Analyzing and developing precise pointing analysis tool to reduce image distortion in Earth Observation satellites

Vidhan Vohra

Space Engineering, master's level (120 credits)
2023

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Acknowledgment

I would like to offer my sincere gratitude to Dr. Stefanus Schröder and Mr. Harish Rao Ramavaram for their supervision and guidance throughout the project work. I would like to extend my utmost thankfulness to my friends and family, whose support was crucial for my thesis.

Furthermore, I would like to take this opportunity and thank Luleå University of Technology for accepting me into SpaceMaster program, it helped me learn a lot of valuable lessons in my preferred field of interest.
Abstract

With growing space market and entry of more private companies into the industry, there are companies and stakeholders who would like to have a high order of accuracy mission output requirements. These requirements vary from mission to mission. This simply means that if a company wants an Earth observation mission, the main requirement to be fulfilled would be to have the highest resolution of image possible.

In order to achieve this, the satellite carrying the camera payload would be required to be pointed in the right direction with utmost accuracy. For a satellite to be pointed in the right direction, the noise generated by the sensors and actuators on-board, which determines the attitude of the satellite and helps in changing it, should be minimized. The aim of this thesis is to design a method which could help in determining the right components to be procured so that the pointing requirements of the satellite are fulfilled. This objective is achieved by designing algorithms in python and MATLAB. The values generated by these algorithms, ultimately describe the type of sensor or actuator to be procured. Finally, the noise generated by such individual components act as pointing error source and then PEET is used to translate these error sources to platform error, to generate a pointing budget and ensure that all pointing requirements are satisfied.
Contents

List of Figures iv
List of Tables iv

1 Introduction 1
  1.1 Thesis objectives ........................................... 1
  1.2 History ...................................................... 1
  1.3 Core Concepts .............................................. 2

2 Theory 6
  2.1 Star Trackers .................................................. 6
  2.2 Reaction Wheels ............................................. 12
    2.2.1 Reaction Wheel Force model ........................... 13
    2.2.2 Reaction Wheel Torque model ........................... 16

3 Methodology 19
  3.1 FOV and Pixel noise ..................................... 19
  3.2 Reaction Wheel-Force Tonal noise ........................ 26
  3.3 Reaction Wheel- Torque Tonal noise ....................... 29

4 Case Study 31
  4.1 PES to platform error .................................... 31
  4.2 Case Study: Earth Observation mission .................... 31
    4.2.1 EO mission: 3U platform .............................. 32
    4.2.2 EO mission: 6U platform .............................. 34
  4.3 Case Study: Inter satellite link .......................... 35
    4.3.1 Inter satellite mission: 3 RWs ....................... 36
    4.3.2 Inter satellite mission: 4 RWs ....................... 37

5 Scope for further work 38

References 39

Appendix 41

I Appendix 41
  A Appendix 1 .................................................... 41
  B Appendix 2 .................................................... 42
  C Appendix 3 .................................................... 43
  D Appendix 4 .................................................... 44
  E Appendix 5 .................................................... 44
  F Appendix 6 .................................................... 45
  G Appendix 7 .................................................... 47
List of Figures

1. Pointing relation .................................................. 2
2. PEET GUI interface .................................................. 3
3. Industry Star Tracker ................................................. 6
4. CCD and CMOS sensors .............................................. 7
5. APS Star Tracker ..................................................... 8
6. Angles subtended with Cross-axis and Bore sight axis .... 9
7. Reaction Wheels in a platform .................................... 12
8. Static Imbalance process .......................................... 15
9. Rocking mode dynamics of a reaction wheel ............... 17
10. Dynamic Imbalance process of Reaction wheel ............ 18
11. Graph representing relation between all factors for $PSD_{FOV}$ ............................................. 20
12. Graph representing relation between factors affecting $PSD_{pixel}$ ........................................... 22
13. Power spectral density and error generated ................. 23
14. Relation of Number of Stars and Error ....................... 24
15. Relation between $\omega_{sp}$ and Error ......................... 24
16. Relation between FOV and error ............................ 25
17. Relation between $\beta$ and Error ........................... 25
18. Relation between $\alpha$ and Error ............................. 26
19. Monte carlo output for tonal noise-Force Model ......... 27
20. Relation for broadband noise and error ....................... 28
21. Relation between Static imbalance and error .......... 28
22. Relation between wheel speed and error ..................... 29
23. Monte carlo output for tonal disturbance- RW Torque .... 30
24. PEET GUI system for 3U platform ............................ 32
25. PEET GUI system for 6U platform ............................ 34
26. Inter Satellite antenna gain .................................... 35

List of Tables

1. Accuracy of sensors in a satellite ................................. 3
2. Accuracy of actuators in a satellite ............................. 4
3. Error sources of a Star Tracker .................................. 5
4. Resolution of different Star Trackers ........................... 8
5. Types of error caused by Reaction Wheels ..................... 13
6. Star Tracker parameters generated by algorithm .......... 33
7. Reaction Wheel parameters generated by algorithm ........ 33
8. Pointing budget for EO mission 3 RWs and Star Tracker .... 34
9. Pointing budget for EO mission 4 RWs and Star Tracker .... 35
10. Star Tracker parameters generated by algorithm ............ 36
11. Reaction Wheel parameters generated by algorithm ........ 37
12. Pointing budget for ISL mission: 3 RWs and Star Tracker .... 37
13. Pointing budget for ISL mission: 4 RWs and Star Tracker .... 37
Abbreviations

**AOCS**: Attitude and Orbital Control System

**PEET**: Pointing Error Engineering Tool

**GUI**: Graphical User Interface

**CMG**: Control Moment Gyroscope

**FOV**: Field of View

**HFSE**: High Frequency Spatial Error

**LFSE**: Low Frequency Spatial Error

**CCD**: Charged Coupled Device

**CMOS**: Complementary Metal Oxide Semiconductor

**APS**: Active Pixel Sensor

**PSD**: Power Spectral Density

**PES**: Pointing Error Source

**IV**: Induced Vibrations
1 Introduction

The purpose of this thesis is to introduce pointing analysis by developing the workflow required to get the optimum values which generates pointing error within the required limit. The workflow would be used in designing pointing budgets for future missions which require precise pointing requirements. Furthermore, since pointing analysis is a novel approach and there are significant number of companies and customers who want a better resolution of the final image or a better communication link established between satellites, pointing engineering becomes even more relevant to be studied.

1.1 Thesis objectives

Keeping in mind the above-mentioned limitations, it was decided to design the workflow for reaction wheels and star trackers. The major objectives are:

- Introduce the concepts of pointing requirements and pointing analysis.
- Define the use of Star Trackers and Reaction wheels, how do they contribute to pointing error of a spacecraft.
- Develop mathematical analysis which can help the user understand the factors which would contribute to the pointing error.
- Discuss and analyze the meaning of results generated.
- Discuss case studies for different platforms using the results of the mathematical analysis generated.

1.2 History

The work undertaken by the field of AOCS is crucial to the right functioning of the satellite and in-turn in the functioning of the overall mission. Pointing analysis is being done by engineers in the space world as back as 1991 by the Space Telescope Science Institute, which could help in realizing if there is any error caused due to jittering of components on-board the Hubble telescope, and how much is it affected. This early work in the field of pointing analysis created a strong backbone in understanding the concept behind jitter analysis caused by the individual components of the spacecraft. During the mid-1990s, this work became key in developing the next set of jitter files, by Observation Monitoring System, which could give an idea about the pointing calculations, and pointing error sources. Simulations are of key importance for the AOCS team and development of the sub-system. This is primarily because AOCS architecture consists of either manufacturing or procuring costly hardware components which serve as a fortitude to a successful mission achieving objective. It should be noted that, such components due to its expensive nature and it’s important in a mission design have to be selected very carefully. Simulations help the engineers in understanding how these components act under various real time conditions. This in-turn helps the engineers chose AOCS hardware components which satisfy the mission requirements, while ensuring the cost efficiency of the entire process of selection of such components.
The main requital in a simulation model is between the accuracy of the model in real time and the simulation accuracy. From a lesser level, accuracy can be defined as the results satisfying the definitions of parameters which we know from experience of previous work. In practice, it can also be seen as that it is defined as the amount or value of results generated coinciding with the results derived from previous experiments.

1.3 Core Concepts

This section gives a general idea of pointing analysis, pointing error categories, pointing budget and PEET.

**Pointing analysis**: A spacecraft, especially which has got Earth Observation payload requires mission objectives, which necessitates the payload to be oriented towards the concerned area with the utmost accuracy. This would depend on factors which would include the payload camera properties, the accuracy with which the entire spacecraft has achieved the attitude for the payload to be directed in the right direction. The scope of this thesis involves the study of the factors which would affect the accuracy with which the spacecraft attitude would be adjusted, for the payload to be pointed in the right direction. The study of the factors which contribute to the errors generated by the spacecraft, to point the payload and then in-turn how to effectively reduce these errors, is defined as pointing analysis.

**Pointing Budget**: Pointing budget is defined as a structure which defines logic for components and factors which contribute to the overall accuracy of pointing the satellite. This is done in tabular data format, where all these individual error sources are specified which make up the total pointing budget of the spacecraft. Furthermore, it should be noted that in a pointing budget the error contributed by individual error sources is specified in X, Y and Z direction separately, in order to know the error contributed by the component of a sub-system in every axis and analyze the results accordingly. The pointing budget would essentially help the mission design engineers in understanding the risks associated with choosing a particular actuator and/or sensor for the mission, which would help in designing a mission which has a balanced trade-off between design risks and cost associated with the same. This statement can be very well described in figure 1.

![Figure 1: Pointing relation](image)
PEET: PEET, is software developed by Astos solutions for calculating the pointing budgets from PES. ASTOS stands for "Analysis, Simulation and Trajectory Optimization Software for Space Applications". It is a German company based in Stuttgart. It provides software products for space applications. The software products can be used for trajectory optimization, mission performance analysis, pointing analysis and many more. This software assists the users in the arrangement and finally calculation of the effect which an individual pointing error source would have on the overall system. This software is designed as a toolbox, which can be executed via MATLAB. It provides a specific GUI, which helps in creating the impact an individual PES would have, successively creating the error signal flow. Such an approach along with its GUI feature makes it remarkably simple and clear procedure to calculate pointing budget. A diagram representing the GUI of PEET, and its flow is mentioned in figure 2.

Figure 2: PEET GUI interface

Pointing errors: As mentioned in the above definition of pointing budget, the pointing errors generated in a spacecraft depends upon hardware components which are being used in a spacecraft. Such hardware components include attitude determination sensors such as star tracker, sun sensor, radio frequency beacon and horizon sensor. It should be noted that even actuators contribute to the overall pointing error generated in a platform. The actuators used commonly include reaction wheels, magnetorquer, gyroscopes and thrusters. Out of these sensors and actuators, for the scope of the thesis, the factors which affect the noise generated by Star Trackers and Reaction wheels would be discussed. Both these hardware components are chosen to be discussed in the thesis, because they are the most accurate components, and are required in a mission to satisfy stringent pointing requirements. This could very well be made clear by the table below.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun sensor</td>
<td>1'</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>1'</td>
</tr>
<tr>
<td>Radio Frequency beacon</td>
<td>1'</td>
</tr>
<tr>
<td>Horizon sensor</td>
<td>5'</td>
</tr>
<tr>
<td>Star Tracker</td>
<td>1&quot;</td>
</tr>
</tbody>
</table>

Table 1: Accuracy of sensors in a satellite.
Reaction wheels are considered to be the cleverest choice of an engineer, especially for high accuracy CubeSat missions. This is because as compared to the other actuators, they have advantages in terms of precise attitude correction. Let’s look into this via discussing other actuators. [4]

**Magneto torquers:** Magneto torquers work on the principle of controlling magnetic moment and using it for attitude control. The magnetic moment is generated when an electrical current is passed through the coil. Some of the setbacks which using magneto torquers face are low level of torque induced which might account to not achieving the correct attitude and also are used for coarse attitude correction.

**Control Moment Gyros:** The working principle of a CMG is very similar to that of a reaction wheel, which means it has a spinning wheel controlled by an electric motor. For commercial missions, CMGs are less preferred over reaction wheels despite having similar working principle and higher efficiency, because of the phenomenon known as gimbal lock. Gimbal motions can produce certain orientations in relative terms such that the torque produced by the CMG is in a plane and not in any of the axis of the CMG. This makes it challenging to effectively control the attitude of the spacecraft both mathematically and computationally. [5]

**Thrusters:** Thrusters work on the principle of burning combustible fuel and generating force to adjust or change the attitude. One of the major shortcomings of thrusters is that they carry with them the factor of fuel limitations, which makes the overall system more difficult to control. Below mentioned table would make it easier to understand why reaction wheels are preferred over other actuators.

<table>
<thead>
<tr>
<th>Actuators</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magneto torquers</td>
<td>1°</td>
</tr>
<tr>
<td>CMG</td>
<td>0.1°</td>
</tr>
<tr>
<td>Thrusters</td>
<td>0.1°</td>
</tr>
<tr>
<td>Reaction Wheels</td>
<td>0.01°</td>
</tr>
</tbody>
</table>

Table 2: Accuracy of actuators in a satellite.

Now that, it is discussed about how Star Tracker and Reaction wheels are the preferred choice of selection for a mission, due to their exceptional degree of accuracy for operational use, it is important, to understand about what are the different ways in which these hardware components act as pointing error source and contribute to the pointing budget.

The error sources by which Star Tracker contributes to the platform error are Assembly misalignment, thermal distortion, FOV and Pixel noise. [6] This thesis elaborates on the mathematical modelling and the degree of effect of individual factors which contribute to the overall FOV and Pixel noise for a Star Tracker. FOV and Pixel noise are generated due to the intrinsic properties of a Star Tracker sensor. When a star tracker is pointed towards the stars, these stars are captured by the individual pixels and the entire sensor FOV as well. The error generated due to the intrinsic properties or noise by the sensor result in the distortion of the image captured by sensor FOV and by individual pixels. These two values are defined as FOV and Pixel noise. FOV noise represents the error generated due to the entire sensor FOV and Pixel noise represents the error generated due to individual pixels. The aim of the project is to understand the effect of these factors on the overall error, so that the right star trackers can be procured for any upcoming missions with the necessary knowledge to reduce the error.
contributed by Star Tracker to the overall platform. Furthermore, it should also be noted that FOV and Pixel noise generate and contribute the highest to the overall platform error among all the other error sources responsible.

The table below gives an idea of the amount of contribution made by these sources on the overall platform error, based on an example simulation in PEET. It should be noted that these are not the quintessential values and are mentioned as an example to point out that among misalignment, thermal distortion and FOV/Pixel noise, the maximum contribution is made by FOV/Pixel noise.

<table>
<thead>
<tr>
<th>Error source</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly misalignment</td>
<td>3.98&quot;</td>
</tr>
<tr>
<td>Thermal distortion</td>
<td>1.568&quot;</td>
</tr>
<tr>
<td>FOV and Pixel noise</td>
<td>36.07&quot;</td>
</tr>
</tbody>
</table>

Table 3: Error sources of a Star Tracker

**Assembly Misalignment error:** These errors are caused as a result of not aligning the instruments or sensors while mounting them immediately on the spacecraft bus. This misalignment would result in an error in controlling the spacecraft’s attitude determination. This error would be passed down to the spacecraft via a control system. Furthermore, all of the misalignment errors are calculated twice, that is once at the beginning while fixing the instruments with the satellite bus and second after the vibration testing. This test would help in understanding what might be the worst-case error scenario which could exist for the particular spacecraft. The most significant and unknown misalignment error which engineers have to consider is misalignment errors due to launch. In order to consider these errors and put methods in place to estimate them, there are two main methods which are being used. The first process which is being used is the tolerance stack up analysis. In this method, calculations are made which help the engineer understand the total effect of part tolerance in accordance with the final assembly. The second process which is being used for calculating the misalignment error during launch is by testing the flight hardware assembly through vibration testing. This method would essentially push the process requirements through vibration testing against loads which would either be equal to that during the launch or greater than the launch loads. The difference between pre-launch condition vibration testing and post-launch condition vibration testing would help in determining the misalignment due to launch. [7]

**Thermal Distortion errors:** The errors which occur due to in-orbit differences in temperature, caused by the relative separation between spacecraft components, resulting in thermal distortion errors. This relative motion of components mean that the instruments and sensors which act as pointing error sources also move, which effects in sun being observed on different parts of the spacecraft. This leads to thermal distortion experienced by the spacecraft and its components. To compute thermal distortion errors, engineers require structural and thermal model of the spacecraft. The thermal model which is generated helps in calculating the temperature of every node of the spacecraft model. This would then be added to the structural model. This process would help the engineers in understanding the temperature variations which the spacecraft would face due to rotation of the spacecraft. [7]

**FOV and Pixel noise:** FOV and Pixel noise are also known as spatial errors and are caused due to the relative motion of stars on FOV and pixels. Another point which should be noted is that in the datasheet of a star tracker, FOV noise might be termed as low frequency
spatial error and Pixel noise as high frequency spatial error. Both HFSE and LFSE are modelled using the 1st order Gaussian-Markov process. HFSE in a star tracker is caused due to dark signal and pixel response non-uniformity errors. LFSE in a star tracker is caused due to imbalances in calibration of sensor optics. FOV and Pixel noise contribute the highest among the mentioned error sources to the platform error. In the next section of the thesis, I will be discussing the factors which contribute to the FOV and Pixel noise of a star tracker.

2 Theory

This section describes the factors which contribute to the pointing error generated by Star Tracker and Reaction wheels, which ultimately lead to the error generated by the overall platform. Once a star tracker and reaction wheels are in operation mode, the error generated by them flows via a system. This system of how the error is translated from the error source to the overall platform will be discussed in the later part of the thesis. The section is further divided into two primary sub-sections which would discuss Star Trackers and Reaction wheels and the theory behind the pointing error generated by them due to the factors which contribute to them.

2.1 Star Trackers

Star Trackers: A star tracker is a sensor which determines the relative position of stars using a system of camera and electronics circuit. The working principle of a star tracker in a simple way can be stated as; that it creates an image/map of stars in the area which it can scan from the spacecraft. There has been extensive study done on finding the accurate positioning of the stars in sky. Hence, this knowledge acts as a validating tool, and this can help in finding the attitude of the spacecraft in reference to the stars. It must also be noted that the star trackers contain a processor which is capable of identifying the stars by assessing the sky with the pre-fed known pattern of stars. An image of a star tracker is mentioned below in figure 3.
Star trackers are considered to be a costly and heavy affair in the space world. Thanks to its method of obtaining the star image and comparing it to an accurate map of pre-determined stars in the sky, Star trackers are currently one of the most accurate sensors in the market. From [10], Star Trackers are digital cameras which have either of the two types of sensors at its focal point CCD or CMOS. CCD sensors are considered to be of a high quality standard as compared to CMOS sensors, this mainly comes down to the technology which is used to reckon the charge generated due to capturing photons into image. For CCD sensors, the electronic charge generated is carried across the sensor chip, then examined at the array corner, furthermore, passed through an analog-to-digital converter to get the digital images. As for the CMOS device, they are based on the technology of the electronic charge being carried by wires after they are amplified by transistors. Figure 4 is a good example of the difference in working of CCD and CMOS sensor.

![Figure 4: CCD and CMOS sensors](image)

As an engineer our main focus is to compare the advantages which each sensor offers and then choose it accordingly. This would essentially mean that the selection would be a tradeoff between the cost and the quality of image produced. The main advantage which CCD sensors have over CMOS is that they produce a high quality image and are less sensitive to noise. One of the other characteristics of CMOS sensor which plays in its disadvantage, is that it has got a lower light sensitivity because the photons might fall on the transistor instead of the photosite. While we discussed the advantages which CCD has over CMOS sensors, it is important to note that it is this technology of CCD as well which makes it more expensive than CMOS sensors and is one key factor to be kept in mind while procuring it. It should also be noted that CCD sensors consume 100 times more power than CMOS and would have to be accounted for in a spacecraft’s power budget. From the above mentioned description, it can be inferred that CCD sensor based Star Trackers are of more advantage in industry.

**Active Pixel Sensor Star Trackers**: APS based Star Trackers have come out as a potential replacement to the existing CCD based star trackers. These are manufactured using existing CMOS technologies. Its lower power consumption, high dynamic range control, Analog-to-Digital conversion and centroiding operations pose significant advantage over their CCD counterparts. On top of these points, the motivation to replace CCD with APS lies within the technology’s simplified hardware, no blooming effect, which means that the star tracker would be able to function properly in the presence of a bright object in its FOV, and
APS technology being more resilient to radiation. Figure 5 gives an idea of how an APS star tracker assembly looks like.

![APS Star Tracker](image)

In the following part of this sub-section, we will be discussing what are the factors which contribute to the star tracker FOV noise and Pixel noise, and how do they generate the total error contribution. From [6] it can be rightly implied that the following factors are responsible and contribute to the FOV and Pixel noise: Detector size, FOV, spacecraft angular velocity, average number of stars tracked, alpha, beta, noise standard deviation for FOV, size of centroiding window, noise standard deviation for pixel and second order damping coefficient.

**Detector size:** The detector size of a star tracker is the overall number of pixels on the sensor surface covering the overall FOV. Another terminology for detector size which is used in the industry while procuring star tracker is the resolution of a star tracker. This factor governs the total amount of light which would be absorbed by the star tracker to further generate the image of the scanned area of sky. After referring various documents of Star Trackers from different manufacturers, it has been observed that currently, the industry has got Star Trackers with detector size of mainly two different dimensions, which is 512 and 1024 pixels. Below table shows this information:

<table>
<thead>
<tr>
<th>Model</th>
<th>Detector size (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTRO-APS3/Jena Optronik</td>
<td>1024</td>
</tr>
<tr>
<td>ASTRO-APS/Jena Optronik</td>
<td>1024</td>
</tr>
<tr>
<td>TI Star Tracker/TERMA</td>
<td>1024</td>
</tr>
<tr>
<td>TERMA and VST-41M/VECTRONIC Aerospace</td>
<td>512</td>
</tr>
</tbody>
</table>

Table 4: Resolution of different Star Trackers

**Sensor FOV:** It is defined as the extent of angle with which the entire sensor of a star tracker would scan the sky. It should also be noted that there is another terminology which is used for the angle subtended by each pixel of a star tracker through which spot scanning of the sky is completed. This angle is called instantaneous FOV or IFOV. From previous experiments and missions, it has been observed that a smaller FOV results in an easy design of the
camera and higher accuracy. However, it will prove to be disadvantageous for the operation of initial attitude determination, as a smaller number of stars would be scanned.\cite{3} From, \cite{14} Star Tracker to satisfy the pointing error requirements for a nanosatellite, range for sensor FOV should be between 15° to 40°.

**Spacecraft angular velocity:** The spacecraft angular velocity is a parameter which depends on the flight operation which is taking place at the particular moment. In the later part of this section, it would be clearer that this parameter should be a non-zero value, and if the spacecraft is at rest, then a small non-zero value should be considered. After a personal conversation with GomSpace AOCS team, I decided that the range for spacecraft angular velocity for any mission stays between 0-1°/s.

**Average number of stars tracked:** The number of stars tracked by a star tracker in the FOV is defined as the average number of stars tracked. Its value is desirable to be as high as possible. This is primarily due to the reason that more stars would be included in the star image generated by star catalog and be helpful in matching with the pre-existing star catalog. The range defined for this factor is between 1-12 stars tracked for a star tracker.

\( \alpha \) and \( \beta \): In order to understand the angles \( \alpha \) and \( \beta \), we need to understand the terminology of cross-axes and boresight axis in a star tracker. The detector of a star tracker is a two dimensional planar array. The axis which essentially is at 90° to the detector frame is called boresight axis. While the axes which are coplanar to the detector frame are termed as cross-axes. Figure \[6\] depicts these axes with better understanding. \( \alpha \) is defined as the angle between the star motion on the detector and the detector reference axes. While \( \beta \) is defined as the angle between boresight axis and the axis of rotation of the spacecraft. Both these angles have a range defined between 0° to 90° for all star trackers.

![Figure 6: Angles subtended with Cross-axis and Bore sight axis](image)

**Noise standard deviation (FOV):** When measuring the attitude using a star tracker, every quaternion measurement gives inaccurate measurement due to the noise. This noise measurement is considered as only white noise. In case of taking the images of static stars, for a star tracker there is a correspondence between frames in the entire sensor FOV of a star tracker. The FOV and the subsequent image of stars do not overlap completely, and hence certain edge
of the image can be ignored. The weighted average method is used, where the sensor FOV is divided into four different areas. The standard deviation of the image noise for the entire FOV which is calculated by adding all four areas is defined as noise standard deviation for FOV. However, when we are dealing with precise pointing requirements, such noise cannot be neglected. This white noise is called low frequency spatial error. LFSE is generated due to calibration imbalance of sensor optics and the error is displayed using 1st order Gaussian-Markov process. Essentially in the process of noise standard deviation for FOV is generated for the time proportional for a star to pass the whole sensor FOV. It should be noted that we get the value of standard deviation of noise in all the three axes for the FOV, and they all contribute to the overall pointing error source of the star tracker. After referring to various set of values of noise standard deviation of FOV for a star tracker, I have decided that the right range of values which would suffice with the industry standards and can be used for the algorithm which I have designed. This value should be between 1-51 arcseconds.

**Noise standard deviation (Pixel):** In case of taking the images of static stars, for a star tracker there is a correspondence between frames in a pixel of a star tracker. The pixels and the subsequent image of stars do not overlap. This would then be solved by weighted average method, where the pixel is divided into four different parts or frames. The standard deviation of the image noise in a single frame is defined as noise standard deviation for pixel. Noise standard deviation for pixel is classified under HFSE. This means that the errors which are generated due to dark signals and pixel non-regularity errors. Noise standard deviation due to pixels is present both in cross-axes and boresight axes and is also considered as white noise. Noise standard deviation due to pixel falls under the category of HFSE. It is calculated over the time it takes for the star to pass one pixel in a star tracker. The range which I have decided for the algorithm in accordance with the industry standards would be 1-7 arcsec for noise standard deviation for cross-boresight axis, and the range for boresight axis would be 8-40 arcsec.

**Size of centroiding window:** This parameter essentially means what is the scalar value of centroiding window present in the star tracker. This allows the engineer to understand the amount of light from stars which would be absorbed by the individual pixel. This scalar value has its units as pixels. From the range for the size of centroiding window should be between 3-7 pixels.

From we can understand how the above mentioned factors which contribute to the pointing error generated by Star tracker FOV and Pixel noise, are mathematically related and contribute to the overall pointing error.

\[
PSD_{totalax} = \sqrt{G_{FOV ax} * G_{FOV ax} + G_{pixel ax} * G_{pixel ax}}
\]  

Where PSD is the power spectral density, generated by Star Tracker.

\[
\begin{pmatrix}
G_{FOV x} & G_{FOV y} & G_{FOV z} \\
(PSD_{FOV x} & PSD_{FOV y} & PSD_{FOV z}) (PSD^*_{FOV x} & PSD^*_{FOV y} & PSD^*_{FOV z})
\end{pmatrix}
\]

\[
\begin{pmatrix}
G_{Pixel x} & G_{Pixel y} & G_{Pixel z} \\
(PSD_{Pixel x} & PSD_{Pixel y} & PSD_{Pixel z}) (PSD^*_{Pixel x} & PSD^*_{Pixel y} & PSD^*_{Pixel z})
\end{pmatrix}
\]

\[
PSD_{FOV ax} = \sqrt{(T_{FOV} * n_{FOV ax})/(1 + s * T_{FOV} / 2)}
\]

10
\[ T_{FOV} = N/\left(v_{star} \times \sqrt{N_{star}}\right) \] (5)

\[ V_{star} = \left(\omega_{s/c} \times N \times \sin \beta \times \cos \alpha\right)/FOV \] (6)

The error generated due to Pixel noise is formulated as below:

\[ PSD_{Pixelax} = \left[(\omega_0)^2 \times \sqrt{TPixel \times n_{Pixelax}}\right]/\left[s^2 + 2\zeta \times \omega_0 \times s + \omega_0^2\right] \] (7)

\[ \omega_0 = 4\zeta/TPixel \] (8)

\[ TPixel = N_{Pixel}/V_{star} \] (9)

\[ PSD_{FOVax}(arcsec/\sqrt{Hz}) = FOV \text{ Power Spectral Density for axis (ax=x,y,z)} \]

\[ PSD_{FOVax}^T(arcsec/\sqrt{Hz}) = \text{Complex conjugate transpose of FOV Power Spectral Density for axis (ax=x,y,z)} \]

\[ \omega_{s/c}(°/s) = \text{Spacecraft velocity} \]

\[ N(pixels) = \text{Detector size} \]

\[ \beta(°) = \text{Bore-sight angle} \]

\[ \alpha(°) = \text{Cross-axes angle} \]

\[ FOV(°) = \text{Field of View} \]

\[ N_{Pixel} = \text{Size of pixel/centroiding window} \]

\[ n_{FOV}(arcsecond) = \text{Noise standard deviation (FOV)} \]

\[ n_{Pixel}(arcsecond) = \text{Noise standard deviation (Pixel)} \]

\[ \eta = \text{Second order damping coefficient} \]

In the above mentioned equations, PSD stands for Power Spectral Density, which is defined as the average power of a signal in the frequency domain. The total PSD is generated over a frequency range. A plot is then created between the frequency values of the specified range and PSD corresponding to every value of the frequency. The area under the curve is then calculated which represents the variance. Star Tracker FOV and Pixel noise error is modelled using zero mean gaussian noise which is calculated by the formula:

\[ Error_{ax} = (\mu + 3\sigma) \] (10)
In the above mentioned equation, \( \mu \) represents the mean, which is zero, this essentially means the error is \( 3\sigma \). The value of \( \sigma \) can be calculated by performing the square root operation of the area under the curve. An algorithm has been designed, which would be discussed in section 3 of this report, wherein based on the value of error, which is required by the mission designer, the factors which would account to that error for a Star Tracker would be generated. These factors would essentially help the company or the engineer in procuring the star tracker from the market required for the satellite and achieve the mission pointing requirements.

2.2 Reaction Wheels

A reaction wheel also referred to as a flywheel is an important component of a satellite, which does not require any means of fuel/propellant in order to exercise its function in a spacecraft. To understand and put forward the working principle of a reaction wheel it can be stated that it operates on the basic physics principle of conservation of angular momentum. The reaction wheel is made up of a mass, which can spin around its axis, and henceforth would have strong amount of inertia present in itself. This primarily implies that there would be an extra mass of the reaction wheel, added to the spacecraft/satellite. Henceforth, this means that the equation which would describe the conservation of the angular momentum would be:

\[
h_{total} = I_{satellite} \cdot \omega_{satellite} + \sum I_{wheeli} \cdot \omega_{wheeli}
\]  

(11)

Due to the presence of vacuum and weightless environment of space, once a satellite is inserted into its orbit, it should maintain the same attitude throughout the mission. This, however, is not the case because of the presence of solar radiation pressure and aerodynamic drag, there is constant perturbation which is induced in the x, y, and z-directions of the satellite. This has to be countered by executing a correcting counter torque, using a reaction wheel. An example of a reaction wheel is shown in figure 7.

Figure 7: Reaction Wheels in a platform

The reaction wheel works on the principle of conservation of angular momentum, with a rotating mass which is connected to an electric motor. The rotational speed or the wheel speed is differed under control. Usually, the speed of the wheel is in thousands of rpm. As discussed above, the rotating mass of a reaction wheel spins on high speed. This would cause micro-vibrations, also known as jittering in the reaction wheel, which would be translated to
the platform and ultimately affect the accurate attitude of a satellite which can cause image distortion. The sources through which reaction wheels contribute to the pointing error of a satellite are Force and Torque caused by wheel jittering and external environment bias i.e., reaction wheel alignment with platform.

Reaction wheel misalignment error is caused due to calibration shortcoming while integrating the reaction wheel onto the platform. This misalignment further results in micro vibrations caused by the reaction wheels during their operational time. As a result of this, it might lead to the target attitude being not achieved precisely by the reaction wheel and furthermore these micro vibrations travel via the control system making the image to be taken by the payload camera distorted, especially for an earth observation satellite. However, it should be noted that these errors do not contribute significantly to the overall jittering induced by the reaction wheels, which could be made clear by the below mentioned table. [18]

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Error contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reaction Wheel control jitter</td>
<td>9.3&quot;</td>
</tr>
<tr>
<td>Reaction wheel imbalance</td>
<td>0.8&quot;</td>
</tr>
<tr>
<td>Environmental noise</td>
<td>1.2&quot;</td>
</tr>
</tbody>
</table>

Table 5: Types of error caused by Reaction Wheels.

From the above table it can be seen and furthermore concluded that the control jitter and wheel imbalance which comprises of the total error contributed by the reaction wheel jittering is 10.1 arcseconds, while the misalignment error is 1.2 arcsecond. This makes it furthermore confirmed that misalignment error caused by reaction wheel contributes less to the platform error than the wheel jittering caused due to operational use of the reaction wheel. Now that it has been discussed about how major contribution to the platform by reaction wheel is due to jittering, the next part of this subsection would describe about how is this jittering error generated and the factors which affect it. In a reaction wheel, the source of jittering is classified into two major groups that is tonal and broadband disturbance.

According to [19], the broadband noise is not contributing significantly to magnitude as much as tonal disturbance. However, during an IV test conducted by NASA, it was found out that the tonal disturbance noise had the largest micro-vibration reaction, but the broadband noise was continuously producing jitter at all wheel speeds. This makes it important to understand the modelling of how both the tonal and broadband noise is done and then understand the factors which contribute to them respectively. In PEET there are two models which are used to model the jittering contributed by reaction wheels, these are the force and torque models.

2.2.1 Reaction Wheel Force model

**Reaction wheel-Force model:** The force jittering model of a reaction wheel would point to the mathematical analysis of all the factors which lead to translational mode jittering along all the three axes of the reaction wheel. The factors which affect this jittering model are mass of the flywheel, speed of the wheel, translation frequency, translation mode damping constant, noise deviation and static imbalance coefficient.
Mass of the flywheel: This represents from the name itself, the mass of the wheel which when in operation, rotates at the specified speed and leads to induction of jittering.

Wheel speed: The speed at which the flywheel is used in operation, in order to change the current attitude of spacecraft and achieve the target attitude.

Translation mode frequency: When a reaction wheel is put into operation at a certain speed, it leads to induction of vibrations due to defects during manufacturing and leads to force being induced in the wheel in all axes at a certain periodic time interval. The reciprocal of this periodic interval at which force is induced is called translation mode frequency. From [20] it can be seen that the interval which would be used for this parameter would be between 380-480Hz

Translation damping ratio: The damping ratio of a system exhibits the oscillations in a system decay after disturbance. In case of a reaction wheel translation model, the disturbance is the vibrations induced due to operational use. The range of this frequency is between 0.15-0.2

Noise standard deviation: From [19] it is seen that broadband noise is the biggest contributor of the overall jitter caused in the reaction wheel. This broadband noise is calculated using the noise standard deviation. Broadband noise is caused due to bearing components and ripple and cogging phenomenon which happens in the motor of the reaction wheel. The noise force which is generated due to broadband noise is modelled by considering it to be white noise. This implies that in PEET it would be modelled as random variable noise component. From [6] the formula for modelling is:

\[
\begin{pmatrix}
S_{RVoutputx} & S_{RVoutputy} & S_{RVoutputz}
\end{pmatrix} = ||H_{inf gain}(f)|| \begin{pmatrix}
S_{RVinputx} & S_{RVinputy} & S_{RVinputz}
\end{pmatrix}
\]

In the above equation \( S_{RVinputx} \) represents the individual value of the noise standard deviation which is being input by the user, for individual axis i.e. x,y and z. \( H_{inf gain} \) is the infinite gain of the transfer function which is used to model the reaction wheel system, and is a scalar value. To understand this, it should be kept in mind that a reaction wheel model for jittering analysis is done by developing it as a mass-spring-damper system. Similarly, \( S_{RVoutputx} \) is the output value of noise standard deviation value for x,y and z axis where ax=(x,y,z).

The second order differential equation which represents the mass spring damper system is:

\[
Mx'' + bx' + kx = F
\]

Now let’s take a laplace transform of the above equation:

\[
Ms^2x(s) + bsx(s) + kx(s) = F(s)
\]

\[
H(s) = x(s)/F(s)
\]

\[
H(s) = k/(Ms^2 + bs + k)
\]
The getPeakGain() function in matlab helps in calculating the infinite gain value of the function. This particular code will be further used in understanding the impact of individual factors on the overall error generated, which will be discussed in the coming chapters.

Once the transfer function from matlab is calculated, it is then multiplied with the noise standard deviation value which is being input by the user. The resulting output represents the \( S_{RVoutput} \).

Since broadband noise is modelled as a white noise, the error is then calculated using zero-mean gaussian white noise model. Considering this, the final value of the error contributed by the broadband noise would be \( 3 \times S_{RVoutput} \). This implies the error for broadband noise as:

\[
Error_{Broadbandnoise_{ax}} = 3 \times S_{RVoutput_{ax}} \tag{17}
\]

**Static imbalance coefficient:** Static imbalance coefficient is a parameter which affects the noise contributed to the reaction wheel from tonal disturbance. It originates from the unevenly split mass of the rotor and the limited reaction wheel assembly tolerance. The static imbalance coefficient is calculated as a product of the uneven mass value and the distance of this mass from the center of gravity of the reaction wheel. It is represented by the symbol \( U_s \).

The range for static imbalance coefficient is found to be between 0.16 to 5 g-cm.

![Static Imbalance process](image)

**Figure 8: Static Imbalance process**

Tonal noise is modelled in PEET as a periodic signal. This means that for a periodic signal the following formula will give the result [6]:

\[
S'_{A_{ax out}} = H_{ax}(f) \times S'_{A_{ax input}} \tag{18}
\]

In the above mentioned formula, \( S'_{A_{ax input}} \) is calculated as the product of static imbalance coefficient with the square of the wheel speed of the flywheel.

\[
S'_{A_{ax input}} = U_s \times (\text{wheelspeed})^2 \tag{19}
\]

The next parameter which is to be discussed is the value of the transfer function at frequency \( f \), which is the frequency at which the flywheel rotates in a reaction wheel. The transfer function which represents the mass spring damper system as previously mentioned is:

\[
H(s) = k/(Ms^2 + bs + k) \tag{20}
\]
\[ s = j\omega \]  \hspace{1cm} (21)

\[ H(f) = \frac{k}{(M(j\omega)^2 + b(j\omega) + k)} \]  \hspace{1cm} (22)

\[ H(f) = \frac{k}{[b^2\omega^2 + (-M\omega^2 + k)^2]} \]  \hspace{1cm} (23)

\( s = \) complex frequency

\( \omega = 2\pi f \)

where \( f \) is the frequency of the wheel speed.

These equations would be of utmost use, in the next chapter, where algorithms which help us understand the effect of each factor on the overall error generated.

\section*{2.2.2 Reaction Wheel Torque model}

\textbf{Reaction wheel-Torque model:} When a reaction wheel is made operational, the jittering induced leads to rocking mode as well, and creates a whirl. This jitter torque produced by the reaction wheel is dependent on factors which would be discussed further down below. From \cite{19}, rocking dynamics of a reaction wheel i.e., dynamics imbalance in a reaction wheel is dependent on rotational inertia of a flywheel perpendicular to spin axis, the total stiffness caused due to bearings, shaft, case, and the effective damping constant. Finally, another factor which is introduced in the modelling is gyroscopic torque and angular wheel spin rate.

Before discussing and understanding these factors individually it is important to learn about the mathematical modelling of all these factors which lead to the final output torque.

\[ I_{rr}\theta'' + c_{\text{rocking}}\theta' + I_{zz}\Omega\theta' + k_{\text{rocking}}\theta = \tau_{ax} \]  \hspace{1cm} (24)

In the above equation, the term \( I_{zz}\Omega \), represents the gyroscopic effect on the torque induced in a reaction wheel. This term can be ignored because \( I_{rr}\omega \gg I_{zz}\Omega \). \cite{23} This would mean that the equation is now transformed to:

\[ I_{rr}\theta'' + c_{\text{rocking}}\theta' + k_{\text{rocking}}\theta = \tau_{ax} \]  \hspace{1cm} (25)

I have formulated a transfer function of the above mentioned equation, to then calculate the tonal and broadband disturbance due to torque effect of a reaction wheel.

\[ H(s) = \frac{1}{(I_{rr}s^2 + c_{\text{rocking}}s + k_{\text{rocking}})} \]  \hspace{1cm} (26)

\textbf{Moment of inertia perpendicular to spin axis (}I_{rr}\text{):} Moment of inertia is the mass analog for linear motion. It is defined as the entity which would decide the amount of torque required for a desired angular acceleration. This quantity is provided by the manufacturer of the reaction wheel. Below figure makes it clear to understand.
Rocking mode frequency: From the above figure it can be observed that during the operation of a reaction wheel, it would rotate about its axis at a certain speed, which would lead to induction of vibrations. This frequency at which the vibrations are induced due to rotation of wheel is called rocking mode frequency. From [19] it can be seen that the interval which would be used for this parameter would be between 290-400Hz.

Rocking mode damping ratio: In case of a reaction wheel rocking model, the vibrations induced due to the whirl induced in a reaction wheel, while in operation furthermore leads to system decay with every passing oscillation, this is rocking mode damping ratio. The range of this ratio is between 0.02-0.15. [24]

Noise standard deviation: This broadband noise is calculated using the noise standard deviation. Broadband noise is caused due to bearing components and ripple and cogging phenomenon which happens in the motor of the reaction wheel. The noise standard deviation remains the same entity for both the force and torque model, the main point of difference is the value of infinity gain for torque model, as the transfer equation is different. The formula to generate the value of broadband noise remains same i.e.

\[
\left( S_{RV \, output \, x} \quad S_{RV \, output \, y} \quad S_{RV \, output \, z} \right) = \left| H_{\text{inf gain}}(f) \right| \left( S_{RV \, input \, x} \quad S_{RV \, input \, y} \quad S_{RV \, input \, z} \right)
\]

To calculate the infinity gain of this function, I have used the same approach as mentioned for the force model of reaction wheel, which is using the peakGain() function of matlab. The final value of the error contributed by the broadband noise would be 3*\( S_{RV \, output} \). This implies the error for broadband noise as:

\[
\text{Error}_{\text{Broadband noise ax}} = 3 \times S_{RV \, output \, ax}
\]

Dynamic imbalance coefficient: Dynamic imbalance coefficient is a parameter which affects the noise contributed to the reaction wheel for rocking mode from tonal disturbance. The dynamic imbalance coefficient is calculated as a product of the uneven mass value and the distance of this mass from the center of gravity of the reaction wheel. It is represented by the symbol \( U_d \). After referring to various research articles the range for dynamic imbalance coefficient is found to be between 1 to 11.09 g – cm\(^2\).
Before we discuss the modelling of tonal disturbance for torque model of a reaction wheel, it is important to note that, from [21] the effect of rocking dynamics on the reaction wheel is negligible in the z-direction and therefore the torque produced in z-direction is considered zero, upon conducting an induced vibration test of a reaction wheel to model the effect of torque on the reaction wheel. The modelling of tonal disturbance for torque model is the same as force model of a reaction wheel and therefore the formula which is used is:

\[ S'_{Aaxout} = H_{ax}(f) \ast S'_{Aaxinput} \]  

(29)

In the above mentioned formula, \( S'_{Aaxinput} \) is calculated as the product of dynamic imbalance coefficient with the square of the wheel speed of the flywheel.

\[ S'_{Aaxinput} = U_d \ast (\text{wheelspeed})^2 \]  

(30)

The next parameter which is to be discussed is the value of the transfer function at frequency ‘f’, which is the frequency at which the flywheel rotates in a reaction wheel. The transfer function which represents the mass spring damper system as previously mentioned is:

\[ H(s) = \frac{1}{I_{rr}s^2 + c_{rocking}s + k_{rocking}} \]  

(31)

\[ s = j\omega \]  

(32)

\[ H(f) = \frac{1}{I_{rr}(j\omega)^2 + c_{rocking}(j\omega) + k_{rocking}} \]  

(33)

\[ H(f) = \frac{1}{[c_{rocking}\omega^2 + (-I_{rr}\omega^2 + k_{rocking})]^2} \]  

(34)

s= complex frequency
\( \omega = 2\pi f \), where f is the frequency of the wheel speed.

The next chapter will discuss the algorithms which are designed by me, to select the type of star tracker and reaction wheel to be selected for a particular mission.
3 Methodology

In this section, the mathematical modelling and the algorithm which I have created would be discussed. The final aim of this section would be to make the reader understand the impact of individual factors on the overall pointing error source.

3.1 FOV and Pixel noise

I have first simplified the equations which help in calculating PSD for FOV noise. For the same after, simplification of the three equations, the following equation was derived:

\[
PSD_{FOV} = \sqrt{\dfrac{[FOV/(\omega_{s/c}\sqrt{N_s}\sin\beta\cos\alpha)]n_{FOV}/(1+s[FOV/(\omega_{s/c}\sqrt{N_s}\sin\beta\cos\alpha)]/2)}{2}) \tag{35}
\]

\[
PSD_{FOV_{ax}}(\text{arcsec}/\sqrt{Hz})= \text{FOV Power Spectral Density for axis (ax=x,y,z)}
\]

\[
\omega_{s/c}(°/s)= \text{Spacecraft angular velocity}
\]

\[
N_s= \text{Number of stars detected}
\]

\[
\beta(°)= \text{Bore-sight angle}
\]

\[
\alpha(°)= \text{Cross-axes angle}
\]

\[
FOV(°)= \text{Field of View}
\]

\[
n_{FOV}(\text{arcsecond})= \text{Noise standard deviation (FOV)}
\]

From the above equation, it can be concluded that the detector size or the resolution of the Star Tracker, does not contribute to the FOV noise. Now that the final formula which includes all the factors which affect the FOV noise is derived, the next step is to design an algorithm which considers them and calculates the PSD\textsubscript{FOV}.

To achieve this, I have designed a monte carlo algorithm. The reason behind this is because the monte carlo method has the capabilities of accepting all the possible values which I have introduced and then generating the corresponding PSD for the same. This would help us in identifying the value of individual factors corresponding to the value of PSD which we have mentioned. This is a key function of the algorithm because it will help in deciding the type of star tracker which the mission requires in order to achieve the required pointing accuracy of the mission.

Before, we discuss the code, which is used to model the algorithm, an important step needs to be addressed. The values of FOV, \(\omega_{s/c}, N_s, \beta, \alpha\) and \(n_{FOV}\) all represent the properties of the Star Tracker and spacecraft. Their values are constant at a particular moment. This means that in PEET and also in the monte carlo algorithm, it is modelled using this concept. Now, the value of complex frequency at \(s=0\) Hz would generate the maximum value of PSD and all the values further from zero up to the maximum value of the frequency range would result in a decreasing value of the PSD generated.
This means that the PSD generated corresponding to \( s = 0 \) Hz generates the worst case value for a PSD. So, with this in mind, the monte carlo algorithm is modelled at \( s = 0 \) Hz, which would give us the values of all the individual factors contributing to the pointing error, and they can be further used to convert into final statistical value of error generated by star tracker.

The code mentioned in appendix A is programmed by me, and explains how the monte carlo algorithm works for FOV noise. The aim of this algorithm is to generate the values of factors which generate the particular value of PSD. When the code is executed successfully it leads to the generation of graph which describes the relation between all the factors and how they contribute to the overall PSD generated based on the user input. The graph which is generated, and which describes the relation is as mentioned below:

![Figure 11: Graph representing relation between all factors for \( PSD_{FOV} \)](image)

In the above graph, which is generated after executing the monte carlo algorithm, it can be seen the effect of individual factors on the overall PSD generated and their inter-dependability. The y-axis represents the total number of PSD generated for the value less than the one set by the user. The x-axis represents the value of all the individual parameters. It can be deduced from the graph, that if we are taking the case where the user inputs a PSD value of less than 36, then spacecraft angular velocity has to be as close to its minimum value in the specified range, field of view value has to be closer to its minimum value of the specified range. 

\( \beta \) follows a trend where the higher value of the angle satisfies the criteria of having PSD which satisfies the user input. \( \alpha \) follows the trend where a greater number of PSD values for the user input condition would be achieved when it is close to its minimum value of the range.

The factor \( N_s \), which represents the number of stars observed by the star tracker, satisfies the user input criteria when it has a value which is closer to its maximum value of the specified range.
range. Finally, the noise standard deviation for FOV, follows the trend where lesser the value it has in its specified range, more would be the number of PSDs which would satisfy the user input.

Now the next part of the code which is in appendix [B] would focus on deciding the PSD generated due to pixel noise. As mentioned in section [2], $PSD_{Pixel}$ value is dependent on many of the same factors which $PSD_{FOV}$ is dependent upon. This would essentially imply that, in the monte carlo algorithm which decides the value of the remaining factors, the previous factors which are common in the formula would be fixed from the previous algorithm. I have simplified the formula for $PSD_{Pixel}$ and it has taken the following form:

$$PSD_{Pixel_{ax}} = \sqrt{(N_p * FOV / (\omega_{s/c} * N * \sin \beta * \cos \alpha) * n_{Pixel_{ax}}) (36)}$$

$PSD_{Pixel_{ax}}(arcsec/\sqrt{Hz}) = \text{Pixel Power Spectral Density for axis (ax=x,y,z)}$

$\omega_{s/c}(^{\circ}/s) = \text{Spacecraft velocity}$

$N(pixels) = \text{Detector size}$

$N_p = \text{Size of pixel/centroiding window}$

$\beta(^{\circ}) = \text{Bore-sight angle}$

$\alpha(^{\circ}) = \text{Cross-axes angle}$

$n_{Pixel_{ax}}(arcsecond) = \text{Noise standard deviation (Pixel)}$

From the above formula, it can be concluded that $N_p$ and $n_{Pixel}$ which represent the size of centroiding window and noise standard deviation for Pixel in a star tracker are the two factors whose value have to be determined by the monte carlo algorithm created for Pixel noise.

When the algorithm for pixel noise is successfully executed, it would generate a graph which would then make it possible to understand the relation of the factors affecting pixel noise. The graph generated is as follows:
From the above mentioned graph, it can be concluded that size of centroiding window i.e. $N_p$, if the value is more towards the upper value of the specified range, there would be a greater number of PSDs which are present for the given user input. And as for noise standard deviation of pixel noise ($n_{Pixel}$) represented by $n$ in the graph, it has to be towards the smaller magnitude value of its range, so that the user input is satisfied.

Another important reasoning which needs to be addressed is the selection of the value for number of samples in the monte carlo algorithm. I have selected 500,000 as the number of samples which have to be generated in the monte carlo. This is because the considering the limitations which I have in terms of computational power of my computer and also ensuring the accuracy of the result generated, I have taken the residual error value should be less than 0.1. This means that, when executed for the first time the algorithm generated a probability of PSD to be less than the input value as 6.618 percentage and when executed for the second time again with the same number of samples, the value of probability generated came to be 6.575%. This very clearly satisfies the condition of probability to be less than 0.1 which made me choose 500,000 as the number of samples for the monte carlo algorithm.

For the monte carlo algorithm which generates value for $PSD_{Pixel}$, the total number of samples selected is 100,000. This is because the value of the total number of samples satisfies the condition for residual error to be less than 0.1. Now that we have discussed how to generate the values of various parameters which define the PSD value for both FOV and Pixel noise, the next step is about converting PSD into the form of statistical error value.

The process of converting the PSD value to error is by calculating the area under the curve. For the same, I have first created the code as mentioned in appendix C to calculate the PSD from the parameters derived from the monte carlo algorithm, which satisfies the user require-
ment and then calculate the error which will be generated from these factors, and in-turn from the star tracker. The parameters $T_{FOV}$, $n_{FOV}$, $T_{Pixel}$ and $n_{Pixel}$ are derived from the monte carlo algorithms, zeta is second order damping constant and it has a fixed value of 0.6 according to [6].

The code mentioned in appendix C finally generates an output in the form of a PSD graph and the statistical value of the error generated. Since, the output which is generated from the code is derived from the parameters which determine the type of star tracker to be used, this would help the engineer in deciding the star tracker which is to be procured.

The output of the code is as follows:

![Power spectral density and error generated](image)

Figure 13: Power spectral density and error generated

Furthermore, to show the impact of every factor which contributes to the overall error generated by Star Tracker, I have generated graphs which would help in understanding the impact of every factor on the error individually, as the previous graphs generated from monte carlo algorithm, describes the inter-dependability of all the factors together. The graphs generated from the code mentioned in appendix D are as follows:
Figure 14: Relation of Number of Stars and Error

Figure 15: Relation between $\omega_{sc}$ and Error
Figure 16: Relation between FOV and error

Figure 17: Relation between $\beta$ and Error
As mentioned in \[2.2.1\] reaction wheel force model has got two contributing factors, which are tonal and broadband noise disturbance. I have designed a monte carlo algorithm which will determine the value of all the parameters which would decide the type of reaction wheel to be used. The monte carlo algorithm is based on the calculation of tonal noise disturbance and also depicts the dependability of factors affecting the jitter in a reaction wheel.

The value 100,000 is used as the total number of values for which the monte carlo will be executed. This value is taken to be 100,000 because the residual error for 100,000 cases is less than 0.1, and thus it can be concluded that this value of the total number of samples will generate the values which are accurate. After successful execution of the code a graph is generated which depicts the relation between all the individual factors which contribute to the value of tonal noise induced, which is input by the user. The graph is as follows:
The above mentioned graph depicts the relation of factors when the user defined condition for tonal disturbance to be 0.11N. From the graph, we can infer that when the wheel speed is on the minimum side of the specified range, there are the greatest number of cases possible for less jittering in a reaction wheel. Mass of the wheel, if is on the minimum side of its range then, jittering induced is minimum. For damping constant and static imbalance coefficient, the co-relation is not clearly visible because of the values of the range being really small in magnitude. So, I have created graphs of individual factors in the later part of this section.

Finally, the radial frequency, the value remains almost constant with the lower end of the range having slightly a greater number of cases where jittering induced is less. Now that the values of factors which contribute to the tonal noise are known, they help us in determining the reaction wheel which the mission engineer needs to choose and helps in determining the tonal disturbance noise. This is because from Sec. 2.2.1 it can be inferred that the same factors are used to calculate tonal and broadband noise for a reaction wheel jittering model. For broadband disturbance, from Sec. 2.2.1 noise standard deviation is a user input and is then multiplied by the maximum/infinity gain of the transfer function. To calculate the maximum gain of the transfer function, I have written a code in MATLAB as mentioned in appendix G which upon execution generates the value of maximum gain.

I have created graphs of individual factors affecting the overall noise generated for both tonal and broadband disturbances, and the graphs are mentioned below:
Figure 20: Relation for broadband noise and error

Figure 21: Relation between Static imbalance and error
3.3 Reaction Wheel- Torque Tonal noise

As mentioned in section 2.2.2 reaction wheel torque model has got two contributing factors, which are tonal and broadband noise disturbance. I have designed a monte carlo algorithm which will determine the value of all the parameters which would decide the type of reaction wheel to be used. It is important to note that, in both the cases, speed of reaction wheel and noise standard deviation is same for both force and torque model, however other factors which are rocking mode damping constant, rocking mode frequency, dynamic imbalance coefficient and moment of inertia changes.

For the algorithm which decides the value of parameters for rocking mode of a reaction wheel, the value 100,000 represents the total number of cases for which the monte carlo is be executed. This value is taken to be 100,000 because the residual error for 100,000 is less than 0.1, and thus it can be concluded that this value will generate the values which would contribute to procuring the reaction wheel accurately.

The algorithm which I have created generates the values of the parameters for a reaction wheel which would have the necessary values contributing to the tonal disturbance for rocking dynamics within the limit set in by the user. The graph generated by the algorithm is as follows:
For broadband disturbance, from section 2.2.2 noise standard deviation is a user input. In order to calculate the value of broadband noise contribution in the total jittering of a reaction wheel, the method remains same as discussed in section 2.2.1.
4 Case Study

4.1 PES to platform error

In this section of the thesis, I will be using the algorithms which I have designed and explained in the previous section, to generate the value of parameters for Star Tracker and Reaction wheels, and then use them as input parameters. I have used PEET, as a tool to generate pointing budget for different platforms with different mission requirements.

In order to generate a pointing budget for a platform which translates pointing error sources to platform error, a system is designed. After the various input parameters for Star Trackers and Reaction wheels are set-in, the reaction wheel force model is passed through a matrix which determines the wheel orientation. For a 3U platform, the reaction wheels are in an orthogonal arrangement and for a 6U platform the wheels are arranged in a tetrahedron arrangement.

This multiplication through the matrix, converts force to torque. Then, the torque generated by dynamic imbalance of a reaction wheel is added to this value generated from multiplication with wheel orientation matrix. Both the error generated from Star Tracker and the total torque are then forwarded to a system.

The error from Star Tracker, which is the attitude error, is then passed through a PID controller, wherein the value of proportional, differential gains has to be input. PID controllers are used very commonly in industry, because the PID structure can be simplified by reducing the gain of proportional, integral or differential gain to be zero. The three gains which I discussed earlier, have another terminology called modes, and thus a PID controller is also called three mode controller. Integral gain \( K_i \) is called reset, differential gain \( K_d \) is known as rate. It should be noted that there is a condition called reset windup, where the error saturates the loop and increases the integral. The values of these gains are \( K_p = 0.03 \), \( K_d = 0.11 \), \( K_i = 0 \). The three parameters represent gain values and are unit-less. This controller then converts the attitude to torque, which is then added to the torque generated by reaction wheels.

Next operation would be to find the effect of this torque on the platform, from the relation:

\[
\tau = I \alpha
\]  

\( I(kg/m^2) \) = Moment of inertia  
\( \alpha(rad/s^2) \) = Angular acceleration

From the above equation could be converted to its transfer equation. 

\[
G(s) = 1/(I s^2)
\]  

Then, from this, the transfer function is multiplied with the function of 1/s and then with the complex conjugate of the G(s). Finally, the output is generated, and this output is platform error.

4.2 Case Study: Earth Observation mission

I have taken up two platforms one being 3U and the other a 6U CubeSat, with the aim of designing the platform with the right hardware, which furthermore satisfies the pointing requirements. For both the cases of 3U and 6U platform, the pointing error in all three axes
is taken to be 0.25°\(^{\circ}\)[27]. This value is the target value of pointing error for the platform, star trackers and reaction wheels would be chosen which would generate error, when translated to platform error would stay within 0.25°.

I have used the algorithms which I have designed and explained in Sec. 3. The values of parameters generated from these algorithms are then used as an input in PEET to generate a pointing budget.

### 4.2.1 EO mission: 3U platform

This particular case is about a 3U platform with 0.25° pointing error in all three axes.\[28\][29] For a 3U platform I have used 3 reaction wheels which are placed in a pyramidal arrangement. The values of pyramidal arrangement along with the moment of inertia of the satellite taken is as below:

Wheel orientation matrix:

\[
\begin{bmatrix}
0.7877 & -0.2113 & -0.5774 \\
-0.2113 & 0.7877 & -0.5774 \\
0.5774 & 0.5774 & 0.5774
\end{bmatrix}
\]

\[I_{sat} = \begin{bmatrix}
0.0067 & 0 & 0 \\
0 & 0.0333 & 0 \\
0 & 0 & 0.0333
\end{bmatrix} \text{ km}^2\]

With these values, I have generated a pointing budget in PEET. The PEET GUI interface which is designed for a 3U platform is as below:

![Figure 24: PEET GUI system for 3U platform](image)

The values of all the parameters for a Star Tracker which are generated by the algorithm are mentioned in the table.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_{sc}$</td>
<td>$0.1007^\circ/s$</td>
</tr>
<tr>
<td>FOV</td>
<td>$30.04^\circ$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$5.064^\circ$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$50.373^\circ$</td>
</tr>
<tr>
<td>$N_s$</td>
<td>3.363</td>
</tr>
<tr>
<td>$n_{FOV}$</td>
<td>33.147&quot;</td>
</tr>
<tr>
<td>$N_P$</td>
<td>6</td>
</tr>
<tr>
<td>$n_{Pixel}$</td>
<td>4.1608&quot;</td>
</tr>
</tbody>
</table>

Table 6: Star Tracker parameters generated by algorithm

The values of all the parameters for a Reaction Wheel which are generated by the algorithm are mentioned in the table.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel speed</td>
<td>2623.44 rpm</td>
</tr>
<tr>
<td>Mass of wheel</td>
<td>5.66 kg</td>
</tr>
<tr>
<td>Translational frequency</td>
<td>477.88 Hz</td>
</tr>
<tr>
<td>Translation damping ratio</td>
<td>0.156</td>
</tr>
<tr>
<td>Static Imbalance coefficient</td>
<td>$6.34 \times 10^{-6} \text{ kg-m}$</td>
</tr>
<tr>
<td>Moment of inertia ($I_{rr}$)</td>
<td>$0.0511 \text{ kg - m}^2$</td>
</tr>
<tr>
<td>Rocking mode frequency</td>
<td>328.517 Hz</td>
</tr>
<tr>
<td>Rocking mode damping ratio</td>
<td>0.121</td>
</tr>
<tr>
<td>Dynamic imbalance coefficient</td>
<td>$1.8069 \times 10^{-6} \text{ kg - m}^2$</td>
</tr>
<tr>
<td>Noise Standard deviation</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 7: Reaction Wheel parameters generated by algorithm

From these Star Trackers and Reaction wheels the pointing error of the platform is generated after using these values as an input in PEET and executing the system designed. The final platform error generated is:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>2'16.2&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>2'16.2&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>2'22.32&quot;</td>
</tr>
</tbody>
</table>

The Worst case scenario error considering the misalignment error:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>2'22.92&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>2'22.92&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>2'29.04&quot;</td>
</tr>
</tbody>
</table>

From the above mentioned tables it can be concluded that the Star Trackers and Reaction Wheels satisfy the pointing requirements of the mission and thus these hardware components can be procured for the mission.
4.2.2 EO mission: 6U platform

In the next Earth observation mission, the platform used is a 6U satellite, with the pointing accuracy to be \(0.25^\circ\). There are two main changes from the previous case. That is the total number of reaction wheels being four which are positioned in a tetrahedron arrangement and the moment of inertia of the platform being different.

Wheel orientation matrix:

\[
\begin{bmatrix}
0.9428 & 0 & 0.3333 \\
0.4714 & 0.8165 & 0.3333 \\
0.4714 & 0.8165 & 0.3333 \\
0 & 0 & 0.3333
\end{bmatrix}
\]

\[
I_{sat} = \begin{bmatrix}
0.026 & 0 & 0 \\
0 & 0.0594 & 0 \\
0 & 0 & 0.086
\end{bmatrix} \text{ kgm}^2
\]

The PEET GUI interface for a 6U platform would be as follows:

![PEET GUI system for 6U platform](image)

The same star trackers and reaction wheels were used to check if they satisfy the pointing requirements of the mission for a 6U platform. The PEET system was executed which then led to the calculation of platform error, the result as follows:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>2'14.1&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>2'27.48&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>2'27.48&quot;</td>
</tr>
</tbody>
</table>

Table 8: Pointing budget for EO mission 3 RWs and Star Tracker

The below mentioned table is the error value for worst case scenario for a 6U platform which is achieved by adding the misalignment error as well.
<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>2°20.82&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>2°34.26&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>2°34.26&quot;</td>
</tr>
</tbody>
</table>

Table 9: Pointing budget for EO mission 4 RWs and Star Tracker

From the above mentioned results it can be concluded that the Star Trackers and Reaction Wheels satisfy the pointing requirements of the mission for a 6U platform and thus these hardware components can be procured for the mission.

4.3 Case Study: Inter satellite link

The next case study which I have taken into account is about establishing a successful communication link between two satellites. In order to do this, both the antennas of the satellites have to be pointed in such a way that the pointing loss is minimum. This implies that the better the pointing accuracy, the higher are the chances of the two antennas being able to establish a communication link successfully.

![Inter Satellite antenna gain](image)

Figure 26: Inter Satellite antenna gain

In the above figure it can be seen that angle 1 and angle 2 combined form the pointing loss which has to be minimized to establish a communication link. The mathematics which helps in understanding the relation between gain and pointing accuracy is: \[30\]

\[
\Delta G = 12\left(\frac{\theta_p}{\theta_{3dB}}\right)^2
\]  

(39)

\(\theta_p\) = Pointing accuracy  
\(\theta_{3dB}\) = Half power beam width

From [31], it can be concluded that \(\theta_{3dB}\) accounts for the beam width of a parabolic antenna. This implies that in order to establish a communication link for an inter satellite mission, it is independent of the distance and instead depends on the diameter of the antenna, along with gain.
\[
\theta_{3dB} = 70 \ast \frac{\lambda}{d}
\] (40)

\(\lambda(\text{metre}) = \text{Wavelength}\)

\(d(\text{metre}) = \text{Diameter of antenna}\)

Considering the above mentioned formulas I have designed two pointing budgets for a gain of 0.088dB, pointing accuracy of 0.1°, frequency of 6GHz and diameter of antenna being 3m. The first mission discusses about the budget with 3 RWs and the next discusses about a budget with 6RWs. The PID controller in PEET, has the value of proportional, differential gains as:

\[K_p:\begin{bmatrix}
581.23 \\
543.97 \\
227.71 \\
1976.2 \\
1849.5 \\
774.2
\end{bmatrix}\]

\[K_d:\begin{bmatrix}
1976.2 \\
1849.5 \\
774.2
\end{bmatrix}\]

4.3.1 Inter satellite mission: 3 RWs

Wheel orientation matrix:

\[
\begin{bmatrix}
1.9445 & 0 & 0 \\
0 & 0.18385 & 0 \\
0 & 0.4 & 1.5
\end{bmatrix}
\]

\[I_{sat}:egin{bmatrix}
4600 & 0 & 0 \\
0 & 4300 & 0 \\
0 & 0 & 1800
\end{bmatrix} kgm^2\]

The values of all the parameters for a Star Tracker which are generated by the algorithm are mentioned in the table.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\omega_{sc})</td>
<td>0.545°/s</td>
</tr>
<tr>
<td>FOV</td>
<td>33.668°</td>
</tr>
<tr>
<td>(\beta)</td>
<td>32.911°</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>48.652°</td>
</tr>
<tr>
<td>Ns</td>
<td>10.9</td>
</tr>
<tr>
<td>n_{FOV}</td>
<td>22.2413&quot;</td>
</tr>
<tr>
<td>N_P</td>
<td>5</td>
</tr>
<tr>
<td>n_{Pixel}</td>
<td>9.772&quot;</td>
</tr>
</tbody>
</table>

Table 10: Star Tracker parameters generated by algorithm

The values of all the parameters for a Reaction Wheel which are generated by the algorithm are mentioned in the table.

36
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel speed</td>
<td>2931.6 rpm</td>
</tr>
<tr>
<td>Mass of wheel</td>
<td>4.14 kg</td>
</tr>
<tr>
<td>Translational frequency</td>
<td>468.875 Hz</td>
</tr>
<tr>
<td>Translation damping ratio</td>
<td>0.152</td>
</tr>
<tr>
<td>Static Imbalance coefficient</td>
<td>$7.51 \times 10^{-6}$ kg-m</td>
</tr>
<tr>
<td>Moment of inertia ($I_{rr}$)</td>
<td>0.0568 kg$\cdot$m$^2$</td>
</tr>
<tr>
<td>Rocking mode frequency</td>
<td>314.037 Hz</td>
</tr>
<tr>
<td>Rocking mode damping ratio</td>
<td>0.098</td>
</tr>
<tr>
<td>Dynamic imbalance coefficient</td>
<td>$5.441 \times 10^{-7}$ kg$\cdot$m$^2$</td>
</tr>
<tr>
<td>Noise Standard deviation</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 11: Reaction Wheel parameters generated by algorithm

The above parameters for Star Trackers and Reaction wheel were generated by the algorithm as mentioned in appendix.

Based on these parameters, the pointing budget is designed using PEET, with the same system, and the result generated is as follows:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1'12.88&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>1'12.85&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>1'12.84&quot;</td>
</tr>
</tbody>
</table>

Table 12: Pointing budget for ISL mission: 3 RWs and Star Tracker

4.3.2 Inter satellite mission: 4 RWs

In this case, the mission target conditions are same, however there is an extra reaction wheel and the reaction wheels are set in a tetrahedron arrangement. With the same Star Tracker and Reaction wheels as used in 4.3, the pointing budget is generated in PEET. The results are as follows:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1'12.92&quot;</td>
</tr>
<tr>
<td>Y</td>
<td>1'12.85&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>1'12.84&quot;</td>
</tr>
</tbody>
</table>

Table 13: Pointing budget for ISL mission: 4 RWs and Star Tracker
5 Scope for further work

The work undertaken in this project, focuses upon the method which can be used to find the right Star Tracker and Reaction Wheels, based on the mission objectives. One of the key areas of focus for further research work could be about coming up with the mathematics and logic for sensors such as radio frequency beacon, sun sensors and magnetometers. This is primarily because, such sensors are cost efficient and could be of utmost use for start-up companies and student satellite projects. A detailed study can be conducted in a lab with how each of this sensor behaves under various conditions, and then a mathematical relation could be drawn among all the factors of these sensors which contribute to the noise generated by them.
References


import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt

def m_c_sim_uniform(init, final, s, sc):
    result = np.array([])
    result = np.append(result, sc * rnd.uniform(init, final, s))
    return np.array(result)

om = m_c_sim_uniform(0, 1, 500000, 1)
fov = m_c_sim_uniform(15, 40, 500000, 1)
b = m_c_sim_uniform(0, 90, 500000, 1)
a = m_c_sim_uniform(0, 90, 500000, 1)
Ns = m_c_sim_uniform(1, 12, 500000, 1)
n = m_c_sim_uniform(1, 51, 500000, 1)

PSD = (np.sqrt(fov / (om * np.sin(np.deg2rad(b)) * np.cos(np.deg2rad(a)) * np.sqrt(Ns)))) * n

# To show values for PSD less than 'val' and the corresponding value of parameters
val = float(input("Enter a value under which the probability is to be found (in float): "))
i = np.array([])
i = np.append(i, np.where(PSD < val))

om_min = np.array([])
fov_min = np.array([])
b_min = np.array([])
a_min = np.array([])
Ns_min = np.array([])
n_min = np.array([])
for j in i:
    om_min = np.append(om_min, om[int(j)])
    fov_min = np.append(fov_min, fov[int(j)])
    b_min = np.append(b_min, b[int(j)])
    a_min = np.append(a_min, a[int(j)])
    Ns_min = np.append(Ns_min, Ns[int(j)])
    n_min = np.append(n_min, n[int(j)])

print ("Minimum value of PSD: ", min(PSD), ", Maximum value of PSD: ", max(PSD))
# To print the probability of PSD occurring less than a particular value

\[
\text{probability} = \left( \frac{\text{np.sum}(\text{PSD} < \text{val})}{\text{len}(\text{PSD})} \right) \times 100
\]

\text{print} \left( \"The probability of PSD to be less than \text{val}, for 500000 samples is = \text{probability}, \%\" \right)

plt.figure(1)
plt.title('Histogram for PSD < ' + str(val))
plt.xlabel("Value of parameters")
plt.ylabel("Number")
plt.hist(om_min, histtype='step')
plt.hist(fov_min, histtype='step')
plt.hist(b_min, histtype='step')
plt.hist(a_min, histtype='step')
plt.hist(Ns_min, histtype='step')
plt.hist(n_min, histtype='step')
plt.legend([\'omega\', \'fov\', \'beta\', \'Alpha\', \'Ns\', \'n\'],loc=9)

B Appendix 2

import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt

def m_c_sim_uniform(init,final,s,sc):
    result = np.array([])
    result = np.append(result,sc*rand.uniform(init,final,s))
    return np.array(result)

Np= m_c_sim_uniform(3,7,100000,1)
n= m_c_sim_uniform(1,10,100000,1)

PSD = (\text{np.sqrt}(\text{Np} \times 30.04 / (0.1007 \times \text{np.sin}(\text{np.deg2rad}(5.064)) \times \text{np.cos}(\text{np.deg2rad}(50.373)) \times 1024))) \times n

# To show values for PSD less than 'val' and the corresponding value of parameters

\text{val} = \text{float}(\text{input}(\"Enter a value under which the probability is to be found (in float): \)}

i = np.array([])
i = np.append(i,\text{np.where}(\text{PSD}<\text{val}))

Np_min=\text{np.array([])}
n_min= \text{np.array([])}
for j in i:
    n_min= np.append(n_min, n[int(j)])
    Np_min= np.append(Np_min, Np[int(j)])

print ("Minimum value of PSD: ", min(PSD), " Maximum value of PSD: ", max(PSD))

# To print the probability of PSD occurring less than a particular value
probability = (np.sum(PSD<val)/len(PSD)) * 100

print ("The probability of PSD to be less than", val ," for 100000 samples is = ", probability, ")

plt.figure(1)
plt.title('Histogram for PSD < ' + str(val))
plt.xlabel("Value of parameters")
plt.ylabel("Number")
plt.hist(Np_min, histtype='step')
plt.hist(n_min, histtype='step')
plt.legend(['Np','n'],loc=9)

C Appendix 3

import numpy as np
import matplotlib.pyplot as plt
from numpy import trapz
Tfov= 2080
nfov= 0.8
eta= 0.6
Tpixel= 21.1
npixel= 12
w= (4*eta)/Tpixel
f= np.arange(0.00001,1000,0.00001)
complex_ls = []
for i in f:
    complex_ls.append(complex(0,6.28*i))
s= np.array(complex_ls)
PSDfov= np.sqrt(((Tfov*(nfov**2))/(1+(s*Tfov/2)**2))
PSDfov_conj= PSDfov.conjugate()
PSDpixel= np.sqrt(((w**4)*(npixel**2)*(Tpixel))/((s**2+ 2*eta*w*s + w**2)**2))
PSDpixel_conj= PSDpixel.conjugate()
Gfov= np.sqrt(PSDfov*PSDfov_conj)
Gpixel= np.sqrt(PSDpixel*PSDpixel_conj)
PSD_final= np.sqrt((Gfov**2+Gfov**2) + (Gpixel**2+Gpixel**2))
PSD_total= PSD_final.real
plt.figure()
plt.semilogx(f, PSD_total) # Bode magnitude plot
plt.figure()
area = trapz(PSD_total, dx=0.0005)
print("Total error generated_", 3*np.sqrt(area))
D  Appendix 4

Ns = 5
Tfov_ls = []
Tpixel_ls = []
eta = 0.6
error_ls = []
x_ls = []

for i in range(0,101,1):
    x = (Ns+(Ns*(i/100)))
    x_ls.append(x)
    Tfov_ls.append(30/(0.004167*np.sin(np.deg2rad(89))*np.cos(np.deg2rad(1))*np.sqrt(x)))
    Tpixel_ls.append((90/(0.004167*np.sin(np.deg2rad(89))*1024*np.cos(np.deg2rad(1)))))

for Tfov,Tpixel in zip(Tfov_ls,Tpixel_ls):
    f = np.arange(0.00001,1000,0.0001)
    complex_ls = []
    for i in f:
        complex_ls.append(complex(0,6.28*i))
    s = np.array(complex_ls)
    PSDfov = np.sqrt(((Tfov*(nfov**2))/(1+(s*Tfov/2)**2)))
    PSDfov_conj = PSDfov.conjugate()
    w = (4*eta)/Tpixel
    PSDpixel = np.sqrt(((w**4)*(npixel**2)*(Tpixel))/((s**2+2*eta*w*s+w**2)**2))
    PSDpixel_conj = PSDpixel.conjugate()
    Gfov = np.sqrt(PSDfov*PSDfov_conj)
    Gpixel = np.sqrt(PSDpixel*PSDpixel_conj)
    PSD_final = np.sqrt((Gfov**2+Gfov**2) + (Gpixel**2+Gpixel**2))
    PSD_total = PSD_final.real
    area = trapz(PSD_total, dx=0.005)
    Error = 3*np.sqrt(area)
    error_ls.append(Error)

E  Appendix 5

import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt

def RW_F_sim_uniform(init,final,s,sc):
    result = np.array([])
    result = np.append(result,sc*rnd.uniform(init,final,s))
    return np.array(result)

Ws = RW_F_sim_uniform(33.33,50,100000,1)
m = RW_F_sim_uniform(1.33,7.55,100000,1)
fr = RW_F_sim_uniform(415,480,100000,1)
er = RW_F_sim_uniform(0.15,0.2,100000,1)
Us = RW_F_sim_uniform(0.16E−05,1.33E−05,100000,1)
\[ F_t = \frac{((m \cdot 6.28 \cdot fr)^2 \cdot Us \cdot (Ws^2))}{np.sqrt((4 \cdot 3.14 \cdot er \cdot fr \cdot m)^2 \cdot (6.28 \cdot Ws)^2 + (-m \cdot (2 \cdot 3.14 \cdot fr)^2) \cdot 2)} \]

# To show values for Force less than 'val' and the corresponding value of parameters

val = float(input("Enter a value, under which the probability is to be found (in float): "))

i = np.array([])
i = np.append(i, np.where(F_t < val))

Ws_min = np.array([])
m_min = np.array([])
fr_min = np.array([])
er_min = np.array([])
Us_min = np.array([])

for j in i:
    Ws_min = np.append(Ws_min, Ws[int(j)])
    m_min = np.append(m_min, m[int(j)])
    fr_min = np.append(fr_min, fr[int(j)])
    er_min = np.append(er_min, er[int(j)])
    Us_min = np.append(Us_min, Us[int(j)])

print("Minimum value of Force: ", min(F_t), 
      "Maximum value of Force: ", max(F_t))

# To print the probability of Force occurring less than a particular value
probability = (np.sum(F_t < val) / len(F_t)) * 100

print("The probability of Force to be less than ", val, "for 100000 samples is ", probability, ")%")

plt.figure(1)
plt.title('Histogram for Force < ' + str(val))
plt.xlabel("Value of parameters")
plt.ylabel("Number")
plt.hist(Ws_min, histtype='step')
plt.hist(m_min, histtype='step')
plt.hist(fr_min, histtype='step')
plt.hist(er_min, histtype='step')
plt.hist(Us_min, histtype='step')
plt.legend(['Ws', 'm', 'fr', 'er', 'Us'], loc=9)

F Appendix 6

import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt
def RW_T_sim_uniform(init, final, s, sc):
    result = np.array([])
    result = np.append(result, sc * rnd.uniform(init, final, s))
    return np.array(result)

Ws = 50
I_rr = RW_T_sim_uniform(0.0001, 0.063, 100000, 1)
fr_d = RW_T_sim_uniform(290, 400, 100000, 1)
er_d = RW_T_sim_uniform(0.02, 0.15, 100000, 1)
Ud = RW_T_sim_uniform(5.0E-07, 11.09E-07, 100000, 1)

F_t_r = ((Ud * (Ws ** 2)) / np.sqrt((4 * 3.14 * er_d * fr_d * I_rr) ** 2 * (6.28 * Ws) ** 2 + (I_rr * (2 * 3.14 * Ws) ** 2 + I_rr * (2 * 3.14 * fr_d) ** 2) ** 2))

# To show values for Force less than 'val' and the corresponding value of parameters
val = float(input("Enter a value under which the probability is to be found (in float): "))
i = np.array([])
i = np.append(i, np.where(F_t_r < val))

I_rr_min = np.array([])
fr_d_min = np.array([])
er_d_min = np.array([])
Ud_min = np.array([])

for j in i:
    I_rr_min = np.append(I_rr_min, I_rr[int(j)])
    fr_d_min = np.append(fr_d_min, fr_d[int(j)])
    er_d_min = np.append(er_d_min, er_d[int(j)])
    Ud_min = np.append(Ud_min, Ud[int(j)])

print("Minimum value of Force: ", min(F_t_r), "Max value of Force: ", max(F_t_r))

# To print the probability of Force occuring less than a particular value
probability = (np.sum(F_t_r < val) / len(F_t_r)) * 100
print("The probability of Force to be less than ", val, "for 100000 samples is ", probability, ")%")

plt.figure(1)
plt.title('Histogram for Force < ' + str(val))
plt.xlabel("Value of parameters")
plt.ylabel("Number")
plt.hist(I_rr_min, histtype='step')
plt.hist(fr_d_min, histtype='step')
plt.hist(er_d_min, histtype='step')
plt.hist(Ud_min, histtype='step')
plt.legend(["I_rr", "fr_d", "er_d", "Ud"], loc=9)
G Appendix 7

\[ G(s) = \text{tf}(157.24,[1,50.24,157.24]) \]
\[ \text{gpeak} = \text{getPeakgain}(G(s)) \]