0, P(t_f) = S \\ &u = -R^{-1}B^T P(t)x \rightarrow K(t) = R^{-1}B^T P(t) \end{aligned} \tag{3.15}$$

이때, Steady-State Algebraic Riccati equation을 고려하면,

$$\begin{aligned} &\dot{P}_\infty = 0,\ A^T P + PA + Q - PBR^{-1}B^T P = 0 \\ &K = R^{-1}B^T P \end{aligned} \tag{3.16}$$

수식(2.44)와 같이 Kalman Gain을 구할 수 있다.

구체적인 Steady-State Algebraic Riccati equation의 해를 구하는 알고리즘인 Potter's method 수식(2.45)와 같다.

$$
\begin{aligned}
&A^T P + PA + Q - PBR^{-1}B^T P = 0 \\
&A^T P + P(A - BR^{-1}B^T P) + Q = 0 \\
&A^T P + PA_c + Q = 0 \rightarrow A^T P + Q = -PA_c \\
&A^T PF + QF = -PA_c F \\
&\text{Let } PF = E,\ A_c F = F\Lambda \rightarrow (A - BR^{-1}B^T P)F = F\Lambda \\
&\rightarrow AF - BR^{-1}B^T E = F\Lambda \\
&A^T E + QF = -PF\Lambda = -E\Lambda \\
&\rightarrow -A^T E - QF = E\Lambda
\end{aligned}
\tag{3.17}
$$

$$
\begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} F \\ E \end{bmatrix} = \begin{bmatrix} F \\ E \end{bmatrix} \Lambda \rightarrow H\Phi = \Phi\Lambda,\ H \triangleq \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{3.18}
$$

수식(2.45)는 수식(2.46)의 Hamiltonian matrix의 eigenvalue/eigenvector를 구하는 문제로 동일시 할 수 있다. 또한, H-matrix는 Real non-symmetric matrix이므로, symmetric matrix의 eigenvalue/eigenvector보다 알고리즘이 복잡하다는 단점이 있다.

일반적으로 MATLAB에서는 시스템의 고유치, 고유벡터를 구하는 알고리즘이 공개되어 있지는 않지만, Fortran기반으로 시스템의 고유치 고유벡터를 구하는 알고리즘이 소개 되어있다. eigenvalue/eigenvector problem 은 다음의 알고리즘을 이용하여 구하게 된다[3]. 이를 Table 2-2에 소개하였다.

Table 2-2 Eigenvalue/eigenvector calculation algorithm in C

Function	Function Role
balanc	Balances matrix and isolates eigenvalues when possible
balbak	Forms eigenvectors by back transforming those of the corresponding matrices determined by BALANC
elmhes	Reduces matrix to upper Hessenberg form by using elementary similarity transformations
eltran	Accumulates transformations used in the reduction to upper Hessenberg form done by ELMHES
hqr2	Finds eigenvalues and eigenvectors by QR method
normalize	Normalize eigenvectors

다음은, eig 함수의 결과물로 2n 개의 complex eigenvalue와 eigenvector가 나오게 된다 이 중 n개의 음수인 eigenvalue를 선택하여, eigenvector를 선정한다.

$$\Phi = \begin{bmatrix} F \\ E \end{bmatrix}_{2n \times n} \rightarrow P = EF^{-1} \rightarrow K = R^{-1}B^{T}P \qquad (3.19)$$

이때, complex eigenvalue, eigenvector가 real canonical form 이기 때문에 complex inversion을 real canonical form 형태에서 취해주면 된다.

예를 들어, 2개의 real eigenvalue와 2개의 complex eigenvalue를 갖는 시스템을 수식(2,48)에 표시하였고 이를 수식(2.49)와 같이 real canonical form으로 eigenvector를 표현할 수 있다.

$$\begin{aligned}\Lambda_{complex} &= diag\left[\lambda_1 \quad \lambda_2 \quad \tilde{\lambda}_2 \quad \lambda_4 \quad \tilde{\lambda}_4 \quad \lambda_6\right] \quad \text{(complex form)} \\ \Phi_{complex} &= \left[\phi_1 \quad \phi_2 \quad \tilde{\phi}_2 \quad \phi_4 \quad \tilde{\phi}_4 \quad \phi_6\right] \quad \text{(compelx form)}\end{aligned} \tag{3.20}$$

$$\begin{aligned}\Lambda_{real} &= diag\left[\lambda_1 \quad \lambda_{2R} \quad \lambda_{2I} \quad \lambda_{4R} \quad \lambda_{4I} \quad \lambda_6\right] \text{ (real canonical form)} \\ \Phi_{real} &= \left[\phi_1 \quad \phi_{2R} \quad \phi_{2I} \quad \phi_{4R} \quad \phi_{4I} \quad \phi_6\right] \quad \text{(real canonical form)}\end{aligned} \tag{3.21}$$

다음 수식(2.50)에서 n by n inversion을 통하여 Kalman gain을 구할 수 있다.

$$P_{n\times n} = E_{n\times n}F^{-1}{}_{n\times n} \rightarrow K = R^{-1}B^T P \tag{3.22}$$

제 3 장 Processor in the Loop Simulation

Figure 3-1 On Board Computer (OBC)

PILS(Software In the Loop Simulation)의 필요성

먼저, SILS(Software In the Loop Simulation) 알고리즘은 MATLAB기반의 Space simulator와 MATALB기반의 ADCS알고리즘을 분리하여 ADCS 알고리즘의 성능을 단독적으로 검증하는데 목표가 있으며, 다음과정인 PILS는 OBC Processor에 맞도록 추정 및 제어 알고리즘을 C기반의 알고리즘으로 변경하여 FM(Flight Model)에 탑재되게 될 알고리즘을 검증하는데 그 목적이 있다.

HILS(Hardware In the Loop Simulation)의 필요성

HILS 단계에서는 직접 ADCS알고리즘을 지상에서 검증하기 위한 과정이며, 마찰이 없는 Air-bearing을 이용하여 가상의 우주 환경을 구성하여 ADCS 알고리즘이 우주공간에서 어떻게 적용될지를 검증하는 단계이다. 하지만, 본 논문에서는 단일 축 HILS 실험 검증을 통하여 실제 우주에서의 적용되는 ADCS알고리즘을 지상에서 단일 축 차원에서 검증 및 ADCS

성능을 지상에서 검증하고자 하는데 의미가 있다.

제 1 절 센서 오차 모델링 및 보정

1. 지자계(Magnetoemter) 보정

Magentometer는 OBC에 내장되어 있는 Honeywell사의 HMC5843 모델을 사용하고 있으며, 12bit ADC와 I2C통신을 이용하고 있다. Full Scale은 default로 $\pm 1.0[\text{Gauss}]$ 이는 datasheet에 있는 값이며, 실제 Resolution을 이용한 Full Scale한 값과는 차이가 있다. Resolution은 1/1300[Gauss/count] (default)이며, 출력이 mG 단위로 나오기 때문에, 1/1.3[mG/count] =0.769[mG/count] 값을 갖는다. 이를 이용하여, 구한 Full Scale의 값은 $2^{12}/1.3 = 3150.77[\text{mG}]$ 이므로, 약 $\pm 1.5[\text{Gauss}]$ 의 Full Scale을 갖는다고 할 수 있다.

지자계 보정은 절대크기를 보정하기 위한 과정과 Hard Iron Compensation과 Soft Iron Compensation과정으로 나눌 수 있다.

먼저, 지구자기장 절대자기장의 크기를 보정을 하기 위해서 주변 금속의 영향이 없는 환경에서 보정을 수행한다. 다음은 서울대학교 학군단 운동장에서 타원체 보정을 수행한 내용이며, NGDC-NOAA 홈페이지 에서 이 위치에서의 절대자기장 크기를 알 수 있다. 2017년 6월 학군단 운동장에서의 절대 자기장 크기는 510.37[mG](NGDC 기준)이며, 측정치로는 406.28[mG]이기

때문에 Z-축 기준으로 절대 자기장의 Scale Factor는 $SF = 510.37 / 406.28 = 1.26$로 구할 수 있다.

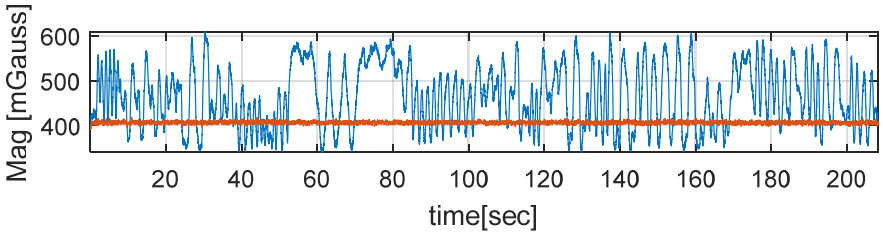

Figure 3-2 절대자기장 크기보정(406.28[mG])

다음으로 지자계 센서의 오차는 지구자기장에 의한 영향으로 Soft Iron에 의 타원오차를 유발하며, 주변의 유도된 금속 영향에 의한 Hard Iron에 의한 Bias오차를 유발하는 것으로 생각할 수 있다.

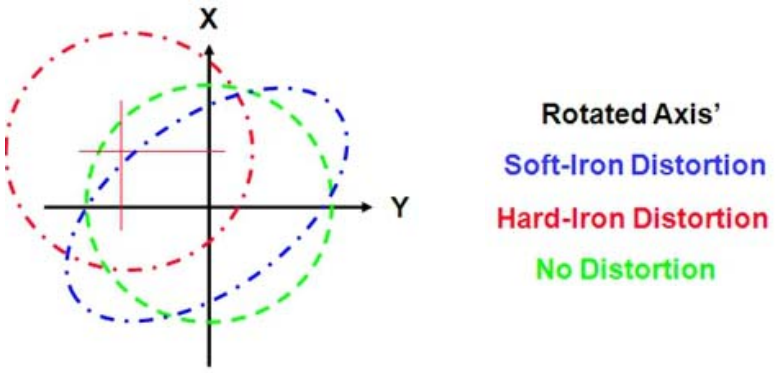

Figure 3-3 Soft / Hard Iron Distortion

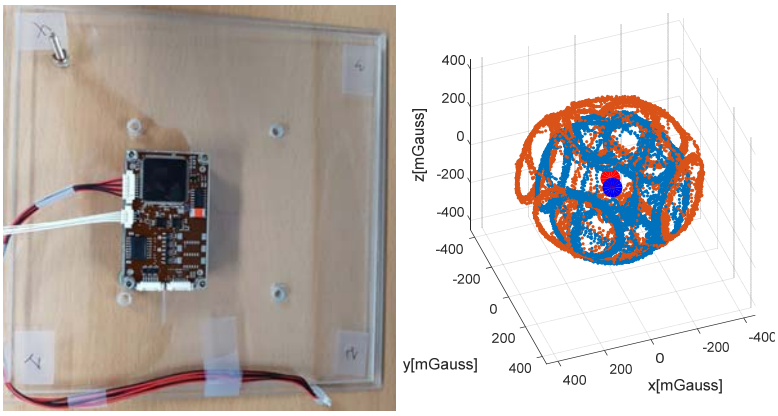

Figure 3-4 주변 금속이 없을 때, 보정 전 / 후 그림

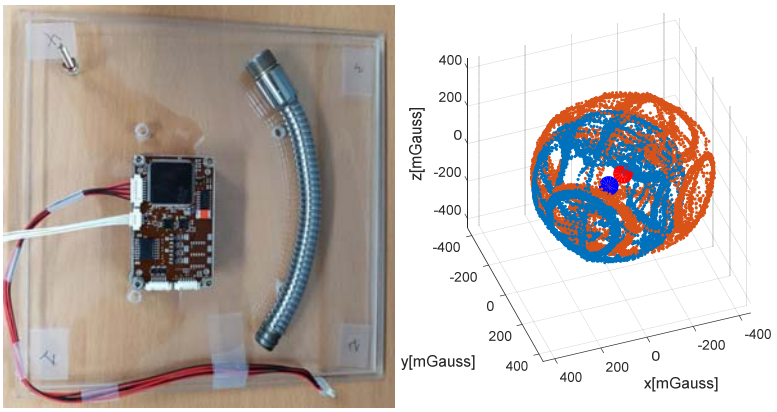

Figure 3-5 주변 금속이 있을 때, 보정 전 / 후 그림

위와 같이 금속이 있을 때와 없을 때 측정치가 달라지는 모습을 볼 수 있다. (보정 전(Red), 보정 후(Blue))

다음은 위의 오차를 보상하기 위한 Hard Iron과 Soft Iron Compensation 과정이다. 보정 전의 측정치는 위의 그림과 같이 지구자기장과 주변 금속에 의해 타원체 형상과 타원체의 중심이 원점에서 벗어난 모습을 볼 수 있다. 이런 이유 때문에, 오차가 발생된다.

보정 측정치를 수집할 때 주의하여야 할 점은, 모든 측정치들을 타원체형태로 고르게 받아야 하며, 자기장이 실내에서는 방향과 크기가 변하기 때문에 최대한 한 점에서 보정을 수행하여야 한다.

세부적인 보정 알고리즘은 모든 자기장 측정치들을 Least square기법으로 fitting한 타원체를 구하며, 타원체의 회전과 주축의 길이 보정을 위한 eigenvector/eigenvalue를 추출하고, 원점으로부터의 bias를 구하게 된다. 보상 파라미터로는 Hard Iron(center- r_{center}), Soft Iron(eigenvector, radii- R_{eig}, r_{radii}) 구체적인 보상과정은 다음과 같다.

1) Hard Iron Compensation

$$B_{hard} = B_{meas} - r_{center} \tag{3.23}$$

2) Soft Iron Compensation (기준 축 설정 – z축 기준)

$$\begin{aligned} B_{phase} &= R_{eig} B_{hard} \\ B_{scale,x} &= \frac{r_{center,z}}{r_{center,x}} B_{phase,x} \\ B_{scale,\mathrm{y}} &= \frac{r_{center,z}}{r_{center,y}} B_{phase,y} \\ B_{scale,z} &= B_{phase,z} \\ B_{soft} &= R_{eig}^{T} B_{scale} \end{aligned} \tag{3.24}$$

Hard Iron Compensation 된 데이터를 이용하여, eigenvector (회전 matrix) 방향으로 회전해준 후 한 축을 기준으로 eigenvalue(주축 길이)를 이용한 타원체를 구 형태로 계산해주는

보상과정을 거친다.

최종적으로 B_{soft} 값이 Soft Iron과 Hard Iron이 보상된 측정치이다. 실험을 위한 실내에서의 보상된 측정치는 다음의 그림과 같다.

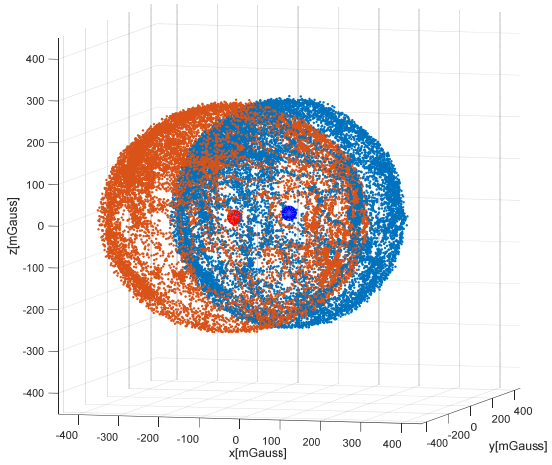

Figure 3-6 Ellipsoid Compensation(3D) (calibrated – blue)

이를 2차원 평면에서 보면 다음과 같이 타원형상이 원 형태로 보상이 된 결과를 볼 수 있다.

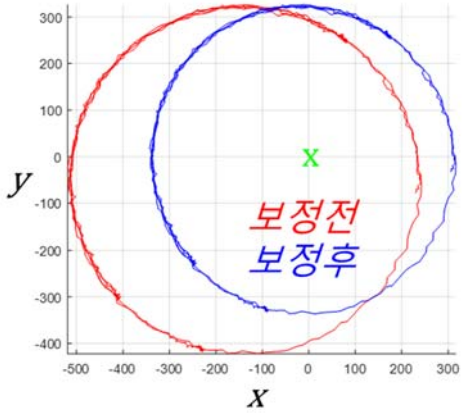

Figure 3-7 Ellipsoid Compensation(2D)

보정하지 않았을 때와, 보정했을 때의 Yaw 방향으로 오차가 얼마나 나는지 확인해보았다. 벡터 각도측면에서 보면, 2차원 평면일 때 RMS 26° 최대 48.8° 만큼 벡터의 각도 차이가 나는 것을 볼 수 있다.

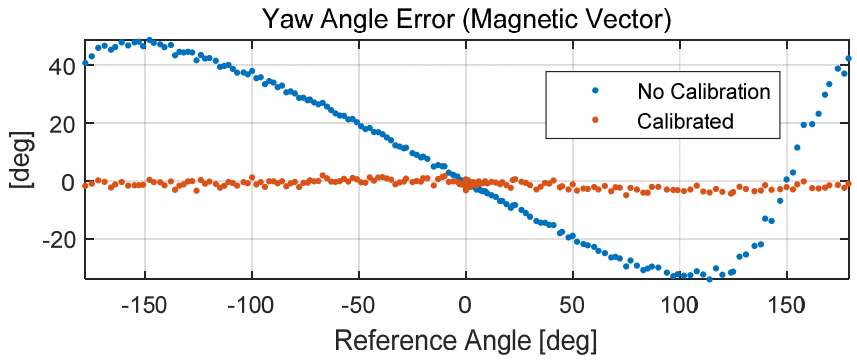

Figure 3-8 Yaw angle error of magnetic vector

Table 3-1. RMS and Max of yaw angle error of magnetic vector

	No Calibration	Calibrated
RMS	26.0 °	**1.7 °**
Max	48.8 °	**3.4 °**

2. 비정밀 태양센서(Coarse Sun Sensor) 오차보정

태양센서는 태양전지판에 부착 되어있는 Photodiode타입 비정밀 선센서로서, 태양의 빛의 세기에 따라 태양광이 전류로 변환이 되고, 이를 I2V amplification회로를 거쳐서 전압으로 변환이 되어 최종적으로 10-bit ADC로 변환이 되게 된다. 단위는 count값이며, OBC와 I2C 통신을 통해 각 태양센서의 값을 읽게 된다.

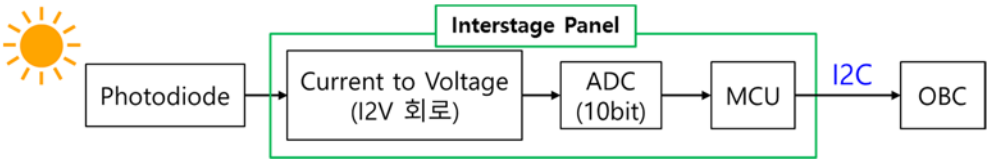

Figure 3-9 Signal Block diagram of coarse sun sensor

태양센서는 5면에 부착이 되어있으며, (+)z면에 따라서 태양벡터를 구하는 3축 알고리즘과 2축 알고리즘이 분리되어 적용된다.

3축 알고리즘의 경우에는 상대적인 크기를 이용하여 다음의 알고리즘을 통하여 태양벡터를 구하게 된다. (Tridiagonal Method)

$$\begin{aligned} I_x &= I_{\max}\sin(\alpha_x)\cos(\alpha_z) \\ I_y &= I_{\max}\sin(\alpha_y)\cos(\alpha_z) \\ I_z &= I_{\max}\sin(\alpha_z) \end{aligned} \quad \vec{\mathbf{s}}^B = \begin{bmatrix} \sin(\alpha_x)\cos(\alpha_z) \\ \sin(\alpha_y)\cos(\alpha_z) \\ -\sin(\alpha_z) \end{bmatrix} = \frac{1}{I_{\max}} \begin{bmatrix} I_x \\ I_y \\ -I_z \end{bmatrix} = \begin{bmatrix} I_x \\ I_y \\ -I_z \end{bmatrix} \quad (3.25)$$

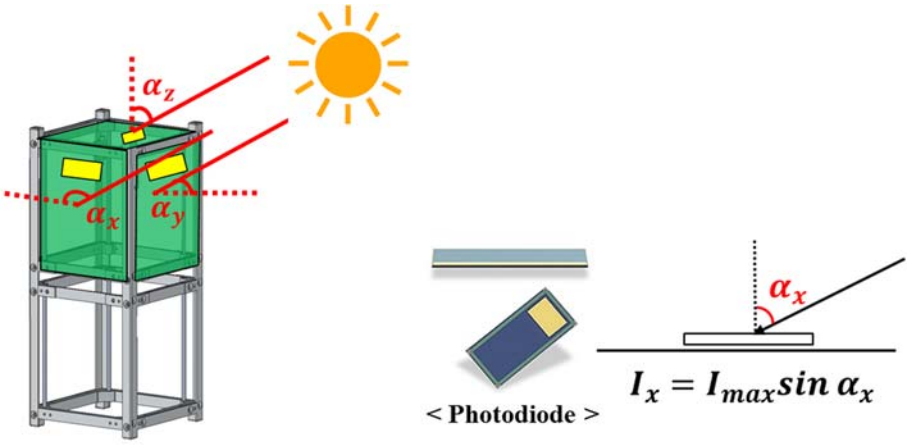

Figure 3-10 Sun vector calculation principle from sun sensor

이때, 태양센서는 태양의 각도에 대하여 Sine 관계를 가져야 위와 같은 상대적인 크기로 방향벡터를 구할 수 있다. 따라서, 태양센서가 태양과의 각도에 대해서 sine관계를 갖도록 센서 오차 모델링을 수행해 주어야 한다.

2축 센서를 이용하여 3축 벡터를 구하는 알고리즘은 수식(3.4)와 같다. (Conical shell model)

$$\vec{\mathbf{s}}^B = \begin{bmatrix} x \\ x\cot(\alpha_x)\sin(\theta_x) \\ x\cot(\alpha_x)\cos(\theta_x) \end{bmatrix} = \begin{bmatrix} 1 \\ \cot(\alpha_x)\sin(\theta_x) \\ \cot(\alpha_x)\cos(\theta_x) \end{bmatrix}$$

$$\alpha_x = \sin^{-1}\frac{I_x}{I_{\max}}, \quad \theta_x = \sin^{-1}\left(\pm\sqrt{\frac{\cot^2(\alpha_x)+1}{\cot^2(\alpha_x)(\cot^2(\alpha_y)+1}}\right) \qquad (3.26)$$

(1) 실외보정(태양)

태양센서는 궁극적으로 우주에서 사용될 것이기 때문에, 직접 태양의 밝기가 어느정도 되는지 실험하기 위해 야외에 나가서 직접 실험하였다. 2축 실험의 경우에는, 다음과 같이 수평계를 이용하여 수평을 맞추었으며, 삼각자를 이용해 45도가 되도록 기울였다. 이때, 데이터는 1[Hz]로 받았으며, 이를 GMAT 프로그램과 비교하였다. 실험환경은 2월 23일 서울대학교 정밀기계연구소 옥상에서 11:50am 부터 5:00pm 까지 진행하였다.

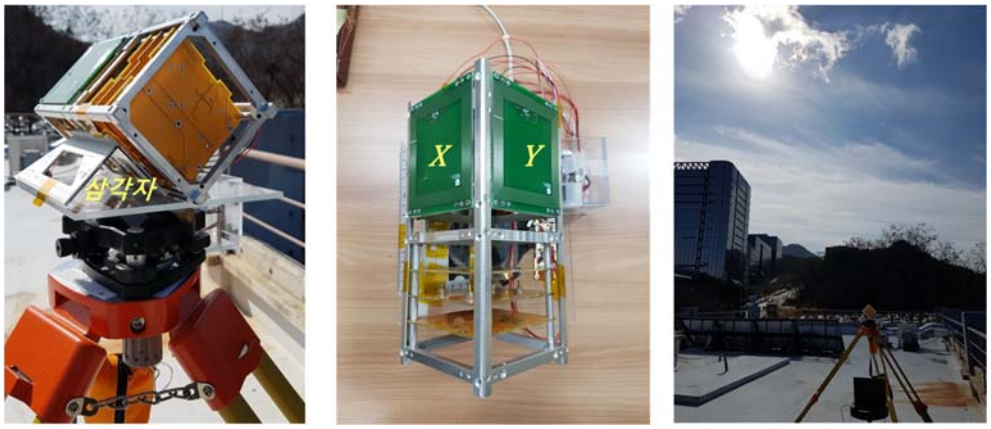

Figure 3-11 태양센서 실외(태양) 성능 실험

GMAT에서는 UTC 시간을 기준으로 ECEF의 태양좌표를 주기 때문에, 정밀기계연구소 옥상의 좌표를 기준으로 ENU 좌표로 바꾸었다. 이를 DCM를 이용하여, Body의 sun vector 측정치와 비교해주었다.

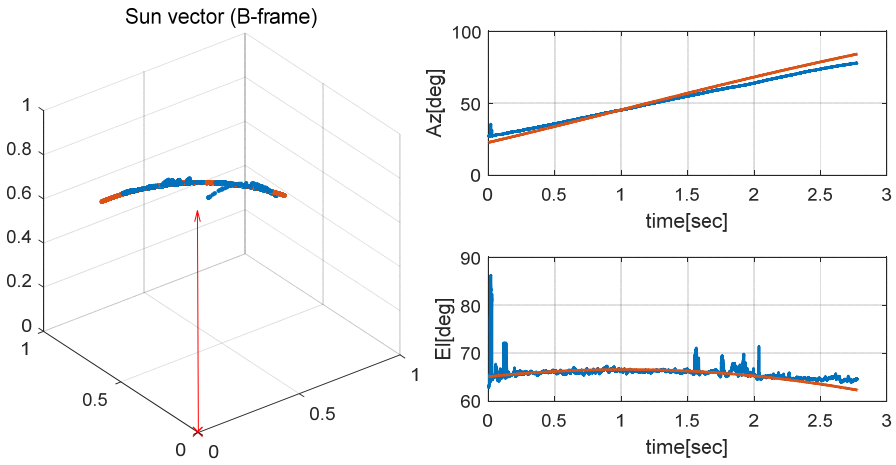

Figure 3-12 Sun vector (measure, simulation)

위의 그래프는 Azimuth 방향과 Elevation방향의 시뮬레이션 측정치와 2축 선센서를 이용하여 측정치를 나타낸 그림이다. 오차를 비교해보면 아래의 그래프와 같다.

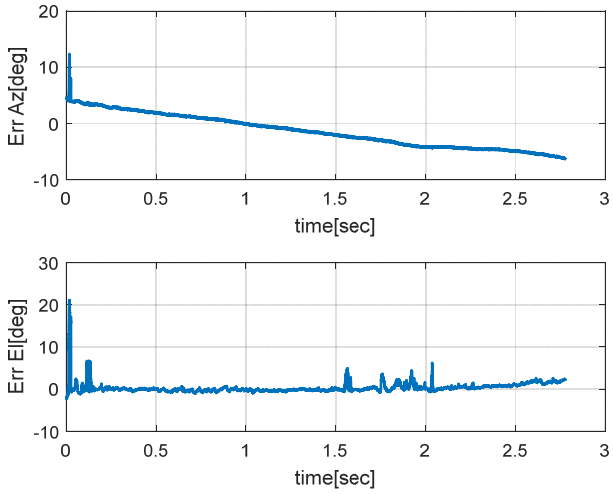

Figure 3-13 Azimuth, Elevation error of sun vector

Azimuth 방향으로 약 5도 정도의 오차와 Elevation방향으로는 2도 정도의 rms 오차를 보였다. 따라서, Sun sensor의 정확도는 약 5 도라고 할 수 있으며, 더욱 정확히 모델링을 하기 위해서는 $I_{\max}$ 의 값을 정확히 알아야 한다.

(2) 실내보정(할로겐 램프)

단일 축 HILS 실험검증을 위해서는 실내에서 실험을 진행해야 하므로, 태양벡터를 할로겐램프의 측정치를 이용하여야 한다. 이때, 할로겐 램프의 특성이 실제 태양보다 좋지 않기 때문에(Noise 수준이 10배 좋지 않음) 태양으로 실험하는 방법보다는 더 열악한 환경이라고 생각할 수 있다. 또한, 할로겐 램프의 특성이 태양과 다르기 때문에, 각도에 따른 센서 모델도 다르다. 할로겐 램프를 이용하여, Static 상황에서 측정치를 받아보면 다음 그림과 같다.

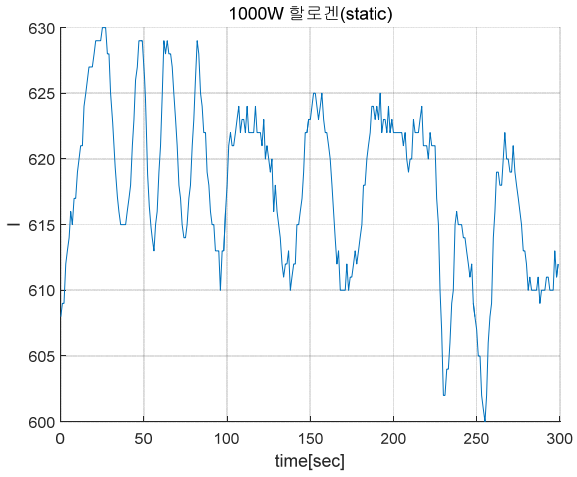

Figure 3-14 Static sun measurement from halogen lamp

3분 동안, 포토다이오드 타입 태양 센서를 빛을 최대로 받을 수 있도록 직각으로 위치해 놓고 측정을 해 보았다. 광량이 약 20~30초 정도 비 주기적으로 진폭이 최대 15정도로 감소 및 증가하는 모습을 볼 수 있으며, 이는 태양의 측정치와 비교할 때, 광원에 의한 오차가 상당히 큰 값 임을 알 수 있다.

태양의 광원의 거리를 정해 놓고, 광량이 최대로 입사될 때의 intensity를 I_{max}로 놓고, 그 값을 기준으로 태양센서 모델을 이용한 보정을 수행한다. Figure3-15와 같이 시간에 따른 광량을 나타낸 그림이다.

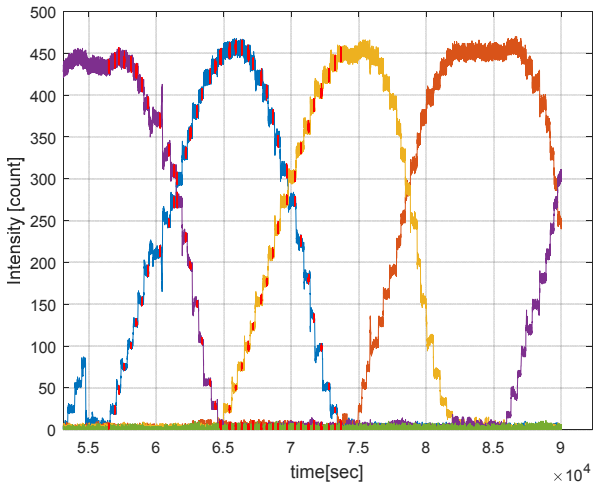

Figure 3-15 Time history of sun sensor of 5-axis

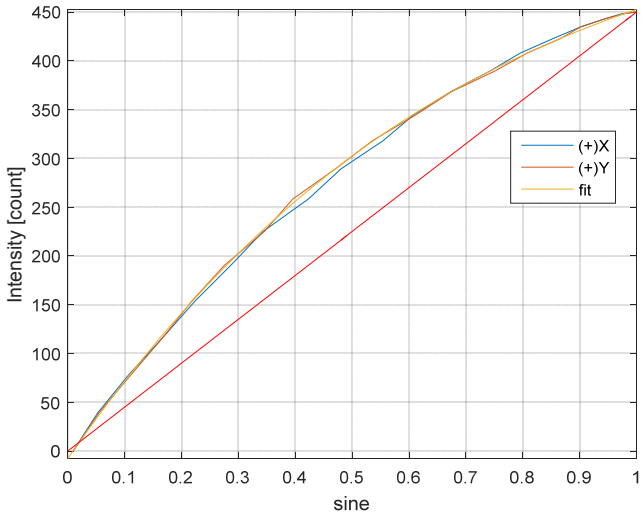

Figure 3-16 Sine v.s. Intensity for Calibration

위의 그래프처럼 $I_{max} = 450[count]$ 일 때, Sine그래프를 x축으로 측정치를 y축으로 설정한 그래프이다. 측정치가 직선이 Sine과 동일한 output을 내주어야 하기 때문에, 3차 함수로 fitting을 하여 모델링을 해 주었다.

$$y = 36e^{-6}x^3 + 2.51e^{-3}x^2 + 1.88x - 7.78$$

다음은 태양센서 실내 보정 실험 사진이며, 다음의 그래프에서 보정 전 / 후 의 Yaw방향 각도 오차를 볼 수 있다.

Figure 3-17 태양센서 실내(할로겐 램프) 보정사진

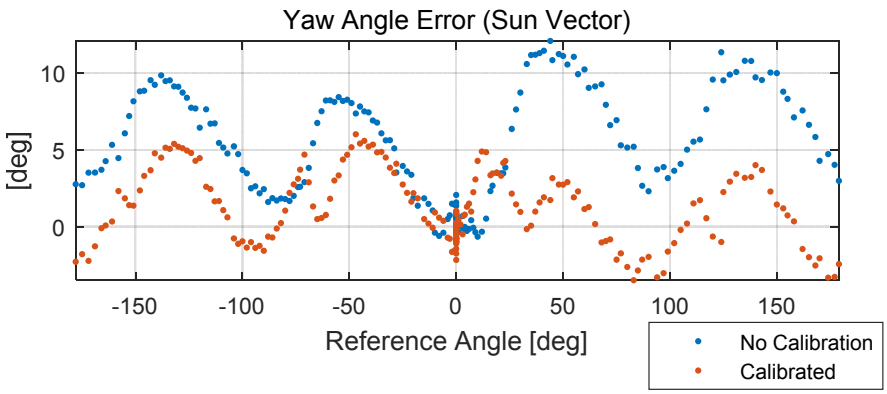

Figure 3-18 Yaw angle error of sun vector

Table 3-2 Yaw angle error of sun vector

	No Calibration	Calibrated
RMS	6.1 °	**2.6 °**
Max	12.1 °	**6.0 °**

3. 각속도계(Gyroscope) bias 모델

먼저, Gyroscope는 OBC에 내장되어 있는 invenses사의 MPU3300 모델을 사용하고 있다. 옵션으로 Full Scale : 225°/s, 450°/s 두개의 Full scale을 설정할 수 있으며, 내부의 Low pass filter Bandwidth를 설정할 수 있다. 큐브위성이 각속도가 큰 기동을 하지 않으므로, Full scale이 작은 범위를 선택하는 것이 유리하므로 Full Scale : 225°/s로 선택을 하였고, noise가 특성이 가장 작은 5Hz (default)값으로 설정해 주었다. Full scale이 정해지면, 측정치의 Resolution을 다음과 같이 확인할 수 있다. 16bit ADC이므로, Resolution은 다음과 구할 수 있으며 이를 측정치의 resolution과 함께 확인한 그래프이다. ($\text{Resolution} = 450 / 2^{16} = 0.00686[^\circ / \text{s} / \text{count}]$)

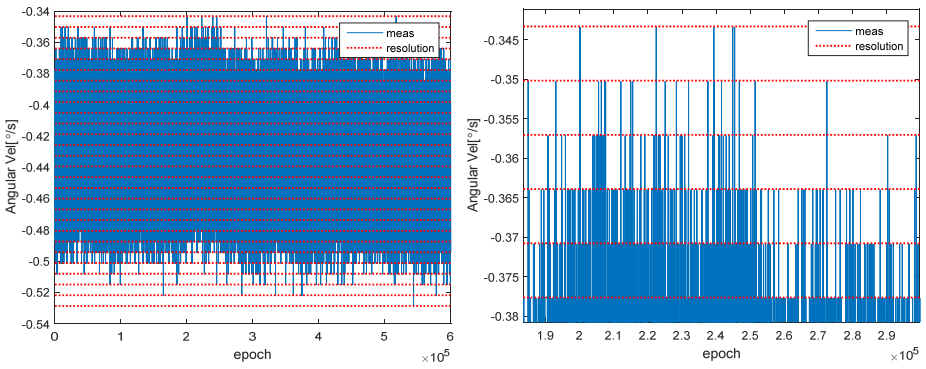

Figure 3-19 Resolution check of Magnetometer

각속도계는 수식 (3.5)과 같은 수학적 모델로 표현할 수 있다.

$$\omega_{meas}(t) = \omega_t(t) + b_0 + b(t) + w(t) \qquad w \sim N(0, \sigma^2_{ARW})$$
$$\dot{b}(t) = -\frac{1}{T_c} b(t) + w_b(t) \qquad w_b \sim N(0, \sigma^2_{RRW}) \tag{3.27}$$

b_0 : const. bias, $b(t)$: time-varying bias

$w(t)$: Gaussian noise

각속도 측정치는 다음과 같이 true 측정치 $(\omega_t(t))$에서 constant bias, b_0와 time-varying bias인 $b(t)$, Gaussian noise 인 $w(t)$로 측정치를 나눌 수 있다. 다음으로는 각속도의 constant bias성분을 알아보기 위해서 다음과 같이 static 상황에서의 데이터를 분석해 보았다. 정적상태일때 자이로는 다음과 같이 offset 되어있는 constant bias성분을 볼 수 있다. 엄밀히 말하면 이 b_0 성분은 전원을 on/off 할 때마다 바뀌는 파라미터로 써, 여러 번의 정적상태일때의 측정치의 평균을 이용하여 수식(3.6)과 같이 평균을 빼 준 측정치를 이용하여 센서를 사용하게 된다.

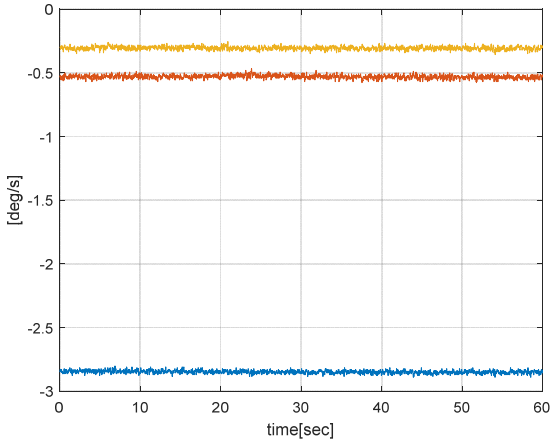

Figure 3-20 Gyroscope measurement at static (b_0)

$$b_0 = \begin{bmatrix} -2.8461 \\ -0.5282 \\ -0.3096 \end{bmatrix} \tag{3.28}$$

정적상태일때의 측정치의 평균을 이용하여 $\omega_t(t) = \omega_{meas}(t) - b_0$ constant bias를 보상하여 준다.

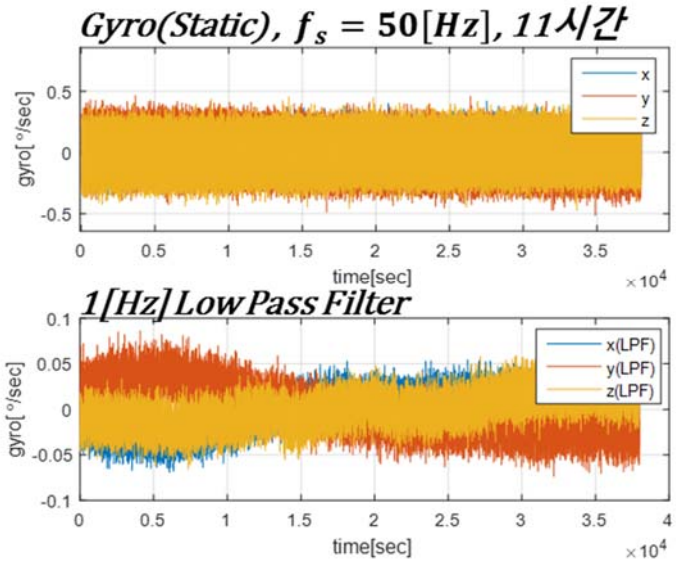

Figure 3-21 Gyroscope static bias measurement

다음으로 시간에 따라 변화하는 성분의 bias를 분석하기 위하여 Figure 3-9-1와 같이 11시간 동안 static gyro 측정치 받았으며, 이를 1[Hz]의 Low pass filter를 거쳤을 때, Figure 3-9-2그래프가 됨을 볼 수 있다. 이는 Bias가 시간에 따라 변화하는 것을 나타내며, 이를 시간에 따라 1차 Gauss markov process 모델로 Bias 모델을 선정하였다.

$$\dot{b}(t) = -\frac{1}{T_c} b(t) + w_b(t) \qquad w_b \sim N(0, \sigma_{RRW}^2) \tag{3.29}$$

T_c : Time constant

수식((3.6)의 모델 변수들 중 b_0 은 static 상태의 측정치를 통하여, σ_{ARW}, σ_{RRW}, T_c 는 Allan Variance 그래프를 통한 noise분석을 통하여 실험적으로 구할 수 있다. Allan Variance 수식은 (3.8)과 같이 average bin을 설정하여 noise의 평균을 이용한 $y(\tau)$ 값을 이용하여 bias의 variance를 구하는 방법이다.

$$\sigma^2 = \frac{1}{2(n-1)} \sum_i \left(y(\tau)_{i+1} - y(\tau)_i \right)^2 \tag{3.30}$$

$y(\tau)$: average of bin

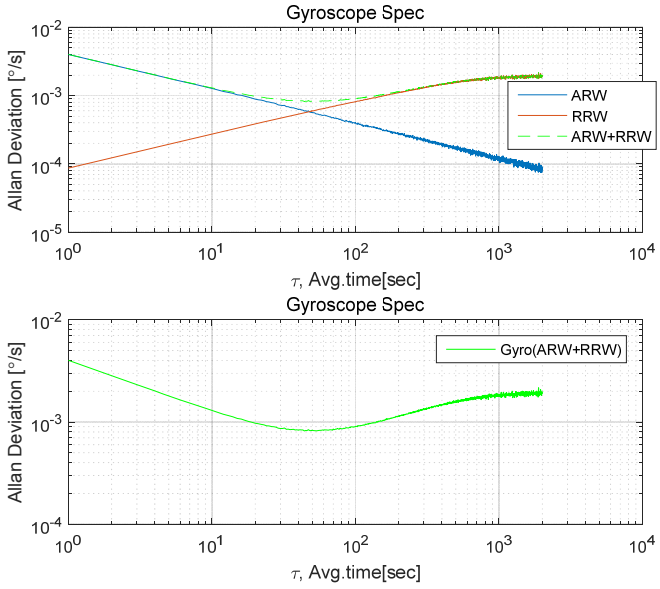

Figure 3-22 Angular / Rate random walk of Gyroscope

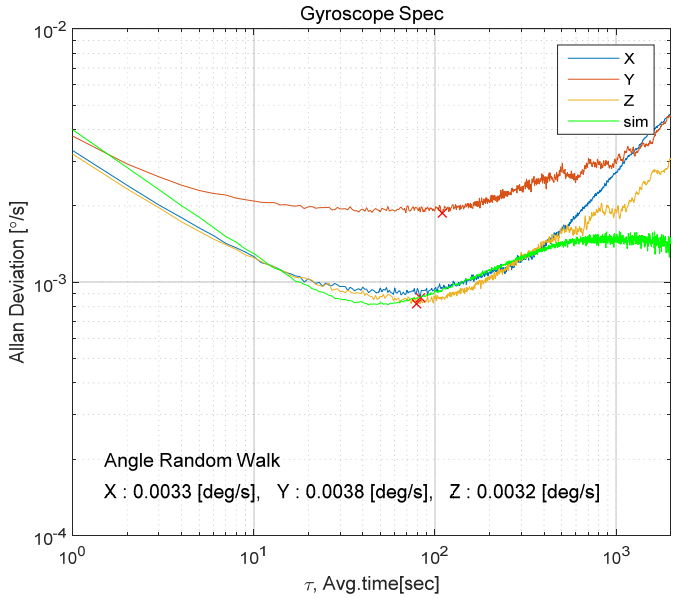

Figure 3-23 Allan Variance graph with Simulation data

위 그림은 Allan Variance 그래프를 각 축에 대하여 그린 그림이다. 이를 Simulation data와 함께 그려 σ_{ARW}, σ_{RRW}, T_c 를 도출 할 수 있다.

$$\sigma_{ARW} = 0.02°/s$$
$$\sigma_{RRW} = 0.0011°/s$$
$$T_c = 500s$$

제 4 장 단일 축 HIL 검증 실험

4장에서는 Engineering Model의 제작과정을 상세히 설명하며 이를 바탕으로 자세 추정 및 제어 알고리즘을 검증하기 위한 단일 축 HILS 검증을 위한 실험 구성과 실험결과에 대해서 설명한다.

아래 그림은 Flight Model의 Signal Block Diagram을 나타낸 그림이다. 실제 위성은 GPS의 측정치를 이용하여 reference vector인 자기장모델(IGRF-12)와 태양모델(JPL, DE405)을 사용하여 위성 자세를 추정하게 된다.

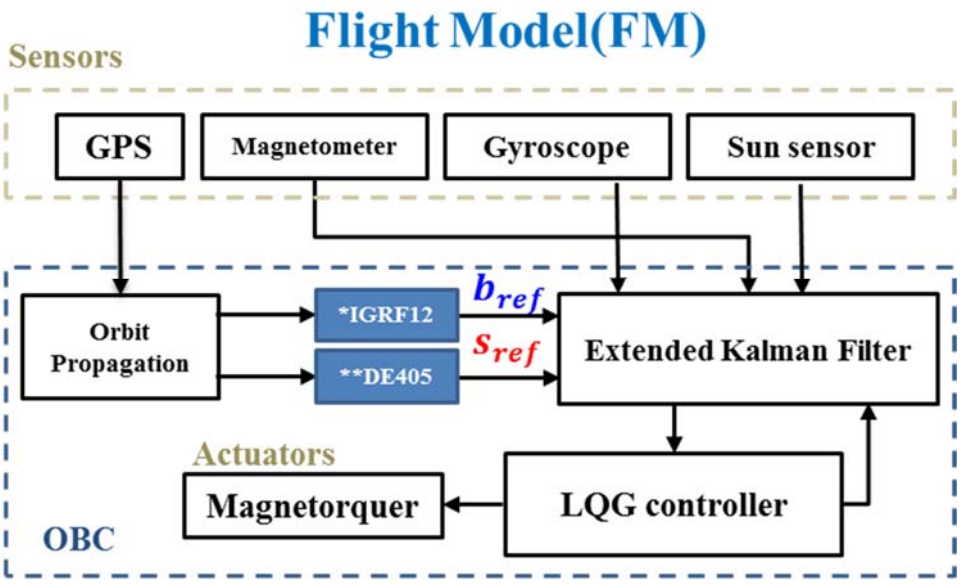

Figure 4-1 Flight Model signal block diagram

하지만 이를 검증하기 위한 단일 축 HILS 검증 실험에서는 다음과 같이 reference vector를 GPS를 이용하지 않고, 처음의 원점을 설정해주는 것으로 다음과 같이 입력의 3초간의 평균을 이용한다.

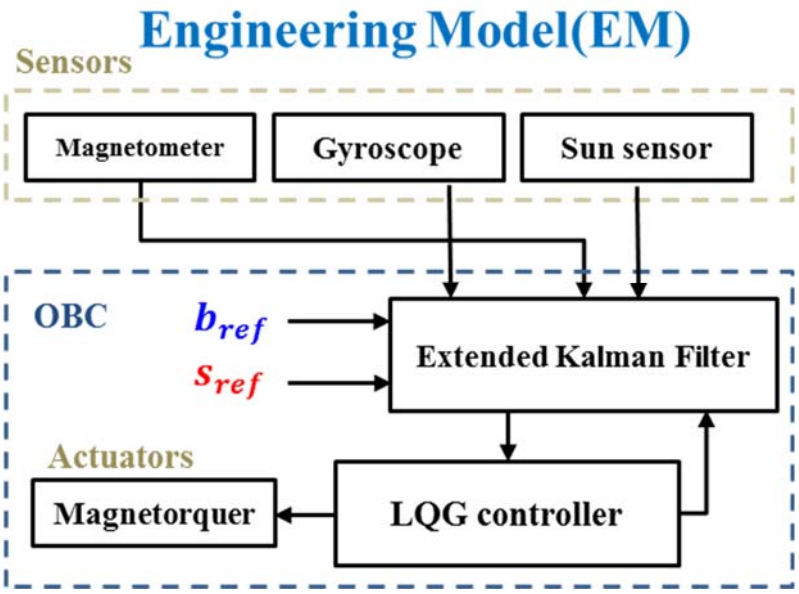

Figure 4-2 Engineering Model signal block diagram

제 1 절 Engineering Model 큐브위성 제작

Engineering Model(EM)은 Flight Model(FM)의 시제품으로서 기능과 성능 측면에서 거의 동일하도록 제작되었으며, PILS와 단일축 HILS 실험 검증을 위한 용도로서 제작이 되었다.

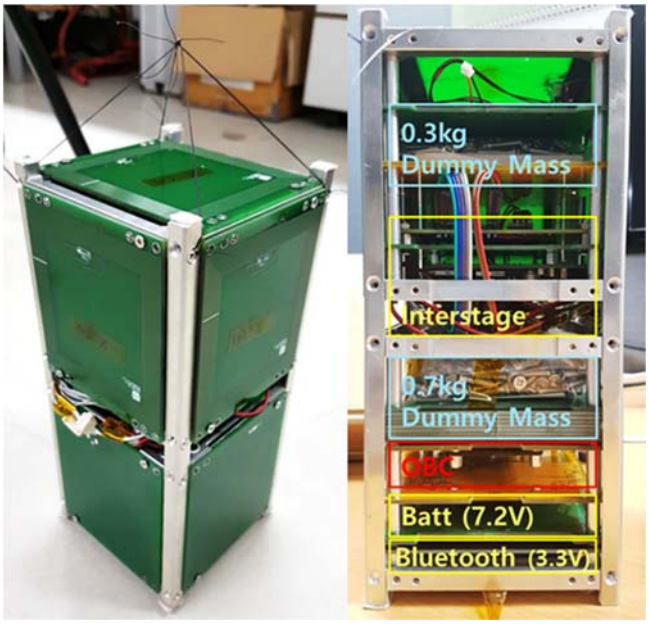

Figure 4-3 Engineering Model(EM) Cubesat

약 1.7kg 의 2U size의 큐브위성이며 FM모델과 동일한 OBC를 탑재하고 있고, 태양 전지판에는 엑추에이터인 마그네토커와 포토다이오드 타입 태양센서를 탑재하고 있다.

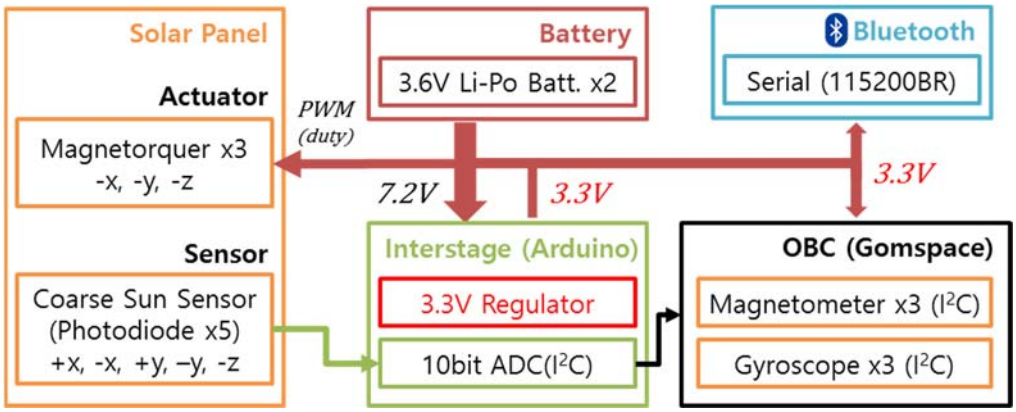

Figure 4-4 Signal block diagram of Engineering Model

큐브위성의 신호블록선도는 위의 그림과 같다. 3.6V의 리튬폴리머 배터리 2개를 이용하여 7.2V를 생성하여 주고, 아두이노의 레귤레이터를 통하여 3.3V를 각 모듈에 공급하여 주고 있다.

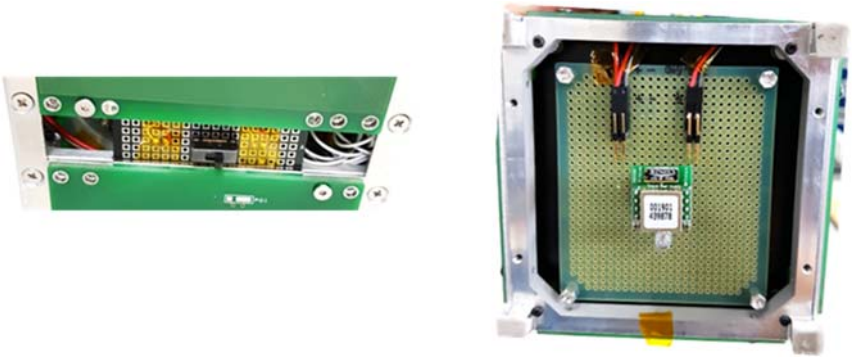

Figure 4-5 Switch and bluetooth for wireless environment

또한, 큐브위성은 아주 약한 마그네토커를 이용하여 제어하기 때문에 위와 같은 무선환경에서 추정 및 제어 알고리즘을 수행할 수 있는 환경을 제작하였다.

1. Magnetorquer 설계(Actuator)

Gomspace의 태양전지판에 내장되어 있는 Magnetoquer는 면적 1.55[m^2]에 10 layer로 20~25턴수로 감겨있는 형태로 제작이 되어 있다. Flight Model과 동일하게 제작하기 위하여 다음과 같이 PCB 패턴 폭은 0.2[mm] 패턴 너비는 7[μm](2once)로 제작하여 설계 후 저항이 130[Ω]이 되도록 설계하였다.

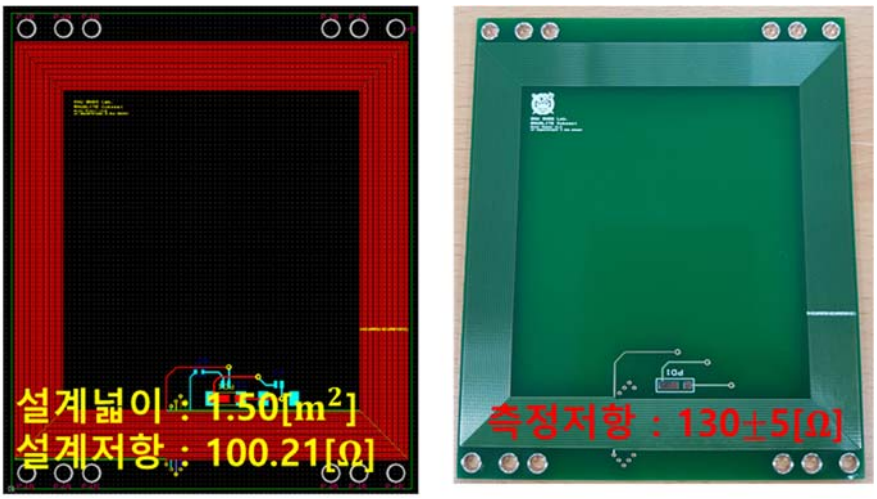

Figure 4-6 Magnetorquer Design Layout

설계된 입력은 $\mu = 3.3[\mathrm{V}]/130[\Omega]*1.50[\mathrm{m}^2] = 0.038[Am^2]$ 이 된다. 이는 FM모델과 거의 비슷한 스펙을 갖는 것을 알 수 있다. 이를 FM 모델과 비교한 실험이다. PWM 신호를 이용하여, 거의 동일한 자기장의 크기가 나오는지 확인하였다.

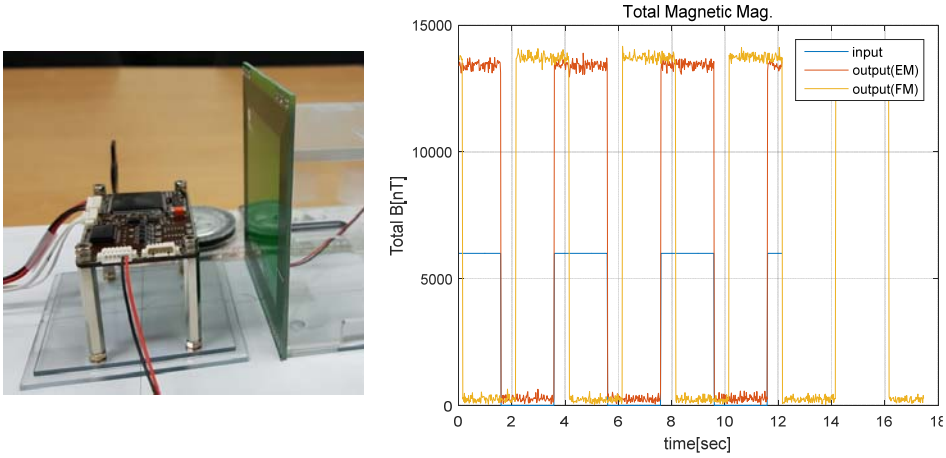

Figure 4-7 Magnetorquer Design Validation Check

Flight Model과 Engineering Model이 거의 동일한 성능을 보임을 알 수 있으며, 토커에 전압을 가했을 때, 최대 자성에 도달하는 시간은 50[Hz]이하이기 때문에, 추정 및 제어 Bandwidth가 1[Hz] 이기 때문에 이러한 출력에 대한 rising time은 고려하지 않아도 된다.

2. 태양센서 설계

저가의 포토다이오드 타입의 SLCD-61N8 모델을 이용하여 (-)Z 축 제외한 5면에 센서가 부착되어 있다. 측정치의 아날로그 신호를 읽어오기 위한 ADC interstage panel은 Arduino 로 다음과 같이 제작을 하였다.

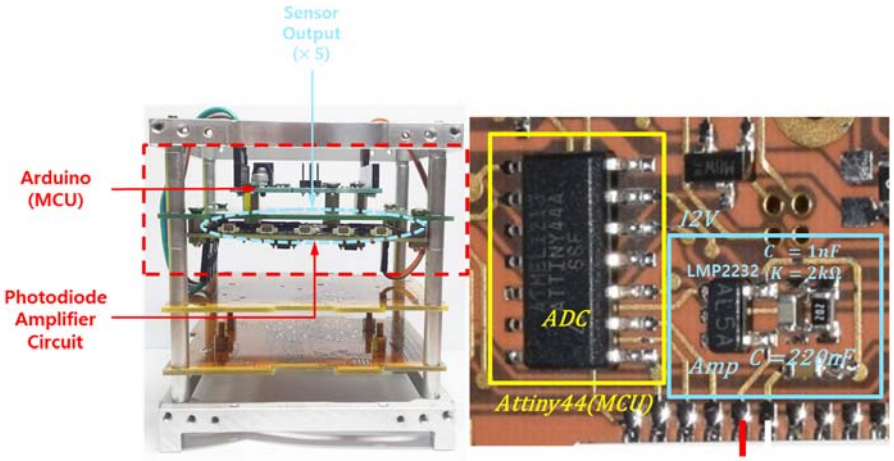

Figure 4-8 Interstage Panel Design using Arduino

Arduino 에서는 200[Hz]로 태양센서 값을 Sampling하며, 이를 OBC에 I2C통신으로 전송해 주고 있다.

블루투스는 펌테크의 FB155BC 모델을 이용하여 OBC의 debugging을 위한 serial 통신에 연결하여 115200 buadrate의 속도로 무선으로 데이터를 전송해 주고 있다. 이는 단일 축 HILS 실험시, 마찰이나 외력이 없어야 하므로 무선으로 구현을 하였다.

제 2 절 단일 축 HILS 검증 실험 환경

단일 축 HIL 검증 실험은 Z축(Yaw 방향)으로 자세 추정 및 제어 검증을 하기위한 실험환경은 다음과 같이 큐브위성의 (-)Z방향의 4면에 낚싯줄을 이용하여 큐브위성이 제어하려는 방향인 Z축으로만 회전할 수 있도록 실험환경을 구성하였다. 아래 그림과 같이 실내의 지구자기장을 이용하며, 태양벡터를 이용하여 큐브위성의 자세를 추정 및 제어를 수행하게 된다.

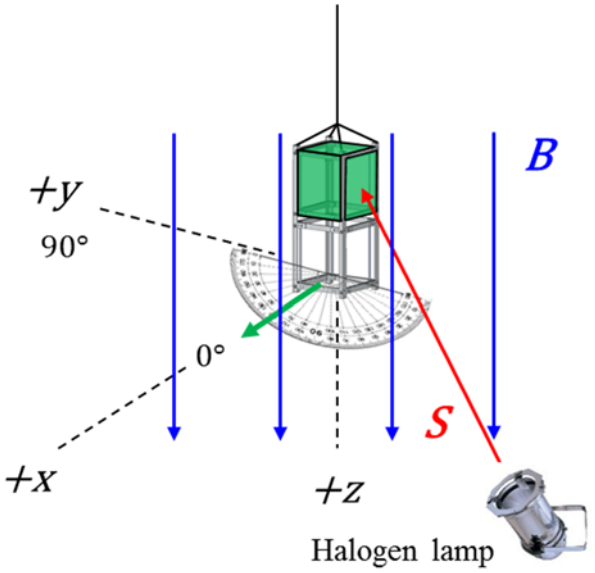

Figure 4-9 Experiment Setup of Single-Axis HIL Experiment

이 실험에서 고려해야할 중요한 부분은 SNUGLITE의 actuator인 magnetorquer는 아주 약한 힘으로 움직이기 때문에

낚싯줄의 꼬임에 의한 힘과 탄성에 의한 힘이 제어하려는 torquer의 입력보다 크게 되면 원하는 자세로 제어를 할 수 없게 된다. 따라서 먼저, 낚싯줄에 의한 꼬임과 탄성에 의한 힘이 어느 정도 되는지 이론적으로 계산하여 보았다.

낚싯줄은 다음과 같이 각도에 대한 탄성(C)과 꼬임(κ)을 갖는 1차 모델로 생각할 수 있다.[4]

$$\begin{aligned}
&\tau = -(\kappa + C)\theta \\
&\tau = I \cdot \alpha = I \cdot \frac{d^2\theta}{dt^2} \\
&I \cdot \frac{d^2\theta}{dt^2} + (\kappa + C)\theta = 0,\ s^2\Theta + \frac{(\kappa + C)}{I} = 0 \\
&\theta(t) = \theta_0 \cos(\omega t + \phi) \\
&\omega = \sqrt{\frac{(\kappa + C)}{I}}
\end{aligned} \tag{4.1}$$

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{I}{\kappa}}$$

따라서, 낚싯줄에 매달린 큐브위성의 주기 T를 구할 수 있으면, 각도에 따른 외력이 얼만큼의 크기로 작용하는지 구할 수 있게 된다.

외력의 크기는 지상실험 환경에서의 복각은 55도라고 실험적으로 구했으며, 입력최대 토크는 다음과 같이 구할 수 있다.

$$\begin{aligned}
&\boldsymbol{\mu} = 0.038[Am^2] \\
&\mathbf{B} = 0.5[Gauss] \rightarrow 0.5e^{-4}[Gauss] \\
&\mathbf{B}_{x,y} = 0.5e^{-4} \cdot \cos(55°) = 2.87e^{-5}[T] \\
&\boldsymbol{\mu} \times \mathbf{B}_{x,y} = 0.038 \cdot 2.87^{-5} = 1.09e^{-6}[Nm]
\end{aligned} \tag{4.2}$$

따라서, 약 큐브위성이 360도 회전했을 때, 입력의 0.2 정도의 외력의 크기가 되도록 주기를 선정하였다.

$$\tau = -(\kappa + C)\theta$$
$$\kappa + C = \frac{(1.09e^{-6})}{2\pi * 5} = 3.4696e^{-8}[\mathrm{Nm}] \qquad (4.3)$$
$$T = 2\pi\sqrt{\frac{I}{(\kappa + C)}} = 2\pi\sqrt{\frac{0.0028}{3.4696e^{-8}}} = 1780[\mathrm{sec}] = 30[\mathrm{min}]$$

따라서 약 30분의 주기를 갖는 낚싯줄을 선정하여 큐브위성의 자세 추정 및 제어 알고리즘을 수행하였다.

제 3 절 MATLAB 기반 실시간 통신 프로그램

단일 축 HILS 지상실험을 하기 위하여 큐브위성 EM모델과 명령 및 데이터 송수신을 위하여 위와 같은 MATLAB기반 실시간 통신 프로그램을 제작하였다.

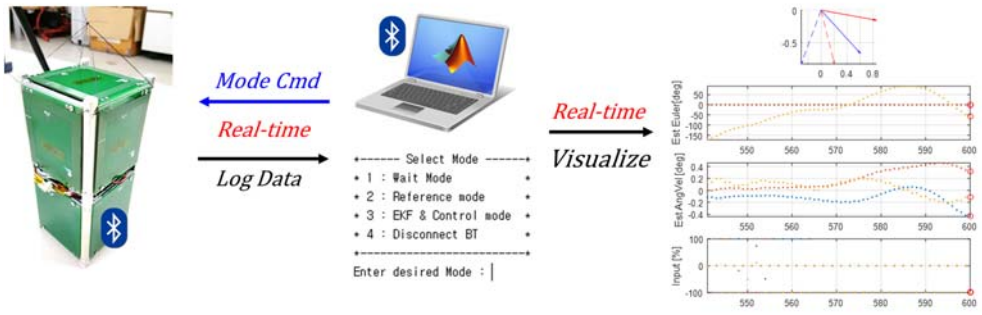

Figure 4-10 Realtime data logging program based on MATLAB

모드 구성은 다음과 같다.

1. Reference Mode 설정 : local(reference) vector를 설정하기 위한 모드로써, Local의 태양벡터와 자기장 벡터를 설정한다. 기준각도(0°)이며,
2. EKF & Control Mode : 추정 및 제어 알고리즘을 수행하는 모드이다.
3. Mag calibration Mode : 자기장을 보정하기 위한 모드이다.
4. Disconnect Mode : 블루투스 연결을 종료하는 모드이다.

HILS실험은 Reference Mode 설정 후, EKF & Control Model로 진행되며, 시간 및 sampling frequency는 MATLAB에서 설정할 수 있다. 통신 Baudrate는 115200이다.

제 4 절 단일 축 HIL 검증 실험 및 분석

다음은 지상에서의 큐브위성의 자세 추정 및 제어 알고리즘을 Yaw 방향의 단일 축 HILS 검증 실험을 하였고, 추정알고리즘만 이용했을 때, 추정오차가 얼마나 되는지 확인한 후에, 추정 및 제어는 0도와 30도 제어를 수행한 결과를 reference와 비교하였다. 또한 이를 추정목표인 ADCS 요구사항과 비교 분석하였다.

1. 자세추정결과, Attitude Estimation Result

다음은 0~360 도까지의 Euler각의 추정결과 결과 이며, 첫번째 그래프를 Yaw 방향의 추정각을 나타내며, 두번째는 reference angle과 Yaw 방향의 오차를 그린 그래프이다. 이때, reference angle은 영상을 통하여 1초, 1도 간격으로 구해주었다.

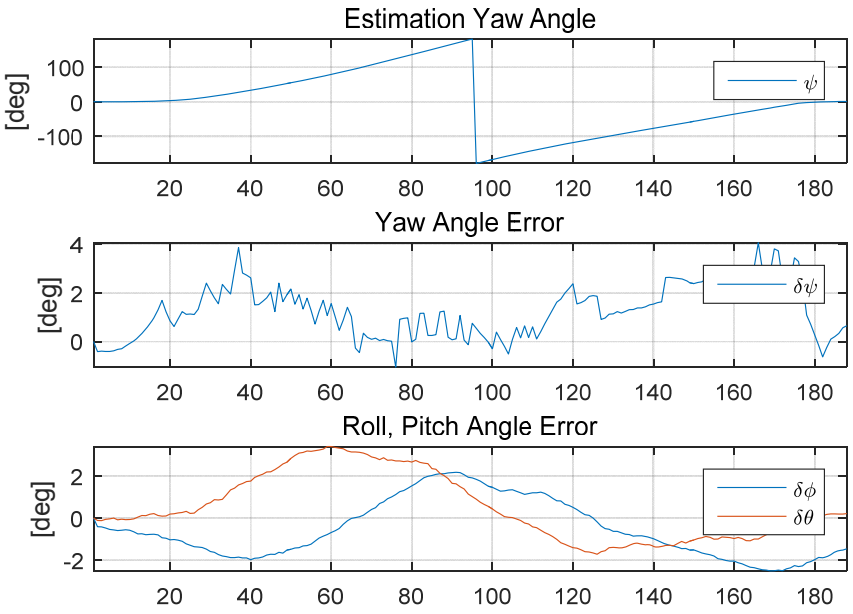

Figure 4-11 Euler angle estimation result

Table 4-1 Estimation error of Euler angle

	Roll	Pitch	Yaw
RMS	1.5°	1.6°	1.7°
Max	2.5°	3.4°	4.0°

요구조건과의 point angle을 비교하기 위해서 Euler angle 3축의 오차의 크기를 확인을 하였다.

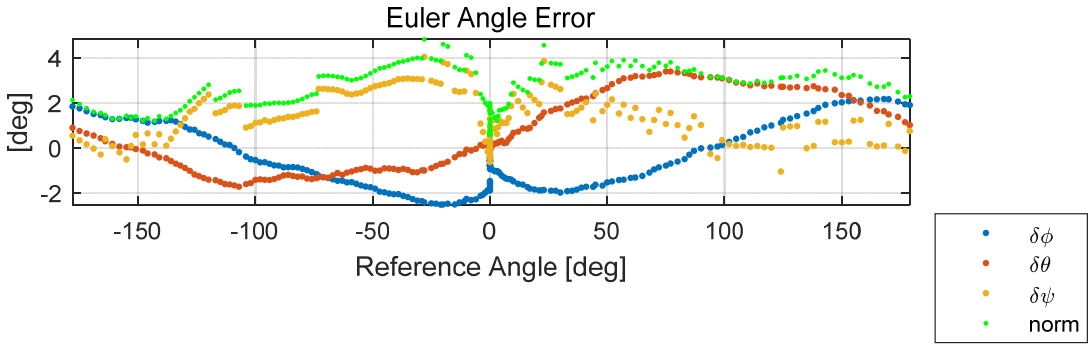

Figure 4-12 Estimation error of Euler angle

위의 오차그래프를 살펴보면, roll, pitch, yaw 3축이 각각 rms 1.5, 1.6, 1.7도 임을 알 수 있으며, 최대 4도 안으로 오차가 들어오는 것을 확인할 수 있었다. 또한 이를 requirement와 비교하면 다음과 같이 만족함을 알 수 있다.

Table 4-2 Estimation result compared to requirement

Error	RMS	Max	Requirement
Estimation	2.7°	4.8°	< 5.0°

2. 자세제어결과, Attitude Estimation and Control result

(1) $\psi_c = 0°$

위의 추정결과를 바탕으로 $\psi_c = 0°$ 로 큐브위성 추정 및 제어 알고리즘을 단일 축 HILS 실험의 결과이다. 초기조건은 $\psi_0 = 40°$ 이다.

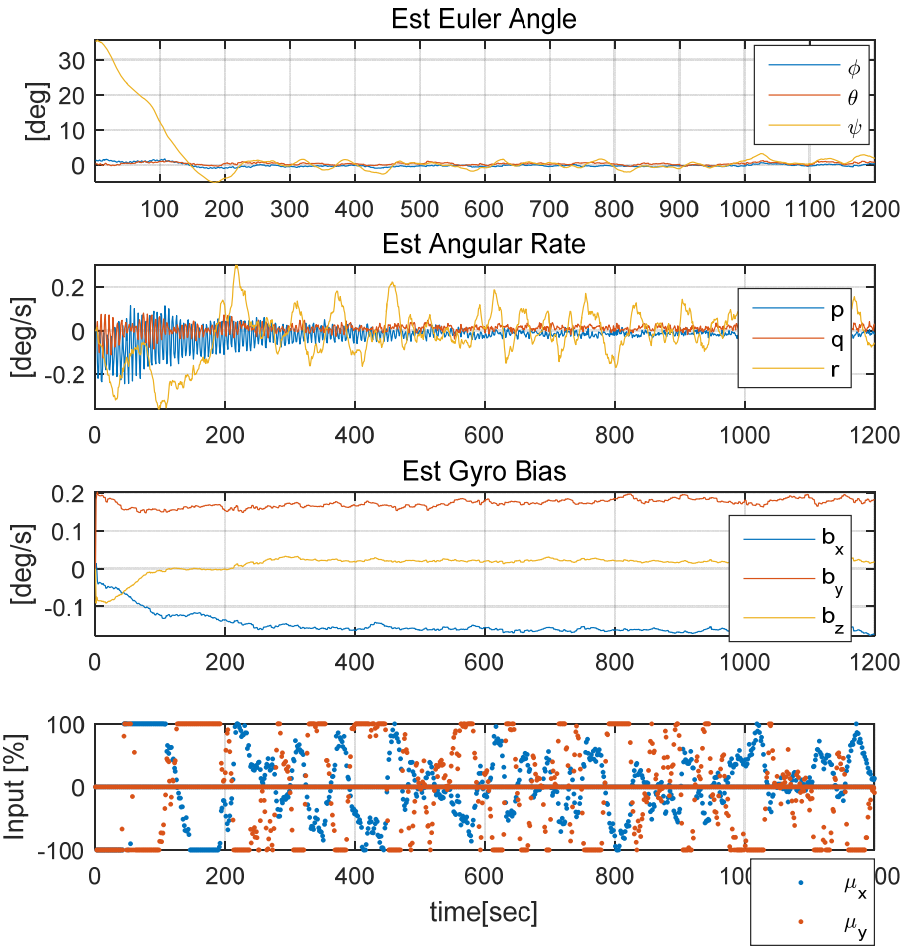

Figure 4−13 Attitude estimation and control result ($\psi_c = 0°$)

$\psi_c = 0°$ 로 제어하는 경우에는, 약 5분(300초) 수렴하는데 결렸으며, 수렴 후 계속 자세를 제어를 하는 모습을 볼 수 있으며 input그래프를 확인해보면, 수렴 전까지는 입력의 100[%]를 사용하며 saturation 된 결과를 볼 수 있으며, 이후 제어 입력이 줄어드는 것을 볼 수 있다.

Yaw방향의 추정 및 제어 성능을 확인하기 위해 Yaw 추정 결과만 따로 Figure 4-10에서 확인할 수 있으며, yaw reference는 영상(수렴 후, 10초 간격)을 통하여 획득한 결과를 이용하였다(초록색으로 표시)

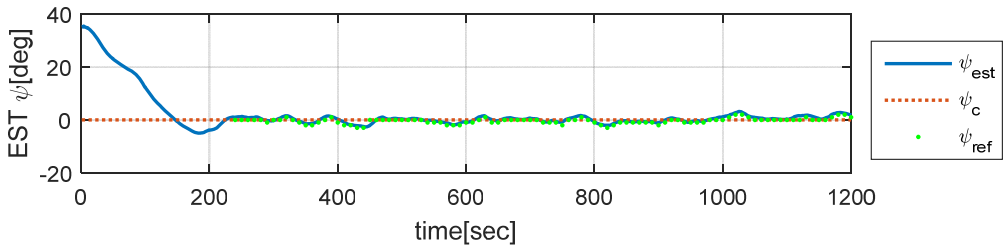

Figure 4-14 Yaw direction estimation and control result ($\psi_c = 0°$)

또한 성능을 수치적으로 확인하기 위하여 reference의 결과를 이용하여 RMS와 max값을 확인하여 요구조건 3축 10°, 1축 5.77°(= 10°/sqrt(3)) 을 만족함을 다음의 Table 4-3에서 확인하였다.

Table 4-3 Control performance with requirement ($\psi_c = 0°$)

Error	RMS	Max	Requirement
Est+Ctrl(0°)	1.1°	2.0°	< 5.77°

(2) $\boldsymbol{\psi_c = 30°}$

다음은 제어 목표 Yaw angle :$\psi_c = 30°$, 초기값 : $\psi_0 = 0°$ 를 수행한 단일 축 HIL 자세 결정 및 제어실험 결과이다.

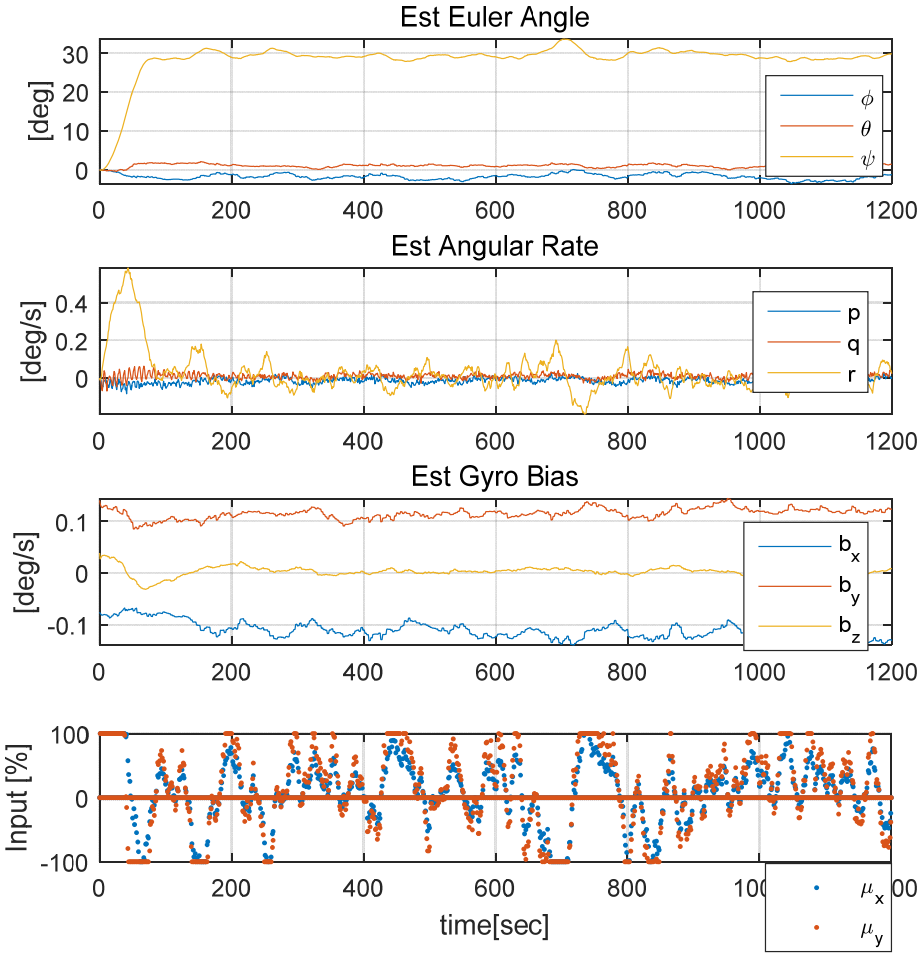

Figure 4−15 Attitude estimation and control result ($\boldsymbol{\psi_c = 30°}$)

첫번째 실험결과와 마찬가지로 약 5분 안에 수렴함을 볼 수 있으며, 추정 및 제어 성능을 확인하면 다음과 같다.

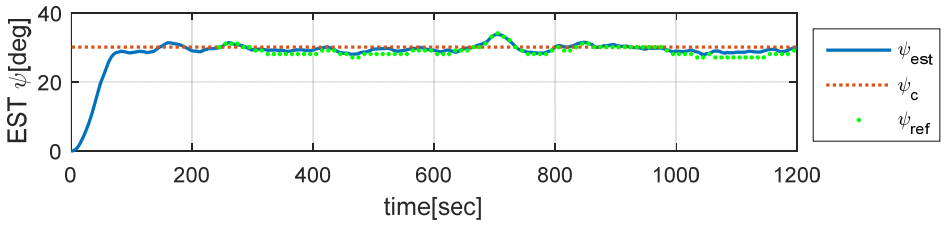

Figure 4−16 Yaw direction estimation and control result ($\boldsymbol{\psi_c = 30°}$)

Table 4−4 Control performance with requirement ($\boldsymbol{\psi_c = 30°}$)

Error	RMS	Max	Requirement
Est+Ctrl(30°)	1.12°	3.0°	< 5.77°

$\psi_c = 30°$의 경우에도 거의 동일한 성능을 보였으며, 실험 1, 2 모두 요구조건을 만족하는 것을 확인하였다. 이러한 추정오차에 대한 원인으로 측정치의 오차가 있는데 이는 다음을 통해서 확인 가능하다.

3. 측정치 오차분석

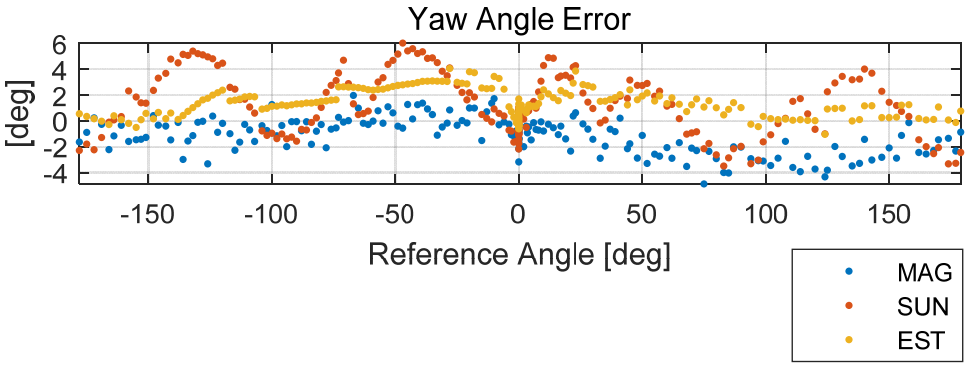

Figure 4-17 Yaw angle error with sensor error

앞서, 3장에서 센서 보정 후 그래프를 함께 Yaw 방향 오차와 그린 그래프이다. 센서 성능을 고려한다면 추정결과가 더 좋은 것을 확인할 수 있다. 센서의 noise 수준은 이러한 오차 수준대비 작은 것을 다음의 그림에서 확인할 수 있다.

이러한 측정치의 오차는 다음과 같은 이유에서 유발되는 것으로 생각할 수 있다. 자기장 벡터의 경우 OBC와 Frame간의 Alignment가 맞지않아서 이러한 오차가 생길 수 있으며, 또한 센서가 가지고 있는 Alignment가 정밀하게 보정되지 않을 수 있다. 또한 타원보정이 가지고 있는 한계로 인한 오차로 생각할 수 있다. 다음으로 태양 센서의 경우에는 직접 Sine 그래프를 통한 모델 보정을 이용하여 보정하였는데, 이도 정확히 센서의 모델과 일치하지 않을 수 있는 보정의 한계를 가지고 있다.

제 5 장 결 론

본 논문에서는 큐브위성의 자세 추정 및 제어 알고리즘(ADCS)를 SILS의 결과와 비교하여 PILS 검증을 통한 실제 우주환경에서의 알고리즘을 검증하였으며, 실제 센서의 성능을 통하여 추정성능과 정확도를 지상에서 단일 축 HIL 검증 실험을 통하여 확인하였다. 또한 실제 큐브위성에 탑재되게 될 센서들의 스펙을 확인하고 이를 EKF와 LQG의 필터의 Covariance값으로 설정하여 주었고, 측정치를 최대한 오차 없이 정확하게 얻기 위하여 센서 오차 모델링 및 보정을 수행하였으며 이후에 이를 ADCS 알고리즘을 지상에서 검증하는 내용 및 실험으로서, HILS 과정을 수행하여 알고리즘 검증을 수행하였다. 특히 HILS는 실용적이고 비용측면에서 유리한 단일 축 HILS 실험검증을 통하여 알고리즘 검증을 하였다.

최종적으로 자세 추정 및 제어 결과를 수치적으로 요구사항과 비교했을 때, 추정 정확도는 point angle 3축을 고려했을 때 최대 4.8°의 오차를 보였으며, 제어 정확도는 최대 3°도 이내로 추정 및 제어가 가능함을 보였다. 이는 실내에서 할로겐 램프를 이용하여 실험한 결과로, 실제 큐브위성이 궤도를 돌면서 우주에서 임무를 수행 시, 직접적인 태양 측정치를 이용할 것이므로, 추정 정확도는 향상 될 수 있을 것이라고 기대된다.

참고 문헌

[1] 김영두, “Attitude determination and control of cubesat using magnetic torquer and LQR controller”, 국내석사학위논문, 서울대학교, 2015.

[2] 장주영, “Attitude determination and control system for low earth orbit cubeSat considering operation scenario”, 국내석사학위논문, 서울대학교, 2016.

[3] S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, "Numerical Recipes in C: The Art of Scientific Computing", 3rd, (Chapter 11.2, Real Nonsymmetic Matrices)

[4] I. Gaponov, “Twisted String Actuation Systems: A Study of the Mathematical Model and a Comparison of Twisted Strings”, *IEEE/ASME TRANSACTIONS ON MECHATRONICS,* VOL. 19, NO. 4, August 2014

[5] M. Nygren, Using Solar Panels as Sun Sensors on NTNU Test Satellite, TTK4550 – Specialization Project

[6] S. Theil, Low Cost, Good Accuracy – Attitude Determination Using Magnetometer and Simple Sun Sensor, 17th Annual AIAA/USU Conference on Small Satellites

[7] Biswa Datta, 2003, Numerical Methods for Linear Control Systems, Academic Press, (Chapter13. Numerical Solutions and Conditioning of Algebaic Riaccati Equations)

[8] Basile Graf, “SwissCube Control Algorithm Design and Validation”, Laboratoire d’Automatique EPFL, February 23, 2007

[9] J. Reijneveld and D. Choukroun, “Attitude Control

System of the DELFI-N3T Satellite", Progress in Flight Dynamics, GNC, and Avionics 6 (2013) 189-208

[10] Kasper Fuglsang Jensen, et. al, "Attitude Determination and Control System for AAUSAT3", Master Thesis, Aalborg University, 2009

[11] N. Kemal Ure, "The Development of a Software and Hardware-in-The-Loop Test System for ITU-PSAT II Nano Satellite ADCS", Massachusetts Institute of Technology, Aerospace Controls Lab

[12] Princeton Satellite Systems, Inc., "Attitude And Orbit Control Using The Spacecraft Control Toolbox v4.6", Princeton Satellite Systems

[13] Roger R. Bate, "Fundamentals of Astrodynamics"

[14] James R. Wertz, "Spacecraft Attitude Determination and Control", Volume73

[15] Takaya Inamori, et. Al., "Compensation of time-variable magnetic moments for a precise attitude control in nano- and micro-satellite missions", Advances in Space Research, vol.48, no.3, 2011, pp.432-440.

[16] Erwan Thebault, et. Al., "International Geomagnetic Reference Field: the 12th generation", Earth, Planets and Space (2015)

[17] Andrew J. Turner, "An OpenSource Extensible Spacecraft Simulation And Modeling Environment Framework", Virginia Polytechnic Institute and State University, 2003.

[18] 김병수 김유단 방효충, "비행동역학 및 제어", 경문사

[19] Yaguang Yang, "Quaternion based model for momentum biased nadir pointing spacecraft", Aerospace Science and

Technology 14 (2010) 199-202

[20] Chris Hall, “Chapter 4. Attitude Determination”, March 3, 2003.

[21] J. R. W. a. W. J. Larson, Space Mission Analysis and Design, Third ed, Microcosm Press and Springer, 1999.

[22] E. Babcock, "CubeSat Attitude Determination via Kalman Filtering of Magnetometer and Solar Cell Data”, 25th Annual AIAA/USU Conference on Small Satellites

[23] G. F. Franklin, J. D. Powell and A. Emami-Naeini, Feedback Control of Dynamic Systems 6th edition, PEARSON, 2010.

[24] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," Journal of Guidance Control and Dynamics, vol. 5, pp. 417-429, 1982.

[25] J. P. Hespanha, "Lecture notes on LQR/LQG controller design," Lecture Notes, University of California, Santa Barbara, California, 2005.

[26] A. E. Bryson, JR, Control of Spacecraft and Aircraft, Princeton University Press, 1994.

[27] R. G. Brown and P. Y. Hwang, Introduction to Random Signals and Applied Kalman Filtering, 4th edition, John Wiley & Sons, 2012.

[28] Y. Bulut and O. Bayat, "Kalman Filtering with Model Uncertainties," in *Proceedings of the IMAC-XXX*, Jacksonville, Florida, USA, 2012.

[29] Y. Bulut, D. Vines-Cavanaugh and D. Bernal, "Process and Measurement Noise Estimation for Kalman Filtering," in *Proceedings of the IMAC_XXVIII*, Jacksonville, Florida, USA, 2010.

[30] G. Gede, "Introduction to Kalman Filtering – An Engineer's Perspective," 20, January, 2011.

[31] V. A. Bavdekar, A. P. Deshpande and S. C. Patwardhan, "Identification of process and measurement noise covariance for state and parameter estimation using extended Kalman filter," *Journal of Process Control,* vol. 21, pp. 585–601, 2011.

Abstract

CubeSat Attitude Determination and Control System(ADCS) is essential algorithm to control and maintain space vehicle(SV)'s attitude to nadir pointing. SNUGLITE (Seoul National University GNSS Laboratory Satellite) is 2U-size Low Earth Orbit(LEO) cubesat, which is be scheduled to be launched in the first half of 2018. For estimating and control cubesat's attitude, Extended Kalman Filter(EKF) and Linear Quadratic Gaussian(LQG) algorithms are applied. Generally, Verification of ADCS algorithm in ground needs complicated and costly HILS(Hardware In the Loop Simulation) simulator with air-bearing that provides closest space environment (enable cubesat to rotate three directions without friction).

As sensors for ADCS, 3-axis MEMs gyroscope and 3-axis MEMs magnetometer are mounted in On Board Computer(OBC) and 3-axis Photodiode type Coarse Sun sensor in solar panel and dual frequency GPS receiver is loaded. Also, 3-axis Magnetic torquer is PCB embedded as an actuator. In particular, actual Flight Model(FM) sensors are needed to calibrate properly in verification of HILS, so sensor error modeling and calibration should be required.

In this paper, single-axis HILS(Hardware in the loop simulation) is proposed and estimation and control performance are verified through the experiment.

Keywords : Cube Satellite, ADCS, PILS(Processor in the Loop Simulation), HILS(Hardware in the Loop Simulation), EKF, LQG control, Attitude determination and control system, sensor calibration, single-axis HIL simulation

Student Number : 2015-22738