DP3: Spacecraft with Star Tracker

George Thomas and Varsha Krishnakumar
University of Illinois at Urbana-Champaign

The purpose of this project is to design, implement, test, and experiment with a controller that will enable a spacecraft to maintain a stable, fixed orientation. This will have to be done despite noisy sensor measurements from the star sensor on the spacecraft and an obstacle in the form of a shooting star. This report will discuss the method of linearizing the equations of motion (EOMs) of the spacecraft and the sensor measurements and deriving a stable controller. Next, simulation experiments will be conducted to test the validity of the designed controller. The report will define the requirement and verification to further analyze the experiment results. Also, the experimental method will be explained in detail, which allows other researchers to repeat the experiments. Finally, the results of the experiments will be displayed and analyzed.

I. Nomenclature

ψ	=	yaw angle, rad
ϕ	=	roll angle, rad
θ	=	pitch angle, rad
w_x	=	angular velocity about x axis, rad/s
w_y	=	angular velocity about y axis, rad/s
w_z	=	angular velocity about z axis, rad/s
τ_1	=	wheel 1 torque, N•m
τ_2	=	wheel 2 torque, N•m
τ_3	=	wheel 3 torque, N•m
τ_4	=	wheel 4 torque, N•m
α	=	right ascension of star, rad
δ	=	declination of star, rad

II. Introduction

The concept of keeping a vehicle in outer space as stable as possible is an incredibly important control concept in the field of aerospace engineering. Outer space can be unpredictable, and learning how to stabilize a spacecraft in such an environment is an incredibly useful tool going forward in control design.

For this project, we were given a spacecraft with four reaction wheels, arranged in a pyramid structure. Actuators allow us to apply torques to each of the wheels, up to a maximum value of ±1 N•m. The spacecraft also has a star tracker sensor, which measures the position of each of the stars that are in the spacecraft's field of view.

Our main goal for this project is to keep the spacecraft as stable as possible by keeping it's yaw, roll, and pitch angles as close to 0 rad for as long as possible, even with the obstacles of noisy sensor measurements and a colliding shooting star. The simulation will also break down if any of the stars leaves the field of view of the spacecraft sensor, and if the angular velocities of any of the reaction wheels exceeds ± 50 rad/s. It seems inevitable that at least one of the wheels angular velocities will exceed the value, so our goal is to not have that happen for as long as we can.

The spacecraft has a system of ordinary differential equations that govern the rotational motion of the vehicle, and a set of equations that describe the position of the stars as described by the star tracker sensor. We can linearize these EOMs of the spacecraft and the equations relating to the sensor to derive a stable controller and observer, and use those to run simulations to see how long we can keep the spacecraft stable.

III. Theory

A. State-Space and Sensor Model

The EOMs of the spacecraft are as follows:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = \begin{bmatrix} \frac{w_y \sin\phi + w_z \cos\phi}{\cos\theta} \\ w_y \cos\phi - w_z \sin\phi \\ w_z + w_y \sin\phi \tan\theta + w_z \cos\phi \tan\theta \\ -\frac{55\sqrt{2}\tau_1}{1484} + \frac{55\sqrt{2}\tau_2}{1484} - \frac{150 w_y w_z}{371} \\ -\frac{55\sqrt{2}\tau_3}{1484} + \frac{55\sqrt{2}\tau_4}{1484} + \frac{150 w_x w_z}{371} \\ -\frac{55\sqrt{2}(\tau_1+\tau_2+\tau_3+\tau_4)}{2084} \end{bmatrix} \tag{1}$$

The satellite is equipped with a sensor that measures the position of each star. It can then return that position as a 2D location. In this project, the applicability of our controller in varying star positions within the field of view will be analyzed.

$$\begin{bmatrix} y_{\text{star}} \\ z_{\text{star}} \end{bmatrix} = g(\phi, \theta, \psi, \alpha_0, \delta_0, \alpha_1, \delta_1, \alpha_2, \delta_2, \alpha_3, \delta_3, \alpha_4, \delta_4, \alpha_5, \delta_5, \alpha_6, \delta_6, \alpha_7, \delta_7) \tag{2}$$

In order to implement a controller, the equations of motion are used for a linearized state-space model. A state must be selected to denote the positions, velocities, and other physical details of the system. For this project, the states and inputs can be defined as:

$$\dot{x} = \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix} \qquad u = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} \tag{3}$$

1. Equilibrium Points

Each variable must have an equilibrium point to be linearized about. Correspondingly, the following equilibrium points were chosen:

$$\dot{\psi}_e = 0 \tag{4}$$
$$\dot{\theta}_e = 0 \tag{5}$$
$$\dot{\phi}_e = 0 \tag{6}$$
$$\dot{\omega}_{xe} = 0 \tag{7}$$
$$\dot{\omega}_{ye} = 0 \tag{8}$$
$$\dot{\omega}_{ze} = 0 \tag{9}$$

The time derivatives of these states are set to 0 to force the orientation of the spacecraft to stay motionless and fixed with respect to the stars.

2. Linearization and Controllability

The system can be made easier to solve by linearizing it. This can be done by calculating the Jacobians of the function f with respect to x and u, resulting in A and B matrices respectively.

$$A = \frac{\partial \dot{x}}{\partial \dot{x}_i}(x_e, u_e)$$
$$B = \frac{\partial \dot{x}}{\partial u}(x_e, u_e)$$

(Values of A and B are in the code)

3. Verifying Controllability

To ensure that arbitrary poles can be placed, a controllability matrix W_c can be calculated:

$$W_C = \begin{bmatrix} B & AB & A^2B & A^3B & \dots & A^{n-1}B \end{bmatrix} \tag{10}$$

In the above matrix, n represents the size of the state. In this case, $n = 6$, resulting in a 6x6 W_c matrix. Controllability was tested by calculating the rank of W_c and verifying that it is equal to the size of matrix A. Matrices A and B yield a controllability matrix W_c of rank 6, indicating that system is controllable.

B. Controller and Observer

The LQR method was selected for analysis of the controller. This method allows for close tuning of a number of variables, allowing for adjustment for how the controller will attempt to respond to observed states. Since there is noise from the sensors, finding the correct responsiveness to the inputs is critical.

1. Observer Equations of Motion and Derivation of C

The equations of motion are set as a function of satellite's attitude and the locations of the reference stars. As we are using seven stars, and that each star will have a respective α (right ascension) and δ (declination), there will be 14 measurements in y. We can define a matrix G that combines our equations of motion so that the nonlinear version of y is G.

$$G = g(\phi, \theta, \psi, \alpha_0, \delta_0, \alpha_1, \delta_1, \alpha_2, \delta_2, \alpha_3, \delta_3, \alpha_4, \delta_4, \alpha_5, \delta_5, \alpha_6, \delta_6, \alpha_7, \delta_7),$$
$$\text{with } \alpha_0 = -0.10, \delta_0 = -0.15, \alpha_1 = 0.00, \delta_1 = -0.15, \alpha_2 = 0.10, \delta_2 = -0.15, \alpha_3 = 0.00, \delta_3 = 0.00$$
$$\alpha_4 = -0.10, \delta_4 = 0.15, \alpha_5 = 0.00, \delta_5 = 0.15, \alpha_6 = 0.10, \delta_6 = 0.15$$

To find C, g was linearized by taking its Jacobian around the set equilibrium points with respect to the inputs m:

$$C = \frac{\partial g}{\partial m} \tag{11}$$

For the set of 7 stars, 7 linearizations must be performed to calculate a full C matrix. *(Value of C is in the code)*

2. Verifying Observability

To ensure that our system is observable in its current state, an observability matrix W_o can be calculated:

$$W_O = \begin{bmatrix} C^T & A^TC^T & (A^T)^2C^T & (A^T)^3C^T & \dots & (A^T)^{n-1}C^T \end{bmatrix} \tag{12}$$

As with the controllability matrix, n represents the size of the state, and n=6 for our case. We can just test observability in a similar way we tested controllability: by calculating the rank of W_o and verifying that the rank is equal to n. Our matrices A and C yield an observability matrix W_o of rank 6, showing that our system is observable.

3. Choosing Q and R

The way the LQR controller is tuned is by adjusting the diagonal entries in the Q and R matrices. Adjusting these relative to each other allows for different weights to be placed, thereby controlling each state. A larger relative value for a state leads to a larger deviation from equilibrium.

$$Q = \text{diag}(q_1, q_2, q_3, q_4, q_5, q_6) \qquad R = \text{diag}(r_1, r_2, r_3, r_4) \tag{13}$$

The Q and R matrices were designed to ensure an ideal system. The values were picked through trial and error to have the error be as small as possible.

$$Q = \text{diag}(0.469, 0.469, 0.319, 0.168, 0.243, 0.243) \qquad R = \text{diag}(0.2, 0.2, 0.2, 0.2) \tag{14}$$

4. K Matrix and its Eigenvalues

With the baseline Q and R matrices, the matrix P was solved using a feature from the SciPy library in Python: *linalg.solve_continuous_are*(A, B, Q, R). Then, the gain matrix K was solved for using the equation $K = R^{-1}B^T P$. *(Value of K is in the code)*

5. LQR Observer

Values for the cost matrices Q_o and R_o also need to be picked. Matrix Q_o places an emphasis on the sensor model, whereas matrix R_o places an emphasis on the dynamic model. It was found that placing more weight on the dynamic model led to greater differences between the actual state and estimated state. Correspondingly, by setting the values of R_o lower, the resulting errors were further reduced.

(15)

IV. Experimental Methods

To simulate a spacecraft with reaction wheels and a star tracker, we used the SpacecraftDemo template code provided to the class through the AE353 GitHub [3]. This code included a Simulator class, belonging to Prof. Bretl's spacecraft interface, and a controller class. In addition to the code that ran the spacecraft, there was plotting code that graphed the motion variables associated with the spacecraft as a function of time and that graphed the positions of the stars.

We mainly edited all of the functions inside the Controller class. We changed the initializer function to have access to all the matrices and equilibrium values that we defined earlier so it could use it in the run function. The reset function was also edited to initialize $\hat{x}$, our state estimate. In the run function, we implemented the equation $u = -K\hat{x}$, and were able to extract the torques for the spacecraft's reaction wheels by accessing different elements of the u vector. Before returning the torques in the function, we also update our $\hat{x}$ by using Euler's method for the observer given a time step of $dt = 0.01$ s and an initial guess of $\hat{x}_0 = [0, 0, 0, 0, 0, 0]$.

While the L matrix was kept constant as seen in equation (15), the K matrix was generated randomly using an algorithm detailed in the code associated with the report. The K matrix was selected based on its efficiency in satisfiying this project's set requirements.

The code is written so that during a simulation, if the star tracker on the spacecraft loses track of a star i.e. if one of the stars goes out of the field of view of the sensor, the simulation will immediately stop. However, before we started running simulations, we wanted to define a requirement for our system to satisfy and develop a verification method to verify that our requirement was met:

Requirement:

1) 10 % of all simulations run must last at least 25 seconds before failure. For example, for 500 simulations, 50 simulations must run for at least 25 seconds
2) The yaw, pitch, and roll error must remain below 0.2 radians. This ensures that the spacecraft remains in a fixed orientation for as long as possible.

Each simulation that we run will involve the initial conditions of the yaw, pitch, and roll angles of 0 rad, and angular velocities of 0 rad/s. Each simulation will also be ran with a scope noise of 0.1, since that will be enough to mess with the sensors and test our controller and observer, but not so large that the stars go out of the star tracker very quickly because of inaccurate sensor measurements. Since we do not want the simulation code to take a long time to run, every simulation will also run for 120 seconds each, since that is a large enough time to see how the spacecraft deals with sensor noise.

After the above has been performed, the controller and observer will be tested with sensor noises 0.5, 1, and 5.

Verification: To check that the requirement is fulfilled, we can use code to see that at least 50 out of 500 simulations ran for at least 25 seconds before quitting. We have access to an array of integers that stores how long each simulation ran for before it ended. We can make a loop that goes through the array and counts how many simulations were at least 25 seconds long. If this count is at least 50, that means we have fulfilled our requirement.

V. Results and Discussion

A. Controller (Single Trial)

Firstly, a single trial test was conducted to see if the controller is functional and produces a minimal amount of error. With the previously defined Q, R, Q_o, and R_o matrices, a single trial produced the following data:

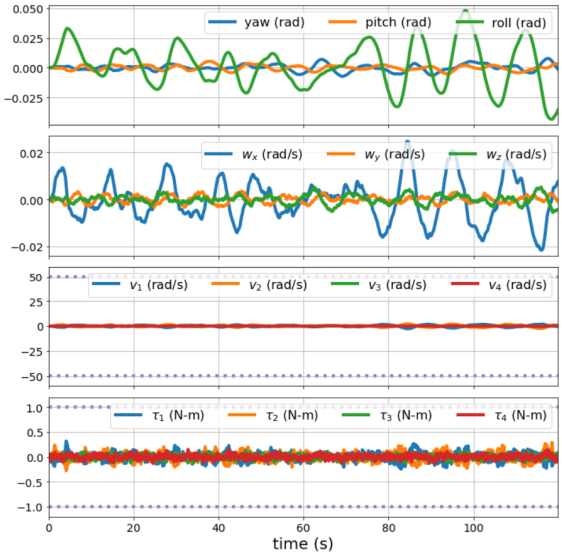

Fig. 1 The First Trial Run Over A 120 Seconds Time Limit.

The yaw, pitch, and roll values seem to stay within the range of [-0.03, 0.05] rad, which is close to 0. Additionally, the torques on the wheels stay within the range of [-0.5, 0.5] N·m.

B. Controller (Multiple Trials)

To further test the limits of this exploration, this controller was applied to a multi-trial run. 1000 simulations of our controller were conducted, and the resulting data was plotted in the following histograms:

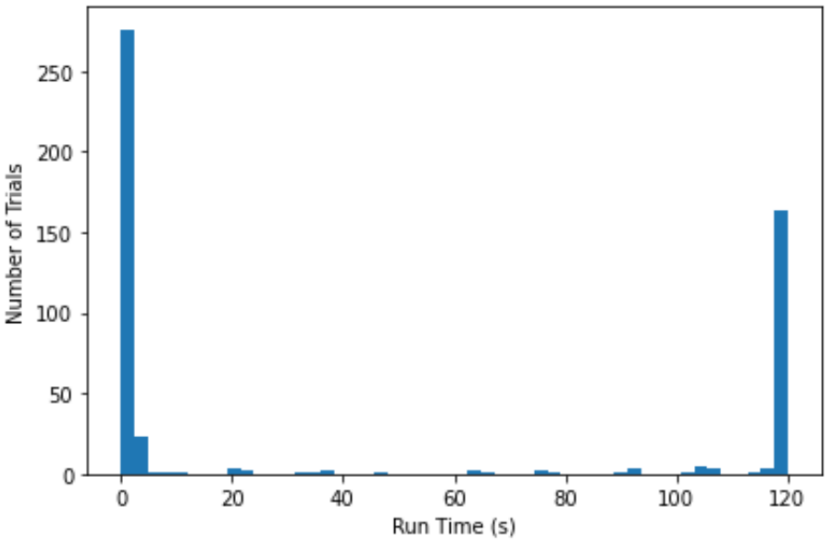

Fig. 2 Simulation Runtime Statistics across 500 simulations

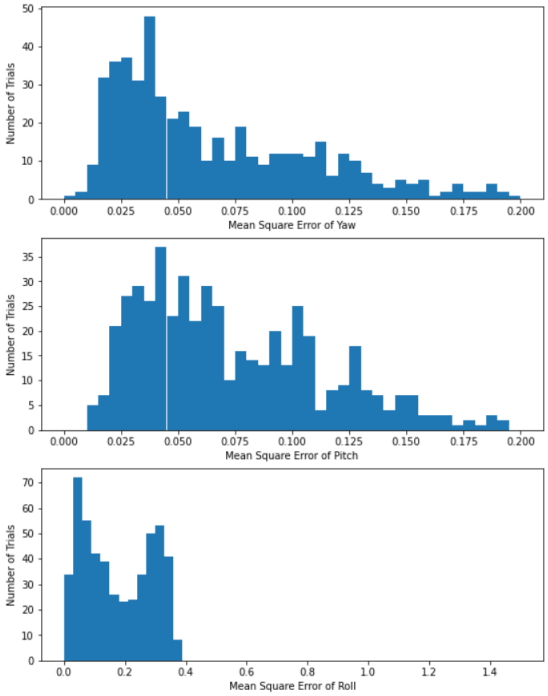

Fig. 3 Mean Squared Error Statistics across 500 Simulations

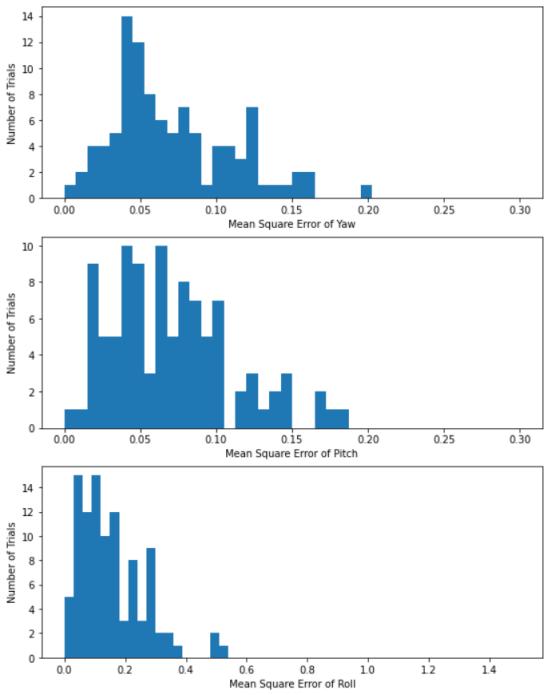

Fig. 4 Mean Squared Error Statistics across 100 Simulations with Sensor Noise = 5.0

As can be seen in Figure 2, the vast majority of simulations had a Mean Square Error of yaw and pitch within the range of [0.0, 0.10] radians. The overall Mean Square Error of roll ranged between [0.0, 0.4] radians. This indicates that there is little variation in these values, and that the controller is efficient in minimizing major orientation change and retaining its fixed orientation.

In Figure 3, it can be seen that even with an increase in the sensor noise, the range of the Mean Square Error of the relevant statistics did not vary to a great extent. The system was also tested for the sensor noises 0.5 and 1.0, which can be viewed in the code associated with this report.

C. Fulfillment of Requirement

After running the simulations, we were able to check whether our verification methods were able to show that we fulfilled our requirement. Running the code that looped through the simulation times, we found that 202 simulations ran for at least 25 seconds before quitting. This more than meets our requirement, which was to get at least 59 simulations out of the 500 we ran that would last at least 25 seconds before becoming unstable.

VI. Conclusion

In the final analysis of our simulation, we can observe that the simulation was a success. We define success in our case as using our verification methods to observe that all our requirements were met after running the experiment.

However, there were definitely ways to improve on our experiments. If we had more time, we could have experimented with more variables that control how stable our simulation is. For example, we could have replicated what we did to find the optimal K matrix to find an optimal L matrix. This would have given us better simulations that ran longer than the ones we ended up with. We could also have experimented with our K optimization method by running more simulations on each viable K matrix, since that would give us more accurate information on which K matrix fits our criteria the best.

Nevertheless, in summation, we were able to successfully linearize the equations of motion of the spacecraft about a set of equilibrium values, use the LQR algorithm to derive a controller and an observer for our system. From there, we defined requirements that we wanted to fulfill and verification methods that would ensure that our requirements were satisfied in simulation. Using the plots and our own python code, we were able to verify that our experiment satisfied our requirements. Thus, we can say that we accomplished what we set out to accomplish from the beginning of this project.

Acknowledgments

G. Thomas thanks Fuad Samhouri and Jake Amoruso for their help in assisting with the content of the paper and in getting started on the paper.

V. Krishnakumar thanks Professor Mariana Silva for learning resources on Sparse Matrices, Conditioning, and Eigenvalues from her class CS 357 (taken Fall 2020).

References

[1] Astrom, K. and Murray, R., 2008. Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press.

[2] Bretl, T., 2022. Reference. [online] AE 353: Aerospace Control Systems. Available at: <https://tbretl.github.io/ae353-sp22/reference> [Accessed 17 February 2022].

[3] Bretl, T., 2022. GitHub - tbretl/ae353-sp22: The website for AE 353 (Spring 2022) at the University of Illinois at Urbana-Champaign.. [online] GitHub. Available at: <https://github.com/tbretl/ae353-sp22> [Accessed 17 February 2022].

[4] Krishnakumar, V., 2022. "Spacecraft with Star Tracker," GitHub Available: https://github.com/varshakrishnakumar/AE-353-Design-Project-3.

Appendix

Day	Task	Person or People
03/21/2022	Creation of Overleaf document, importing of AIAA document format	George Thomas
03/22/2022	Started coding up linearization of EOM in Python notebook, found A, B, and K matrices	George Thomas
03/22/2022	Creation of GitHub repository with ReadtheDocs Documentation and outline of model generating code in Python environment	Varsha Krishnakumar
03/23/2022	Generating LQR matrices for controller and observer and testing various K matrices	Varsha Krishnakumar
03/24/2022	Started filling in Abstract, Nomenclature, and Introduction sections	George Thomas
03/25/2022	Write-up of draft of theory section	Varsha Krishnakumar
03/25/2022	Finished sections, added contributions to appendix table	George Thomas
03/29/2022	Completed draft of final code, including methods for statistical plots	Varsha Krishnakumar
03/29/2022	Fixed mistakes in Introduction and Theory section using feedback	George Thomas
04/01/2022	Completed Experimental Methods section	George Thomas
04/01/2022	Completed draft of Results section	Varsha Krishnakumar
04/08/2022	Completed Conclusion section	George Thomas
04/08/2022	Created logic (code) for LQR matrix generator for the Dynamic Model	Varsha Krishnakumar
04/08/2022	Tested system for different Sensor Noises	Varsha Krishnakumar
04/08/2022	Made edits in the Experimental Methods and Results section	Varsha Krishnakumar
04/08/2022	Recorded and edited project video	George Thomas