J. Korean Soc. Aeronaut. Space Sci. 51(3), 207-216(2023)
DOI:https://doi.org/10.5139/JKSAS.2023.51.3.207
ISSN 1225-1348(print), 2287-6871(online)

큐브위성용 자세결정 및 제어시스템의 실시간 소프트웨어 검증 기법

심한준[1], 배영환[2], 기창돈[3]

Real-time Software Verification Technique of Attitude Determination and Control System for CubeSat

Hanjoon Shim[1], Yonghwan Bae[2] and Changdon Kee[3]
Department of Aerospace Engineering and the Institute of Advanced Machines and Design, Seoul National University, Seoul, Republic of Korea

ABSTRACT

This paper presents a real-time software verification technique for the attitude determination and control system (ADCS) of CubeSats. The on-board computer (OBC) of the CubeSat is equipped with a single core and limited redundancy, making it essential for reliable software to be installed. In consideration of cost, development time, resources, and manpower, an accessible software verification method is necessary. Based on this point of view, this paper first performs a model-in-the-loop simulation (MILS) using MATLAB, a commonly used software in educational institutions for ADCS design. Based on the designed model, software verification is performed by separating the space environment simulator, which provides dynamic models and sensor measurements, and the ADCS module. RS-232 communication is used for data input and output between these modules, and MATLAB-based software-in-the-loop simulation (SILS) and OBC-based processor-in-the-loop simulation (PILS), which is implemented in a real-time operating system (RTOS), are performed. The validity of the implemented software is verified by comparing the results. The proposed technique was validated by presenting the numerical errors of the SILS and PILS results of the SNUGLITE-II CubeSat ADCS.

초 록

본 논문에서는 큐브위성용 자세결정 및 제어시스템의 실시간 소프트웨어 검증 기법을 다룬다. 큐브위성에 탑재되는 OBC(On-Board Computer)는 하나의 단일 코어를 활용하여 다양한 서브시스템을 관리하고, 예비부품(Redundancy) 탑재가 제한되므로 효율적이고 신뢰 가능한 소프트웨어가 탑재되어야 한다. 또한 큐브위성 프로젝트의 비용적 측면, 개발 기간, 시설, 인력 투입 조건의 제약을 고려하여 일반적으로 쉽게 접근 가능할 수 있는 소프트웨어 검증이 수행되어야 한다. 이러한 관점에 기반하여 본 논문에서는 교육기관에서 보편적으로 자세결정 및 제어시스템 설계에 사용되는 MATLAB 소프트웨어를 활용하여 MILS (Model-In-the-Loop Simulation)를 수행하고, 설계된 모델을 바탕으로 동역학 모델 및 센서 측정치를 제공하는 우주환경 시뮬레이터, 그리고 자세결정 및 제어시스템을 분리하여 소프트웨어 검증을 수행한다. 이를 위해 분리된 모듈 간 데이터 입출력을 RS-232 통신을 활용하여 MATLAB 기반의 SILS(Software-In-the-Loop Simulation), OBC상의 RTOS(Real-Time Operation System) 기반의 PILS(Processor-In-the-Loop Simulation)를 각각 수행하고, 결과물을 비교함으로써 구현된 소프트웨어의 수치적 유효성을 검증한다. 제안된 기법은 SNUGLITE-II 큐브위성에 탑재되는 자세결정 및 제어시스템의 SILS, PILS 결과의 수치 오차를 제시함으로써 그 유용성을 확인하였다.

† Received : January 3, 2023 Revised : February 8, 2023 Accepted : February 9, 2023
[1] Graduate Student, [2] Graduate Student, [3] Professor
[3] Corresponding author, E-mail : kee@snu.ac.kr, ORCID 0000-0002-8691-7068

Key Words : CubeSat(큐브위성), Attitude Determination and Control System(자세결정 및 제어시스템), Processor-In-the-Loop Simulation(PILS), Software-Test-Bed(STB), Real-time Software (실시간 소프트웨어), Flight Software(비행 소프트웨어)

Ⅰ. 서 론

큐브위성은 표준 1U의 정육면체 규격을 단일 또는 다중 구조로 가지는 위성으로[1], 1-10[kg]의 나노위성(Nano-Satellite)으로 분류된다. 기존의 중대형 위성과 달리 가볍고 단순하며 저비용으로 개발할 수 있는 이점 때문에 인력 양성, 기술검증, 지구 관측, 통신, 항법, 심우주 탐사 등 큐브위성의 다양한 임무를 수행하기 위한 연구가 진행되고 있다[2,3]. 이러한 큐브위성의 다양한 임무 응용을 위해서는 한정된 크기 내에 구성된 센서를 활용하여 3축 능동 자세를 유지할 수 있는 자세결정 및 제어시스템(ADCS, Attitude Determination and Control System)의 소프트웨어 검증이 필수적이다.

잘 알려진 바와 같이 위성 비행 소프트웨어에 탑재되는 모든 하위시스템은 운용 환경 및 그 특수성으로 인해 헤리티지(Heritage)가 있는 소프트웨어를 활용하거나 높은 신뢰도를 확보하는 검증 활동들이 이루어진다. 하지만 큐브위성의 경우 기존 위성과 비교하여 개발 환경 구축부터 설계와 검증을 짧은 기간 내에 수행하여야 하고, 우주에서 정상적으로 동작된 소프트웨어의 헤리티지 또한 큐브위성의 짧은 임무 수명으로 인해 신뢰도를 평가하기 어렵다. 또한 상당수의 큐브위성에 탑재되는 상용 OBC(On-Board Computer)의 제공된 소프트웨어 아키텍처를 활용하는 경우 발사 후 코드 오류로 발생되는 문제들을 패치를 통해 해결하는 것이 현실적으로 한계가 있다. 게다가 큐브위성에 탑재된 단일 코어에 내장된 비행 소프트웨어가 모든 하위 시스템을 관할하기 때문에 소프트웨어의 치명적인 결함은 위성 운용이 불가능한 상황에 놓인다고 해도 과언이 아니다[4,5]. 이러한 문제를 방지하기 위해 위성에 탑재되는 소프트웨어는 실제 위성 컴퓨터의 하드웨어 모델과 유사한 인터페이스를 제공하는 STB(Software Test Bed), VSTP(Verification Test Script Parser) 시스템 등의 매우 다양한 환경을 활용하여 작성된 코드에 대한 유효성 검사 및 검증을 수행한다[6,7]. 하지만 전술한 환경을 구축하기 위해서는 실제 위성에 탑재되는 비행 소프트웨어의 실행환경과 유사한 인터페이스를 모사해 주어야 하기 때문에 프로젝트 비용 및 개발 기간 증가 등의 요인이 되며, 헤리티지가 확보되지 않은 큐브위성의 하위 시스템 개발에는 적합하지 않다. 최근에는 열거한 방법들을 탈피하고, 큐브위성의 비행 소프트웨어 개발을 위한 소프트웨어 시뮬레이터, cFS(core Flight System) 등의 오픈소스 플랫폼이 개발되어 위성 비행 소프트웨어의 개발 및 검증이 각 개발자들의 컴퓨터를 통해서 용이하게 수행되고 있으나[8], 소프트웨어 구축을 위한 추가 비용, 개발 환경 구축, 플랫폼에 지나치게 의존적인 환경의 한계점들이 존재한다.

자세결정 및 제어시스템의 소프트웨어를 검증하기 위한 기법인 SILS(Software-In-the-Loop Simulation), PILS (Processor-In-the-Loop Simulation)는 작성된 소프트웨어를 가상 환경에서 검증하는 전통적인 검증 방법이다. 가장 보편적인 알고리듬 검증 방법인 SILS는 가상환경을 컴퓨터를 활용하여 구축한 후 소프트웨어를 검증할 수 있기 때문에 단순하고 손쉽게 위성 탑재 소프트웨어를 검증할 수 있다는 장점이 있다. 하지만 실제 하드웨어의 특성과 실시간 처리에 따른 스케줄링 문제를 고려하지 못한다는 단점이 있기 때문에 비행 소프트웨어에 탑재하기 위해서는 PILS 검증이 동반돼야 한다[7]. 지금까지 자세결정 및 제어시스템의 성능시험을 위한 PILS 방법이 제시되어 왔지만, 프로세서와의 연동시험을 위해 MIL-STD-1553B, 이더넷 등의 통신 규격을 활용하거나[9], 복잡한 시뮬레이터 구성[10,11], 코드 변환에 사용되는 자동 코드 생성기의 정밀도 문제[12], cFS 플랫폼 기반의 오픈소스 활용[13,14], 그리고 다수의 컴퓨터를 활용해야 하는 특징을 가지고 있었다[15]. 이 방법들은 다수의 컴퓨터인 시뮬레이터 호스트, 시뮬레이션 컴퓨터, 탑재 컴퓨터, 소프트웨어 검증용 컴퓨터 등의 다중 프로세서를 각각 구축하고, 자세결정 및 제어시스템의 동역학, 센서, 구동기, 우주환경, 외란 모델을 개별적으로 구성하여야 하는 복잡한 문제를 지니고 있다. 또한 실제 위성 프로세서에 탑재하기 위한 알고리듬의 코드 변환 단계에서 오픈 소스 기반의 자동 변환 프로그램에 의존하는 경우 편의성이 증대되지만 구현된 자세결정 및 제어시스템이 최종적으로 OBC를 통해 계산되는 결괏값에 대한 수치적 정확도와 정밀도가 제시된 바가 없다. 또한, 오픈소스에 의존한 소프트웨어는 국방 분야와 같이 보안이 요구되는 위성에는 활용될 수 없다. 더욱이 대부분의 큐브위성 상용품들이 UART(Universal Asynchronous Receiver and Transmitter) 통신만을 지원하고 있기 때문에 제시된 PILS 환경을 활용하여 큐브위성의 자세결정 및 제어시스템의 검증에 적용하기에는 적합하지 않다.

본 논문에서는 큐브위성을 개발하는 데 제약 조건이었던 헤리티지, 개발 환경 구축, 접근성, 통신 인터페이스 그리고 비용 측면의 한계를 최소화하면서 효율적이고 명확한 큐브위성 자세결정 및 제어시스템의 소프트웨어 검증 기법을 제시한다. 즉 일반적으로 접근 가능한 한 대의 컴퓨터와 큐브위성에 탑재되는 OBC만을 활용하여 구현된 자세결정 및 제어시스템 알고리듬을 검증하는 동시에 일반적으로 사용되는 소프트웨어와 통신 방법을 활용하여 접근성과 효율성, 그리고 정확성을 모두 만족시키는 검증 방법을 제안한다. 이때, 전통적으로 활용되는 자세결정 및 제어시스템의 SILS, PILS를 수행하여 체계적이고 검증된 구현 절차를 따른다. 이를 위해 보편적으로

자세결정 및 제어시스템의 알고리듬 설계를 위해 사용되는 MATLAB을 활용하여 동역학 모델 및 센서 측정치를 제공하는 우주환경 모듈과 자세결정 및 제어시스템 모듈로 분리하고, 모듈 간 RS-232 규격 통신을 활용한 SILS을 수행한다. 검증된 MATLAB 기반의 자세결정 및 제어시스템 모듈은 임베디드 프로세서에 적용하기 위하여 동일한 알고리듬이 C언어로 구현돼야 하므로 Visual Studio 기반 C언어 구현 환경을 조성하여 PILS을 수행한다. 최종적으로 리눅스 기반 개발 환경을 활용하여 테스크를 구성한 실시간 비행 소프트웨어에 자세결정 및 제어시스템을 탑재하고, MATLAB 모듈과 OBC에 탑재된 모듈의 출력을 비교함으로써 그 유효성을 검증한다. 이때 각 단계에서 산출된 값의 수치적 연산 결과를 비교함으로써 구현된 코드가 설계 단계에서 예상했던 출력과 수치적으로 일치되는가에 대한 정확성을 확인한다. 제안된 방법의 유용성을 확인하기 위해 SNUGLITE-II 큐브위성에 탑재되는 자세결정 및 제어시스템 소프트웨어의 SILS 및 PILS를 수행하고 OBC에 탑재하여 그 출력물을 비교함으로써 제안된 기법이 단순하고 효율적으로 큐브위성용 소프트웨어를 검증할 수 있음을 보였다.

II. 자세결정 및 제어시스템 소프트웨어 검증 환경

2.1 시뮬레이터 구성 및 검증방법

자세결정 및 제어시스템의 소프트웨어를 검증하기 위해 다음 Table 1과 Fig. 1에 도시된 4단계의 절차를 따라 시뮬레이터를 구성한다.

먼저 1단계로, 큐브위성 자세결정 및 제어시스템의 알고리듬 설계를 위해 MILS(Model-In-the-Loop Simulation)를 수행한다[6]. MILS는 시뮬레이션 환경에서 동역학 기반 모델을 통한 알고리듬 검증 방법으로써 본 논문에서는 교육 목적의 큐브위성 자세결정 및 제어시스템 설계를 위해 접근이 용이한 MATLAB을 활용하였다. 이 단계에서는 모델 검증 및 설계된 자세결정 및 제어시스템의 성능과 분석이 이루어지며, 시스템 요구사항에 대한 목표 성능을 만족하면 비로소 본격적인 소프트웨어 구현을 수행할 수 있게 된다.

2단계는 검증된 MILS 모델로부터, 가상 우주환경의 모델과 입출력 데이터를 관리하는 우주환경 시뮬레이터 그리고 실제 하드웨어 구현을 수행하는 자세결정 및 제어시스템 모듈로 구분하여 SILS를 수행한다. 구분된 우주환경 시뮬레이터와 자세결정 및 제어시스템에 대한 개념도는 Fig. 1에 2-4단계로 구분된 바와 같다. 이 단계에서는 실제 하드웨어에서 구현되는 알고리듬 모델을 분리하여 추후 단계에서 구현된 산출되는 결괏값의 허용 가능 여부를 판단한다. 여기서 모듈 간 통신은 일반적으로 접근이 쉽고 큐브위성 OBC에서 지원 가능한 RS-232 규격을 활용한다. 본 논문에서는 한 대의 데스크톱 컴퓨터에서 2대의 MATLAB 프로그램을 중복으로 실행하고 가상의 RS-232 포트를 실행하여 SILS를 수행하였다.

Table 1. ADCS software development plan and objective for each phase

Phase		Target software		Objective
1	MILS	MATLAB	.m	Algorithm design
2	SILS			Module separation
3	PILS	Visual Studio	.c	Software implementation
4		Linux Eclipse (OBC, FreeRTOS)		Real-time software verification

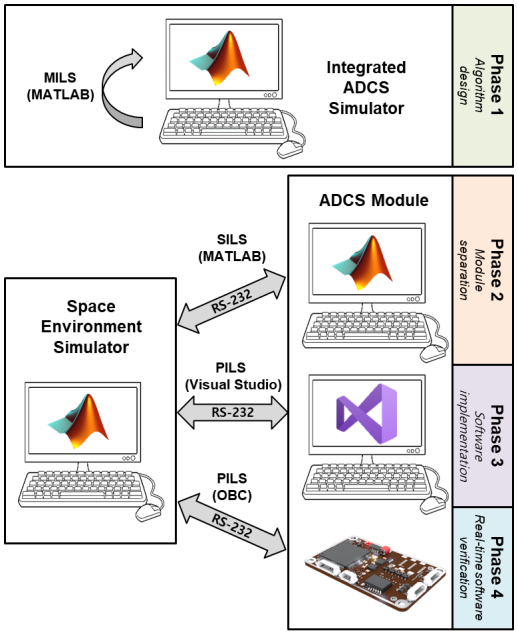

Fig. 1. Configuration of ADCS software verification environment for each phase

3단계는 분리된 자세결정 및 제어시스템 모듈을 임베디드 프로세서에서 실행하기 위해 C언어로 구현하는 단계이다. 2단계에서 수행된 SILS에서 자세결정 및 제어시스템의 모듈이 C언어를 통해 구현되고, 동일한 입력을 인가하는 환경을 조성한다. 이때 본 논문에서는 구현의 편의성을 도모하고자 Visual Studio를 활용하여 프로그램을 구현하고 PILS를 수행하였다. 일반적으로 임베디드 프로세서에서 수행하는 PILS 대신 Visual Studio에서 자세결정 및 제어시스템 모듈을 구현하면 한 대의 컴퓨터만으로 소프트웨어 검증이 용이하게 된다. 임베디드 프로세서에서 실행되는 코드를 구현하기 위해서는 작성된 코드를 부트로더에 업로드하고, 프로세스를 부팅하여 알고리듬을 검증하여야 한다. 그러나 큐브위성에 탑재되는 상용 OBC의 경우 대부분 에뮬레이터와 같은 디버거를 지원하지 않기 때문에 메모리 접근, 중단점 등의 디버깅이 어렵다. 이러한 문제점은 Visual Studio에서 지원하는 디버거 기능을 활용하면 소프트웨어를 비교적 용이하게 검증할 수 있으며, 2단계에서 수행된 SILS와 동일 입력에 대한 동일 출력을 확인함으로써 신뢰성 있는 소프트웨어를 구현할 수 있다.

마지막 단계인 4단계에서는 3단계에서 구현된 소프트웨어를 큐브위성 OBC에 탑재한다. 여기서 큐브위성에 탑재되는 소프트웨어는 Linux Eclipse 기반의 GCC(GNU Compiler Collection) 컴파일러를 활용하며, RTOS(Real Time OS)는 FreeRTOS를 사용하여 1초 주기의 테스크를 구성하였다. 3단계에서 수행한 PILS와 마찬가지로, RS-232 통신 규격에서 제공받는 입력에 대해 자세결정 및 제어시스템 모듈에서 산출되는 데이터 값을 우주환경 시뮬레이터로 전송하여 PILS를 수행한다. 최종적으로 우주환경 시뮬레이터에서 제공되는 동일한 입력에 대해 MATLAB에 작성된 자세결정 및 제어시스템 모듈과 OBC의 실시간 자세결정 및 제어시스템 모듈 출력을 비교함으로써 구현 소프트웨어의 유효성을 검증한다.

2.2 우주환경 시뮬레이터

전체 시뮬레이터에 대한 블록선도는 우주환경 시뮬레이터와 자세결정 및 제어시스템 모듈로 분리되어 다음 Fig. 2에 도시된 바와 같이 구성된다.

본 논문에서 구성된 우주환경 시뮬레이터는 궤도 전파기, 센서 시뮬레이터, GPS 시뮬레이터, 구동기 모델, 자세 전파기로 구성된다. 각각의 시뮬레이터를 구성하기 위해 적용된 모델은 Table 2에 도시되었다. Table 2에서 확인할 수 있는 바와 같이 우주환경 시뮬레이터는 성능검증을 위해 정밀한 모델이 활용된 반면, 자세결정 및 제어시스템은 자세 제어기를 구성하기 위한 가정이 포함되어 계산량을 줄이고, 비선형성 불확실성에 대해 확률적 오차가 고려된 것을 확인할 수 있다.

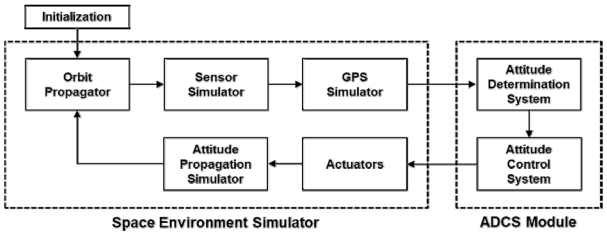

Fig. 2. Overall simulation block diagram

Table 2. Comparison of applied simulation models

Simulation Model	Space Environment Simulator	ADCS Module
Moment of inertia	$I_{xy}, I_{yz}, I_{zx} \neq 0$	$I_{xy}, I_{yz}, I_{zx} = 0$
Gravity	EGM-96 (360x360)	Two-body equation
Air drag density	Harris-Priester model	Extended Kalman filter process noise
Solar radiation pressure	Spherical shadow model	
Magnetic field	World Magnetic Model (WMM)	IGRF-13 (Reduced 11^{th} order)
Solar system	JPL DE405	Simplified DE405

우주환경 시뮬레이터는 전체 시뮬레이션의 시나리오에 대하여 입출력 데이터를 생성 관리하고, 구성된 각 모델에 대해 자세결정 및 제어시스템으로 인가되는 측정치 오차들을 설정할 수 있다. 여기서 생성된 측정치는 태양센서, 자기장 센서, 자이로센서 그리고 GPS 측정치로 구성된다. 이는 큐브위성의 하드웨어에서 자세결정 및 제어시스템에 적용되는 측정치를 모사한 것이며 실제 입력과 동일한 자료형과 사양을 적용한다. 입력에 대해 자세결정 및 제어시스템으로부터 산출 값은 우주환경 시뮬레이터로 제공되어 참값에 대한 오차 성능을 검증할 수 있다. 우주환경 시뮬레이터에서 생성되는 측정치와 자세결정 및 제어시스템으로부터 산출되는 상태변수, 구동기 입력은 Tables 3, 4에 정리된 바와 같다. 여기서 실제 큐브위성의 자세결정 및 제어시스템으로 활용되는 하드웨어는 Table 5에 정리하였다.

분리된 시뮬레이터를 통해 본 논문에서 검증하고자 하는 자세결정 및 제어시스템의 소프트웨어 성능을 효율적으로 검증한다. 이때 Table 3에 정리된 측정치는 각 시뮬레이션에서 동일 입력으로 인가된다. 또한 자세결정 및 제어시스템으로부터 산출되는 출력은 Table 4에 정리된 바와 같이 추정 상태변수, 제어 입력으로 구성되어 있다.

Table 3. Variables and data types provided by the space environment simulator

Module	Variable	Data		Target
		Type	Size	
Sensor Simulator	Sun sensor	uint16	5x1	ADCS
	Magnetometer	float	3x1	
	Gyroscope	float	3x1	
GPS Simulator	Position, velocity, time	int32	6x1	
		uint16	1x1	
	Number of satellite	uint8	1x1	
	PRN	uint8	Nx1	
	Pseudorange	double	Nx1	
	Carrier phase	double	Nx1	
	Doppler	float	Nx1	
	Carrier to noise power density	uint8	Nx1	

Table 4. Variables and data types provided by the ADCS module

Module	Variable	Data		Target
		Type	Size	
Attitude Determination	Estimated state	double	10x1	Attitude Control
Attitude Control	Actuator input	double	6x1	Actuators

Table 5. Hardware configuration

Component	Model	Specification
Sun sensor	Silonex SLCD-61N8	Low-cost planer photodiode type sun sensor
Magnetometer	Honeywell HMC5843	3-axis digital compass
Gyroscope	InvenSense MPU3300	3-axis MEMS gyroscope
GPS receiver	SNU self-development	L1/L2C dual-frequency GPS receiver[16]
Reaction wheel	Cubespace CubeWheel Small	maximum torque 0.23 [*m*Nm], momentum storage 8000[RPM]
Magnetorquer	Cubespace CubeTorquer Small	Rod type, maximum dipole moment 0.24[Am^2] at 5[V]

제공된 추정 상태변수는 자세결정의 필터 추정 성능을 확인하기 위함이며, 제공된 제어입력은 우주환경 시뮬레이터로 제공되어 위성의 자세를 갱신하고, 갱신된 자세로부터 측정치가 생성된다. 본 논문에서는 일정 시간 반복된 시뮬레이션으로부터 얻은 자세결정 및 제어시스템의 출력에 대해 SILS, PILS 결과를 비교함으로써 소프트웨어를 검증한다.

2.3 자세결정 및 제어시스템 모듈

본 논문에서 검증하기 위한 자세결정 및 제어시스템 모듈은 SNUGLITE-II 큐브위성에 탑재된 시스템으로써 1[Hz] 주기로 RTOS의 스케줄러에 의해 테스크가 운용된다. 구성된 자세결정 및 제어시스템은 큐브위성의 운용을 효율적으로 수행하기 위한 방편으로 각각 2가지, 4가지의 모드로 구성되어 있으며, 각 모드에 대한 알고리듬 및 블록선도는 Table 6, Fig. 3에 도시되었다. 본 논문은 구현 소프트웨어의 검증을 목표하므로 알고리듬에 대한 자세한 설명은 생략한다.

자세결정(AD) 모드는 MEMS 센서를 활용한 확장칼만필터(MEMS EKF) 모드와 3개의 GPS 안테나를 활용하여 자세를 결정하는 GPS 자세결정(GPS) 모드로 구성된다. 먼저 MEMS EKF 모드는 태양센서, 자기장센서, 자이로스코프를 결합하여 큐브위성의 자세를 추정하는 모드로써 저전력으로 위성을 운용하는 자세결정 방법이다. GPS 모드는 다수의 mm급 GPS 반송파 위상 신호를 활용하여 정밀 자세 추정을 수행하는 방법으로, SNUGLITE-II 큐브위성의 기술적 임무를 달성하기 위해 시도되는 방법이다.

자세제어(AC) 모드는 초기 큐브위성의 각속도를 감쇠시키기 각속도 감쇠(B-dot) 모드, 임무를 수행하기 위한 지구 지향(Nadir) 모드, 임무 데이터를 전송하기 위한 추적(Tracking) 모드 그리고 반작용휠이 고장 나는 경우 자기토커만을 활용하는 자기토커(MTQ) 모드로 각각 구

Table 6. ADCS mode and applied algorithm

Module	Mode	Algorithm
Attitude Determination (AD)	1	MEMS EKF
	2	GPS
Attitude Control (AC)	1	B-dot
	2	Nadir
	3	Tracking
	4	Magnetorquer (MTQ)

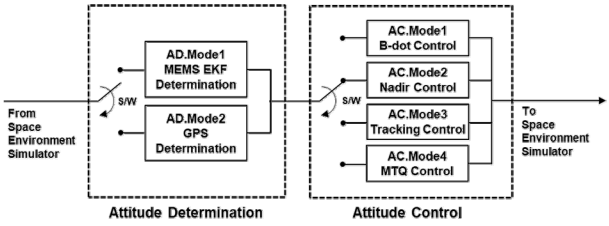

Fig. 3. Configuration of ADCS

성되어 있다. 자세제어 구동기로는 반작용휠이 활용되며, 반작용휠의 포화를 방지하기 위해 자기토커가 활용된다. 구성된 각각의 모드는 한 가지만 선택이 가능하며 CDH (Command and Data Handling) 시스템에서 관리된다.

구성된 자세결정 및 제어시스템을 위성에 탑재하기 위해 구현되는 소프트웨어는 C언어를 기반으로 작성된다. 그러나 2.1절에서 전술한 바와 같이 자세결정 및 제어시스템의 설계는 MATLAB에 내장된 함수를 기반으로 성능 분석이 이루어지기 때문에 수치적 해를 요구하는 알고리듬을 C언어로 구현하여야 한다. 적용된 MATLAB 내장함수에서 동일하게 구현이 불가능한 함수에 대해 C언어로 구현된 수치해석 알고리듬은 다음 Table 7에 도시된 바와 같다. 따라서 MILS, SILS에서 활용된 MATLAB 함수는 C언어 구현 단계에서 설정한 수치적 정밀도에 따라 큐브위성의 자세결정 및 제어시스템 모듈 출력값의 성능이 좌우된다. 본 논문에서 구현된 자세결정 및 제어시스템 알고리듬은 큐브위성의 효율적 계산을 도모하기 위한 수치해석 알고리듬을 채택하고, MATLAB 기반의 SILS와 큐브위성 OBC 기반의 PILS 결과를 비교함으로써 그 성능을 검증하였다.

Table 7. Comparison of key functions implemented in ADCS

Algorithm	MATLAB function	Implemantation numerical method
Singular value decomposition	svd()	Jacobi method
Matrix inverse	inv()	Gauss-Jordan method
LQR solution	lqr()	QR decomposition (Householder method)
		Potter's method

Ⅲ. 소프트웨어 검증 시뮬레이션 결과

3.1 소프트웨어 검증 환경

제안한 큐브위성용 자세결정 및 제어시스템 소프트웨어 검증 기법의 유용성을 검증하기 위해 SNUGLITE-II 큐브위성에 탑재되는 자세결정 및 제어시스템의 MILS, SILS, PILS를 각각 수행하였다. 소프트웨어 검증 환경은 다음 Fig. 4에 도시된 바와 같다. 자세결정 및 제어시스템의 소프트웨어 검증은 자세결정, 자세제어 모듈을 각각 검증하게 된다. 각 모듈별로 각각 검증하는 경우 모듈에 대한 시스템 특성 파악이 용이해지므로 소프트웨어의 오류에 대한 디버깅이 수월해진다. 최종적으로 구현된 자세결정 및 제어시스템의 소프트웨어 검증은 동일 입력에 대한 SILS, PILS 결과를 각각 비교함으로써 그 유효성을 확인한다. 이는 본 논문에서 제안하는 검증 기법 절차에서 MILS, SILS 단계의 출력물이 같기 때문이다. 즉 동일한 프로세서와 환경에 대해 모듈 분리를 수행하였으므로 결과물이 일치한다. 따라서 OBC에 탑재되는 자세결정 및 제어 소프트웨어의 PILS 출력과 MATLAB 기반 SILS를 기준값으로 설정하여 출력을 비교한다. 이때 각각의 검증 결과에 대해 활용된 프로세서, 구현 언어는 Table 8에 정리하였다.

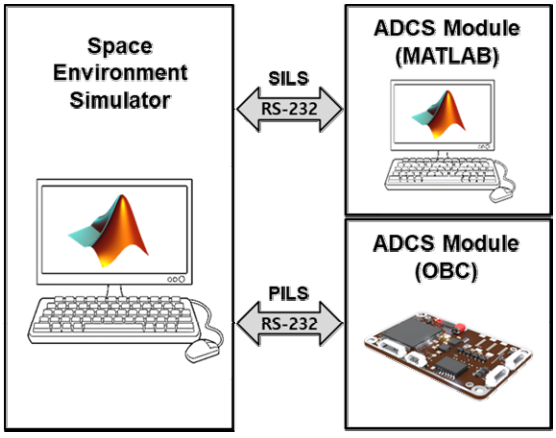

Fig. 4. ADCS software verification environment

Table 8. Comparison of programming languages and processors used for each module

Module	Phase	Language	Processor / RAM
Space Environment Simulator	SILS	MATLAB	Desktop PC (Intel i5-9600KF 3.70GHz) / DDR4 32GB
	PILS		
ADCS	SILS	MATLAB	
	PILS	C	Gomspace A3200 (Atmel AT32UC3C 66MHz) / SDRAM 32MB

3.2 자세결정 소프트웨어 검증 결과

자세결정 소프트웨어를 검증하기 위해 SILS, PILS를 수행한다. 자세결정 소프트웨어는 확장 칼만필터 알고리듬이 탑재되어 메모리 누수에 대한 검증이 면밀히 분석되어야 한다. 따라서 모든 자세결정 모드는 각각 24시간 동안 시뮬레이션이 수행되어 메모리 누수가 관찰된다. 최종 결과물은 Table 4에 도시된 추정 상태변수 비교를 통해 소프트웨어 검증을 수행한다. 본 논문에서는 편의상 추정된 상태변수인 쿼터니언, 각속도, 바이어스 3가지 중 쿼터니언, 각속도만 도시하도록 한다. Table 6에 도시된 MEMS EKF, GPS 자세결정 모드 각각에 대한 소프트웨어 검증 결과는 다음 Figs. 5~12에 도시된 바와 같다.

먼저, Figs. 5~8은 MEMS EKF 자세결정 모드에 대한 소프트웨어 검증 결과를 나타낸다. 소프트웨어 환경에서 센서 측정치의 불확실성을 설정하기 위해 큐브위성의 자세제어 성능이 가장 저하되는 MTQ 자세제어 모드를 우주환경 시뮬레이터에 추가하여 자세결정 모듈의 소프트웨어 검증을 수행하였다. Fig. 5는 MEMS EKF 모드의 추정 쿼터니언, Fig. 7은 각속도에 대한 MATLAB 기반의 SILS, 실시간 RTOS 기반의 PILS 결과를 각각 비교한 결과를 나타낸다. 이에 대한 수치적 오차는 Figs. 6, 8에 도시된 바와 같다. 그림을 확인해 보면 MATLAB을 통해 산출된 SILS 결과와 OBC에서 연산된 PILS의 쿼터니언 그리고 각속도 값의 수치적 오차가 각각 $10^{-10}[-]$, $10^{-8}[\mathrm{deg}/s]$ 수준으로 매우 정밀한 유효 숫자에 대해 소프트웨어가 구현되었음을 확인할 수 있다.

마찬가지로 Figs. 9~12는 GPS 자세결정 모드에 대한 소프트웨어 검증 결과를 나타낸다. GPS 자세결정 모드는 큐브위성의 정상적인 운용을 수행하는 상황을 가정하여 반작용휠을 활용하는 정밀 지구지향 제어 환경에 대한 시뮬레이션을 수행하였다. 시뮬레이션 결과 Figs. 9, 11은 GPS EKF 모드의 추정 쿼터니언, 각속도에 SILS, PILS에 대한 비교 출력을 나타내며 Figs. 10, 12는 각 결과의 수치적 오차를 도시한 결과이다. 그림에서 확인할 수 있는 바와 같이 SILS와 PILS의 쿼터니언 그리고 각속도 값의 수치적 오차가 각각 $10^{-15}[-]$, $10^{-13}[\mathrm{deg}/s]$ 수준을 보이며, 이는 MEMS EKF 자세결정 모드보다 상대적으로 더 정밀하게 소프트웨어가 구현된 것이 확인된다. GPS 자세결정 모드는 MEMS EKF 자세결정 모드와 비교하여 필터의 상태변수의 개수가 적고 연산 부담이 적을 뿐만 아니라 큐브위성의 지구지향 제어 성능이 월등한 반작용휠을 활용한 시뮬레이션 조건을 설정하였으므로 상대적으로 정밀한 결과물이 산출됨을 보인다. 이를 통해 자세결정 모듈이 설계 모델과 동일하게 구현되었으며, 큐브위성 비행 소프트웨어에 탑재될 수 있음을 수치적 오차를 통해 입증할 수 있다.

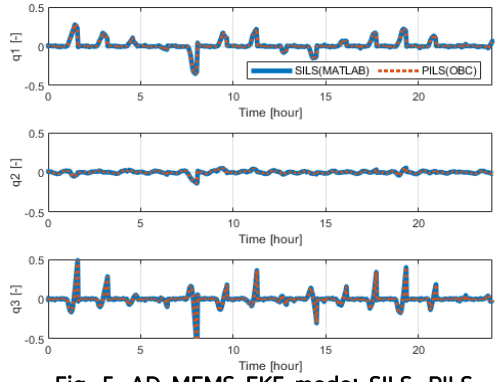

Fig. 5. AD MEMS EKF mode: SILS, PILS estimated quaternion

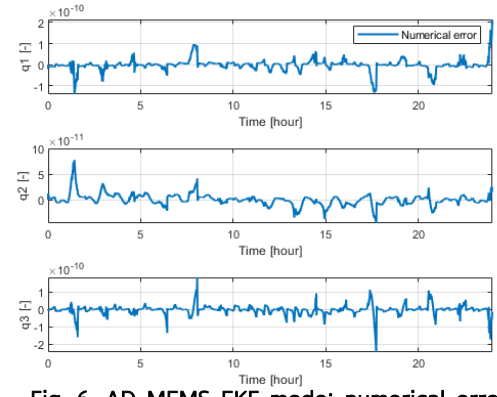

Fig. 6. AD MEMS EKF mode: numerical error in estimated quaternion

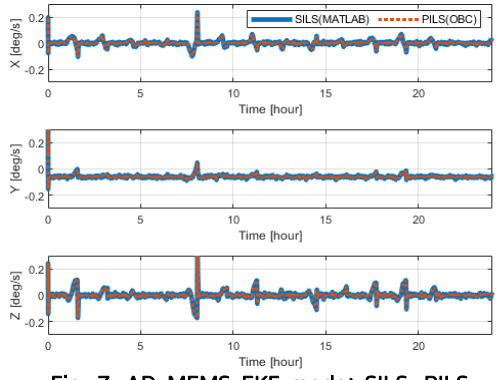

Fig. 7. AD MEMS EKF mode: SILS, PILS estimated angular velocity

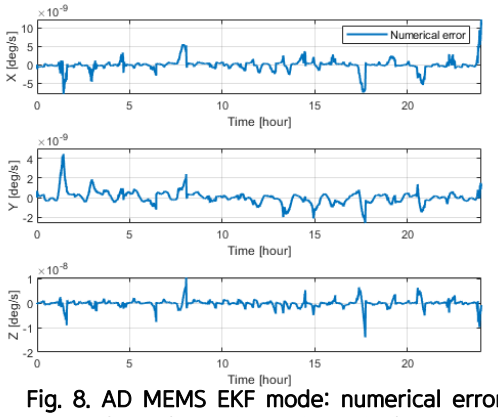

Fig. 8. AD MEMS EKF mode: numerical error in estimated angular velocity

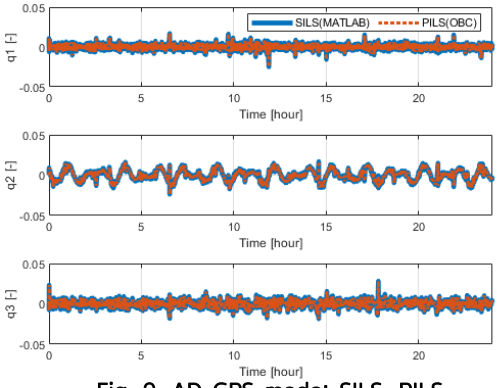

Fig. 9. AD GPS mode: SILS, PILS estimated quaternion

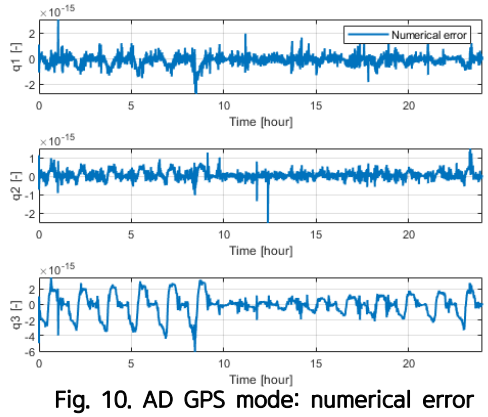

Fig. 10. AD GPS mode: numerical error in estimated quaternion

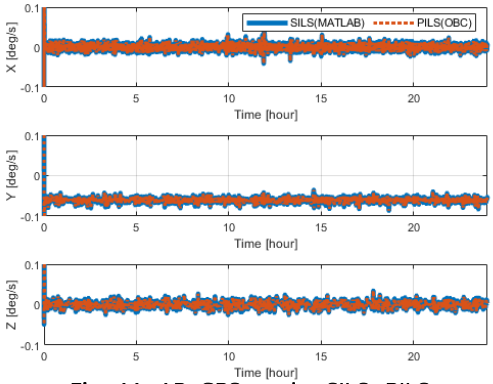

Fig. 11. AD GPS mode: SILS, PILS estimated angular velocity

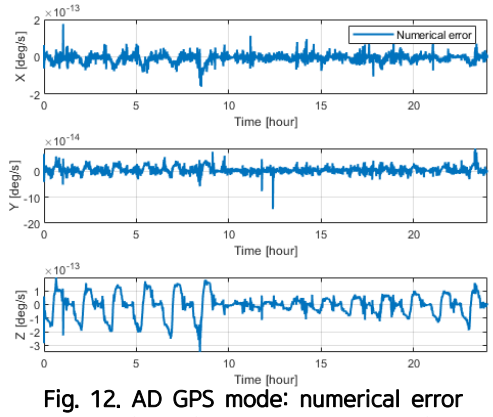

Fig. 12. AD GPS mode: numerical error in estimated angular velocity

3.3 자세제어 소프트웨어 검증 결과

자세제어 모듈의 구현된 소프트웨어를 검증하기 위해 앞서 자세결정 모듈 검증에 적용된 절차를 동일하게 수행한다. 자세제어 소프트웨어는 큐브위성의 위치, 자세결정 모듈로부터 제공되는 상태변수, 그리고 설계된 제어이득을 통해 입력을 산출하는 비교적 단순한 형태의 소프트웨어로 구성된다. 구현된 소프트웨어에 오차 요소로써는 Table 7에 도시된 함수가 있으며 변수의 메모리 정적 할당을 수행하여 메모리 누수가 발생하지 않는 소프트웨어를 구현하였다. 따라서 자세제어 소프트웨어는 Table 6에 기술된 4가지의 제어 모드에 대해 궤도 한 주기 동안의 SILS, PILS를 수행하고 그 정밀도에 대한 분석을 수행한다. 최종 결과물은 Table 4에 도시된 자기토커 및 반작용휠의 제어 입력 비교를 통해 소프트웨어 검증을 수행한다. 본 논문에서는 각 제어 모드의 주요 구동기의 입력에 대한 검증 결과를 도시하도록 한다. Table 6에 도시된 B-dot, Nadir, Traking, MTQ 자세제어 모드 각각에 대한 소프트웨어 검증 결과는 다음 Figs. 13~20에 도시된 바와 같다.

먼저, Figs. 13, 14는 초기 큐브위성의 각속도 감쇠를 위한 B-dot 자세제어 모드의 소프트웨어 검증 결과를 나타낸다. Fig. 13은 B-dot 자세제어 모드에서 산출되는 자기토커의 입력을 MATLAB 기반의 SILS와 실시간 RTOS 기반의 PILS 결과에 대해 각각 비교한 것이다. B-dot 자세제어 모드는 자기장 측정치의 시간 변화율에 대해 단순한 제어 이득이 연산되는 단순한 구조로 구현되어 있다. 따라서 Fig. 14의 수치 오차를 확인하였을 때 SILS, PILS의 산출물이 동일함을 확인할 수 있다.

다음으로 반작용휠을 활용하는 지구지향(Nadir) 및 추적(Tracking) 자세제어 모드의 검증 결과는 Figs. 15~18에 도시된 바와 같다. 두 모드는 소프트웨어적으로 동일한 구조를 택하고 있으나, 시스템 요구사항에 대한 성능을 만족하기 위해 제어기의 이득 값이 다르게 설계되어 있다. Figs. 15, 17의 그림에서 SILS 결과와 PILS 결과가 동일한 결과로 연산되고 있음이 확인되며, 마찬가지로 수치 오차를 확인할 수 있는 Figs. 16, 18에서 $10^{-21}[Nm]$, $10^{-18}[Nm]$ 수준의 수치 오차를 보였다.

마지막으로 반작용휠이 고장이 난 상황을 가정한 자기토커(MTQ) 모드에 대한 검증 결과는 Figs. 19, 20에 도시되었다. 반작용휠을 대체하여 자기토커를 활용하기 위해서는 저궤도에서 큐브위성의 위치에 대해 변화하는 지구 자기장을 고려하여 제어 이득을 산출하여야 한다. 따라서 설계된 가중치에 대한 제어 이득이 가변적으로 변하며 본 논문에서 채택된 LQR 제어 이득을 매초 산출하여야 한다. 따라서 구현된 소프트웨어는 Table 7에서 제시된 LQR 제어이득 산출에 대한 MATLAB의 내장함수와 OBC 구현 수치 함수에 대한 오차 특성으로 반영된다. Fig. 19에서 확인할 수 있는 바와 같이 SILS, PILS 결과의 경향성이 같음을 확인할 수 있다. 하지만 Fig. 20에서 제시된 수치 오차는 $10^{-6}[Am^2]$ 수준으로 다른 자세제어 모드보다 상대적으로 수치 오차가 존재함이 확인된다. 이는 리카티 대수 방정식으로부터 LQR 제어이득을 산출하기 위해 구현된 QR 분해 알고리듬의 최대 반복(iteration)을 100회로 제한한 결과로, OBC의 연산 부담을 최소화하고 소프트웨어의 안정성을 확보하기 위한 결과이다. 발생되는 자기토커의 쌍극자 모멘트에 대한 수치 오차는 지구 자기장과의 외적으로 산출되는 입력 토크의 관점에서 큐브위성의 자세결정 및 제어 성능에는 무시할 만한 수준의 영향을 미친다.

시뮬레이션 결과, 구현된 자세제어 소프트웨어가 설계 모델과 동일함을 수치적 오차를 통해 입증할 수 있었다. 따라서 본 논문에서 제안된 방법을 활용하면 큐브위성의 자세결정 및 제어시스템 소프트웨어의 검증을 위해 일반적인 데스크톱 컴퓨터 환경에서 RS-232 통신이라는 보편적인 방법으로도 효과적으로 실시간 소프트웨어를 구현할 수 있으며, 구현된 코드의 안정성 및 신뢰성을 확보할 수 있음을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 큐브위성용 실시간 자세결정 및 제어시스템의 소프트웨어의 구현과 검증 기법을 제시하였다. 일반적인 교육 목적의 큐브위성 프로젝트의 제약 조건을 고려하여 보편적으로 쉽게 구축이 가능하면서도 효율적이고 명확한 큐브위성 자세결정 및 제어시스템의 소프트웨어 검증 방법을 제안하였다. 이를 위해 먼저 MILS 기반의 자세결정 및 제어시스템의 설계를 수행한 후 설계된 모델을 바탕으로 동역학 모델 및 센서 측정치를 제공하는 우주환경 시뮬레이터, 그리고 소프트웨어 검증 목표 모듈인 자세결정 및 제어시스템을 분리하였다. 분리된 모듈은 MATLAB 기반의 SILS와 Visual Studio 기반의 PILS 그리고 실시간 RTOS의 테스크를 구성한 큐브위성 OBC 상의 PILS를 수행하여 소프트웨어 검증을 수행하였다. 이때 자세결정 및 제어시스템 모듈로 제공되는 데이터는 우주환경 시뮬레이터에서 RS-232 통신을 통해 제공함으로써 실제 하드웨어 데이터를 모사하였다. 이를 통해 SNUGLITE-II 큐브위성에 탑재되는 자세결정 및 제어시스템의 실시간 OBC 기반의 PILS 결과가 MATLAB 기반의 SILS 결과가 동일함을 보였다. 결과적으로 제안된 방법을 활용하면 별도의 시스템을 구축하지 않고도 효과적으로 큐브위성용 자세결정 및 제어시스템의 구현과 검증을 수행할 수 있음을 확인하였다. 제안된 방법은 고가의 장비, 헤리티지, 오픈소스 그리고 별도의 검증 시스템에 의존하지 않고도 일반 데스크톱 컴퓨터만으로도 큐브위성에 탑재되는 소프트웨어 검증을 체계적으로 수행할 수 있다는 특징이 있다. 제안된 소프트웨어 검증 기법을 적용함으로써 교육용 목적의 큐브위성 소프트웨어를 중심으로 다양한 임무를 수행하는 위성 서브시스템 소프트웨어의 구현 및 검증 방법으로 활용될 수 있을 것으로 기대된다.

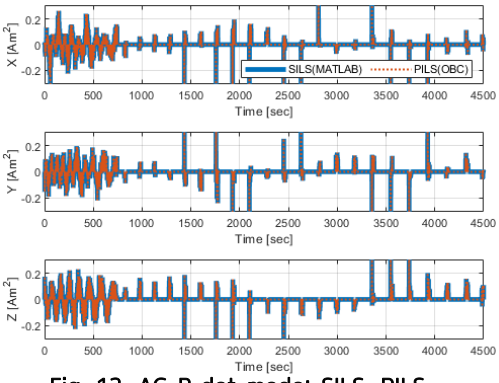

Fig. 13. AC B-dot mode: SILS, PILS dipole moment input

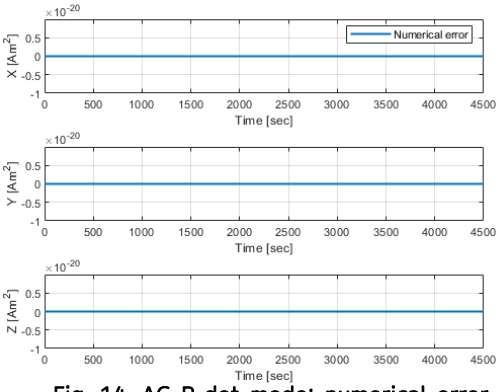

Fig. 14. AC B-dot mode: numerical error in dipole moment input

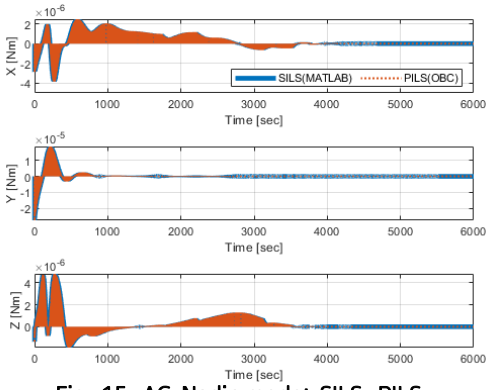

Fig. 15. AC Nadir mode: SILS, PILS reaction wheel input

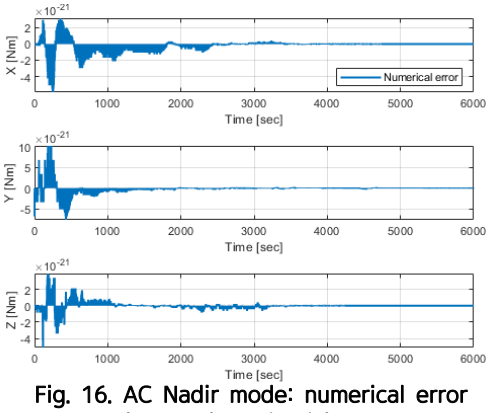

Fig. 16. AC Nadir mode: numerical error in reaction wheel input

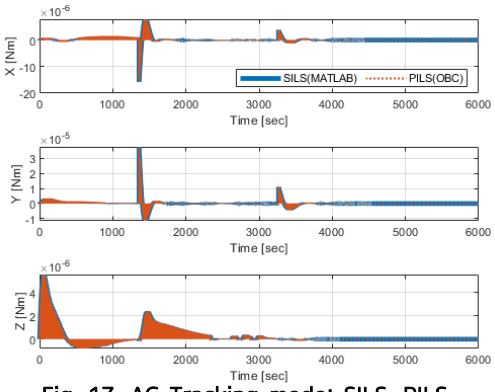

Fig. 17. AC Tracking mode: SILS, PILS reaction wheel input

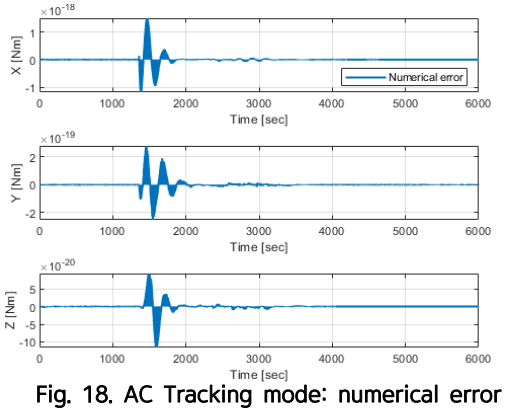

Fig. 18. AC Tracking mode: numerical error in reaction wheel input

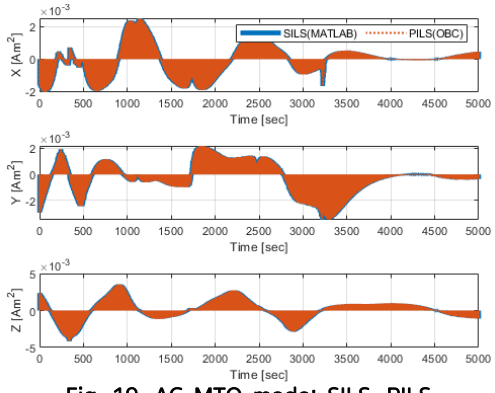

Fig. 19. AC MTQ mode: SILS, PILS dipole moment input

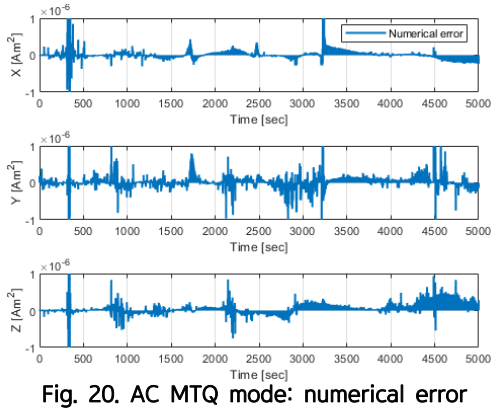

Fig. 20. AC MTQ mode: numerical error in dipole moment input

후 기

본 연구는 과학기술정보통신부 주최 한국항공우주연구원 주관 2022 큐브위성 경연대회의 일환으로 수행되었으며, 이에 관계 기관의 지원에 감사를 드립니다. 그리고 서울대학교 항공우주신기술연구소 및 서울대학교 공학연구원의 시설 지원에도 감사를 드립니다.

References

1) Chin, A., Coelho, R., Nugent, R., Munakata, R. and Puig-Suari, J., "CubeSat: The Pico-Satellite Standard for Research and Education," *Proceeding of The AIAA Space 2008 Conference and Exposition*, September 2008.

2) Poghosyan, A. and Golkar, A., "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, Vol. 88, 2017, pp. 59~83.

3) Bouwmeester, J. and Guo, J. "Survey of Worldwide Pico- and Nanosatellite Missions, Distributions and Subsystem Technology," *Acta Astronautica*, Vol. 67, No. 7-8, 2010, pp. 854~862.

4) Souza, K. V. C. K., Bouslimani, Y. and Ghribi, M., "Flight Software Development for a CubeSat Application," *IEEE Journal on Miniaturation for Air and Space Systems*, Vol. 3, No. 4, 2022, pp. 184~195.

5) Alanazi, A. and Straub, J., "Statistical analysis of cubesat mission failure," *Proceeding of The 32nd Annual AIAA/USU Conference on Small Satellite*, July 2018, pp. 1~8.

6) Jacklin, S. A., "Survey of Verification and Validation Techniques for Small Satellite Software Development," *Proceeding of* The *Space Tech Expo Conference*, May 2015.

7) Park, Y., Koo, C., Park, B., Joo, J. and Choi, J., "Development of VDS for Geosynchronous Satellite and Verification using PILS & HILS," *Journal of the Korean Society for Aeronautical and Space Sciences*, Vol. 34, No. 1, 2006, pp. 103~109.

8) Choi, W., Kim, J. and Kim, H., "A Study on developing Flight Software for Nano-satellite based on NASA CFS," *Journal of the Korean Society for Aeronautical an Space Sciences*, Vol. 44, No. 11, 2016, pp. 997~1005.

9) Park, Y., Han, J., Bang, H. and Han, H., "Development of Real-Time Attitude Control Simulator using Multi-processor : PILS(Processor-In-the-Loop Simulation) System Architecture," *Journal of the Korean Society for Aeronautical and Space Sciences*, Vol. 26, No. 5, 1998, pp. 162~169.

10) Eickhoff, J., Falke, A. and Roser., H., "Model-based design and verification—State of the art from Galileo constellation down to small university satellites," *Acta Astronautica*, Vol. 61, No. 1-6, 2007, pp. 383~390.

11) You, H., Kim, K. and Jung, D., "A Real-Time Simulator for Processor-In-the-Loop Simulation of Small Satellites," *Proceeding of The 2018 International Conference On Control Automation & Information Sciences*, October 2018.

12) Polo, O. R., Esteban, S., Cercos, L., Parra, P. and Angulo, M., "End-to-end validation process for the INTA-Nanosat-1B Attitude Control System," *Acta Astronautica*, Vol. 93, 2014, pp. 94~105.

13) Cho, D., Choi, W., Yoo, P. and Sim, E., "Development Method of Attitude Determination and Control System for Cubesat by using NASA cFS SIL," *Proceeding of The Korean Society for Aeronautical and Space Sciences Fall Conference*, November 2019, pp. 649~650.

14) Morris, J., Zemerick, S., Grubb, M. and Lucas, J., "Simulation-To-Flight (STF-1): A Mission to Enable CubeSat Software-based Validation and Verification," *Proceeding of The AIAA SciTech Forum*, January 2016.

15) Hu, M., Zeng, G., Yao, H. and Tang, Y., "Processor-in-the-loop demonstration of coordination control algorithms for distributed spacecraft," *Proceeding of The 2010 IEEE International Conference on Information and Automation*, June 2010, pp. 1008~1011.

16) Kim, O., Shim, H., Yu, S., Bae, Y., Kee, C., Kim, H., Lee, J., Han, J., Han, S. and Choi, Y., "In-Orbit Results and Attitude Analysis of the SNUGLITE Cube-Satellite," *Applied Sciences*, Vol. 10, No. 7:2507, 2020.