NAZARBAYEV UNIVERSITY

School of Engineering and Digital Sciences

Bachelor of Engineering in
Mechanical and Aerospace Engineering

Design of a Low Earth Orbit Origami Satellite Prototype

(Final Capstone Project Report)

by

Aitdina Turekhanova, Bauyrzhan Shakulov, and
Dias Issabek

Lead Supervisor: Dmitriy Sizov

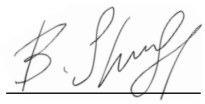
Co-Supervisor: Altay Zhakatayev

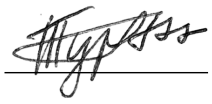
April, 2025

Declaration

We hereby declare that this report entitled “Design of a Low Earth Orbit Origami Satellite Prototype” is the result of our own project work except for quotations and citations that have been duly acknowledged. We also declare that it has not been previously or concurrently submitted for any other degree at Nazarbayev University.

Signatures:

Bauyrzhan Shakulov 

Aitdina Turekhanova 

Dias Issabek 

Date: April 30, 2025

Abstract

The idea of applying origami-inspired design methodologies to satellite design creates a possibility for dealing with the challenges imposed by Low Earth Orbit distance, which is understood to be within the range of 160-2,000 kilometers above the Earth's surface. This project looked into the possibility of creating such a prototype, particularly a 3U CubeSat satellite with integrated aerodynamic stabilization and foldable origami structures.

In this project, a 3D model of the satellite was designed using such advanced modeling tools as SolidWorks and Geogebra, with the numerical simulation of orbital and attitude motion done by means of Newtonian mechanics. Important components for out project includes hinges made from polyamide material and linear actuators chosen for the purpose of folding with strength and precision. The simulations based on our mathematical model have proven that both aerodynamic and magnetic attitude control function and work to maintain the satellite's stability. Furthermore, the PID controller was built in order to dampen the pitch oscillations that are induced by shape transformation during magnetic attitude control.

The results demonstrated that origami-based structures could offer low-cost, scalable solutions for deployable satellites. The limitations of our research include reduced maneuverability and dependence on low-orbit operation. The current research offers the starting point for further development and fine-tuning of an origami-based satellite design.

Contents

List of Figures

List of Tables

Introduction and Problem Statement

It is the pace of development in satellite technology that greatly and continuously enhances our ability to monitor, communicate, and interact with Earth and space. In modern context of satellite technology, Low Earth Orbit (LEO) – the orbital range extending from 160 to 2,000 km above the Earth's surface – becomes increasingly crucial in such applications as Earth observation, telecommunications, and scientific research [1, 2, 3]. The Low Earth Orbit coincides with lower costs of launch due to to proximity to Earth. The communication delays associated with it are also much smaller, and Earth can be watched continuously. However, new ideas for satellites design and launch are needed due to an increase in demand for satellite services.

Recently, Origami, the traditional Japanese art of folding paper, has become a creative solution for the problems associated with spacecraft engineering. The ideas from origami help create structures that are small, flexible, and can fold and unfold accurately [4]. By using origami in satellite technology, engineers are able to create systems that save space and can be easily set up in tough and cramped spatial conditions. Origami-based designs can also help solve many of the problems with large deployable structures such as solar panels and antennas, which are able to fold neatly into compact forms and then unfurl once in orbit. This is crucial for satellites operating in LEO, as space is at the highest points of importance.

The next big step in making smaller satellites is the implementation of CubeSats. These are small satellites that have changed space missions because of their cheap costs and modularity of design [5]. Each CubeSat measures 10x10x10 cm (1U), but each component can be assembled into larger sizes, like the 3U model, for more complex space missions. These satellites are ideal for low orbit applications, as they create cost-effective and small-scale solutions for space research and Earth monitoring. With CubeSat capabilities, major challenges associated with satellite implementation will be resolved, improving aerodynamic stability, operational life, and the flexibility of deployment mechanisms.

Even with modern progress, there is still some limitation to technology with regards

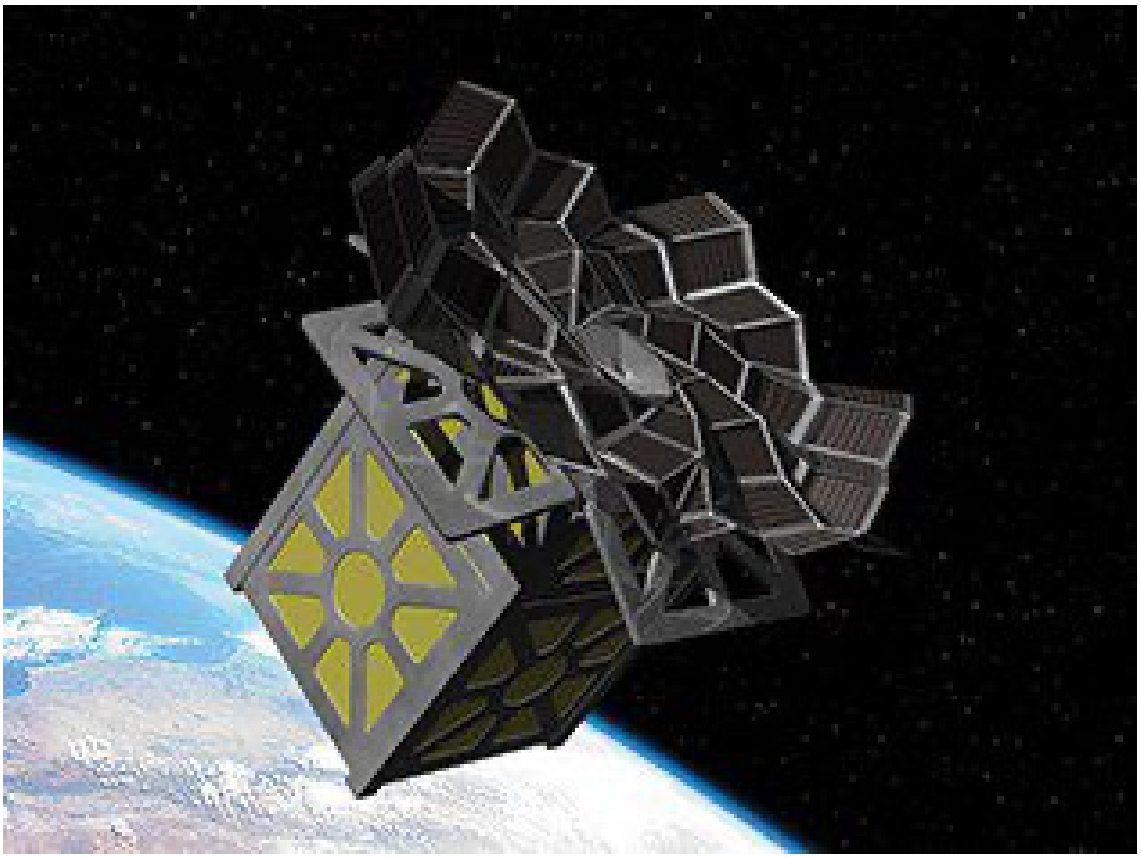


Figure 1: Zirbel's Origami Satellite

to combining the aspects of aerodynamic stabilization and deployable structures [6]. It finds its application with aerodynamic attitude stabilization, since tiny nanosatellites are highly influenced by the density of air molecules within low Earth orbit. Therefore, the developed satellite should be able to maintain stable orientation within space.

This report aims to evaluate the feasibility of a design for a low Earth orbit satellite, incorporating aerodynamic stabilization and an origami-inspired unfolding mechanisms. The origami satellite by Zirgel, as shown in Figure 1 below, demonstrates one of the examples of our prototype [7]. Proposed design plans include creation of a 3D model and associated technical drawings, a mathematical model for the satellite's movement, system for position control to simulate the planned missions, as well as building a physical prototype with a foldable shell.

Only a few studies investigated the idea of utilizing air molecules for aerodynamic stabilization in LEO. However, no solution to the problems regarding reliable deployment mechanisms was given. The origami principle was applied to the proposed design in order to achieve optimum compactness and adaptability toward maximization of stability and functionality in environmental and operational conditions.

Literature Review

The topic of investigation is an aerodynamically stabilized Low Earth Orbit satellite of variable shape. With such a specific topic in mind, the main scope of secondary research was limited to published works with designs of similar structure, and has shown only a number of projects that were developed or proposed. This section of the report will provide the review of existing sources in chronological order of their publication.

An EGG (re-Entry satellite with Gossamer aeroshell and Gps/Iridium) is a 3U CubeSat nanosatellite with a deployable aeroshell [8]. The satellite's deployment consists of the inflation of its shell membrane triggered by an injection of gas. The simplified and variable design makes it a compact and cost effective variant of an orbital satellite, and the shape allows for passive stabilization. However, despite these advantages, this model also possesses important disadvantages that need to be considered before usage. The EGG possesses limited maneuverability due to the structure of its design and quickly loses stability in higher orbits. The other factor of consideration is the relatively quick time of orbital decay that occurs to the satellite due to acting of the aerodynamic force, which also imposes limits on the amount of time the satellite can spend in space. Overall, the successful deployment of the EGG has demonstrated that satellites of similar structure are a feasible new alternative to conventional models.

The "HIBARI" satellite is another microsatellite that utilizes a torque-based attitude control system for orientation [9]. This model is aimed at high-accuracy positioning that is allowed with the usage of a newly developed "Variable Shape Attitude Control (VSAC)", which rotates the solar arrays of the satellite to orient it in the z-axis. The main advantage of this model is the level of active control over the satellite's position and stabilization, which allows for maneuvering, but is quite energy intensive. This, coupled with the model's propensity for mechanical wear, makes "HIBARI" an accurate but energy-inefficient model.

Operation in a very low Earth orbit is considered a challenge due to the high atmospheric density present at such altitudes, which affects the stability and positioning of the satellite. Munoz et al. investigated the issue by analyzing and comparing the performance of two geometric models, the 3-axis control feather configuration and a

2-axis control shuttlecock configuration. The models were tested for maneuverability within simulations of gravitational, magnetic, and aerodynamic torque disturbances. As the results have shown, both configurations possess relative aerodynamic stability and are capable of self-stabilization, but lack range in general maneuverability, unable to perform pointing maneuvers [10]. Although this research demonstrated that operating a spacecraft in a very low orbit is achievable, it has also pointed toward the importance of mathematical modeling and material selection of the satellite.

The current design of the satellite prototype requires 3D printing of several bearing components, which includes hinges to provide connection between plates. Therefore, on the basis of the sources in the field, the material chosen for such hinges is polyamide. This is due to a number of factors associated with the characteristics of the material. Polyamide has been shown to have relatively high resistance to fatigue compared to aramid and photopolymer, and able to maintain its mechanical properties up to 10^6 cycles of folding during testing [11]. This characteristic in particular is useful for the facilitation of repeated folding and unfolding, which will be present in an origami-based satellite.

In conclusion, while secondary research in the field of origami-based aerodynamic stabilized satellites has yielded useful results, it has also shown a general lack of research conducted, especially in relation to 3U CubeSat structures.

Methodology

In order to define the prototype's dimensions, a mathematical model was created using Geogebra's adjustable and easy-to-use 3D software [12]. As a second step, the aerodynamic characteristics of the shape in polar orbit were calculated using Newtonian mechanics of attitude motion in polar orbit. Following this, we designed a PID controller to implement magnetic altitude control. A full 3D assembly of the satellited was modeled using Solidworks 3D Computer Aided Design tool.

3.1 Geogebra 3D to choose satellite dimensions and simulate kinematics

The kinematics of the prototype was observed and adjusted in Geogebra 3D. Geogebra 3D is a flexible and easy to design software, where position and state of the vertices and planes can be controlled and exported from the coordinate system. The kinematics calculation was done mainly on the deployable parts of CubeSat motion. The movement and positioning of foldable panelling and hinging were simulated in GeoGebra so that the function and reliability of the deployment could be guaranteed. Additionally, this simulation data further allowed us to make an aerodynamic simulation for the symmetric folded, symmetric deployed, and asymmetric deployed shapes.

3.2 Newtonian mechanics for simulating attitude motion

Newtonian mechanics is applied to obtain equations of motion for satellite's translational and rotational dynamics in orbit [3]. The external forces and satellite's states given by the equations helped model accurate satellite movement. Considering the external torques and deriving them by the true anomaly, the equation of the circular orbit is derived as following:

$$\theta'' = \alpha(h)C_m(\theta) + \gamma sin2\theta + \mu(\nu)(4sin\nu cos\theta - 2cos\nu sin\theta) \tag{1}$$

In equation (1), the first term is caused by the aerodynamic torque, the second term is caused by the gravitational torque, and the third is caused by the control magnetic torque, where θ is pitch angle, ν is true anomaly, and C_m is the aerodynamic torque coefficient. Shape-dependent C_m coefficients are defined by:

$$C_m = \frac{\alpha_0}{2} + \sum_{i=1}^{7} \alpha_i cos(i\theta) + \sum_{i=1}^{7} b_i sin(i\theta) \tag{2}$$

Aerodynamic Torque ($\alpha(h)C_m(\theta)$): the torque generated by the body and panel surfaces of the satellite;

Gravitational Torque (γsin2θ): torque affecting the stabilization of the satellite;

Magnetic Torque ($\mu(\nu)$): the coefficient of the effect magnetic poles for magnetic control system;

Orbital Effects ($TrueAnomaly, \nu$): the position of the satellite along the orbit (polar in this research).

The aerodynamic simulation of the moment and drag coefficients was applied for three deployment positions: symmetric folded, symmetric deployed, and asymmetric deployed.

3.3 Design of PID controller for magnetic attitude control

To keep satellites in the desired attitude, the use of Proportional-Derivative-Integral (PID) controller to control magnetic torque is needed. According to Gordon et al. [13], the use of magnetorquers is an effective method for LEO satellites. It is related to the usage of the magnetic fields around the Earth, which can be utilized even on the orbit to rotate the satellite body to desired altitude effectively.

To fine PID tune let's consider desired pitch angle as θ_f and actual pitch angle as $\theta(\nu)$, and define error, e(ν), as following:

$$e(\nu) = \theta_f - \theta(\nu) \tag{3}$$

A PID consists of the sum of the scaling of error, of its integral over time and of its derivative.

$$u(\nu) = K_p e(\nu) + K_i \int_0^{\nu} e(t)d_t + K_d \frac{d}{d\nu} e(\nu) \tag{4}$$

For the pitch control of the nanosatellite, using magnetic fields of the Earth, two perpendicular magnetorquers should be used to create torque. They need separate control inputs, which are defined as u_1 and u_2.

$$u_1(\nu) = K_{p_1} e(\nu) + K_{i_1} \int_0^{\nu} e(t)d_t + K_{d_1} \frac{d}{d\nu} e(\nu) \tag{5}$$

$$u_2(\nu) = K_{p_2} e(\nu) + K_{i_2} \int_0^{\nu} e(t) d_t + K_{d_2} \frac{d}{d\nu} e(\nu) \tag{6}$$

So combining torques created by two separate magnetorquers (Eqn. 5 and 6) will change our system equation (Eqn. 1) as following:

$$\begin{aligned} \theta'' = \alpha(h) C_m(\theta) + \gamma sin2\theta + u_1(4sin(\nu)cos(\theta) - 2cos(\nu)sin(\theta)) \\ + u_2(4sin(\nu - \frac{\pi}{2})cos(\theta) - 2cos(\nu - \frac{\pi}{2})sin(\theta)) \end{aligned} \tag{7}$$

3.4 Solid 3D Modeling

3D CAD designs entailed production of specific designs for the satellite, specifically for hinges, rods, and structural plates. These developed models allowed for a proper visualization of the satellite's structure, and helped improved the design from version to version. When the final design was finalized, the digital mock-up of the solution was used to check whether the fits and functions were accurate, which ensured the manufacturability and durability of the 3D printed components and the design. After reconsideration of all aspects, the final assembly was built by demonstrating the best visualization.

Results

4.1 Satellite dimensions

Geogebra 3D modelling provided the understanding of the panels' and actuator's parallel movement in relation to each other. The deployable umbrella shape consists of 16 identical panels, each comprising of short (s), long (s), and connecting (c) rods, as well as 8 length varying actuators (a) which are connected to the short sides of the panels (s rods).

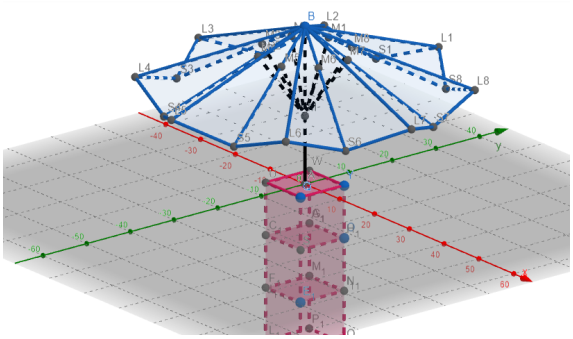


Figure 2: Isometric View of the Geogebra model

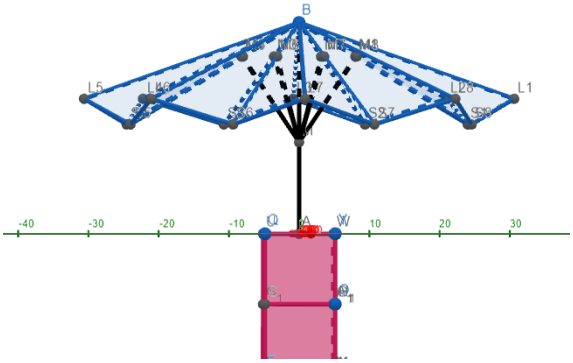


Figure 3: Right View of the Geogebra model

All actuators are connected to one point along the main rod which connects the top of the umbrella to the CubeSat structure. The movement of the short rods is restricted

to a plane parallel to the main rod, while relatively unrestricted movement for the long rods. The resultant coordinates for the nodes and triangular plates, which were used for aerodynamic simulation, are provided within Appendix A. Below are attached fully deployed model dimensions.

Figure 4: CubeSat dimensions in mm

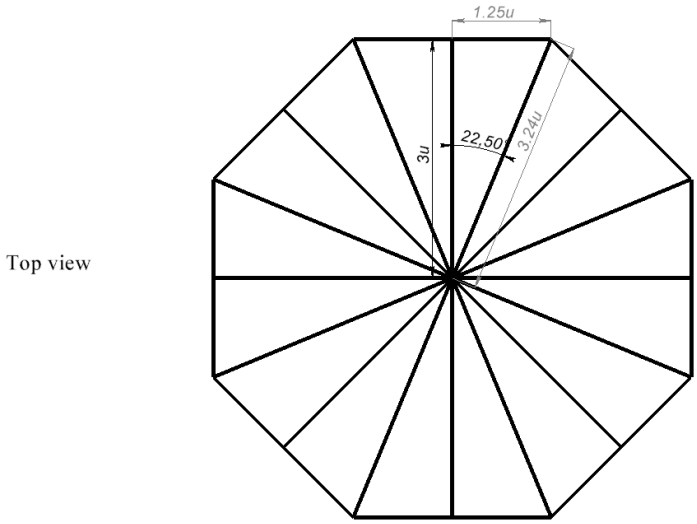


Figure 5: Top View of the technical drawing

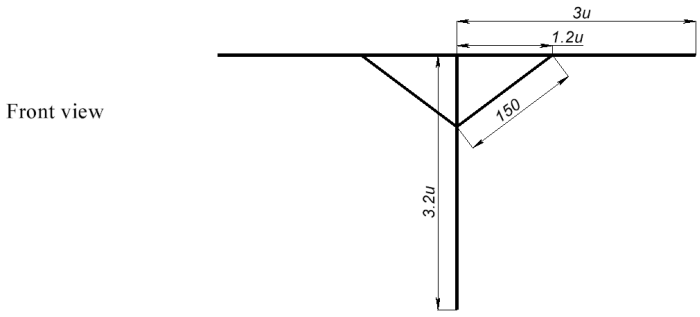


Figure 6: Front View of the technical drawing

4.2 Satellite shapes to be investigated

The nodes given in Appendix A were used to simulate the characteristics of three deployment shapes of the satellite, particularly symmetric deployed, symmetric folded,

and asymmetric deployed. The resultant models are provided below.

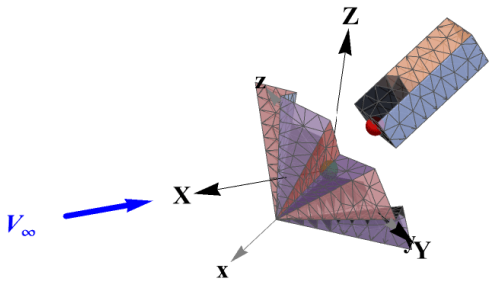


Figure 7: Simulating shape 1

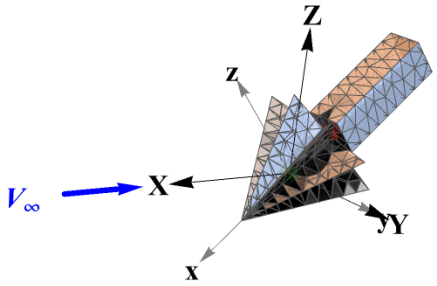


Figure 8: Simulating shape 2

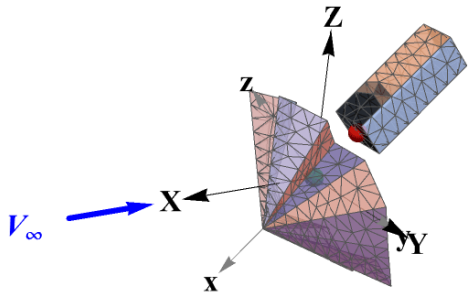


Figure 9: Simulating shape 3

4.3 Aerodynamic characteristics

The aerodynamic characteristics are found by dividing the surface of the body into small flat elements and calculating the pressure and shear stress coefficients for all elements

using Schaaf and Chambre's method[14], except those shielded from the stream by other elements. The sum of the moments of the pressure and shear forces acting on each element over the entire surface gives the moment coefficient C_m. The calculations of the drag and moment coefficients for the following 3 shapes were obtained: symmetric deployed, symmetric folded, and asymmetric deployed. The drag coefficients are shown in Tables 2 and 3, and moment coefficients are given in Tables 4 and 5. These coefficients will be used for Fourier series at Equation 3.

Shape	a_0 **for** C_d	a_0 **for** C_m
Symmetric Deployed	54.76	0
Symmetric Folded	25.02	0
Asymmetric Deployed	61.02	0.134

Table 1: Fourier coefficients a_0 for drag and moment

Shape	**i**						
	1	2	3	4	5	6	7
Symmetric Deployed	0.395	4.396	-0.2234	-1.329	0.09679	1.31	-0.06004
Symmetric Folded	-0.1925	-5.523	-0.1405	-1.324	0.3201	-0.1957	-0.08306
Asymmetric Deployed	0.2687	6.606	0	-0.7844	0.02852	0.6126	-0.09284

Table 2: Fourier coefficients a_i for C_d

Shape	**i**						
	1	2	3	4	5	6	7
Symmetric Deployed	0	0	0	0	0	0	0
Symmetric Folded	0	0	0	0	0	0	0
Asymmetric Deployed	-0.1067	5.903	-0.0482	-1.022	-0.0914	0.6958	0

Table 3: Fourier coefficients b_i for C_d

Shape	**i**						
	1	2	3	4	5	6	7
Symmetric Deployed	0	0	0	0	0	0	0
Symmetric Folded	0	0	0	0	0	0	0
Asymmetric Deployed	5.213	-0.02617	1.258	-0.00997	-0.09907	-0.0446	0.07056

Table 4: Fourier coefficients a_i for C_m

Shape	i						
	1	2	3	4	5	6	7
Symmetric Deployed	-7.203	-0.02843	0.3954	-0.02565	0.7314	0.03299	-0.1382
Symmetric Folded	-7.976	-0.07466	0.6895	0.03107	0.3812	-0.02283	-0.006943
Asymmetric Deployed	-5.89	-0.1004	1.261	0	0.6211	0.01035	-0.1139

Table 5: Fourier coefficients b_i for C_m

4.4 PID controller for satellite magnetic attitude control

To tune the K_p, K_i and K_d parameters, the simulations in MATLAB were run, testing different combinations of K_p, K_i and K_d values. Through iterative testing and analyzing the system's response - such as rise time, settling time, and overshoot - the most efficient gains were identified. All MATLAB codes that were used to find values from Table 6 are provided in Appendix C.

Kp	860
Ki	0
Kd	200

Table 6: Optimal Kp, Ki, Kd values

4.5 Attitude control simulation

Satellite attitude motion simulation for different scenarios were obtained. They were found both with and without magnetic input control. Satellite feedback control was obtained by PID tuning, where we set the satellite to reach 0.765 rad value. Satellite attitude without a magnetic control is affected by the aerodynamics and gravitation. for different altitudes and satellite shapes. Simulation of pitch and control input were built for altitudes of 200, 400, 600, and 800 km, and for panel deployment patterns of symmetric folded, symmetric deployed, and asymmetric deployed.

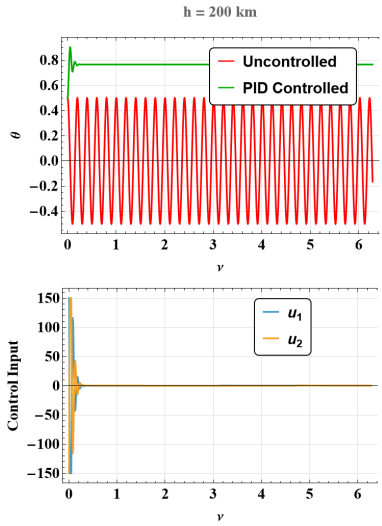


Figure 10: Symmetric Deployed shape at 200 km altitude

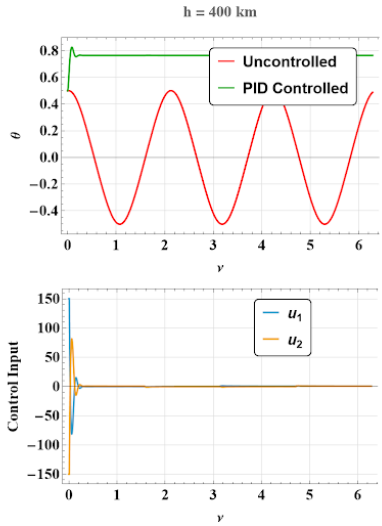


Figure 11: Symmetric Deployed shape at 400 km altitude

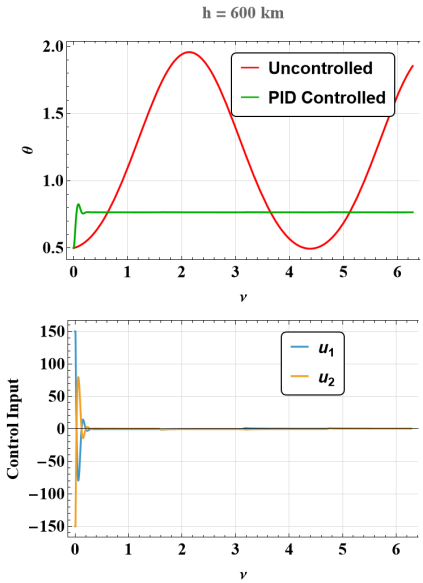


Figure 12: Symmetric Deployed shape at 600 km altitude

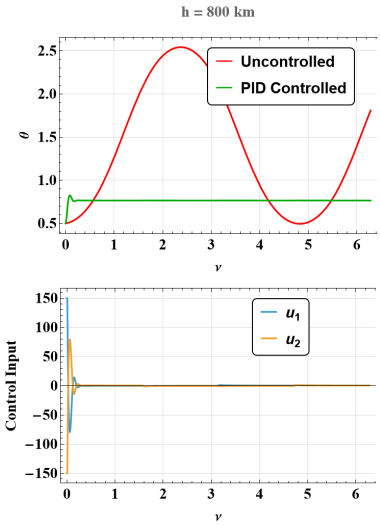


Figure 13: Symmetric Deployed shape at 800 km altitude

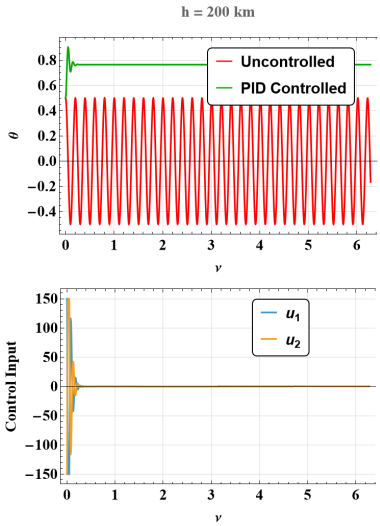


Figure 14: Symmetric Folded shape at 200 km altitude

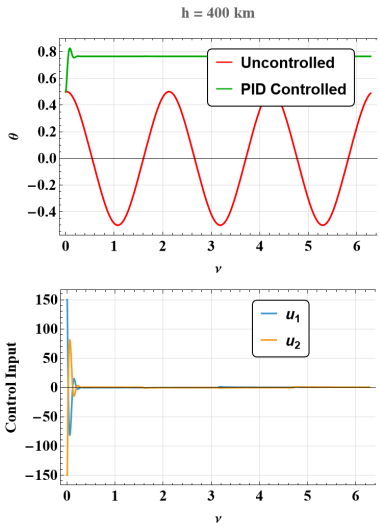


Figure 15: Symmetric Folded shape at 400 km altitude

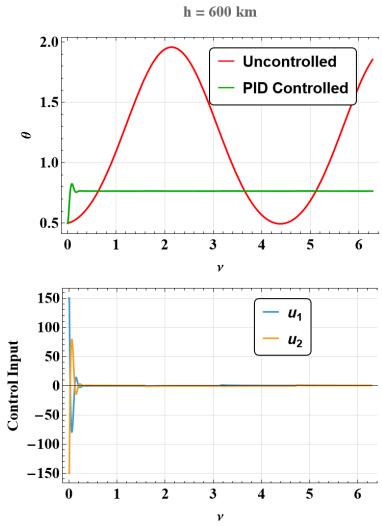


Figure 16: Symmetric Folded shape at 600 km altitude

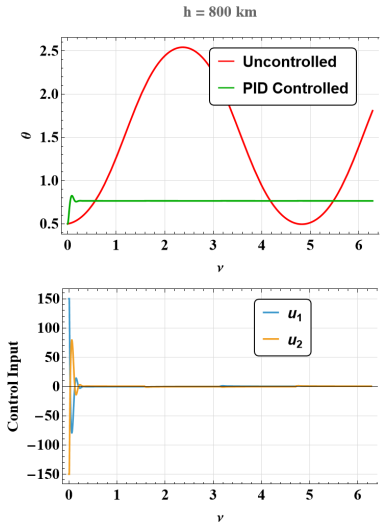


Figure 17: Symmetric Folded shape at 800 km altitude

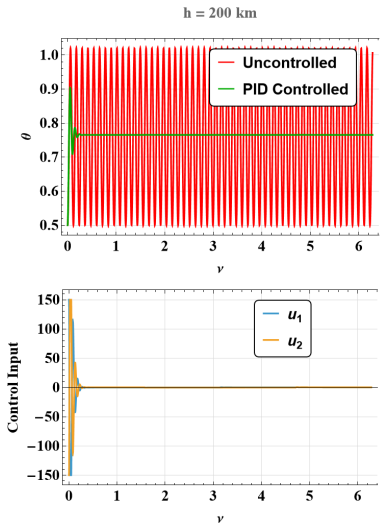


Figure 18: Asymmetric shape at 200 km altitude

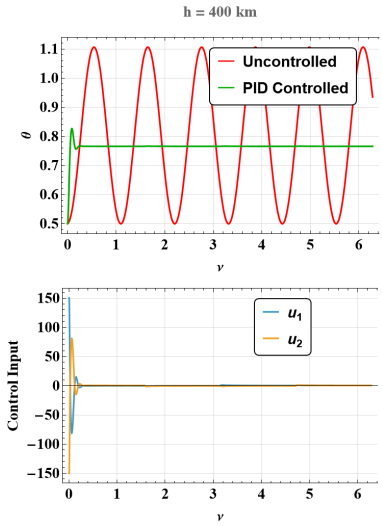


Figure 19: Asymmetric shape at 400 km altitude

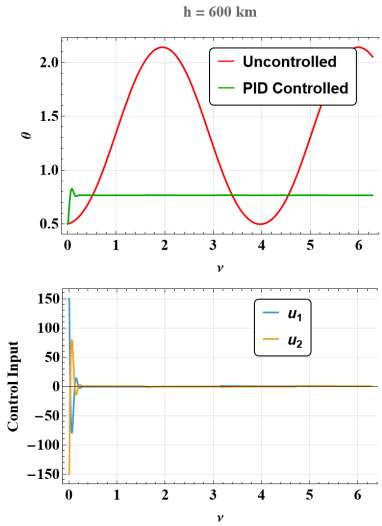


Figure 20: Asymmetric shape at 600 km altitude

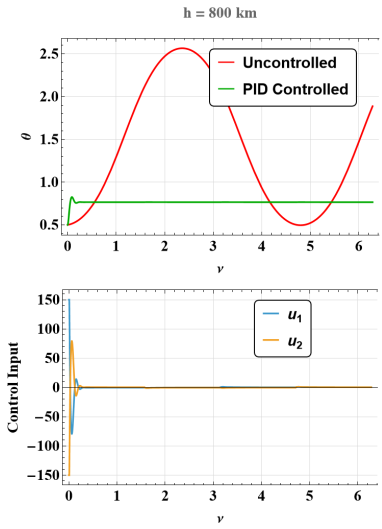


Figure 21: Asymmetric shape at 800 km altitude

Figures 10 and 11 show that the satellite pitch angle is fluctuating around zero when control is not applied. The zero value results from the aerodynamic characteristics of the model, which is very important at 200 km and 400 km, where air particles are dense. However, the air density difference between 200 km and 400 km causes a difference in fluctuation period. Satellites at 200 km fluctuate more frequently. Figure 12 shows that a satellite without PID control at 600 km has a pitch angle fluctuating approximately around 1.25 rad. The center of gravity being located on the CubeSat frame, gravitational force pulls the frame to the planet's center, and it causes a larger gravitational torque compared to the aerodynamic torque. At this altitude air is more rarefied and aerodynamic characteristics are worse. Additionally, air is even more rarefied at 800 km, causing aerodynamic effect to be negligible, which makes the CubeSat body frame tilt, so the satellite will be perpendicular to the Earth surface. According to Figures 10, 11, 12, and 13, applying control for every altitude resulted in the stabilisation of the pitch angle at the desired angle. The u_1 and u_2 plots at h = 200 km are maximum and more frequent at the start of the simulation compared to other altitudes. However, they are almost 0 after stabilisation. According to Figures 14, 15, 16, and 17, it can be observed that symmetrical shapes have the same tendency for uncontrolled cases. The minor difference is in the amplitudes and frequencies of the theta value.

On the other hand, Figures 18 and 19 show that an uncontrolled asymmetric shape does not fluctuate around 0 pitch angle, but around 0.765 rad pitch. This is caused by aerodynamic characteristics of the deployed panels, due to which the dynamic equilibrium angle has shifted. Applying control in these cases requires only small control input values compared to the symmetric cases. Although the satellite is fluctuating at 800 km, it fluctuates around half of a Pi angle, trying to be perpendicular to the Earth surface.

To sum up, asymmetric shape arrangement provides a change of aerodynamic characteristics. It will be used to shift the zero moment of pressure coefficient, C_m, position of the satellite shape to any desired pitch angle. E.g., the asymmetric shape used in our simulations shifted its zero C_m to pitch angle 0.765 rad (was found numerically, Appendix B), so simulations provided in Figures 18, 19, 20, and 21 provide smaller control input interference. Comparing them with control inputs of symmetric shapes, where C_m= 0 is at zero pitch angle, the case for symmetric shapes is more intense at the start. It proves that arranging deployable plates to move C_m= 0 for desired attitude angle will lead to smaller control input values by a magnetic torque control.

4.6 3D CAD Model

The main rod within the 3D CAD model provides connection between the upper plating and the 3U CubeSat. Therefore, this rod is considered to be the axis of symmetry for the whole model. Around the rod, the sixteen plates were connected with hinges, 2

on long and 3 on short sides. This was needed to provide more degrees of freedom for the overall plate movement, while ensuring the connection to the main rod was steady at the point. The telescopic linear actuators were used as length changin rods. Lower down the main rod, the main node bearing provides a connection between the actuators and the plates and their hinges. Additionally, the main rod adds possibility of linear movement, and can be used to move the overall rod with the ball bearings. The full front view of the whole model is provided below.

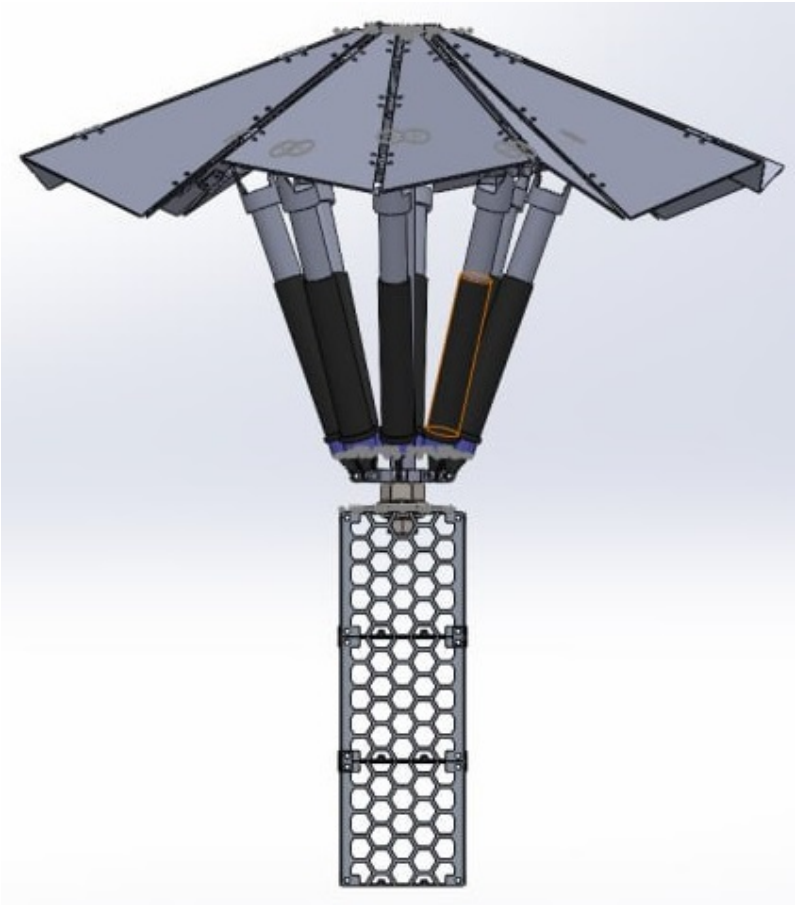


Figure 22: Front view of the Assembly

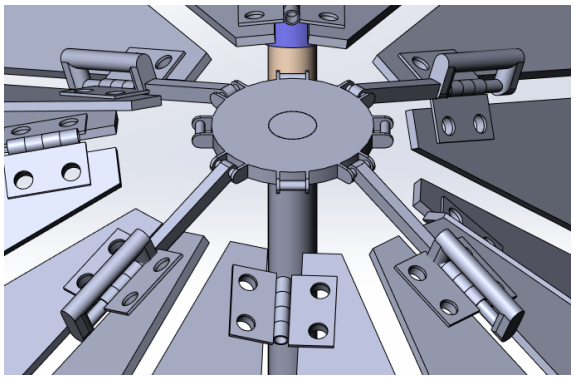


Figure 23: Hinges and Main Rod top connection

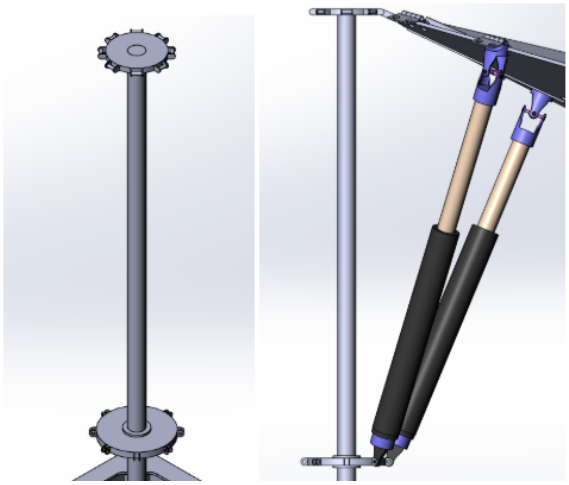


Figure 24: Main rod in relation to Actuators

4.7 Electric control configuration

In this prototype, the eight linear actuators will be controlled, each driven by a DC motor, using an Arduino Uno R3 microcontroller and L298N motor drivers. Due to hardware constraints, the main focus was solely on directional control (forward/reverse) rather than speed or torque modulation. This approach simplifies the system while still achieving reliable actuation for deployment of the panels.

Component	**Quantity**
Actuator	8
Arduino Uno R3	1
Motor Driver L298N	4
Breadboard	1
Jumper Wires	30

Table 7: Electrical components

Since the Arduino has 16 digital I/O pins, two pins per motor (one for each direction) were allocated. This ensures full bidirectional control without requiring PWM (pulse-width modulation) for speed regulation. The wiring can be observed below.

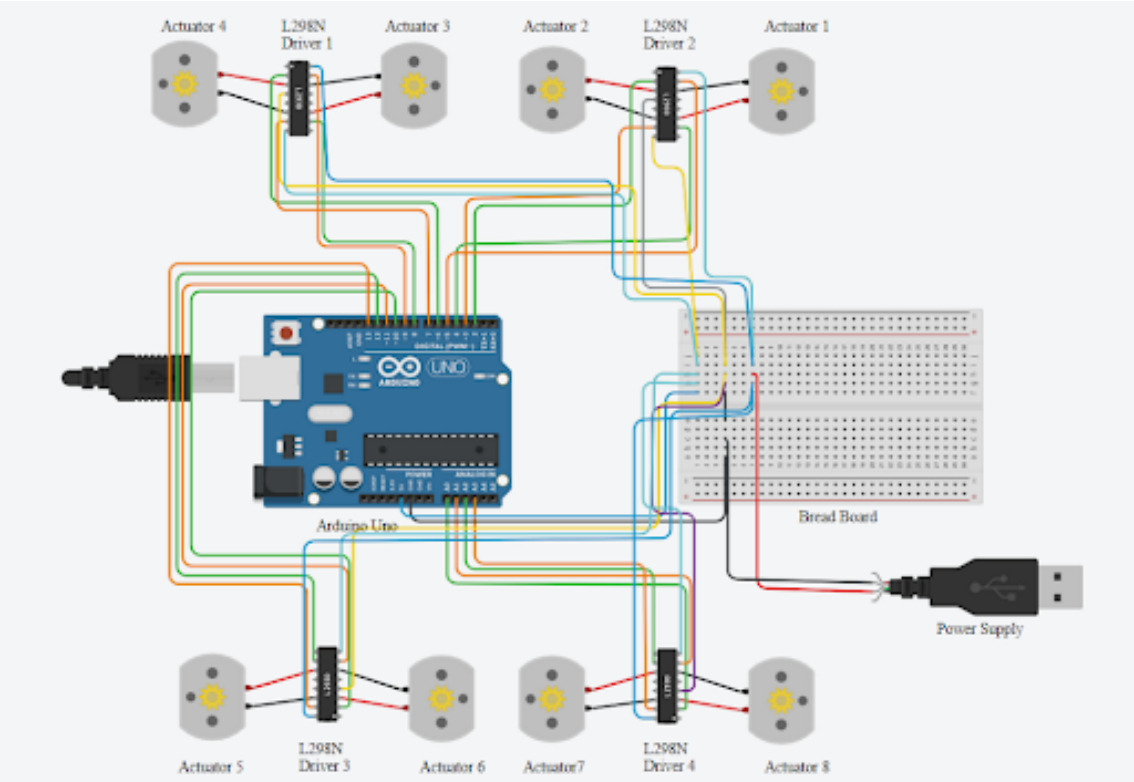


Figure 25: Electrical wiring scheme of the system

L298N Driver	Arduino Pin	Actuator Controls
Driver 1 IN1	D2	Actuator 1 Direction A
Driver 1 IN2	D3	Actuator 1 Direction B
Driver 1 IN3	D4	Actuator 2 Direction A
Driver 1 IN4	D5	Actuator 2 Direction B
Driver 2 IN1	D6	Actuator 3 Direction A
Driver 2 IN2	D7	Actuator 3 Direction B
Driver 2 IN3	D8	Actuator 4 Direction A
Driver 2 IN4	D9	Actuator 4 Direction B
Driver 3 IN1	D10	Actuator 5 Direction A
Driver 3 IN2	D11	Actuator 5 Direction B
Driver 3 IN3	D12	Actuator 6 Direction A
Driver 3 IN4	D13	Actuator 6 Direction B
Driver 4 IN1	A0	Actuator 7 Direction A
Driver 4 IN2	A1	Actuator 7 Direction B
Driver 4 IN3	A2	Actuator 8 Direction A
Driver 4 IN4	A3	Actuator 8 Direction B

Table 8: Electrical wiring of the system

4.8 Technical Drawings

Our 3D model uses not only the standard engineering components such as stopper rings, nuts, and screws, but also unique parts that were designed specifically for the project. These parts include the main rod, top node, main node, top connector, hook, cross, and cube connector. All aforementioned pieces have technical drawings that we created in order to facilitate future reproducibility, assembly, and manufacturing of our satellite prototype in the future. The drawing of the complete assembly with all unique and standard components in place has also been created. We have specified all the necessary dimensions, as well as tolerances and fits, and ensured the drawings are up to standard. The full assembly drawing, as well as all the designed part drawings, are provided within Appendix D. The bill of materials including all parts, their quantity, and materials, are also provided as a table within Appendix D.

4.9 Physical Prototype

The physical prototype was assembled in two stages: first, we ordered the necessary components and materials, and then we have fit every part together. Standard parts such as hex nuts and bolts, as well as linear actuators and Arduino plates were ordered from online manufacturers. At the same time, the plates and designed parts (e.g., main rod and top connectors for the plates) were 3D printed using PET-G material. The assembly was done first by connecting the plates together with hinges and the top node. Then, the main node was joined with the main rod by threading. The eight actuators were connected using the hook and the cross parts to ensure connection with the plates. Additionally, in order to ensure every place stayed locked in place, we applied nuts to prevent unscrewing and added stopper rings and locking wire to our design. The photo results of our assembly can be seen below.

Figure 26: Symmetric Deployed

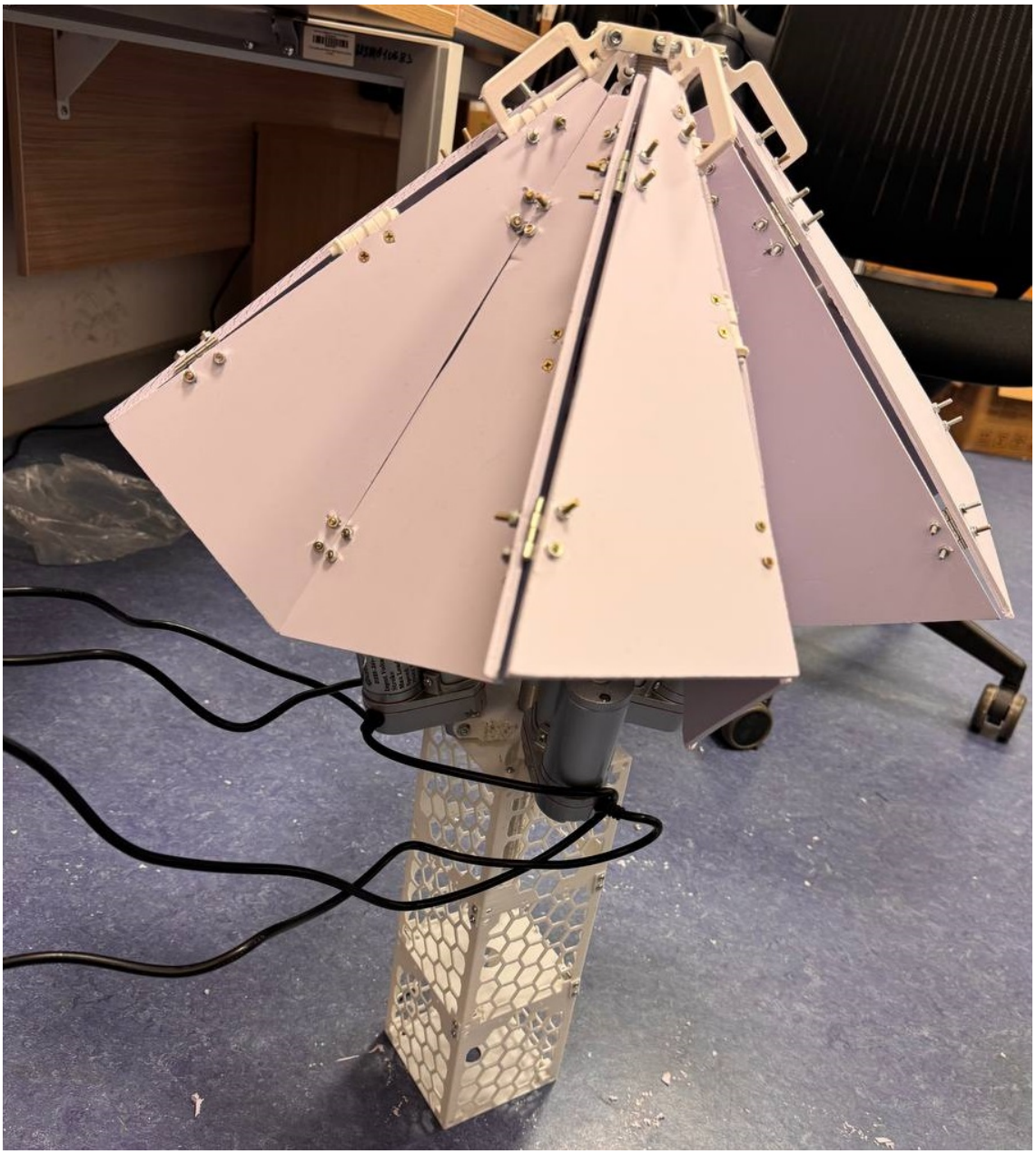


Figure 27: Symmetric Folded

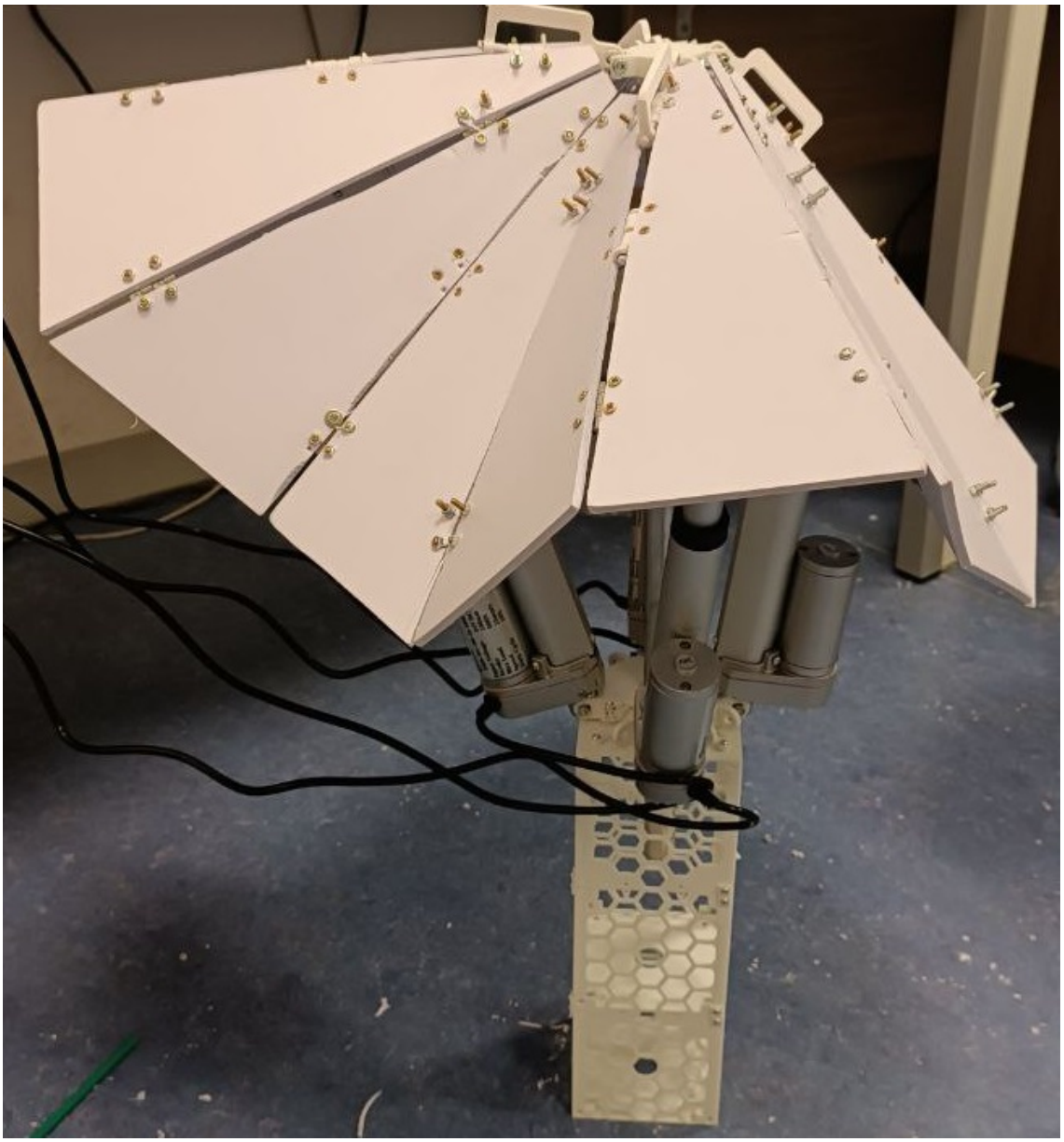


Figure 28: Asymmetric Deployed

Limitations

Considering the project's focus on feasibility yields that several aspects are outside of scope of the design, testing, and implementation.

First, the simulations and calculations only consider circular orbits, which yields that non-circular orbits such as elliptical and parabolic are omitted completely. Moreover, the satellite only has Pitch motion, with Roll and Yaw motions beyond the scope of the current prototype.

Second, the final prototype functions only as a proof of concept for an origami-based Low Earth Orbit satellite and is not designated to physically fly. The physical model is made to demonstrate the folding and unfolding motion by linear actuators, and ignores full satellite deployment and the following costs of real life implementation. The physical prototype also has a control system implemented, but only simulates attitude control. Furthermore, the reliability of the satellite, beyond the structural integrity of its mechanical components, was not be factored during design.

Conclusions

The "Design of Low Earth Orbit Origami Satellite Prototype" project initially received quite a significant development impetus, thereby building a very solid ground for its subsequent work. The present report provides a consolidation of ideas on aerodynamic stabilisation integrated with principles of origami toward the solution of some of the very key problems in miniaturized satellite systems, especially in LEO.

This is where the project presents detailed 3D models developed with SolidWorks and Geogebra to date, helping visualize in depth the structure and mechanisms proposed for the satellite. These models have been critical tools in the understanding of the design feasibility and functionality. Additionally, there has been a deep literature review that allows the development of precious insight on current technologies and methodologies about aerodynamic stability, origami-based deployable structures, and CubeSat design.

Preliminary test results show a promising merging of aerodynamics with the principles of origami to build compact, cost-effective, adaptive satellite solutions. Initial model and analysis studies point towards improvements, especially in the realms of stability and the efficiency of the deployment process. It was shown by simulations, that arranging umbrella shaped control surfaces to use aerodynamics and moment of pressure led to smaller interference of magnetorquers.

The pre-mathematical modeling was fully completed, together with simulations in order to perform the refinement of the control system, the fabrication of a functional prototype with foldable shells working. These steps were crucial in the process of validating the design concept and assessing its practicality against realistic applications. In all, though still a project, the results so far obtained show clear directions to the realization of the project objectives. Fresh ideas, modern powerful modeling tools, and professional research form a very good ground for successful completion.

Going forward, this might just be what will change the face of satellite engineering and lead to further research into origami-inspired aerospace innovations.

Contributions

The project was done by the team, and each member contributed in their way to ensure that development goes well. The contributions are as follows:

- Aitdina Turekhanova made a great deal of literature review, which is the background for better understanding the research and technologies already in existence related to this project. The other task of Aitdina was to do the selection of material for this particular process. The other major and important responsibility taken by Aitdina in the making of mathematical modeling was perhaps the main role of all, where it helps determine the theory behind the whole working mechanism. Additionally, she did the technical and assembly drawings of the designed parts.

- Dias Issabek played a key role in 3D modeling of the satellite in SolidWorks, allowing the creation of a very detailed and accurate representation of the system's main components. He also used GeoGebra for simulations and mathematical modeling, which had contributed to the optimization of the design and verified the numerical aspect of the system. He simulated the satellite movement on polar orbit and built electronics to control linear actuator movements.

- Bauyrzhan Shakulov contributed to the pre-design phase, making the first sketches and conceptual designs on paper that allowed us to better envision what the project was aiming for. Bauyrzhan also worked with GeoGebra in order to assist the mathematical simulations. After finalizing the 3D Model he was working on the 3D printing and collecting all of the materials that had to be buyed. In addition, he was in charge of controlling the provision of the project, ensuring timely completion of all tasks and proper management of resources.

Each member of the team was important in the development of the project, and their combined efforts allowed for the comprehensive development of the system.

Bibliography

[1] NASA, "Commercial space frequently asked questions," April 2024.

[2] H. D. Curtis, *Orbital mechanics for engineering students*. Butterworth-Heinemann, 2019.

[3] Y. Dong, M. Humi, and Z. Zhang, "Satellite orbits and fractional mechanics," *The Romanian Journal of Technical Sciences. Applied Mechanics.*, vol. 65, no. 3, pp. 183–198, 2020.

[4] S. Yue, "A review of origami-based deployable structures in aerospace engineering," in *Journal of Physics: Conference Series*, IOP Publishing, 2023.

[5] I. Siddique, "Small satellites: revolutionizing space exploration and earth observation," *European Journal of Advances in Engineering and Technology*, vol. 11, no. 3, pp. 118–124, 2024.

[6] S. A. Rawashdeh, J. E. Lumpp, *et al.*, "Aerodynamic stability for cubesats at iss orbit," *Journal of Small Satellites*, vol. 2, no. 1, pp. 85–104, 2013.

[7] A. K. Fronk, "Origami in space," Oct 2021.

[8] Y. Nagata, K. Yamada, K. Suzuki, *et al.*, "Flight demonstration of telecommunication system for satellite using iridium satellite communication [j]," *Nihon Kōkū Uchū Gakkai ronbunshū= Journal of the Japan Society for Aeronautical and Space Sciences*, vol. 67, no. 1, pp. 1–8, 2019.

[9] K. Watanabe, Y. Kikuya, Y. Shintani, K. Sasaki, H. Ando, T. Nakashima, K. Miyamoto, K. Matsubara, S. Matunaga, and Y. Yatsu, "Concept design and development of 30kg microsatellite hibari for demonstration of variable shape attitude control," 2019.

[10] V. C. Muñoz, D. González, J. Becedas, R. Dominguez, P. Roberts, N. Crisp, V. T. A. Oiko, S. Edmondson, S. Worrall, S. Haigh, *et al.*, "Attitude control for satellites flying in vleo using aerodynamic surfaces," *JBIS-Journal of the British Interplanetary Society*, vol. 73, no. 3, pp. 103–112, 2020.

[11] M. A. Wagner, J.-L. Huang, P. Okle, J. Paik, and R. Spolenak, “Hinges for origami-inspired structures by multimaterial additive manufacturing,” *Materials & Design*, vol. 191, p. 108643, 2020.

[12] G. GmbH, “Geogebra 3d calculator,” 2001–2025.

[13] R. Gordon, A. Leibak, P. Org, E. Priidel, A. Rassõlkin, and T. Vaimann, “Adcs development for student cubesat satellites – taltech case study,” *Proceedings of the Estonian Academy of Sciences*, vol. 70, p. 268, 08 2021.

[14] V. S. Aslanov and D. A. Sizov, “Chaotic pitch motion of an aerodynamically stabilized magnetic satellite in polar orbits,” *Chaos, Solitons and Fractals*, vol. 164, November 2022.

Appendices

Appendix A. Model Coordinates

Triangle no.	Vertices	Rectangle no.	Vertices
1	(B, S5, L5)	1	(V1, V2, V3, V4)
2	(B, L6, S5)	2	(V1, V2, V5, V6)
3	(B, L6, S6)	3	(V2, V3, V6, V7)
4	(B, S6, L7)	4	(V3, V4, V7, V8)
5	(B, S7, L7)	5	(V4, V1, V8, V5)
6	(B, S7, L8)	6	(V5, V6, V7, V8)
7	(B, L8, S8)		
8	(B, S8, L1)		
9	(B, S1, L1)		
10	(B, L2, S1)		
11	(B, S2, L2)		
12	(B, L3, S2)		
13	(B, S3, L3)		
14	(B, L4, S3)		
15	(B, S4, L4)		
16	(B, S4, L5)		
Triangular plates		Rectangular CubeSat sides	

Figure 29: Aerodynamic Surfaces

	x	y	z
S1	9.06	-7.4	17.86
S2	9.06	-17.86	7.4
S3	9.06	-17.86	-7.4
S4	9.06	-7.4	-17.86
S5	9.06	7.4	-17.86
S6	9.06	17.86	-7.4
S7	9.06	17.86	7.4
S8	9.06	7.4	17.86
L1	13.6	0	26.76
L2	13.6	-18.92	18.92
L3	13.6	-26.76	0
L4	13.6	-18.92	-18.92
L5	13.6	0	-26.76
L6	13.6	18.92	-18.92
L7	13.6	26.76	0
L8	13.6	18.92	18.92
B	32	0	0
V1	0	-5	-5
V2	0	-5	5
V3	0	5	5
V4	0	5	-5
V5	-30	-5	-5
V6	-30	-5	5
V7	-30	5	5
V8	-30	5	-5

Figure 30: Data for medium Symmetric Model

	x	y	z
S1	2.34	-1.72	4.16
S2	2.34	-4.16	1.72
S3	2.34	-4.16	-1.72
S4	2.34	-1.72	-4.16
S5	2.34	1.72	-4.16
S6	2.34	4.16	-1.72
S7	2.34	4.16	1.72
S8	2.34	1.72	4.16
L1	3.95	0	16.37
L2	3.95	-11.57	11.57
L3	3.95	-16.37	0
L4	3.95	-11.57	-11.57
L5	3.95	0	-16.37
L6	3.95	11.57	-11.57
L7	3.95	16.37	0
L8	3.95	11.57	11.57
B	32	0	0
V1	0	-5	-5
V2	0	-5	5
V3	0	5	5
V4	0	5	-5
V5	-30	-5	-5
V6	-30	-5	5
V7	-30	5	5
V8	-30	5	-5

Figure 31: Data for folded Model

	x	y	z
S1	6.52	-6.06	14.63
S2	7.08	-15.43	6.39
S3	16.71	-23.84	-9.88
S4	16.71	-9.88	-23.84
S5	16.71	9.88	-23.84
S6	16.71	23.84	-9.88
S7	7.08	15.43	6.39
S8	6.52	6.06	14.63
L1	10.81	0	24.6
L2	11.1	-17.1	18.03
L3	12.19	-25.68	1.56
L4	20.5	-21.47	-21.47
L5	20.5	0	-30.37
L6	20.5	21.47	-21.47
L7	12.19	25.68	1.56
L8	11.1	17.1	18.03
B	32	0	0
V1	0	-5	-5
V2	0	-5	5
V3	0	5	5
V4	0	5	-5
V5	-30	-5	-5
V6	-30	-5	5
V7	-30	5	5
V8	-30	5	-5

Figure 32: Data for fully Symmetric Model

Appendix B. Mathematica Code

```
In[95]:= ClearAll["Global‘*"]
In[96]:= R=6378000;
\[Mu]o=1.256*10^(-6);
\[Mu]m=7.7*10^22;
\[Mu]G=3.986*10^14;
In[100]:= A=0.01;
L=0.3;
R=6378000;
Temporary:
In[103]:= Jt=0.08; (*kg*m^2*)
Jl=0.04; (*kg*m^2*)
Cm Coefficent definition based on theta and deployment type.
In[105]:= (*Define the Cm matrices*)
Cm0={{0},{0},{0.134}};
Cmalist={{0,0,0,0,0,0,0},{0,0,0,0,0,0,0},
{5.213, -0.02617,1.258,-0.00997,-0.09907,-0.0446,0.07056}};
Cmblist={{-7.203,-00.02843,0.3954,-0.02565,0.7314,0.03299,-0.1382},
{-7.976,-0.07466,0.6895,0.03107,0.3812,-0.02283,-0.006943},
{-5.89,-0.1004,1.261,0,0.06211,0.01035,-0.1139}};

In[109]:= ClearAll[Cm]
Cm[\[Theta]_,i_]:=Part[Cm0[[i]]+ Total[Cmalist[[i]]
*Cos[\Theta]*Range[Length[Cmalist[[i]]]]]]+Total[Cmblist[[i]]
*Sin[\[Theta]*Range[Length[Cmblist[[i]]]]],1]
In[111]:= trend=x/.FindRoot[Cm[x,3]==0,{x,.8}]
Out[111]= 0.765097
Alpha Calculation on height
In[112]:= ClearAll[rho]
rho[h_]:=StandardAtmosphereData[Quantity[h,"Kilometers"],
```

```
"Density"]
In[114]:= ClearAll[alpha]
alpha[h_]:=QuantityMagnitude[(A*L*(1000*h+R)^2*rho[h])/(2*Jt)]
Gamma calculation
In[116]:= gammaeq[Jt_,Jl_]:=3*(Jt-Jl)/(2Jl);
gamma=gammaeq[Jt,Jl];
GENERAL SOLUTION
In[118]:= ClearAll@quad
quad[t_]:=Mod[Floor[t/(Pi/2)],4]
In[120]:= Umax=150; Umin=-150;
In[121]:= ClearAll[kp1, kp2,kd1,kd2]
kp1[t_]:=860*If[(quad[t]==0) ||(quad[t]==1),1,-1]
kp2[t_]:=860*If[(quad[t]==0) ||(quad[t]==3),-1,1]
kd1[t_]:=20*If[(quad[t]==0) ||(quad[t]==1),1,-1]
kd2[t_]:=20*If[(quad[t]==0) ||(quad[t]==3),-1,1]
In[126]:= ClearAll[U1,U2,u1,u2, Ucoef]
Ucoef[h_]:=(\[Mu]o*\[Mu]m)/(8*Pi*\[Mu]G (h+R)^3)
U1[t_, \[Theta]_,\[Theta]d_,setpoint_]:=If[(kp1[t]*(setpoint-\[Theta])(*+ki*i
U2[t_, \[Theta]_,\[Theta]d_,setpoint_]:=If[(kp2[t]*(setpoint-\[Theta])(*+ki*i
u1[t_,setpoint_]:=U1[t,\[Theta][t],\[Theta]'[t],setpoint]
u2[t_,setpoint_]:=U2[t,\[Theta][t],\[Theta]'[t],setpoint]
In[132]:= ClearAll[it]
it[t_]:=1*If[t<2Pi/3,1,If[t<1.5Pi,3,2]]
In[135]:= ClearAll@solPID
solPID[h_, i_,{\[Theta]0_,\[Theta]d0_},{tin_,tout_},setpoint0_]:=NDSolve[{\[T
\[Theta][tin]==\[Theta]0,\[Theta]'[tin]==\[Theta]d0},{\[Theta][t],\[Theta]'[t
In[137]:= ClearAll@sol
sol[h_, i_,{\[Theta]0_,\[Theta]d0_},{tin_,tout_},setpoint0_]:=NDSolve[{\[Thet
\[Theta][tin]==\[Theta]0,\[Theta]'[tin]==\[Theta]d0},{\[Theta][t],\[Theta]'[t
In[139]:=
Symmetric folded
In[140]:= tin=0;
tout=2Pi;
\[Theta]0=0.5;
\[Theta]d0=.1;
desired=trend(*0.7*);
In[145]:= (*  HEIGHT = 200 KM*)
sol200s1=sol[200,1,{0.5,\[Theta]d0},{tin,tout}, desired];
sol200PIDs1=solPID[200,1,{0.5,\[Theta]d0},{tin,tout}, desired];
```

```
In[147]:= (*  HEIGHT = 400 KM*)
sol400s1=sol[400,1,{0.5,\[Theta]d0},{tin,tout},desired];
sol400PIDs1=solPID[400,1,{0.5,\[Theta]d0},{tin,tout},desired];
In[149]:= (*  HEIGHT = 600 KM*)
sol600s1=sol[600,1,{0.5,\[Theta]d0},{tin,tout}, desired];
sol600PIDs1=solPID[600,1,{0.5,\[Theta]d0},{tin,tout}, desired];
In[151]:= (*  HEIGHT = 800 KM*)
sol800s1=sol[800,1,{0.5,\[Theta]d0},{tin,tout}, desired];
sol800PIDs1=solPID[800,1,{0.5,\[Theta]d0},{tin,tout}, desired];
Symmetric deployed
In[153]:= (*  HEIGHT = 200 KM*)
sol200s2=sol[200,2,{0.5,\[Theta]d0},{tin,tout}, desired];
sol200PIDs2=solPID[200,2,{0.5,\[Theta]d0},{tin,tout}, desired];
In[155]:= (*  HEIGHT = 400 KM*)
sol400s2=sol[400,2,{0.5,\[Theta]d0},{tin,tout}, desired];
sol400PIDs2=solPID[400,2,{0.5,\[Theta]d0},{tin,tout}, desired];
In[157]:= (*  HEIGHT = 600 KM*)
sol600s2=sol[600,2,{0.5,\[Theta]d0},{tin,tout}, desired];
sol600PIDs2=solPID[600,2,{0.5,\[Theta]d0},{tin,tout}, desired];
In[159]:= (*  HEIGHT = 800 KM*)
sol800s2=sol[800,2,{0.5,\[Theta]d0},{tin,tout}, desired];
sol800PIDs2=solPID[800,2,{0.5,\[Theta]d0},{tin,tout}, desired];
Asymmetric
In[161]:= (*  HEIGHT = 200 KM*)
sol200s3=sol[200,3,{0.5,\[Theta]d0},{tin,tout}, desired];
sol200PIDs3=solPID[200,3,{0.5,\[Theta]d0},{tin,tout}, desired];
In[163]:= (*  HEIGHT = 400 KM*)
sol400s3=sol[400,3,{0.5,\[Theta]d0},{tin,tout}, desired];
sol400PIDs3=solPID[400,3,{0.5,\[Theta]d0},{tin,tout}, desired];
In[165]:= (*  HEIGHT = 600 KM*)
sol600s3=sol[600,3,{0.5,\[Theta]d0},{tin,tout}, desired];
sol600PIDs3=solPID[600,3,{0.5,\[Theta]d0},{tin,tout}, desired];
In[167]:= (*  HEIGHT = 800 KM*)
sol800s3=sol[800,3,{0.5,\[Theta]d0},{tin,tout}, desired];
sol800PIDs3=solPID[800,3,{0.5,\[Theta]d0},{tin,tout}, desired];
PLOTS ALL
Symmetric deployed
In[169]:= combplot[sol200s1,sol200PIDs1,desired,{0,2Pi},200]
combplot[sol400s1,sol400PIDs1,desired,{0,2Pi},400]
```

```
combplot[sol600s1,sol600PIDs1,desired,{0,2Pi},600]
combplot[sol800s1,sol800PIDs1,desired,{0,2Pi},800]
Symmetrically folded
In[173]:= combplot[sol200s2,sol200PIDs2,desired,{0,2Pi},200]
combplot[sol400s2,sol400PIDs2,desired,{0,2Pi},400]
combplot[sol600s2,sol600PIDs2,desired,{0,2Pi},600]
combplot[sol800s2,sol800PIDs2,desired,{0,2Pi},800]
Asymmetric
In[177]:= combplot[sol200s3,sol200PIDs3,desired,{0,2Pi},200]
combplot[sol400s3,sol400PIDs3,desired,{0,2Pi},400]
combplot[sol600s3,sol600PIDs3,desired,{0,2Pi},600]
combplot[sol800s3,sol800PIDs3,desired,{0,2Pi},800]
EXTRA VISUALISATION
Extra
In[181]:= ClearAll@combplot
combplot[ndsol1_,ndsol2_,desired_,{t0_,t1_},height_]:=Module[{sol1,sol2,style
(*Styling plots*)
sol1=ndsol1;
sol2=ndsol2;
setp=desired;

style={BaseStyle->{FontFamily->"Times New Roman",12,Bold},
Frame->{{True,False},{True,True}},(*Left and bottom frames only*)
FrameStyle->Black,
GridLines->Automatic,
GridLinesStyle->LightGray,
ImageSize->401,
AxesStyle->Directive[Black]} ;

plot1 = Plot[{Evaluate[\[Theta][t]/.sol1],Evaluate[\[Theta][t]/.sol2]}, {t,t0
   Evaluate@style,
PlotStyle->{Red,Darker@Green},
(*PlotLegends->Placed[{Style["Uncontrolled",Background->White],Style["PID Cor

PlotLegends->Placed[LineLegend[{Red,Darker@Green},{"Uncontrolled","PID Contro
PlotLabel->Style[StringForm["h = ‘1.2f‘ km",height],Bold,18],
FrameLabel->{Style["\[Nu]",Bold,18],Style["\[Theta]",Bold,18]},
LabelStyle->18];
```

```
plot2= Plot[Evaluate[{U1[t,\[Theta][t],\[Theta]’[t],setp],U2[t,\[Theta][t],\[
   Evaluate@style,
PlotLegends->Placed[LineLegend[{"Subscript[u, 1]","Subscript[u, 2]"},LegendFu
(*PlotLegends->Placed[{Style["Subscript[u, 1]",Background->White],Style["Subs
PlotRange->All,
FrameLabel->{Style["\[Nu]",Bold,18],Style["Control Input",Bold,18]},
LabelStyle->18];

(*Combine plots vertically with spacing*)
GraphicsColumn[{plot1,plot2},
Spacings->-5,(*Slight overlap to ensure axis continuity*)
Alignment->Center,Frame->None]]
MAIN VIZUALIZATION
In[183]:= ClearAll@MainRot
MainRot[ndsol_,{ti_,t0_,t2_}]:=Module[{s1,s2,s3,sol1, sol2,sol3,gimbal,c3r,ro
(* PLOTTING \[Theta] GRAPH *)
sol1=ndsol;
sol2=ndsol;
sol3=ndsol;
\[Nu]=ti;
ss=Evaluate[\[Theta][t] /.sol3/.t->\[Nu]](*/1\[Degree]*);

graph3=Show[Plot[Evaluate[\[Theta][t] /.sol1/.t->x],{x,t0,t2},PlotLabel->"\[T
(*graph3={Plot[Evaluate[\[Theta][t] /.sol1/.t\[Rule]x],{x,t0,t2}],Evaluate[\[

(* VIZUALIZATION OF THE EULER ROTATIONS *)
o={0,0,0};
x={1,0,0};
y={0,1,0};
z={0,0,1};
la=25;(* length of axis lines *);
lt=27;(* distance to axis labels *);
blank="    ";
rotor={RGBColor[0.8,1,0.2],
Pyramid[{{0,5,5},{0,5,-5},{0,-5,-5},{0,-5,5},{16,0,0}}]};
axis={Black,Thick,Line[{o,la x}],Line[{o,la y}],Line[{o,la z}],Text[Style["X"
axis1={Darker@Green,Thick,Line[{o,la x}],Line[{o,la y}],Line[{o,la z}],Text[S
axis2={Red,Thick,Line[{o,la x}],Line[{o,la y}],Line[{o,la z}],Text[Style["x",
axis3={Blue,Thick,Line[{o,la x}],Line[{o,la y}],Line[{o,la z}],Text[Style["x"
```

```
s1=0(*/1\[Degree]*);
s2=Evaluate[\[Theta][t] /.sol2/.t->ti](*/1\[Degree]*);
s3=0(*/1\[Degree]*);

gimbal=GeometricTransformation[{rotor,axis1},EulerMatrix[{s1,s2,s3},{3,2,1}]]

graph1=Graphics3D[{gimbal,axis},ViewAngle->0.15,ViewVector->{{-255.3471642973


(* POSITION ON POLAR ORBIT *)
center={1.2*Cos[\[Nu]],1.2*Sin[\[Nu]]};
radVecEnd0=center+{-0.15Cos[\[Nu]],-0.15*Sin[\[Nu]]};
tanVecEnd0=center+{-0.2*Sin[\[Nu]],0.2Cos[\[Nu]]};

radVecEnd=center+{-0.15Cos[\[Nu]-s2],-0.15*Sin[\[Nu]-s2]};
tanVecEnd=center+{-0.2*Sin[\[Nu]-s2],0.2Cos[\[Nu]-s2]};

satcord={PointSize[Large],Point[{1.2*Cos[\[Nu]],1.2*Sin[\[Nu]]}],Arrow[{cente
circle={Blue,Circle[{0,0},1],Black,Dashed,Line[{{-1,0},{1,0}}],Black,Dashed,C

graph2=Graphics[{circle,satcord},ImageSize->{400,400},Method->{"ShrinkWrap"->


Grid[{{graph1,Item[graph2,Alignment->{Center,Center}]},{graph3,SpanFromAbove}
In[185]:= ClearAll@\[Theta]Plot
\[Theta]Plot[ndsol_,{t0_,t1_}]:=Plot[Evaluate[\[Theta][t]/. ndsol],{t,t0,t1},
In[187]:= ClearAll@\[Theta]dotPlot
\[Theta]dotPlot[ndsol_,{t0_,t1_}]:=Plot[Evaluate[\[Theta]'[t]/. ndsol],{t,t0,
```

Appendix C. Matlab Code

```
% Satellite model
%Sat.m
clear all; clc; close all; clear;
param.model = 1; % choose what model of Satellite to use, 0 - simple pendulum
param.case = 4; % 1 - steady sta
te case (zero inputs), 2 - open loop case, 3 - bang-bang control, 4 - pid cor
param.time_sim = 0.001; % sample time for simulation output
param.tinit = pi-0.001; % initial time of simulation
tend = 6.5*pi-param.tinit; % end time of simulation
t_out = param.tinit+[0:param.time_sim:tend]';
num_iter = length(t_out); % number of iteration steps for the whole simulatio
param.Nconsam = 100; % number of time steps after which to update the system
%timedist = []; % times at which shape change occurs
timedist = [4*pi/4, 5*pi/4, 4*pi/4+2*pi, 5*pi/4+2*pi, 4*pi/4+4*pi, 5*pi/4+4*p
% for index = 1:100
[param,nlmpcobj] = Sat_init(param); % update param field
x0_int = param.x0; % update x0
u = param.u0; % update u0
opts = odeset('RelTol',1e-4,'AbsTol',1e-4);
x_out = zeros(length(t_out),param.nx); u_out = zeros(length(t_out),param.nu);
cumerror = 0; % cumulative error (used in PID  control)
nloptions = 0;
if param.case == 4
   param.olderror = 0; % old error for PID control
   param.oldt = -0.001; % old time for PID control
   PID_data = zeros(length(t_out),param.nu*3);
elseif param.case == 7
end
tic
```

```
for ind = 1:num_iter
   t = t_out(ind);
   param.t = t;
   y=param.C*x0_int(1:length(param.A));
   error = param.yr-y;
   cumerror = cumerror+error*param.time_sim;
   [u,param,nlmpcobj,nloptions] = Sat_con(t, ind, u, x0_int, error, cumerror,
   PID_data(ind,:) = [param.Kp param.Ki param.Kd];
  if any(abs((t-param.tinit)-timedist(1:2:end))<param.time_sim/2) % abs(t-7*p
       param.shape = 1;
  elseif any(abs((t-param.tinit)-timedist(2:2:end))<param.time_sim/2) % abs(t
       param.shape = 2;
  end
   % Perform simulation:
   [t_o, x_o] = ode45(@(t, x) Sat_dyn(t, x, u, param),[0:param.time_sim/5:par
   x0_int = x_o(end,:)';
   % Record data:
   x_out(ind,:) = x_o(end,:);
   u_out(ind,:) = u';
end
tsim = toc;
% calculate kinetic energy, potential energy, work, etc.
if param.model == 1
   % find kinetic energy:
   KE = 1/2*1*x_out(:,param.nx/2+1:param.nx).^2;
   Work = u_out.*x_out(:,param.nx/2+1:param.nx);
   % find potential energy
   PE = 1*9.81*(param.sma*(1-param.e^2)./(1+param.e*cos(t_out))-6370);
   % find electric current power
   Power = u_out.^2;
   %KW = zeros(length(t_out),1); PW = zeros(length(t_out),1); EW = zeros(leng
   for ind = 2:length(t_out)
       KW(ind,:) = sum(KE(1:ind,:))*param.time_sim; % kinetic work
       PW(ind,:) = sum(PE(1:ind,:))*param.time_sim; % potential work
       EW(ind,:) = sum(Power(1:ind,:))*param.time_sim; % electric work
       WW(ind,:) = sum(Work(1:ind,:))*param.time_sim; % torque work
   end
end
% save(strcat('results/sat_orbital_PIDcof=',num2str(index)));
```

```
% end
%% Plot the obtained results
% plot GC and GV:
figure('position',[100 100 500 500]);
for ind = 1:2
   for ind2 = 1:param.nx/2 % first row are GC, second row are GV
       subplot(2,param.nx/2,(ind-1)*param.nx/2+ind2)
       hold on
       plot(t_out,x_out(:,(ind-1)*param.nx/2+ind2),'-k','LineWidth',2);
       plot(t_out,mean(x_out(:,(ind-1)*param.nx/2+ind2))*ones(length(t_out),1
       xline(param.tinit+timedist,'--g',{'Shape change'},'LineWidth',1);
       xline([fix(t_out(1)/pi):fix(t_out(end)/pi)]*pi,'--b','LineWidth',1);
       if param.C((ind-1)*param.nx/2+ind2) == 1
           plot(t_out,param.yr*ones(length(t_out),1),'--r','LineWidth',2);
       end
       hold off
       xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
       ylabel(param.namestr{(ind-1)*param.nx/2+ind2},'Interpreter','latex','F
       grid on; box on;
   end
end
% plot inputs:
figure('position',[100 100 500 500]);
for ind = 1:param.nu
   subplot(1,param.nu,ind)
   hold on
   plot(t_out,u_out(:,ind),'-k','LineWidth',2);
   plot(t_out,mean(u_out(:,ind))*ones(length(t_out),1),'--k','LineWidth',2);
   xline(param.tinit+timedist,'--g',{'Shape change'},'LineWidth',1);
   xline([fix(t_out(1)/pi):fix(t_out(end)/pi)]*pi,'--b','LineWidth',1);
   hold off
   xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
   ylabel(param.namestr{param.nx+ind},'Interpreter','latex','FontSize',12,'Fo
   grid on; box on;
end
if param.model == 2 || param.model == 3
   x_alter = zeros(length(t_out),param.na); x_alter2 = x_alter; jind = zeros(
   for ind = 1:num_iter
       % for models 2 and 3 find alternative GCs:
```

```
        if param.model == 2
            Rot = rot_x(x_out(ind,3))*rot_z(x_out(ind,2))*rot_x(x_out(ind,1));
            x_alter(ind,:) = [1/2*sqrt(Rot(1,1)+Rot(2,2)+Rot(3,3)+1),(Rot(2,3)
        elseif param.model == 3
            Rot = rot_quat(x_out(ind,1:4)); dumtheta = acos(Rot(1,1));
            if Rot(3,1) ~= 0 && Rot(2,1) ~= 0
                x_cur = [atan2(Rot(1,3)/sin(dumtheta),Rot(1,2)/sin(dumtheta)),
            else
                x_cur = [0,acos(Rot(1,1)),atan2(Rot(2,3),Rot(2,2))];
            end
            x_alter(ind,:) = x_cur+2*pi.*jind;
            x_alter2(ind,:) = x_cur;
            for ind2 = 1:3
                if ind > 1
                    if (x_cur(ind2)-x_old(ind2)) > pi
                        jind(ind2) = jind(ind2)-1;
                        x_alter(ind,ind2) = x_cur(ind2)+2*pi*jind(ind2);
                    elseif (x_cur(ind2)-x_old(ind2)) < -pi
                        jind(ind2) = jind(ind2)+1;
                        x_alter(ind,ind2) = x_cur(ind2)+2*pi*jind(ind2);
                    end
                end
            end
            x_old = x_cur;
        end
    end
    % plot alternative GC and GV:
    figure('position',[100 100 500 500]);
    for ind2 = 1:param.na % first row are GC, second row are GV
        subplot(1,param.na,ind2)
        hold on
        plot(t_out,x_alter(:,ind2),'-k','LineWidth',2);
%         if param.model == 3
%             plot(t_out,x_alter2(:,ind2),'--r','LineWidth',2);
%         end
        plot(t_out,mean(x_alter(:,ind2))*ones(length(t_out),1),'--k','LineWidt
        hold off
        xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
        ylabel(param.namestral{ind2},'Interpreter','latex','FontSize',12,'Font
```

```
        grid on; box on;
    end
end
% plot energies
if param.model == 1
    figure('position',[100 100 500 500]);

    subplot(2,2,1)
    hold on
    plot(t_out,KE,'-k','LineWidth',2);
%     plot(t_out,KW,'-r','LineWidth',2);
    hold off
    xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
    ylabel('KE [J]','Interpreter','latex','FontSize',12);
    grid on; box on;

    subplot(2,2,2)
    hold on
    plot(t_out,PE,'-k','LineWidth',2);
%     plot(t_out,PW,'-r','LineWidth',2);
    hold off
    xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
    ylabel('PE [J]','Interpreter','latex','FontSize',12);
    grid on; box on;
    subplot(2,2,3)
    hold on
    plot(t_out,Power,'-k','LineWidth',2);
    plot(t_out,EW,'-r','LineWidth',2);
    hold off
    xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
    ylabel('EE [W]','Interpreter','latex','FontSize',12);
    grid on; box on;
    subplot(2,2,4)
    hold on
    plot(t_out,Work,'-k','LineWidth',2);
    plot(t_out,WW,'-r','LineWidth',2);
    hold off
    xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
    ylabel('Input work [W]','Interpreter','latex','FontSize',12);
```

```
   grid on; box on;
end
if param.case == 4
   figure
   hold on
   plot(t_out,PID_data(:,1),'-k','LineWidth',2);
   plot(t_out,PID_data(:,2),'-b','LineWidth',2);
   plot(t_out,PID_data(:,3),'-.c','LineWidth',2);
   plot(t_out,PID_data(:,4),'-.m','LineWidth',2);
   plot(t_out,PID_data(:,5),'--r','LineWidth',2);
   plot(t_out,PID_data(:,6),'--g','LineWidth',2);
   hold off
   xlabel('$\nu$ []','Interpreter','latex','FontSize',12);
   ylabel('PID coefficients','Interpreter','latex','FontSize',12);
   grid on; box on;
end

% code for Satellite model dynamics simulation
% Sat_dyn.m
function [dxdt] = Sat_dyn(t, x, u, param)
if param.model == 1
   Cm = param.cma(param.shape,1);
   for ind = 1:(size(param.cma,2)-1)/2
       Cm = Cm + param.cma(param.shape,1+ind)*cos(ind*(x(1)-param.ys));
       Cm = Cm + param.cma(param.shape,1+(size(param.cma,2)-1)/2+ind)*sin(ind
   end
   tnow = param.t+t;
%     dxdt = [x(2);param.alpha*Cm+u*param.beta*(2*sin(tnow)*sin(x(1))+cos(tno
   e = param.e; % eccentricity
   phi = e*sin(tnow)-e^2*sin(tnow)*cos(tnow);
   r = param.sma*(1-e^2)/(1+e*cos(tnow)); % radius vector
   height = round((r-6370)*10)/10; % local height
   alpha = interp1(param.alphahtable(1,:),param.alphahtable(2,:),height,'spli
   var1 = e*cos(tnow)+1;
   acc1 = 2*e*sin(tnow)*(x(2)+1)/var1;
   acc2 = (4*e^3*cos(2*tnow)+2*e^3+e^2*cos(tnow)-e)*sin(tnow)/var1;
   acc3 = alpha*Cm*(e^2+2*e*cos(tnow)+1)/var1^4;
   acc4 = param.gamma*sin(2*(x(1)-phi))/var1/1;
   if param.nu == 1
       acc5 = u*param.beta*(sin(x(1)+tnow+pi/2-phi)-3*sin(x(1)-tnow-pi/2-phi)
```

```
        acc6 = 0;
    elseif param.nu == 2
        acc5 = u(1)*param.beta*(sin(x(1)+tnow-phi)-3*sin(x(1)-tnow-phi))/var1/
        acc6 = u(2)*param.beta*(sin(x(1)+tnow+pi/2-phi)-3*sin(x(1)-tnow-pi/2-p
    end
    dxdt = [x(2);acc1+acc2+acc3+acc4+acc5+acc6];
end
end

% function to calculate the control inputs
%Sat_con.m
function [u,param, nlmpcobj,nloptions] = Sat_con(t, ind, u0, x0_int, error, c
    if rem(ind,param.Nconsam) == 1
        ind
        [A, B, C, D] = Sat_lin(t, x0_int, u0, param);
        param.A = A;
        param.B = B;
        param.C = C;
        param.D = D;
    end
    clear u
    if param.case == 4 % pid control
        % linearize the system if error is large
        % if rem(ind,param.Nconsam) == 1
        %     for ind2 = 1:param.nu
        %         [b,a] = ss2tf(param.A,param.B,param.C,param.D,ind2);
        %         sys2 = tf(b,a);
        %         [Cont,info] = pidtune(sys2,'PID'); % 60
        %         info
        %         [Kp,Ki,Kd] = piddata(Cont);
        %         param.Kp(ind2) = Kp;
        %         param.Ki(ind2) = Ki;
        %         param.Kd(ind2) = Kd;
        %         fprintf("Kp=%d, Ki=%d, Kd=%d", Kp,Ki,Kd);
        %     end
        % end
        if rem(floor(t*2/pi),4) == 0
            param.Kp = [860 860]; param.Ki = [0 0]; param.Kd = [20 20]; % para
        elseif rem(floor(t*2/pi),4) == 1
            param.Kp = [860 -860]; param.Ki = [0 0]; param.Kd = [20 -20]; % pa
```

```
        elseif rem(floor(t*2/pi),4) == 2
            param.Kp = [-860 -860]; param.Ki = [0 0]; param.Kd = [-20 -20]; %
        elseif rem(floor(t*2/pi),4) == 3
            param.Kp = [-860 860]; param.Ki = [0 0]; param.Kd = [-20 20]; % pa
        end
        u = param.Kp*error+param.Ki*cumerror+param.Kd*(error-param.olderror)/(
        u = u';
        u(u>param.umax) = 0; % param.umax;
        u(u<param.umin) = 0; % param.umin;
        param.olderror = error;
        param.oldt = t;
    end
end

% code to obtain linearized model of the satellite
%Sat_lin.m
function [A, B, C, D] = Sat_lin(t, x, u, param)
if param.model == 1
    e = param.e; % eccentricity
    phi = e*sin(t)-e^2*sin(t)*cos(t);
    var1 = e*cos(t)+1;
    r = param.sma*(1-e^2)/(1+e*cos(t)); % radius vector
    height = round((r-6370)*10)/10; % local height
    alpha = interp1(param.alphahtable(1,:),param.alphahtable(2,:),height,'spli
    Cml = 0;
    for ind = 1:(size(param.cma,2)-1)/2
        Cml = Cml - param.cma(param.shape,1+ind)*ind*sin(ind*(x(1)-param.ys));
        Cml = Cml + param.cma(param.shape,1+(size(param.cma,2)-1)/2+ind)*ind*c
    end
%     A = [0 1; param.alpha*Cml+u*param.beta*(2*sin(t)*cos(x(1))-cos(t)*sin(x
%     B = [0; param.beta*(2*sin(t)*sin(x(1))+cos(t)*cos(x(1)))];
%     A = [0 1; alpha*Cml*(e^2+2*e*cos(t)+1)/var1^4+u*param.beta*(cos(x(1)+t-
%     B = [0; param.beta*(sin(x(1)+t-phi)-3*sin(x(1)-t-phi))/var1];
    if param.nu == 1
        A = [0 1; alpha*Cml*(e^2+2*e*cos(t)+1)/var1^4+u*param.beta*(cos(x(1)+t
        B = [0; param.beta*(sin(x(1)+t+pi/2-phi)-3*sin(x(1)-t-pi/2-phi))/var1]
        D = 0;
    elseif param.nu == 2
        A = [0 1; alpha*Cml*(e^2+2*e*cos(t)+1)/var1^4+u(1)*param.beta*(cos(x(1
        B = [0, 0; param.beta*(sin(x(1)+t-phi)-3*sin(x(1)-t-phi))/var1, param.
```

```
        D = [0, 0];
    end
    C = [1 0];
end
end

% code to initialize the Satellite model
%Sat_init.m
function [param,nlmpcobj] = Sat_init(param)
    if param.model == 1    % 1D simple satellite model, polar orbit
        param.nx = 2; % number of states
        param.ny = 1; % number of outputs
        param.nu = 2; % number of inputs
        param.na = 0; % number of alternative variables (0 for this case)
        param.yr = 0.748; % desired output 0.1 or 0.748
        param.ys = 0; % stable output (theta) value due to asymmetric geometry
        height = 600; % current altitude in km
        periapsis = 600; % in km, 200<periapsis<800
        apoapsis = 600; % in km, 200<apoapsis<800, apoapsis > periapsis
        param.sma = 6370+(periapsis+apoapsis)/2; % semi major axis
        param.e = 0; % eccentricity
        % param.e = 1-(6370+periapsis)/param.sma; % eccentricity
        param.alphahtable = [[200:50:800]; 737.3 200.3 67.43 26.19 10.92 4.839
        alpha = interp1(param.alphahtable(1,:),param.alphahtable(2,:),height,'
        param.alpha = alpha; % alpha parameter
        param.alphalooktable(1,:) = [190:0.1:810];
        param.alphalooktable(2,:) = interp1(param.alphahtable(1,:),param.alpha
        param.beta = 1; % beta parameter, amplitude of acceleration due to mag
        %param.beta = (1.256*10^-6*7.7*10^22)/(8*3.1415*3.986*10^14*(1000*heig
        param.gamma = 1.14; % 2.28; % gamma parameter, amplitude of accelerati
    %     param.cma = [0, 0, 0, 0, 0, 0, 0, 0, -50.56 0.4676 -9.785 -0.2731 2.
    %         0, 0, 0, 0, 0, 0, 0, 0, -42.7 0.2556 -5.45 -0.1315 2.246 -0.0170
    %         0, 0, 0, 0, 0, 0, 0, 0, -34.84 0.04363 -1.115 0 1.606 -0.02172 -
        param.cma = [0, 0, 0, 0, 0, 0, 0, 0, -7.203, -0.02843, 0.3954, -0.0256
            0, 0, 0, 0, 0, 0, 0, 0, -7.976, -0.07466, 0.6895, 0.03107, 0.3812,
            0.134, 5.213, -0.02617, 1.258, -0.00997, -0.09907, -0.0446, 0.0705
        param.shape = 3; % shape parameter, 1 - tetrahedral; 2 - transient; 3
        % check if CM is correct
        dumtheta = -pi:0.01:pi;
        Cm = param.cma(:,1);
```

```
        for ind = 1:(size(param.cma,2)-1)/2
            Cm = Cm + param.cma(:,1+ind).*cos(ind*dumtheta);
            Cm = Cm + param.cma(:,1+(size(param.cma,2)-1)/2+ind).*sin(ind*dumt
        end
        figure; plot(dumtheta,Cm(1,:),'-b',dumtheta,Cm(2,:),'-r',dumtheta,Cm(3
        x0_int = [0,0]'; % IC for x
        param.x0 = x0_int;
        param.u0 = zeros(param.nu,1);
        nlmpcobj = 0;
        % obtain linearized model at tetrahedral configuration
        [A, B, C, D] = Sat_lin(param.tinit, param.x0, param.u0, param);
        param.A = A;
        param.B = B; % [0; -1]; %
        param.C = C;
        param.D = D;
        param.umax = 150;
        param.umin = -150;
        eig(param.A)
        Con = ctrb(param.A, param.B); % continuous controllability matrix
        unconvc = length(param.A) - rank(Con); % find number of uncontrollable
        unconvc
        Obc = obsv(param.A,param.C); % continuous observability matrix
        unobsvc = length(param.A) - rank(Obc); % find number of unobservable s
        unobsvc
        if param.nu == 1
            param.namestr = {'$\theta$ [rad]','$\dot{\theta}$ [rad/s]','$u$ [N
        elseif param.nu == 2
            param.namestr = {'$\theta$ [rad]','$\dot{\theta}$ [rad/s]','$u_1$
        end

    end
end
```

Appendix D. Technical Drawings

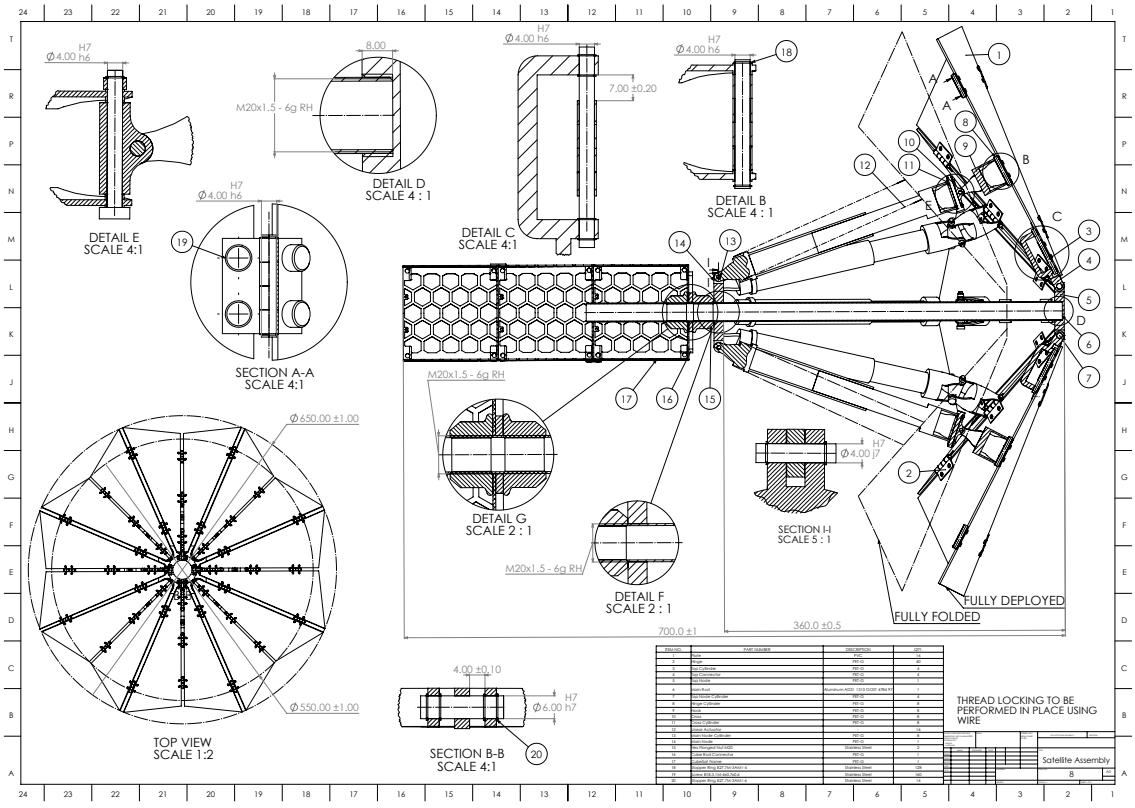


Figure 33: Complete Assembly

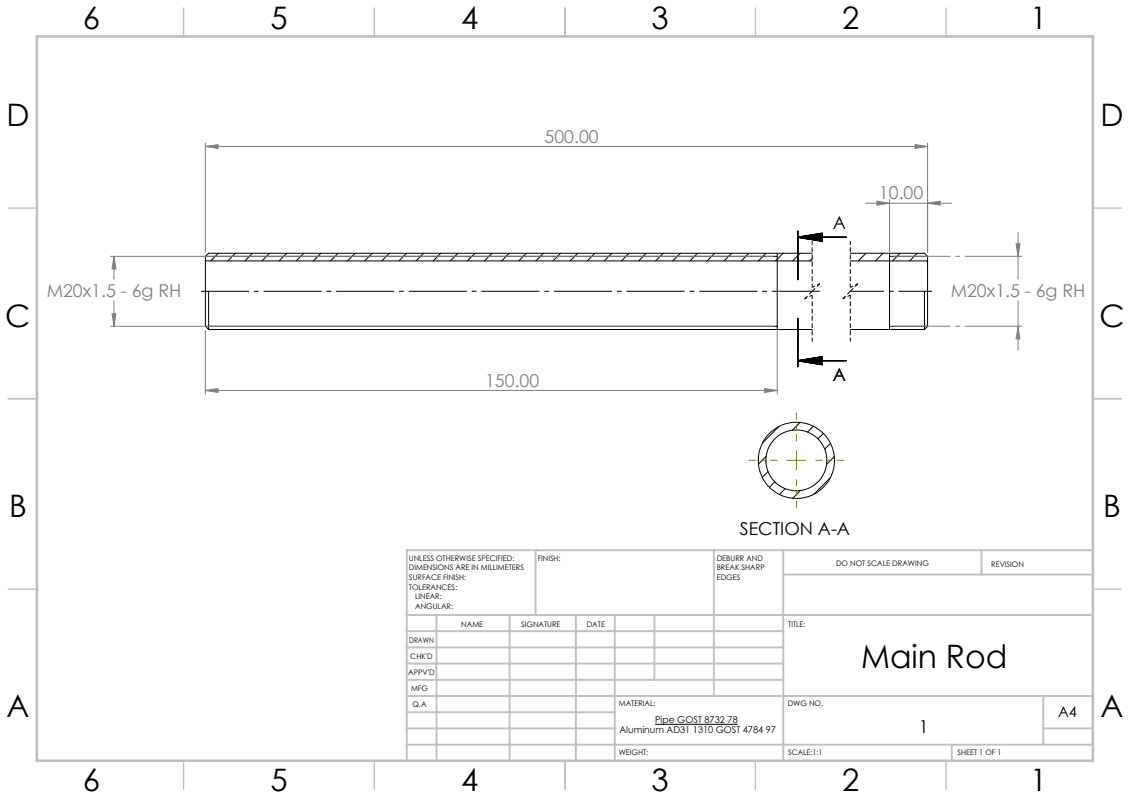


Figure 34: Main Rod

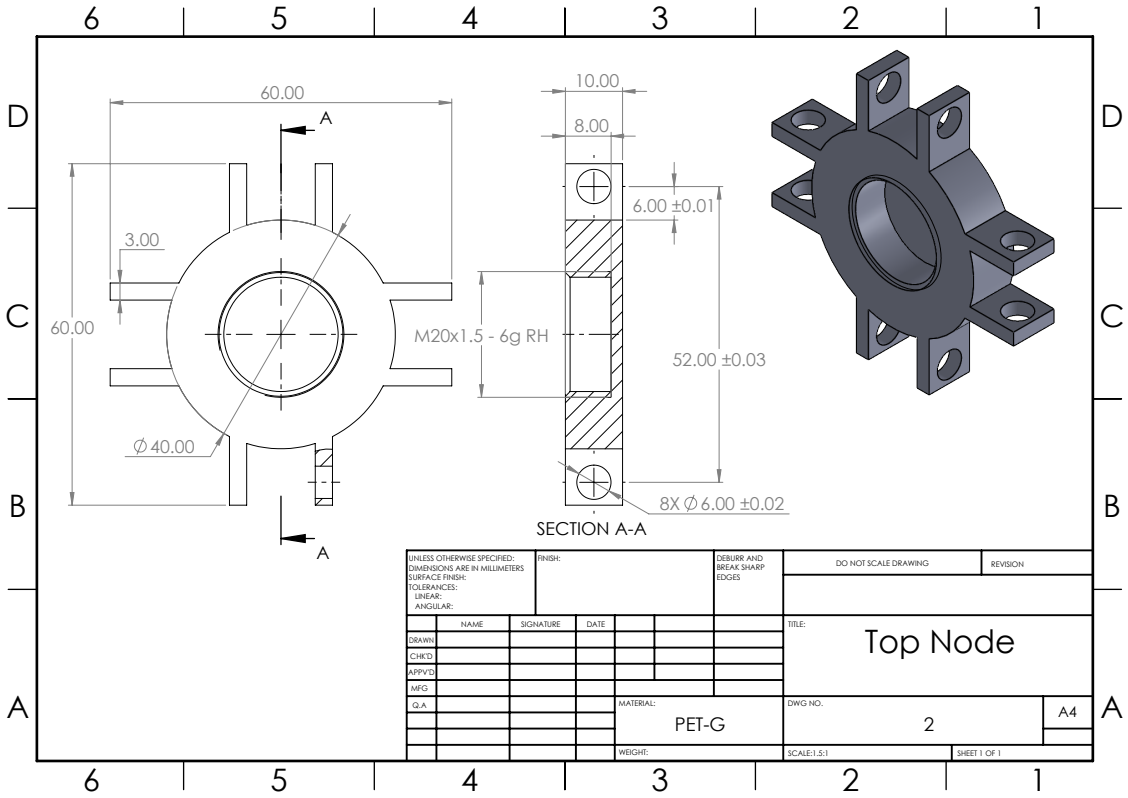


Figure 35: Top Node

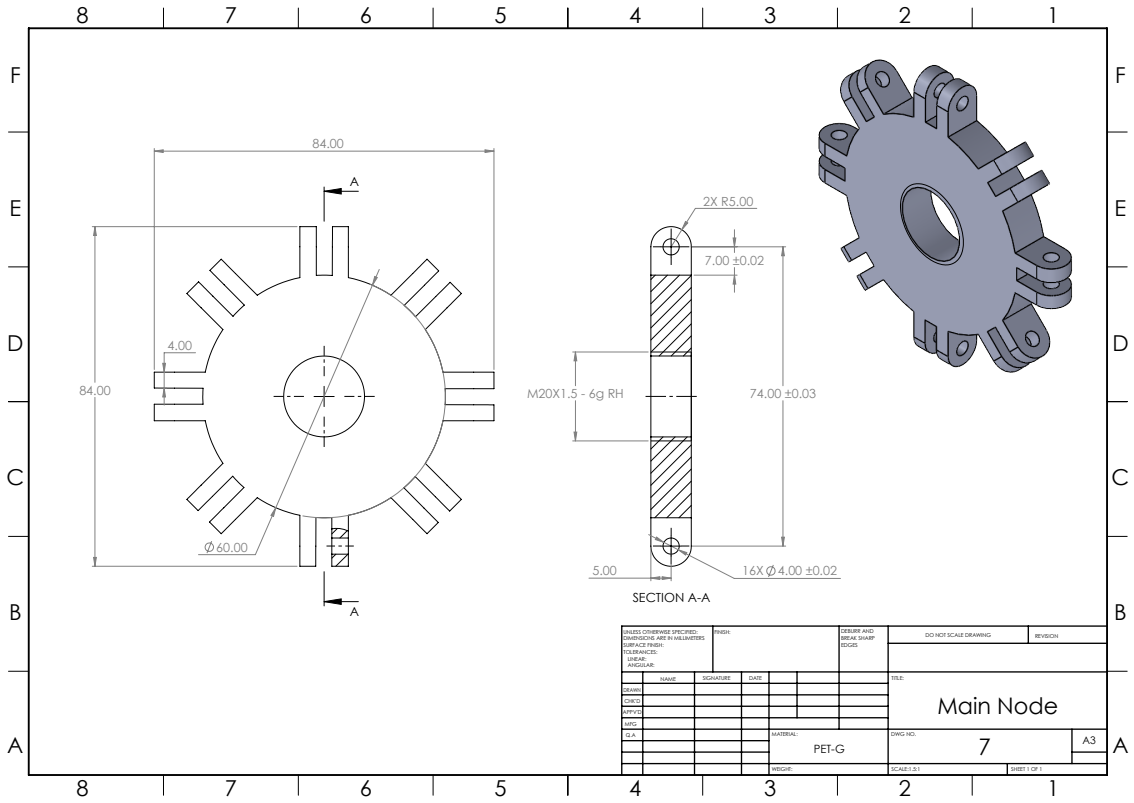


Figure 36: Main Node

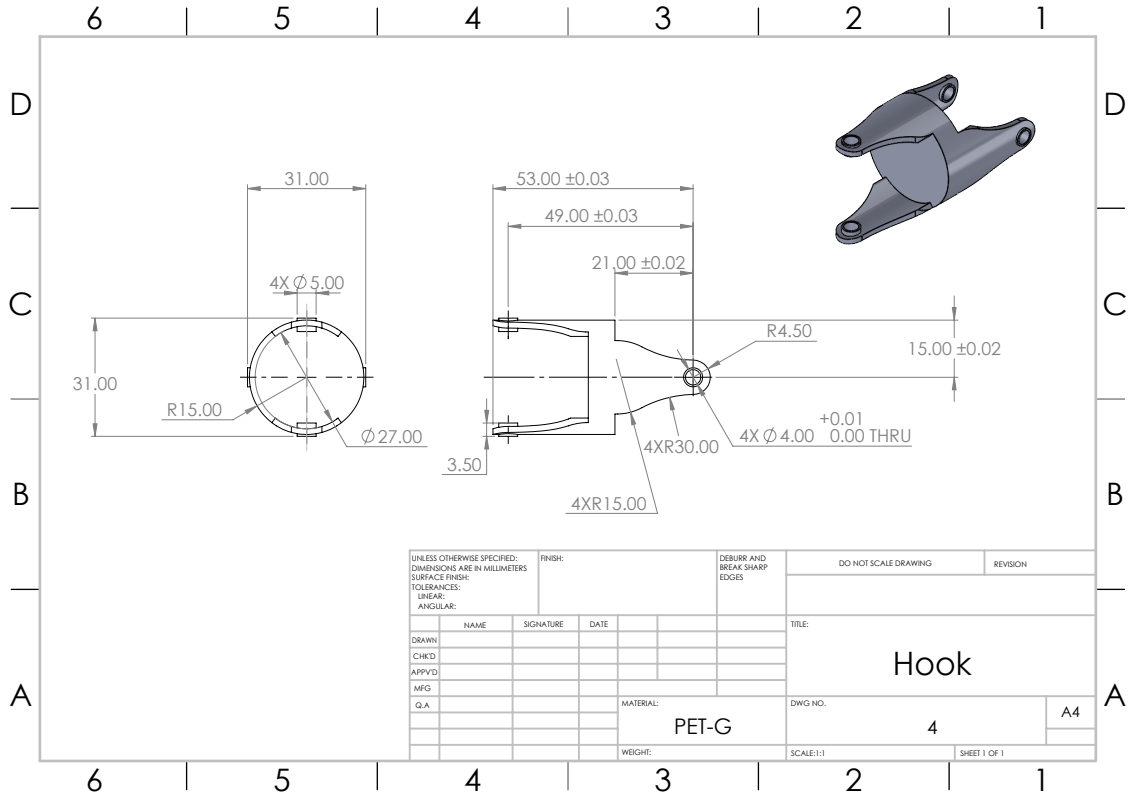


Figure 37: Hook

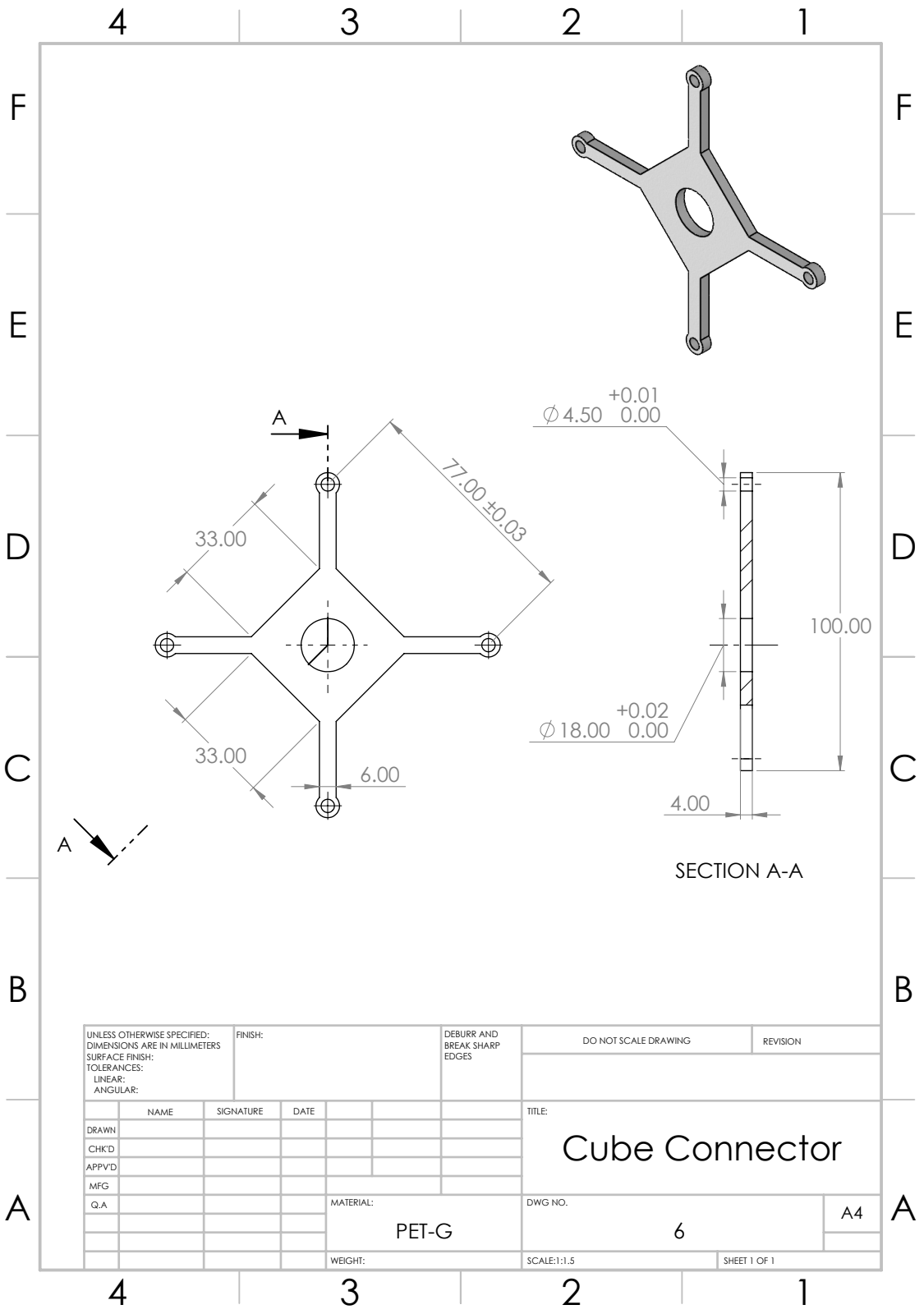


Figure 38: Cube Connector

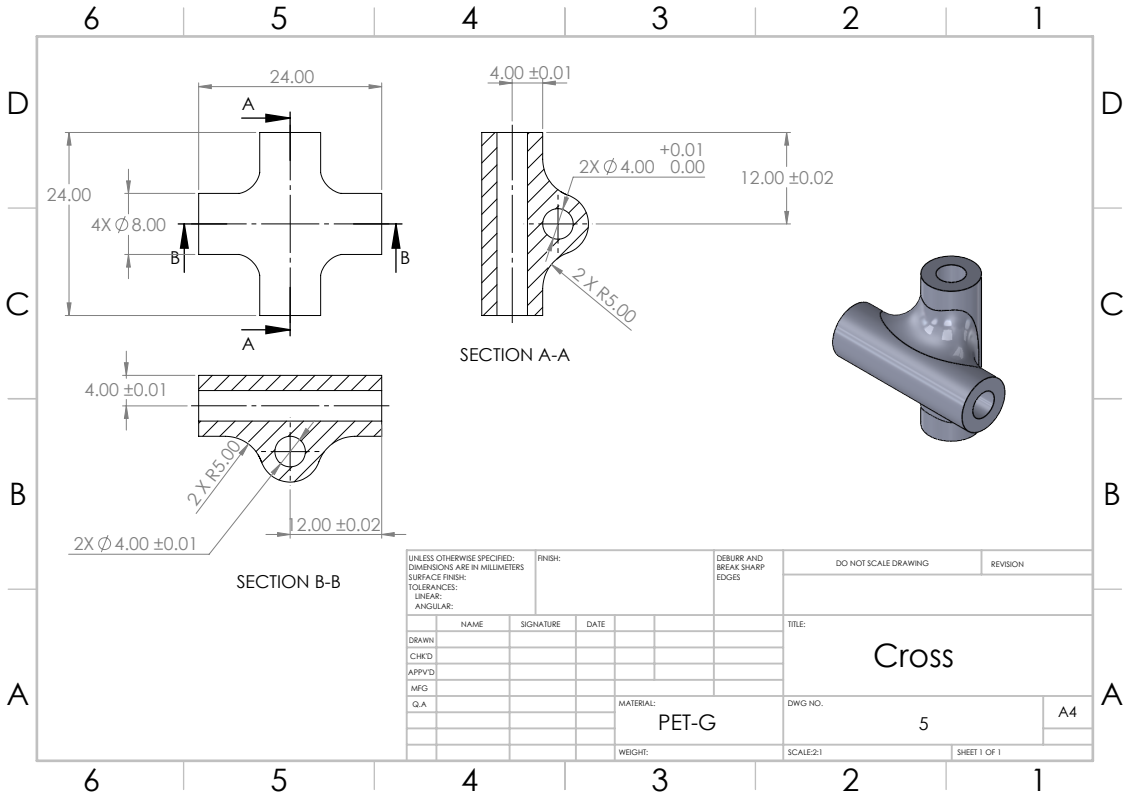


Figure 39: Cross

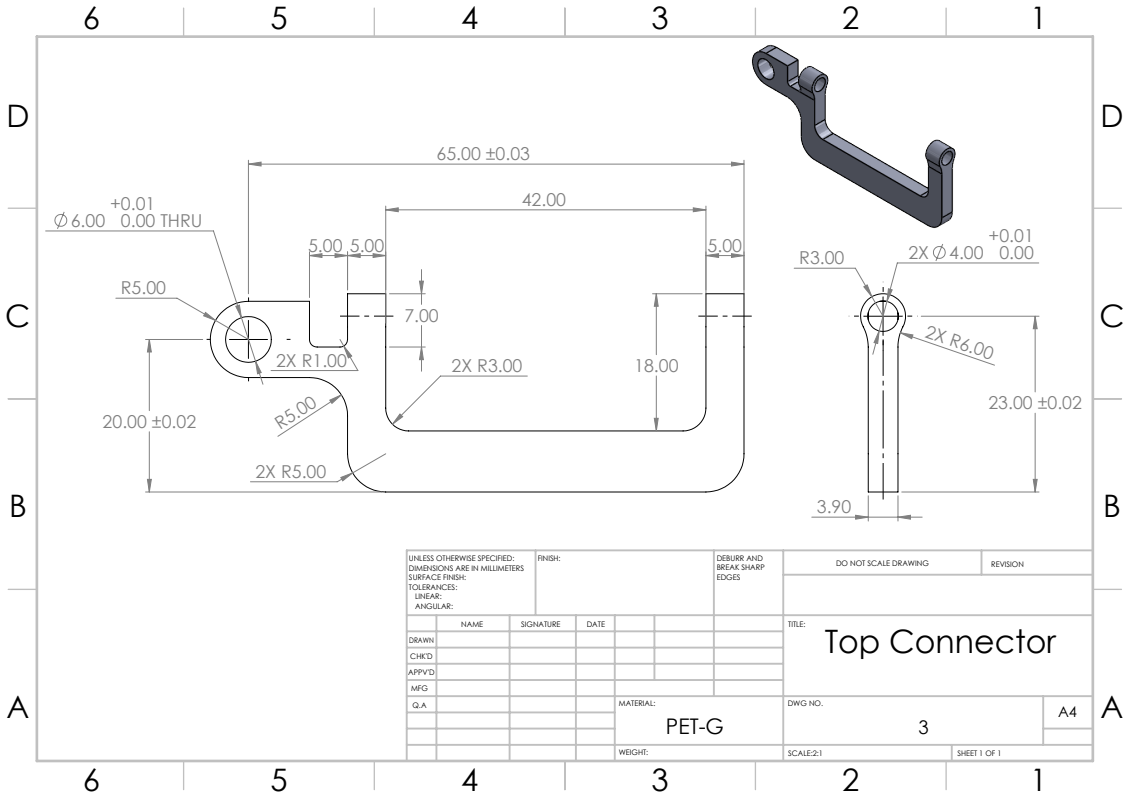


Figure 40: Top Connector

Item №	Part Name	Material	QTY
1	Plate	PVC	16
2	Hinge	PET-G	40
3	Top Cyllinder	PET-G	4
4	Top Connector	PET-G	4
5	Top Node	PET-G	1
6	Main Rod	Aluminum AD31 1310 GOST 4784 97	1
7	Top Node Cyllinder	PET-G	4
8	Hinge Cyllinder	PET-G	8
9	Hook	PET-G	8
10	Cross	PET-G	8
11	Cross Cyllinder	PET-G	8
12	Linear Actuator	-	8
13	Main Node Cyllinder	PET-G	16
14	Main Node	PET-G	1
15	Hex Flanged Nut M20	Stainless Steel	2
16	Cube Connector	PET-G	1
17	CubeSat Frame	PET-G	3
18	Stopper Ring B27.7M-3AM-4	Stainless Steel	128
19	Screw B18.3.1M-4x0.7x0.6	Stainless Steel	160
20	Stopper Ring B27.7M-3AM1-6	Stainless Steel	16

Table 9: Bill of Materials