**Genetic Satellite Control**

CS-455 Section 1: Artificial Intelligence  
Final Project



S. Chatterjee, A. Henderson, M. Gilliam

AE 443, Harshkumar Patel Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32114-3900

Spring 2020

Submitted on: April 23, 2020

**STATEMENT OF ORIGINALITY:**

The authors of this report hereby affirm by affixing their signatures that all work contained within this document is their own to the best of their knowledge. Anyone else’s work, words, or intellectual property referenced in this work are properly given credit to and cited accordingly. All assistance given to the composition of this document is acknowledged.

                   Sayan Chatterjee

S. Chatterjee

                   Marcell Gilliam

M. Gilliam

        Andrew Henderson

A. Henderson

**TABLE OF CONTENTS**

[**ABSTRACT** 1](#_Toc38574485)

[**APPROACH** 3](#_Toc38574486)

[**CONCLUSION** 8](#_Toc38574487)

[**ACKNOWLEDGEMENTS** 9](#_Toc38574488)

[**REFERNCES** 10](#_Toc38574489)

[**Appendix A: Pseudo Code** 11](#_Toc38574490)

[**Appendix B: Individual Summary** 12](#_Toc38574491)

[*Sayan Chatterjee* 12](#_Toc38574492)

[*Marcell Gilliam* 13](#_Toc38574493)

[*Andrew Henderson* 14](#_Toc38574494)

# **ABSTRACT**

The following report explores the implementation of a genetic algorithm on a satellite in space. For a satellite to operate in space, it is desired to have stable kinematics. To help stabilize a satellite, a controller is introduced to possible act as a damper or thruster. The genetic algorithm was utilized to change the controller value to produce a possible stable satellite in space. With the successful implementation of the system, it was found that the system is capable of stabilizing a satellite, but more constraints are needed to improve the results and performance. **INTRODUCTION**

Satellites play a crucial role today. From the transmission of cellular data to observations made in space, it is hard to deny the importance of satellites. Due to the wide use of satellites, it is important to understand the characteristics of satellites. When a satellite is operating in space, it is very important to have full control of a satellite; otherwise, the satellite will prove to be inoperable. For example, the Hubble Space Telescope would be unable to operate if it was constantly oscillating. The satellite is said to be stable if the angular velocity is zero. The satellite is unstable if it has sporadic angular velocity. A controller could be introduced to help mitigate any instability that could be found. The controller could represent reaction wheels or thrusters to help stabilize the satellite. Tuning such a controller could become a tedious process and having a system capable of tuning itself could prove very useful. In addition, when the satellite is introduced to a disturbance, like a gravity gradient, the controller would need to be tuned again.

The objective of this report is to explore the possibility of controlling the gains of a satellite through machine learning. Such a system would take away human interactions. This objective could introduce a long-term goal of making satellites fully autonomous in the future. It is possible to describe the motion of a satellite by using Euler angles, but it was decided to make the system more encompassing by allowing any equations of motion to be entered in state-space form.

To accomplish the goal of tuning a controller, a genetic algorithm was implemented. Genetic algorithms use the principles of Darwinism to explore possible combinations that could solve the problem in question. Only the fittest “gene” will survive while the other possibilities will be discarded. The fittest gene helps to influence the next generation until the values of population will converge or the maximum iterations of the experiment is met [1].

In the context of this experiment, the genetic algorithm was successful in the implementation. After a certain amount of iterations, an acceptable gain amount was able to be reached. Unfortunately, there was some tradeoffs. Although successful in use, the genetic algorithm was very slow to find the gains. Many iterations were needed to get the best possible value. In the scope of this experiment, it would be considered a success, but in context of the future, more work is needed. It may be detrimental to have a satellite stutter in place when performing calculations for a single case scenario. In addition, when another disturbance is introduced, the satellite would need to be repeated. So, the experiment was successful in design parameters, but needs more refinements for performance. Another tradeoff of the genetic algorithm was the discrepancy of the values. Though the values were acceptable and desired, the system tended to exaggerate the solution. If there is ever a future implementation, there must be more constraints to mitigate the large values obtained.

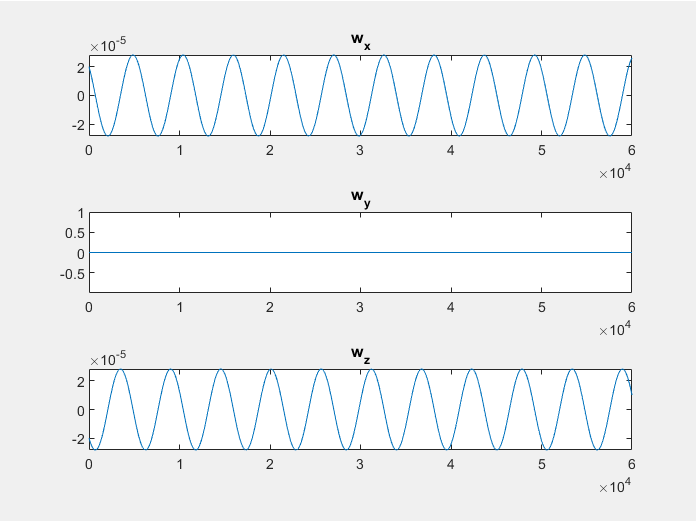


Figure : Example of Possible Angular Velocity of a Satellite

# **APPROACH**

To understand the system, a review of physics is needed. Although the goal is to have the system operating space, the system was designed to take in any equation of motion. Please refer to **Figure 2** for a demonstration.

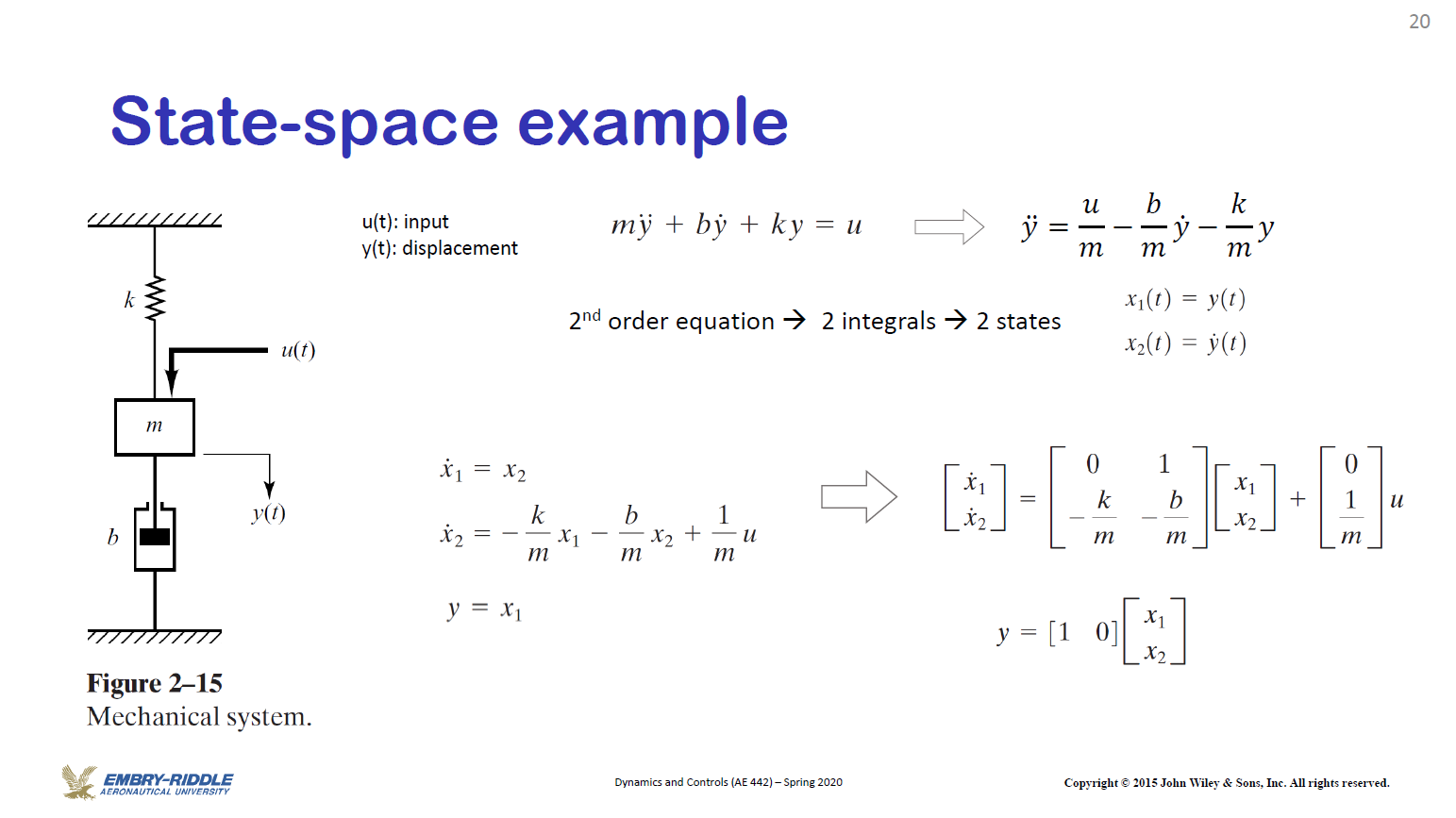


Figure : State-Space Example [2]

The following represents the derivation of state-space equation for a spring mass damper system. The general principle is to start with kinematic equations and formatting the terms into the equations below.

**Equation 2** [5]

**Equation 1** [5]

The variable x represents a defined state space. The values of A, B, C, and D are matrices corresponding to the problem. The value of u represents input of the system. When operating the system, the programme will ask for dimensions and quantity for the A, B, and C matrices. Precautions were done to make sure all the dimensions were the correct size. To simplify the calculations, the D matrix was set to zero.

Mentioned in the previous section, the angular velocity helps to describe the motion of a satellite in space regarding stability. The system can find the angular velocity by converting the state-space equation to a transfer function. In addition, another metric was needed to help find stability. To help understand further, please refer to **Figure 3** regarding the Root Locus.

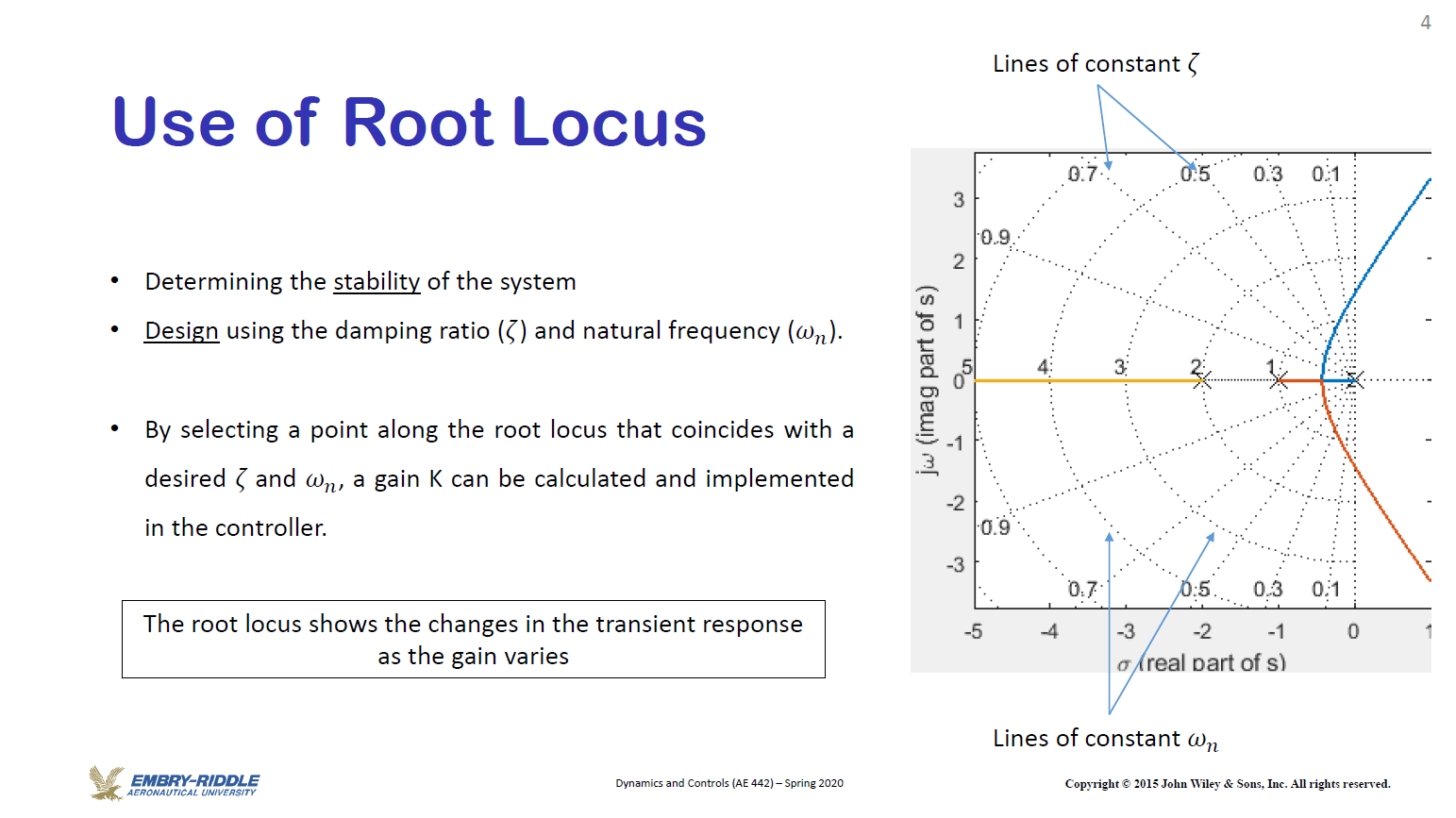


Figure : Root Locus Description [2]

The root locus may be used to analyze the overall system performance. The general rule of thumb is that if the coordinates are on the negative side of the graph, the system may be considered stable. Anything appearing to the right of the graph is unstable. To find the points needed for the root locus, the eigenvalues of the A matrix needs to be calculated. With the combination of the eigenvalues and angular velocity, the overall system stability can be found; helping in decide a fitness metric. When evaluating a gene for the genetic algorithm, a maximum score may be obtained by having a negative eigenvalue and zero angular velocity. In this specific case, each metric will be given 2 points, making the overall score a 4. If the angular velocity is found to be low but nonzero, one point may be awarded. If an eigenvalue of zero is found, a point may be awarded because the satellite is considered marginally stable in that scenario.

The original prospects of the system were designed in MATLAB. In hopes of utilizing machine learning libraries, the system has been transferred to Python. The coding language of Python has many libraries to help aid in the system design. The numpy library was extensively used for matrix algebra. The control library helped to evaluate for the angular velocity and the eigenvalues for the A matrix. In addition, the Control.Matlab library was used to import MATLAB commands into Python [3].

Once fitness was established, there needs to be a way to improve fitness. Referred before, a controller needs to be implemented to help stabilize a system. Recall that the B matrix relates to the input of the system. The system input will be used to stabilize the system. Let the variable K represent controller gain matrix. The K matrix column size needs to match the B matrix row size. In the case of the genetic algorithm, the K matrix values will be randomized. When the controller is implemented, the A matrix will be recalculated from **Equation 3**. Once again, the fitness will be calculated of this new gene. Once done, the genetic algorithm will be running with a specific mutation rate and crossover rate. For this specific implementation, the crossover and mutation rate were 0.90 and 0.035 respectively. For a trivial pseudo code regarding the implementation of the system, please refer to Appendix A.

**Equation 3** [5]

A screenshot of a social media post

Description automatically generated

Figure : General Structure of the System

The following figure above represents the general structure of the system. Jupyter’s Notebook was utilized very heavily in development. Due to the nature of the software, we able to forgo major classes in favor of a top down code with major sections. The first section of the code was regards to the genetic algorithm. These are the basic methods needed to setup generations and introduce breeding. The next class is the satellite class. This is where methods were made for analyzing the state space system. Example methods include finding the eigenvalues or the angular velocity of the satellite. In addition, the fitness calculations were done here. The “Main Method” is referred to where the genetic algorithm and the satellite class are brought together. A state space equation was implemented at random to ensure system consistence when conducting testing. The last section is for user input. It allows for the user to enter their own state-space equation. Special consideration was taken into account to ensure that the matrices were correctly dimensioned when asking for user input.

**RESULTS**

With the code successfully implemented, the results were produced below. It should be noted that the following values are for a very specific state space equation and randomized starting population. A replication of this may produce different results.

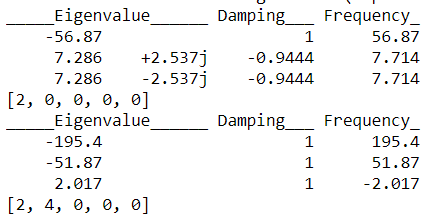


Figure 5: Starting iteration of Genetic Algorithm

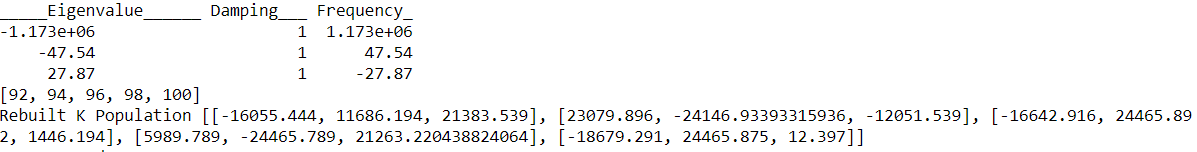


Figure 6: Further Iteration of Genetic Algorithm

**Figure 5** and **Figure 6** demonstrate very specific points of iteration within the code. Inspecting **Figure 5,** it should be noted that the eigenvalues are positive, making the system unstable. As a side note, the damping refers to the natural frequency of the function, or the angular velocity discussed beforehand. In regards to the angular velocity, the system may be considered stable. When it came to the actual implication of the angular velocity, it was decided that the angular velocity does not necessary have to be 0, but at least the threshold of 1. The intention of such a decision was to factor in noise and system precision. After all, any scientific equipment has a sensitivity that could affect precision. Moving towards **Figure 6**, the system has produced very favorable results. The angular velocity reached the threshold of 1, meaning it is not oscillating in space. In addition, the eigenvalues have shifted towards the left of the root locus, signifying the system is starting to stabilize on its own. It yet has to achieve full stability due to the presence of a singular eigenvalue. There could be possible reasons for this result. The first being that the system itself is not fully controllable and therefore stabilizable. There are some state equations that cannot be stabilized regardless of what controller is being used. The second reason could be the number of iterations. For this specific genetic algorithm, there were only 10 generations. Increasing the value could possibly improve the values, but at the trade off of time and resources.

Shifting to overall system performance, it may be argued that the implementation of the genetic algorithm has been successful. Although the ending iteration is not perfect fitness, the system has significantly improved the satellite in space. Hinted earlier in this report, there were some major tradeoffs. The amount of time it took complete this specific implementation of the genetic algorithm was quite high. Though the results may vary, the system took around 2 minutes on average to complete its iterations. Mentioned before, it has already been suggested to increase the iteration to gain a more favorable result, which will significantly increase the time. The time of two minutes may not seem too significant, but in space, there is constant space debris. With the chance of impact being significantly high, the system has to be very responsive and quick to ensure the overall health of the satellite. Finally addressing the last known problem is the overall K values obtained. Referencing back to **Figure 6**, the respective K values are quite high. There was no specific range in regards to the K value, which has become a major oversite of the implementation. To understand why, an anecdotal evidence is needed. Moving outside the scope of satellites, car suspensions may greatly utilize this system with some adaptions. An improper car suspension could produce a lot of oscillations and tend to take a long time to stabilize. To control this scenario, the suspension is introduced to a damping value through either stiffer components or lower ride height. Regardless, such a damping is close to the intention of the K controller for a satellite. A decent amount of damping may decrease oscillation of the vehicle and stabilize the car overall. If the damping was increased beyond this point, it becomes an overdamped scenario. Please refer to **Figure 7** for a visualized example. Though the system may not oscillate, it will translate to a very unpleasant ride experience due to the excessive stiffness. That is exactly what the satellite controller is doing as well. Although the controller was intending to stabilize the system, it caused the overall to reach a very over damped situation. In any case, the goal should be to reach a critically damped scenario. This idea was beyond the scope of the project, and should be considered for future implementations of this system.

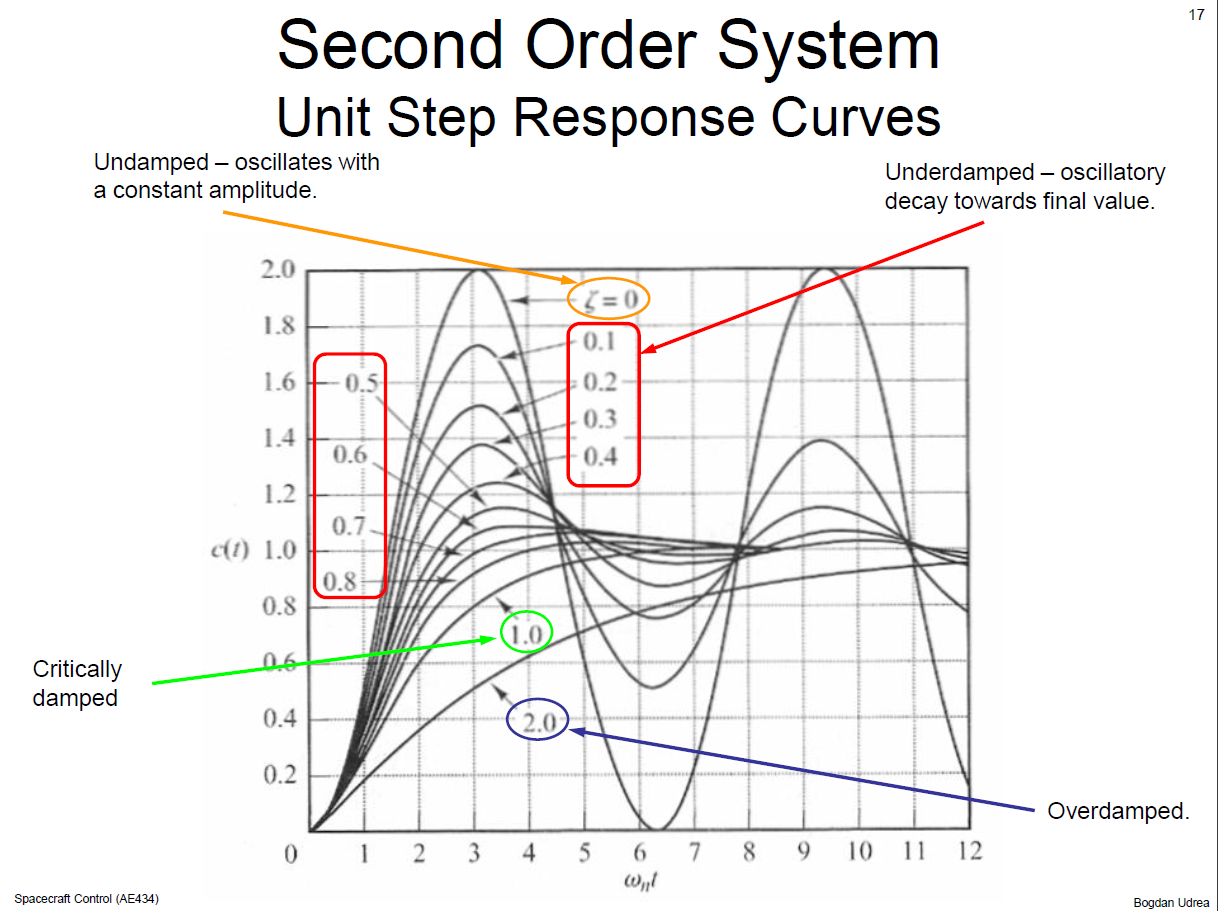


Figure 7: Demonstration of Different Damping Scenarios [4]

# **CONCLUSION**

To end the premise of this report, a genetic algorithm was partially successful in stabilizing a satellite in space. Though the satellite was not able to reach full stability based on the established criteria, the system was able to greatly damp the overall satellite to decrease oscillation. Such action was successful in attempting control, but it was at the cost of speed of the system and over exaggerated values. Overall, the genetic algorithm proved to be a very powerful tool. The genetic algorithm is very random in nature. The starting point is something that is always randomized. In addition, the genetic algorithm is very general. Regardless of what type of scenario a genetic algorithm is being implemented into, the overall premise revolves around Darwinism and survival to the fittest. The only thing that ultimately needs to be changed is the fitness or the performance metric. Otherwise the genetic algorithm was very straight forward to implement. If the number of iterations were increased dramatically, an answer could possibly be always reached. Regarding future implementations, this system needs to be constrained more. In other words, the fitness calculations could be improved. For this specific implementation, the two performance categories was eigenvalues and angular velocity. These two performance metrics should be the bear minimum for the next implementation. A possible performance metric could be settling time or percentage overshoot. These two metrics is more related to overall damping performance of a satellite. Introducing these two metrics produces the drawback of becoming more subjective rather than objective. The original metrics were objective because violations in these metrics would most definitely produce an unstable system. Settling time and percentage overshoot becomes subjective because it mostly depends on specific use scenarios. A satellite could have an excessive settling time, but could still be stable. It depends mostly on the application. Regardless of possible improvements, the genetic algorithm proved to be useful in controlling a satellite in space.

# **ACKNOWLEDGEMENTS**

**Sayan Chatterjee**is credited for the idea behind this report. He was responsible for the control side of the python code and the explanation of the results. He is currently the team lead.

**Marcell Gilliam** is credited for the full implementation of the genetic algorithm. He also assisted in results analysis.

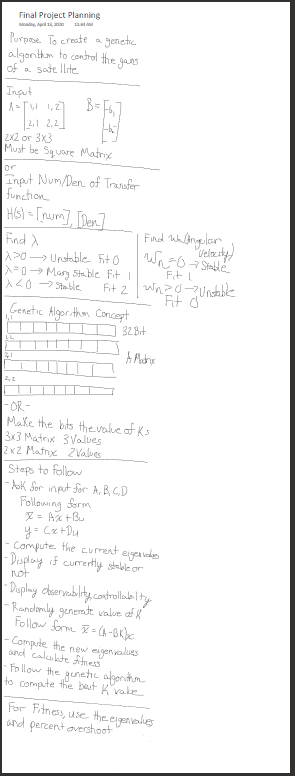
**Andrew Henderson** is credited for help in algorithm implementation.

Special thanks to Dr. Stansbury for his assistance in this project. This report was produced in turbulent times, but help was always provided.

# **REFERNCES**

|  |  |
| --- | --- |
| [1] | R. S. Stansbury, "Genetic Algorithms Background," Embry Riddle Aeronautical University, Daytona Beach, 2020. |
| [2] | H. Y. Moncayo, "Experimental Dynamics and Controls Notes," Embry-Riddle Aeronautical University, Daytona Beach, 2020. |
| [3] | B. Roberto, "Python for Control Purposes," Scuola Universitaria Professionale della Svizzera Italiana, Manno, 2019. |
| [4] | S. Dongeun, "AE434 Spacecraft Control Notes," Embry-Riddle Aeronautical University, Daytona Beach, 2019. |
| [5] | J. Apkarian, P. Karam, A. B. Javid and H. Moncayo, SRV02 Base Unit Experiment for MATLAB /Simulink Users, Markham: Quanser Inc, 2014. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# **Appendix A: Pseudo Code**



# **Appendix B: Individual Summary**

## *Sayan Chatterjee*

This project was mostly my idea. It is pretty evident that I study Aerospace Engineering. I have always had an affinity for satellites when I came to Embry-Riddle. The idea of this project sprouted in my head last semester when I took Spacecraft Controls. My final project of the class involved me tuning the gains of a satellite to produce a stable system. I was successful in the project, but I had one major grip. I had to constantly guess what gain value I needed for the controller to stabilize. Such a task took me almost 2 hours to complete because I had to constantly analysis different scenarios. Around that time, I started to learn more about machine learning due to my experience with Image Recognition with OpenCV. I thought I could simplify the task greatly.

The original final project from my last class was written in MATLAB. The programme MATLAB is very useful for calculations because it has many in built libraries. That became a major hurdle when I was trying to setup this project. My python experience was mostly regulated to writing short scripts in Linux. Upon more research, I found the control library in Python. The library was basically able to replicate all the functions that I needed in MATLAB. It was quite a find, and I started using Python in my Modern Control Design class due to the ease of use. In addition, numpy library in Python is wonderful when performing matrix math. In other languages like Java or C, matrices were essentially just two dimensional arrays. I thought I would have a big issue with dealing with the indexes to do simple matrix math, but Python proved to be a very encompassing language.

Admittingly, I was not as responsible for the genetic algorithm as Marcell. I was mostly regulated to the planning revolving around the project. After all, it was mostly my idea. I setup the project to essentially ask for a state space equation. My intention is to recycle a lot of this code for other courses. I also went about the fitness metric as well. I never had a specific satellite in mind when I was designing the outline of this project. I wanted the code to be as modular as I could make it to apply to different scenario. That is why I went with the eigenvalues and angular velocity. If I kept introducing other factors like settling time or rise time, it was beginning to look complicated. Plus, I had to ensure that the content of the project was able to be understood by the other group members. In regards to the specific genetic algorithm implementation, I mostly revised Gilliam’s work. I made sure the algorithm was solving the context of the problem. I had to play around with the values for crossover and mutation, but my contributions did not stem further from that. I am probably the most excited about the prospects of this project because I was able to relate so much of my classes together.

## *Marcell Gilliam*

I was mostly responsible for the implementation of the genetic algorithm for this project. Truth be told, I was not totally aware of the project details when originally agreeing to join. As a result, I learned quite a bit in regards to controls and satellites. Though I am personally more interested in computer science or cybersecurity, the project was enjoyable for what I have accomplished for myself. I never realized the versatility of python. The project relied heavily on the uses of matrices. Most other programming languages treat matrices as two dimensional arrays, but python was able to do matrix math with relative ease. In addition, I had a chance to use Juptyer’s Notebook in more depth. Originally when I started the class, I was mostly using visual studio, but I slowly decided to make shift to Juptyer. Juptyer’s Notebook proved to be very useful in splitting the code into sections and the overall presentation. Also, I have noticed that Juptyer’s Notebook runs relatively faster than other IDEs I have used in the past.

The genetic algorithm proved to be the biggest learning experience from this project. I originally struggled with the genetic algorithm back in Homework 2 of CS455. The concept was easy to grasp because almost everyone has learned about Darwinism going through public education. The biggest issue could possible stem from my inadequate Python experience. When I originally attempted the homework with another friend, we were trying to approach everything in terms of arrays or lists. There were class examples of better implementations, but I ended up making the genetic algorithm from the ground up. Even though it was a long tedious process, I took away a deep understanding of genetic algorithms. When it came to this project, it became a much easier time to implement. It seemed that the project was much more simpler to understand. The original knapsack problem had a variety of metrics in regards to weight and value, making the binary value pretty large. Due to the simpler nature of this project, I did not have to deal with big binary strings. The main value being chased after was the K matrix. I went ahead and relied more heavily on the in class example. Implementing the genetic algorithm was not the most terrible endeavor. Most of the issues came from my lack of understanding to the scope. Once I actually understood the physics behind the project, it became much easier. Overall, the project went smoothly. The pandemic definitely caused some issues with meeting deadlines, but it was alright at the end.

## *Andrew Henderson*

I was asked to join this project after the idea came about. It was quite a bit of a learning curve to understand what the context of this project was about. Whatever, I learned a lot on the way. My role mostly revolved around basics of principle. I still have a little tough understanding of Python compared to my other languages, but I had a very good fundamental understanding of the genetic algorithm. I mostly helped Sayan to construct the pseudo code used for this project. The genetic algorithm for this project was easier to in concept than our other homework assignments. I think its mostly because the performance metric was not really too strict.

When I was done helping with the planning, I was mostly in charge of the small remedial tasks. I help to construct the report and I did the UML diagram as well. I did have a chance to overview the full code when it was done. I was happy in its successful implementation, but I personally think that another algorithm could have been more useful. When we were originally planning out this project, we tossed around the idea of using tensor flow for this project. This was before we had our lesson about it in class. Ultimately, we decided to throw it out in favor of what we do know. Nonetheless, the project was a great learning opportunity.