Simulation of Spacecraft Damage Tolerance and Adaptive Controls

Scott Nakatani, Capt, USAF
1 Air and Space Test Squadron
806 13th St.
Vandenberg AFB, CA 93437
805-605-7214
scott.nakatani.1@us.af.mil

Timothy Sands, Col, USAF
Air Force Institute of Technology
2950 Hobson Way
Wright Patterson AFB, OH 45433
937-255-3636 x3134
tasands@nps.edu

Abstract—**The nature of adaptive controls, or controls for unpredictable systems, lends itself naturally to the concept of damage tolerant controls in high performing systems, such as aircraft and spacecraft. Recent technical demonstrations of damage tolerant aircraft prove the concept of adaptive controls in an operational environment. Research covered by this paper expands on the topic by discussing the application of adaptive controls to spacecraft and theory behind simulating damage tolerant control implementation. Simulation is then used to demonstrate the stability of adaptive controls when experiencing sudden mass loss and rapid changes in inertia.**

TABLE OF CONTENTS

1. INTRODUCTION

In the 1950s and 1960s, the North American X-15 rocket powered aircraft was pioneering the concepts and principles that would come to define modern powered flight. Among the ground breaking ideas proposed was a system of adaptive controls, or a controller that would take into consideration the changing operational environment to deliver appropriate control to the operator. Limitations of current technology abounded, leaving the X-15 with a successful, but severely limited adaptive control system. Since then, many limitations have fallen away, allowing for the first time employment of adaptive controls on a large scale, especially for space based systems.

Space based operations pose significant operational and technical challenges. The job of maintaining orbit in the space environment is complex, yet operators must also contend with outside threats to their spacecraft. Threats to spacecraft include the space medium itself, conventional weapons, directed energy weapons, and electronic warfare. [1] One threat in particular is growing at an alarming rate; the threat of colliding with other orbiting objects. The advent of space use has introduced more than 39,000 traceable manmade objects into orbit, with over 16,000 large enough to be currently tracked. [2] Of these objects, approximately five percent are functional. [2] This leaves a vast majority of objects orbiting earth that have no means of maneuvering away from a collision.

This population of orbiting objects, including spacecraft, rocket bodies, and debris is continuously growing. The risk of collision in space rises with the number of objects in orbit, and has the potential to create cascading, exponential increases in the number of objects orbiting earth. [3] This increase is best illustrated by the first ever accidental collision between two satellites, Iridium 33 and Cosmos 2251, which left distinct shells of debris across wide orbits. [4] This collision alone produced more than 2,000 traceable objects. [5]

The situation is made worse by the actions of the international community. In 2007, China successfully tested a kinetic kill vehicle against a defunct weather satellite, Fengyun-1C, creating approximately 950 traceable objects and many smaller objects. [6] This debris is generally considered responsible for the sudden breakup of a Russian retroflection satellite in 2013, which was destroyed after passing through the debris cloud left by the Chinese test. [7] As the opportunity for collision increases due to advancing technology and increasing debris, protecting operational spacecraft must become a priority.

Expanding reliance upon space assets and space based capabilities increases the opportunity for and consequences of a disruption. Weapon technology will continue to advance and spread. Orbital debris can take many years to deorbit and only seconds to create. These things will not change. Clearly satellites require greater levels of protection than ever before, and a promising technology may hold some answers for flight within and outside of the atmosphere.

In 2008, the Rockwell Collins Company, sponsored by the Defense Advanced Research Projects Agency (DARPA) demonstrated damage tolerant controls (DTC) in an aircraft scale model, by jettisoning 60 percent of one wing and landing the model, all autonomously. [8] This research, driven largely by increased the United States reliance upon remotely piloted and autonomous aircraft, promises to increase survivability and counteract the input latency inherent to operating an unmanned aerial vehicle (UAV). DTC recovers operational control of a UAV before the operator knows it is damaged, making DTC a tremendous asset. Then, in 2010, Rockwell Collins and DARPA again demonstrated the ability to regain control of a damaged autonomous UAV, this time the operational RQ-7B Shadow, continue the mission, and land successfully. [9] All capabilities were performed via the remaining control surfaces only. This was accomplished using only autonomous software, with modifications called adaptive controls.

Adaptive control is best defined as "…an approach to dealing with uncertain systems or time-varying systems." [10] This infers the use of what is known as an adaptive controller, defined as "…a controller with adjustable parameters and a mechanism for adjusting the parameters." [11] By these definitions, adaptive control is a specific kind of learning system that is generally considered appropriate when a system exhibits time variable dynamics. [12] The demonstration of DTC by the RQ-7B is enabled via the use of adaptive control through an adaptive controller.

Concurrently, with the successes of Rockwell Collins and DARPA, researchers at the University of Illinois successfully modified the programming for a UAV using adaptive controls. [13] Using this modification the UAV was able to search for, and follow a moving ground based target autonomously. These demonstrations represent industry firsts and a tremendous step forward in the use of adaptive controls.

The successful tests by the University of Illinois, Rockwell Collins, and DARPA with autonomous flight and adaptive controls have certain industry parallels. Machine learning has developed many optimized adaptive controls, such as the relatively new reward-weighted regression model. [14] This model was developed specifically to minimize the processing power required, possibly allowing for greater adoption of autonomous machine learning. Adaptive controls have also been designed to adjust for a rapidly shifting center of gravity on light cargo trucks by the Robert Bosch company, dramatically increasing stability and reducing rollover risk. [15] Additionally, adaptive controls are making their way into consumer products in the form of vehicular adaptive cruise control. [16] Adaptive cruise control is performed via the controller maintaining the user set speed of travel and changing to match the speed of slower moving vehicles when encountered. [17] All of these examples utilize adaptive controls to compensate for rapidly changing systems.

Industry success and product availability lends plausibility to the attempt to utilize adaptive controls for space based operations. The advantages of adaptive controls for spacecraft are seemingly boundless. In a study by the University of Florida, adaptive controls are shown to overcome the real world variations in torque seen when spacecraft utilize Control Movement Gyroscope (CMG) gimbals. [18] These variations in torque can make spacecraft attitude control all but impossible, especially for small satellites. Another application is the proposed space tug, which moves between defunct orbiting objects, collects them, and deorbit them. [19] Adaptive controls would enable the tug to compensate for unknown debris mass and maximize its fuel capabilities.

As the space medium becomes more crowded across all orbits, the development of DTC for spacecraft will become a necessity. Adaptive autonomous operations has been called for in order to improve spacecraft survivability. [1] Threats to spacecraft exist at every period of their lifetime; from launch to operations and disposal. DTC and the recent developmental leaps in implementing adaptive controls present a possible answer to these evolving threats. This paper will discuss the history, problems associated with, and a simulated solution for providing damage tolerance via adaptive controls.

The purpose of this research is to simulate a satellite with sudden mass loss and simulate adaptive controls to counteract the loss. DTC would be able to react to the loss much quicker than any human operator, giving the satellite the best opportunity to recover from damage. The question this research seeks to answer is: Are DTC possible for satellites in orbit?

The methodology of this research begins with a literature review of the development of adaptive controls. Following that is a detailed description of the development of the satellite simulation. Last is the data collection and analysis of results.

2. Historical Development

In the 1950s and 1960s, the National Aeronautics and Space Administration (NASA), the United States Air Force (USAF), and the North American X-15 rocket powered aircraft were pioneering the concepts and principles that would come to define modern powered flight. Launched in flight from the wing of a B-52A aircraft, the X-15 would fire its rocket engine, achieve the mission's required altitude, and glide to earth for landing. Among the ground breaking ideas proposed was a system of adaptive controls, or a controller that would take into consideration the changing operational environment to deliver appropriate control to the operator. [20] Primarily what the X-15 proved was the utility of something called Gain Scheduling.

Gain Scheduling is a method of adapting known linear control technique to meet the challenges required of a nonlinear system. [10] By selecting enough points across the changing system and designing linear controls for each portion of the operation, it becomes possible to achieve adaptive nonlinear control. The primary concern when using gain scheduling is the lack of stability that can occur if the system becomes over saturated or departs from the programmed model. In 1967, oversaturated controls of the X-15 that Major Mike Adams was piloting prevented him from regaining control of the aircraft after deviating from the mission trajectory. [21] Major Adams was killed in the resulting breakup, and the X-15 program lost its funding the next fiscal year. NASA reports the lessons from the X-15 were the basis of the flight control system for the Apollo Lunar Excursion Module, and heavily influenced the Space Shuttle program. [21]

Concurrently to the X-15 project, adaptive controls were being proposed and put to the test in many other fields, such as spacecraft control and machine tooling. In partnership with the Bendix Corporation, the USAF experimented with utilizing adaptive controls with feedback for metal cutting, one of the first such attempts. [22] The goal was to account for tool wear in the machining process, both by measurement and by position feedback. At the same time NASA was investigating the possibility of developing adaptive controls for space vehicles, though the stated purpose was to extend the lifetime of spacecraft by minimizing the fuel consumption required to perform the mission. [23] Gain scheduling was heavily favored in these applications, as the required computing power to perform nonlinear adaptive control was yet to come. Gain scheduling requires significant up front processing power, in order to develop all the linear controls needed, which also suffered from the relative lack of computing power of the era.

These projects led naturally to the inclusion of adaptive controls in the development of the International Space Station, to assist with stability during the construction phase and overall attitude control. [24] In this we see the emergence of developing inherently robust controls across the entire mission profile, where only the parameters change between the different phases. Also emerging at this point are many different attempts to limit uncertainty for space based systems, such as the use of Kalman filters, model reference controls, and linear quadratics. [25] Indeed, the realization that slight disturbances can lead to system failure has forced the development these inherently robust controls, and leads developers to calculate uncertainty and feed it into the control parameters adjustment during operations. [26] As late as 1983, the equations required to perform adaptive control for complex space structures were too complicated to be run in real time. [27]

Development up to the 1990s left space operations needing a controller that functions well across a system with changing operational conditions. Serious study at this point focused on utilizing the dominant control method of industry, the proportional integral derivative (PID) controller, which was developed for naval autopilots. [28] The adaptive PID controller works for a wide array of processes without requiring significant characterization of the system, all while outperforming other control methods, such as the generalized predictive control system. [29] This is due in part to the flexibility of the controller, which is demonstrated in the adaptive control scheme used by this simulation. Due to the nature of the PID equation, developers can take specific portions of the controller and adapt them for a specific use, such as the Proportional Derivative controllers proposed for use in space manipulators. [30] PID and PID derivative adaptive controls show significant promise when applied to space based controls.

3. Simulation Methodology

The purpose of the simulation is to answer our research question "Are accurate and stable DTC possible for satellites in orbit?" Simulating ground conditions allows for continuation and validation of any results in the laboratory. The satellite was modeled in SIMULINK and the satellite block diagram is as shown in Figure 1.

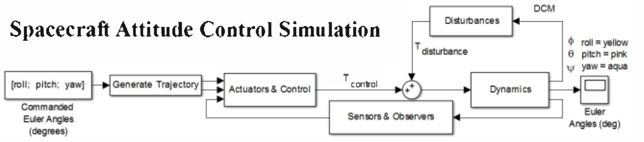

Figure 1. Spacecraft Attitude Control Simulation

The system receives user commanded Euler angles and sends them into the trajectory generator. Trajectory is then sent from the generator to the Actuators and Control block. Inside of the control portion is the ability for the system to select between the PID controller and the modified PID controller, and to enable or disable feed forward control. Leaving the control block is the product of the control selections and the feedback controls, the commanded torques, which are run through the CMG actuators, Dynamics block, and then Kinematics block. The resulting spacecraft Euler angles are finally output from the simulation. Disturbances due to gravity are then fed back into the Dynamics block. Feedback is fed through the Sensors and Observers block, where the system can select between utilizing ideal, sensor, or observer feedback and enable the Kalman and/or low pass filters. The simulation utilizes initial conditions akin to those expected in a laboratory via the model initialization function, which can be found in Appendix A.

Modified PID Controller

The concept of the PID controller comes from the Åström and Wittenmark definition:

$$u(t)=K_C\left(e(t)+\frac{1}{T_I}\int_0^t e(s)ds+T_D\frac{de}{dt}\right) \quad (1)$$

where $u(t)$ is the process or control input, $e(t)$ is the error defined as $e=u_C-y$ where u_C is the reference value, $y(t)$ is the process output, and K_C, T_I, and T_D are constants used to tune the controller). [11] The controller sends torque commands seeking the desired angle to achieve the commanded trajectory. As the torque command is and acceleration, the controller seeks the correct acceleration to produce the desired angle. From Equation 1 we see that the proportional portion depends on current error, the integral portion accumulates past error, and the derivative portion extrapolates future error. The modified PID controller is adapted to accept the selected type of feedback from the feedback block, and outputs the control signal. By passing the derivative action channel of the control to a separate channel, the modified PID controller avoids differentiating noisy signals, increasing control efficiency. The modified PID block diagram is as shown in Figure 2.

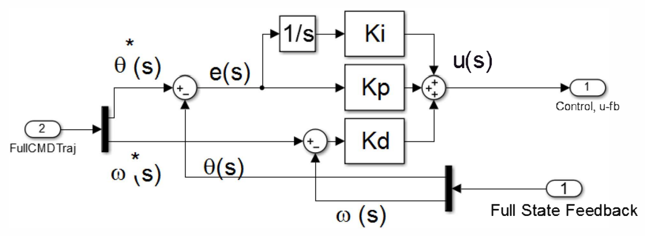

Figure 2. Modified PID Controller

Steering

The spacecraft achieves attitude control via CMG steering, within the CMG Steering subsystem, which is contained in the Actuators and Control subsystem as shown in Figure 1. The simulation uses a minimum, non-redundant CMG array of three CMGs and a balance mass to offset gravity gradient disturbances. The CMG Steering subsystem utilizes the Moore-Penrose Pseudoinverse Steering Law as described by Bong Wie. [31] Wie's description assumes a CMG array as shown in Figure 3.

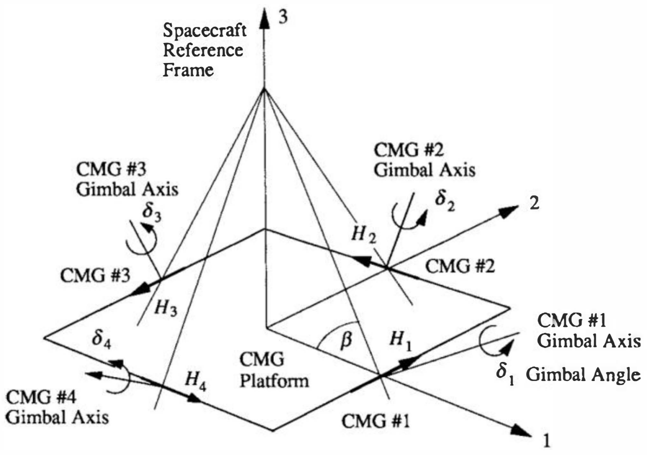

Figure 3. Steering Logic Pyramid Arrangement

The CMG array produces torque by manipulating the angular momentum of the CMGs by changing their angular velocity. The angular momentum vector is defined as:

$$\{h\}=\sum_{i=1}^{4}\{H_i\}\{\delta_i\} \tag{2}$$

$$=\begin{bmatrix}-\cos\beta\sin\delta_1\\ \cos\delta_1\\ \sin\beta\sin\delta_1\end{bmatrix}+\begin{bmatrix}-\cos\delta_2\\ -\cos\beta\sin\delta_2\\ \sin\beta\sin\delta_2\end{bmatrix}+\begin{bmatrix}\cos\beta\sin\delta_3\\ -\cos\delta_3\\ \sin\beta\sin\delta_3\end{bmatrix}+\begin{bmatrix}\cos\delta_4\\ \cos\beta\sin\delta_4\\ \sin\beta\sin\delta_4\end{bmatrix}$$

where β is the pyramid skew angle, $\{H_i\}$ is the angular momentum vector of the ith CMG, and $\{\delta_i\}$ is the gimbal angle of the ith CMG. Taking the time derivative of the angular momentum vector results in the equation:

$$\begin{aligned}\{\dot{h}\} &= \sum_{i=1}^{4}\{\dot{H}_i\}\\ &=\sum_{i=1}^{4}\{a_i\}\{\delta_i\}\{\dot{\delta}_i\}\\ &=[A]\{\dot{\delta}\}\end{aligned} \tag{3}$$

where $\{\delta\}$ is the gimbal angle vector, $\{a_i\}$ is the ith column of $[A]$, and $[A]$ is the Jacobian matrix defined as:

$$[A]=\begin{bmatrix}-\cos\beta\cos\delta_1 & \sin\delta_2 & \cos\beta\cos\delta_3 & -\sin\delta_4\\ -\sin\delta_1 & -\cos\beta\cos\delta_2 & \sin\delta_3 & \cos\beta\cos\delta_4\\ \sin\beta\cos\delta_1 & \sin\beta\cos\delta_2 & \sin\beta\cos\delta_3 & \sin\beta\cos\delta_4\end{bmatrix} \tag{4}$$

For a commanded control torque input $\{u\}$, the CMG momentum rate command $\{\dot{h}\}$ is defined as:

$$\{\dot{h}\}=-\{u\}-\{\omega\}\times\{h\} \tag{5}$$

where $\{\omega\}$ is the angular velocity. Finally the Moore-Penrose Pseudoinverse Steering Law, also known as pseudoinverse steering logic, may be obtained as:

$$\begin{aligned}\{\dot{\delta}\} &= [A]^{+}\{\dot{h}\}\\ &= [A]^{+}\left(-\{u\}-\{\omega\}\times\{h\}\right)\end{aligned} \tag{6}$$

where $[A]^{+}$ is the pseudoinverse of $[A]$ if:

$$A^{+}=A^{T}\left(AA^{T}\right)^{-1} \tag{7}$$

$$AA^{+}A=A \tag{8}$$

$$A^{+}AA^{+}=A^{+} \tag{9}$$

$$\left(AA^{+}\right)^{*}=AA^{+} \tag{10}$$

$$\left(A^{+}A\right)^{*}=A^{+}A \tag{11}$$

where $A=[A]$. Generally speaking, the pseudoinverse $[A]^+$, is the conjugate transpose of $[A]$, a relationship otherwise known as Hermitian matrices. As the simulation uses only 3 CMGs, $[A]$ in the simulation is a 3×3 square, real matrix. Therefore, the conjugate transpose of $[A]$ is equal to the inverse of $[A]$ and the pseudoinverse of $[A]$:

$$A^*=A^{-1}=A^+ \tag{12}$$

for the simulation's specific case only. The subsystem takes the controller commanded torque and removes the current torque, giving the actual torque command. This is multiplied with $[A]^+$ to find the commanded gimbal rate, which is in turn sent to the gimbal actuators. Integrating the gimbal rate gives the gimbal state, which is an input of $[A]$. The subsystem outputs angular momentum rate or torque rate after multiplying $[A]$ with the gimbal rate. The CMG Steering subsystem is as shown in Figure 4.

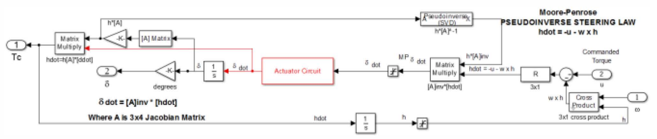

Figure 4. CMG Steering

Dynamics

The Dynamics section was derived from the law of conservation of angular momentum, and the definition of angular momentum:

$$\{H_S\}=[I]\{\omega\} \tag{13}$$

where $\{H_S\}$ is the spacecraft angular momentum, $[I]$ is the moment of inertia matrix, and $\{\omega\}$ the spacecraft angular velocity. From the definition of angular momentum comes Euler's second law of motion:

$$\{M_{IN}\}=\frac{d\{H_S\}}{dt} \tag{14}$$

Where $\{M_{IN}\}$ is the sum of the external moments of force, or torques around an axis in the inertial frame. Euler's second law as applied to rotating frames is:

$$\{M_{ROT}\}=\frac{d\{H_S\}}{dt}+\{M\} \tag{15}$$

where $\{M_{ROT}\}$ is the sum of the torques in the rotational frame and $\{M\}$, the rotation of the body frame, is defined as [32]:

$$\{M\}=\{\omega\}\times\{H_S\}\,. \tag{16}$$

From change in angular momentum we use Equation 13 to find change in angular velocity and angular velocity, outputting both. The spacecraft dynamics block diagram is as shown in Figure 5.

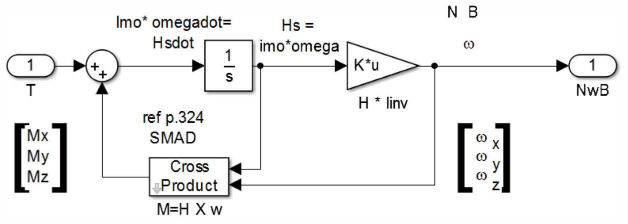

Figure 5. Spacecraft Dynamics

Kinematics

The Kinematics section begins with the angular velocity of the spacecraft, $(\omega_x,\omega_y,\omega_z)$, and adds the angular velocity of the spacecraft's orbit, resulting in $(\omega_1,\omega_2,\omega_3)$, the angular velocity with respect to orbit. This angular velocity is transformed into a quaternion, which is a specific type of complex number, by the relationship:

$$[\dot{Q}]=\frac{1}{2}\{\Omega\}\otimes[Q] \tag{17}$$

where $[Q]$ is the quaternion, $[\dot{Q}]$ is the rate of change of the quaternion, and $\{\Omega\}$ is the angular velocity with respect to orbit. Expanded, Equation 17 becomes the kinematic differential equation for quaternions:

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\\ \dot{q}_4\end{bmatrix}=\frac{1}{2}\begin{bmatrix}0 & \omega_3 & -\omega_2 & \omega_1\\ -\omega_3 & 0 & \omega_1 & \omega_2\\ \omega_2 & -\omega_1 & 0 & \omega_3\\ -\omega_1 & -\omega_2 & -\omega_3 & 0\end{bmatrix}\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix} \tag{18}$$

where q_1 is the scalar part, the values q_2, q_3, q_4 make up a vector part, and $[\ ]$ is a matrix multiplication skew-symmetric version of the cross product in Equation 17 [31] One benefit of using Equation 18 to find the quaternion is that a quaternion calculated via this method will always be normalized. [31] Once the quaternion $[Q]$ is calculated, the direction cosine matrix (DCM) is calculated via the relationships:

$$\{P\}'=[Q]\{P\}[Q]^* \tag{19}$$

$$[Q]^*=\begin{bmatrix}q_1\\ -q_2\\ -q_3\\ -q_4\end{bmatrix} \tag{20}$$

where $\{P\}$ is the point subject to rotation as described by quaternion $[Q]$, $\{P\}'$ is the resulting point, and $[Q]^*$ is the conjugate quaternion [33] When Equation 19 is expanded, the following DCM can be extracted: [34]

$$[DCM]=\begin{bmatrix} 1-2\left(q_2^2+q_3^2\right) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_2q_1-q_3q_4) & 1-2\left(q_1^2+q_3^2\right) & 2(q_2q_3+q_1q_4) \\ 2(q_3q_1+q_2q_4) & 2(q_3q_2-q_1q_4) & 1-2\left(q_1^2+q_2^2\right) \end{bmatrix}. \quad (21)$$

The quaternion block is required in order to avoid singularities during the simulation that result when using Euler angles alone. The Apollo program's inertial measurement unit was highly susceptible to singularities thanks to using Euler angles without quaternions and three axis gimbals, leading to dangerous losses of position accuracy during several missions. [35] The block diagram of the quaternion and DCM calculations is as shown in Figure 6.

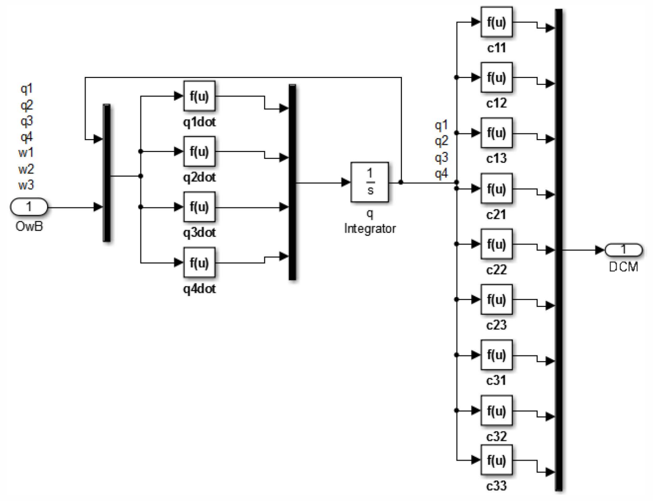

Figure 6. Quaternion and DCM Calculation

The DCM is also defined by writing out the result of the three body axis rotations:

$$[DCM]=\begin{bmatrix} c(\theta)c(\psi) & c(\theta)s(\psi) & -s(\theta) \\ s(\phi)s(\theta)c(\psi)-c(\phi)s(\psi) & s(\phi)s(\theta)s(\psi)-c(\phi)c(\psi) & s(\phi)c(\theta) \\ c(\phi)s(\theta)c(\psi)+s(\phi)s(\psi) & c(\phi)s(\theta)s(\psi)-s(\phi)c(\psi) & c(\phi)c(\theta) \end{bmatrix} \quad (22)$$

where $c(x)=\cos(x)$, $s(x)=\sin(x)$, and assuming the coordinate transformation $\left[C_\phi\right](\phi)\leftarrow\left[C_\theta\right](\theta)\leftarrow\left[C_\psi\right](\psi)$ where $[C_i](x)$ is the rotation matrix about the ith axis with an angle of x. [34] Having used Equation 21 to populate the DCM with values, the simulation uses Equation 22 to find the corresponding Euler angles. Pitch, or θ, can be found by taking the value of the first row, third column of the DCM, DCM_{13}, applying the corresponding portion of Equation 22, and then isolating θ:

$$DCM_{13}=-\sin(\theta)$$

$$-\sin^{-1}(DCM_{13})=-\sin^{-1}(-\sin(\theta)) \quad (23)$$

$$\theta=-\sin^{-1}(DCM_{13}).$$

With θ found, the remainder of the Euler angles can be similarly isolated and solved:

$$\phi=\sin^{-1}\left(\frac{DCM_{23}}{\cos(\theta)}\right) \quad (24)$$

$$\psi=\sin^{-1}\left(\frac{DCM_{12}}{\cos(\theta)}\right) \quad (25)$$

where DCM_{xy} is the element in row x, column y of the DCM. The block diagram of the Euler angle generator is as shown in Figure 7.

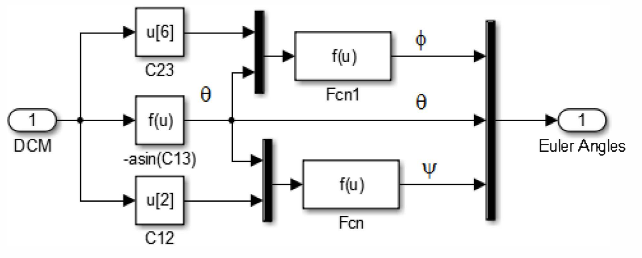

Figure 7. Euler Angle Generator

After calculating the Euler angles, they are output from the Simulation. The block diagram of the Kinematics section is as shown in Figure 8.

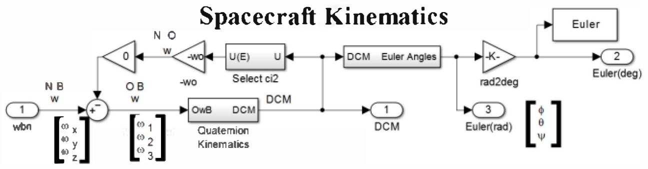

Figure 8. Spacecraft Kinematics.

Disturbances

The only disturbances modeled were the gravity gradient torque, or the torque due to unequal gravitational pull on the disparate parts of the satellite. This disturbance is expected to be observed in a laboratory follow up of this simulation. Gravity gradient torque is defined as:

$$\{T_{gg}\}=\frac{3\mu}{R_0^3}\{u_e\}\times([I]\{u_e\}) \quad (26)$$

where $\{T_{gg}\}$ is the gravity gradient torque, μ is the Earth's gravitational coefficient, R_0 is the distance from Earth's center, $\{u_e\}$ is the unit vector towards Nadir, and $[I]$ is the spacecraft moment of inertia matrix. [32] The unit vector towards Nadir is picked out of the DCM:

$$\{u_e\}=\begin{bmatrix} DCM_{13} \\ DCM_{23} \\ DCM_{33} \end{bmatrix}. \tag{27}$$

The gravity gradient torque is output to the Dynamics Block, and the block diagram is as shown in Figure 9.

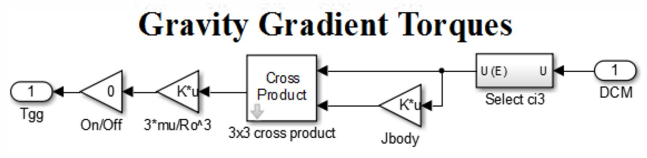

Figure 9. Gravity Gradient Torques

Sensors and Observers

The enabling concept behind the feedback capabilities of the simulation is contained in the Sensors and Observers block. Euler angles and angular rates enter, combined in a state vector, which can be shunted along different paths, from an unedited ideal full state feedback to sensor feedback, observers, state feedback, Kalman filtered sensors, and Lowpass filtered sensors. Once the simulation selects anything other than ideal full state feedback, a sensor is simulated. The sensor is the ideal feedback signal with a random number generator adding noise. Next in the process flow is the Observer Subsystem, containing a Luenberger observer and a PID observer. The purpose of any observer is to create a complete state vector from incomplete and/or noisy observations. The PID observer relies upon the same concept as the PID controller, or measuring current error, accumulating past error, and predicting future error. The PID observer equation, derived from Equation 1, is:

$$\begin{aligned}\{\hat{\theta}\} &= \int\{\hat{\omega}\}dt \\ &= \iint\left(K_P e+K_I\int(e)dt+K_D\dot{e}\right)dtdt\end{aligned} \tag{28}$$

where $\{\hat{\theta}\}$ is the estimated Euler angle, $\{\hat{\omega}\}$ is the estimated angular velocity, $e(t)$ is the error defined as $e=u_c-y$ where u_c is the reference value, $y(t)$ is the process output, and K_P, K_I, and K_D are gain constants used to tune the observer. The PID controller operates by accepting angle data which is used to estimate the angular acceleration used to produce the angle data. Integrating this estimate gives estimated angular velocity $\{\hat{\omega}\}$, integrating again produces estimated Euler angles $\{\hat{\theta}\}$. Tuning the gain constants K_P, K_I, and K_D such that the estimates approach the actual values, or $\{\hat{\omega}\}\to\{\omega\}$ and $\{\hat{\theta}\}\to\{\theta\}$, achieves what is known as full state feedback.

The other choice for observer, the Luenberger observer, also has the ability to receive the control signal from the Dynamics block, labeled TotalControl on the block diagram. The Luenberger observer operates on a system defined as:

$$\dot{x}(t)=Ax(t)+Bu(t) \tag{29}$$

$$\dot{y}(t)=Cx(t) \tag{30}$$

where $x(t)$ is the process state, $u(t)$ is the process inputs, $y(t)$ is the process outputs, and A, B, C, and D are gain constants for tuning the observer. [36] Assuming such a system, the Luenberger observer is defined as:

$$\dot{z}(t)=Az(t)+[L](y(t)-Cz(t))+Bu(t) \tag{31}$$

$$\begin{aligned}\dot{e}(t) &= \dot{z}(t)-x(t) \\ &=(A-[L]C)(z(t)-x(t)) \\ &=(A-[L]C)e(t)\end{aligned} \tag{32}$$

Where $z(t)$ is the estimated process state, $e(t)$ is the observer error, and $[L]$ is the observer gain matrix. The observer gain matrix is tuned such that the observer error converges to zero. The simulated Luenberger observer takes in the Euler angles, finds the observer error compared to the process state, and uses Equation 32 to output full state feedback. The key difference between the PID and Luenberger observer is the Luenberger's ability to sum the derivative portion of the estimation with the velocity component. By doing this, the Luenberger does not put the derivative portion through the additional differentiation used by the PID observer, as shown in Equation 28. By manipulating the derivative portion less, the Luenberger observer propagates less noise than the PID observer. As implemented in the simulation, the Luenberger observer equation is:

$$\{\hat{\theta}\} = \int\left(\int\left(K_P e+K_I\int(e)dt\right)+K_D e\right). \tag{33}$$

The simulated Luenberger observer is enhanced to accept feedback just like the PID controller and features the TotalControl option as previously discussed. The block diagram of the PID and Luenberger observers are as shown in Figure 10.

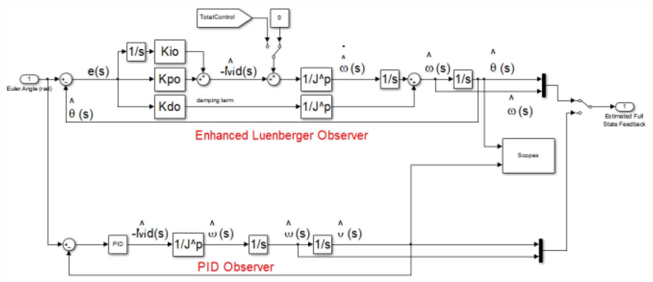

Figure 10. Observer Subsystem

Immediately after the Observer block is the ability to enable a Kalman filter. Kalman filters are a means of isolating a system state buried inside an otherwise noisy signal. The Kalman filter operates on a state defined as:

$$\{x_k\}=[B]\{x_{k-1}\}+[D]\{u_{k-1}\}+\{w_{k-1}\} \tag{34}$$

where $\{x_k\}$ is the state, $[B]$ is the transition matrix, $[D]$ is the control matrix, $\{u_k\}$ is the control, and $\{w_k\}$ is the process noise, all at time k. [37] The filter makes observations defined as:

$$\{z_k\}=[F]\{x_k\}+\{v_k\} \tag{35}$$

where $\{z_k\}$ is the observation, $[F]$ is the observation matrix, and $\{v_k\}$ is the observation noise. Both process and observation noise are assumed to normally distributed with covariance Q and R respectively. With these definitions, the Kalman filter makes estimates via this equation:

$$\{\hat{x}_k\}=K_k\{z_k\}+(1-K_k)\{\hat{x}_{k-1}\} \tag{36}$$

where $\{\hat{x}_k\}$ is the estimated state and K_k is the Kalman gain. Using these relationships a Kalman filter makes a prediction of the state and covariance, calculates gain, and updates its state and covariance model.

In the simulation, the state vector is broken apart, and the Euler angles are run separately from the angular velocity through corresponding Kalman filters. This enables the simulation to find the true error in the Kalman values by comparing the filtered Euler angles to be compared to the true Euler angles. The Kalman Filter block diagram is as shown in Figure 11.

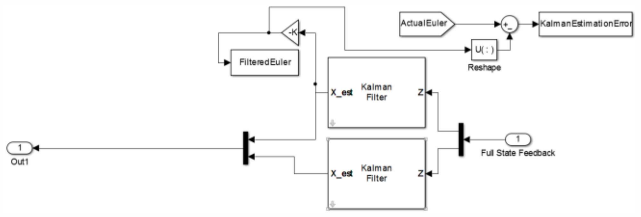

Figure 11. Kalman Filter

The final piece of the Sensors and Observers block is the Lowpass filter, which attempts to eliminate extraneous high frequency noise in our signal:

$$s'=\frac{s^2/\omega_z^2+2\zeta_z s/\omega_z+1}{s^2/\omega_p^2+2\zeta_p s/\omega_p+1} \tag{37}$$

where s is the signal, s' is the filtered signal, and ω_p, ζ_p, ω_z, ζ_z are parameters tuned to the desired filter level [38] When the Lowpass filter is selected, the simulation breaks out each of the six components of the state vector, utilizes individual, separately tunable filters, and pushes the filtered signal to the Controller block. The Lowpass filter block diagram is as shown in Figure 12.

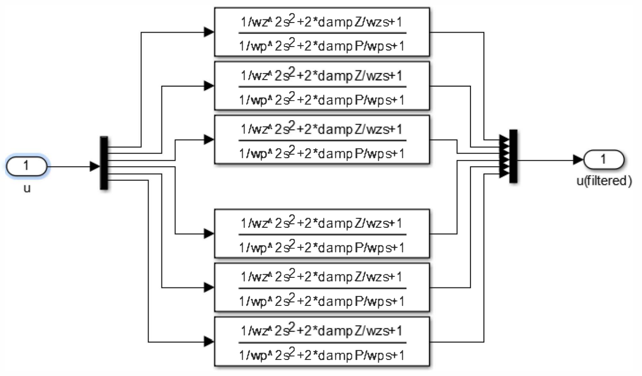

Figure 12. Lowpass Filter

The Sensors and Observers block offers a wide array of abilities for the simulation. After passing through the enabled blocks, the full state feedback is sent to the Dynamics block. The Sensors and Observers block diagram is as shown in Figure 13.

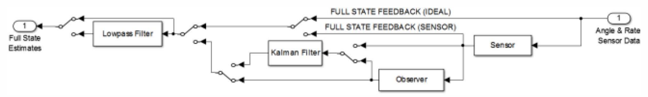

Figure 13. Spacecraft Sensors and Observers

4. Analysis and Results

To answer the research question completely requires that the DTC are effective, accurate, and stable. This requires at least two different methods for analysis. The selected methods are the Monte Carlo analysis and phase portrait analysis. Both methods will demonstrate that DTC are effective, or that the simulated DTC do in fact function. The Monte Carlo analysis will also demonstrate the effects on accuracy due to DTC. The phase portrait analysis will demonstrate the stability of DTC.

Monte Carlo Analysis

The Monte Carlo method, created by Stanislaw Ulam, draws its name from Ulam's uncle, who frequented the Monte Carlo casino. [39] The premise is that by running simulations numerous times and recording the results, the outcome can be accurately characterized. This is akin to visiting a casino many times to determine the odds of any game of chance.

In order to profile the outcome of the DTC in the simulation, a Matlab m-file was written, iteratively calling the simulation 900 times. Each iteration was distinguished by increasing the mass loss of the spacecraft by 0.1 percent, beginning at zero percent and finishing at 90 percent mass loss. Mass loss was simulated by reducing the simulated spacecraft moment of inertia matrix. This assumes that in the case of sudden mass loss, the satellite remains functional. Each iteration, the spacecraft was commanded to perform a 30 degree yaw, and the tracking error was recorded. In addition to the mean error, the standard deviation of the error was also recorded. The m file was written as shown in Appendix B.

The Monte Carlo method applied to tracking error does an excellent job of characterizing the accuracy of the DTC when a spacecraft mass is suddenly lost, and by returning these results, shows that the DTC of the simulation do function correctly. The DTC never experienced greater than a mean 0.9 degree tracking error, with a standard deviation that was always less than 1.1 degrees. The results are as shown in Figure 14, where $\left\|\mu^\circ\right\|$ is the mean measured error, and $\left\|\sigma^\circ\right\|$ is the standard deviation.

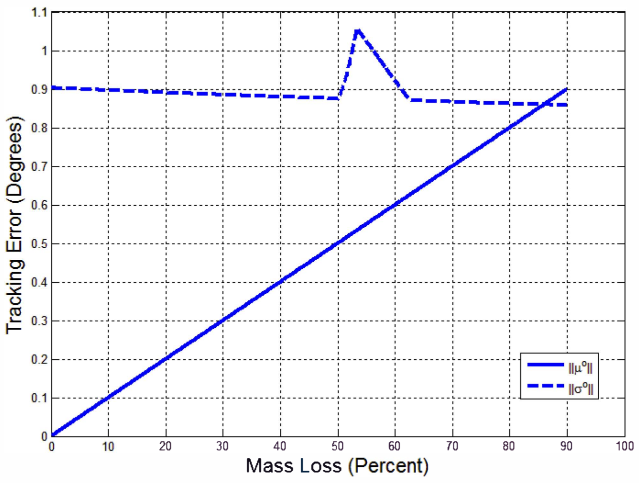

Figure 14. Monte Carlo Analysis Results

Phase Portrait Analysis

The premise behind phase portrait analysis is to characterize the behavior of a system with regard to stability of equilibrium points. In this case, the behavior of the spacecraft DTC was examined around the commanded state vector. In order to demonstrate stability, the axes of angular rate and angular position in the inertial frame were chosen. If the motion described circles into the commanded point, then stability has been achieved.

To create each phase portrait, a Matlab m-file was written which iteratively calls the simulation 36 times. Each portrait reflects fixed mass loss, with many different values of initial position and initial velocity. Mass loss was simulated by reducing the inertia matrix. This assumes that after sudden mass loss, the satellite remains functional. Each iteration changes the initial position and initial velocity, then records the results as the simulation is run for a set time of 100s. The omega value was already calculated by the Dynamics section, as variable w. In order to record the desired results, a new variable had to be captured, the position in the inertial frame. This was accomplished by adding an integrator block to the angular speed in the Dynamics section, and capturing the angle as the variable wAngles, as shown in Figure 15.

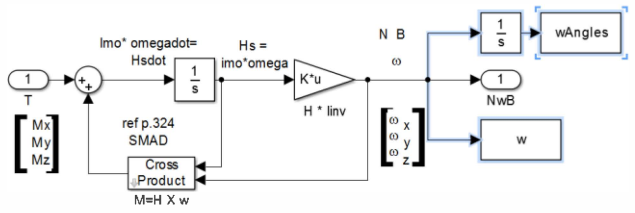

Figure 15. Omega Angle Production

For each iteration the spacecraft was commanded to perform a 30 degree yaw, with the omega angles and omega value recorded, then plotted against each other in graphs for each of the inertial frame dimensions. The m-file was written as shown in Appendix C.

The generated phase trajectory portraits successfully characterizes the stability of the DTC when spacecraft mass is suddenly lost and the spacecraft is imparted with sudden changes in state. Stability is demonstrated by the boundedness of the phase trajectories. [10] Stable trajectories are generally expected to approach the origin, unstable trajectories generally diverge from the origin. Examples by Slotine and Li of phase trajectories with differing stability can be found in Figure 16.

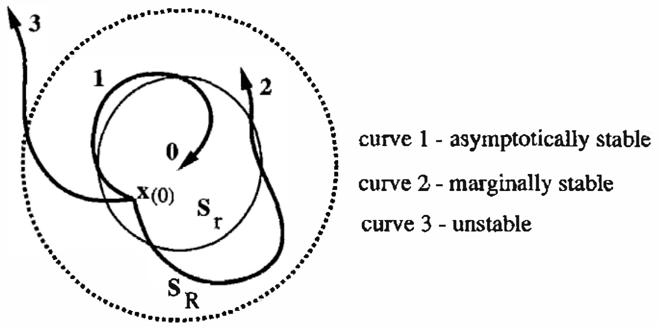

Figure 16. Phase Portrait Stability

By returning these results, the portraits demonstrates that the DTC of the simulation do function correctly. Each set of phase portraits is grouped by the inertial frame dimension it comes from, and the portraits show either zero, 45, or 90 percent mass loss of the spacecraft. The phase portraits are shown in Figures 17 through 25.

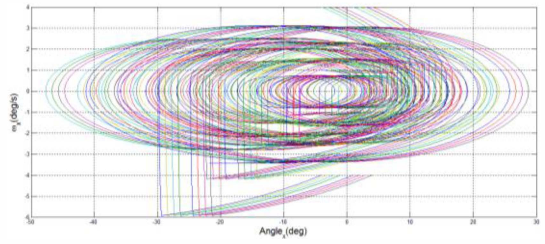

Figure 17. 0% Mass Loss X Phase Portrait

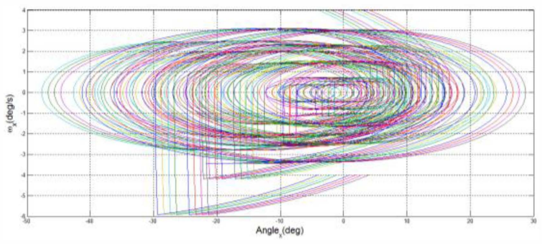
Figure 18. 45% Mass Loss X Phase Portrait

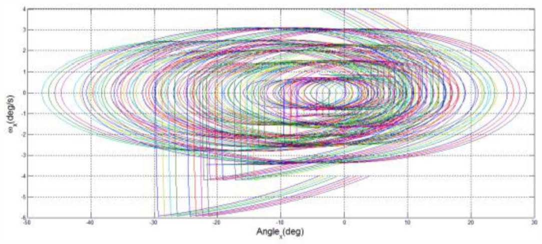
Figure 19. 90% Mass Loss X Phase Portrait

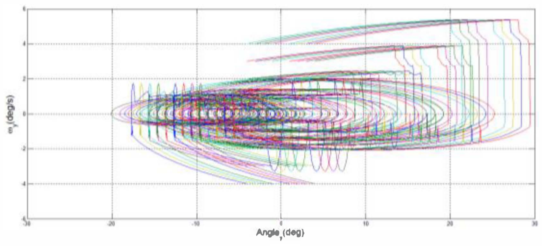
Figure 20. 0% Mass Loss Y Phase Portrait

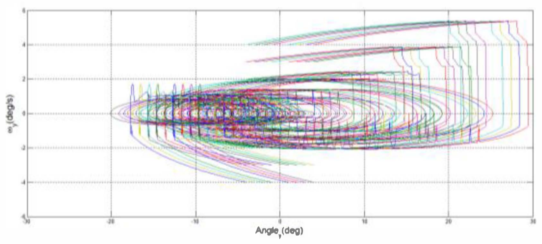
Figure 21. 45% Mass Loss Y Phase Portrait

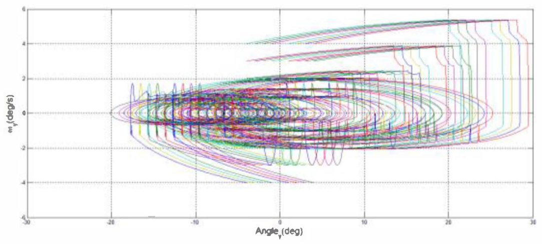
Figure 22. 90% Mass Loss Y Phase Portrait

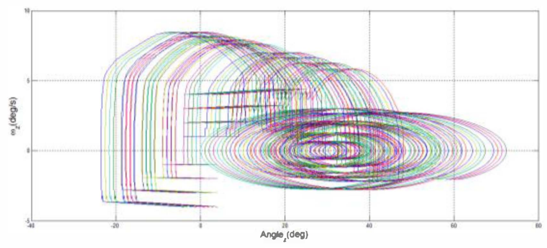
Figure 23. 0% Mass Loss Z Phase Portrait

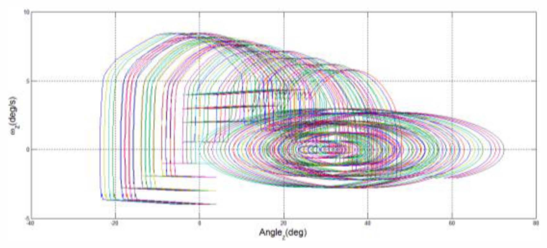
Figure 24. 45% Mass Loss Z Phase Portrait

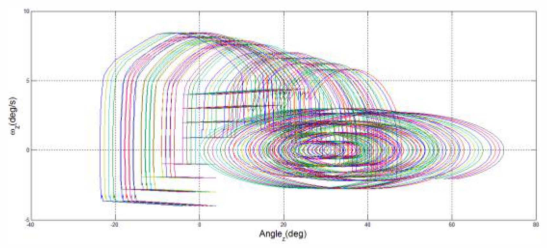
Figure 25. 90% Mass Loss Z Phase Portrait

This chapter covered the analysis methods used to answer the research question. The Monte Carlo method proved effective in characterizing the tracking error of the system when subjected to mass loss. The phase portraits provide insight into the stability of the system when subjected to sudden mass loss, change in velocity, and change in position. In both examples the fewest possible changes were made to the base simulation to protect the integrity of the results. The changes were limited to commenting out, or removing, initial conditions that interfered with data collection, and adding a single integrator and variable output set for phase portrait generation.

5. SUMMARY

The development of adaptive controls and the enabling technology for them has reached a point where new and innovative uses are now becoming possible. The purpose of this research was to determine the plausibility of DTC in satellites. By introducing a modified adaptive PID controller with adaptive feed forward control to this simulated spacecraft, it is demonstrated that the controls have achieved significant damage tolerance. This comes at a time where the need for such tools has never been greater, and the dangers associated with space operations grow daily.

Adaptive Control

Space operations are going to become more difficult and dangerous in the foreseeable future. Significant action is required to protect the space assets that the United States has become dependent on. Adaptive control holds the key for many of the solutions currently proposed to mitigate these risks.

The damage tolerant, adaptive controller as implemented in the simulation provides adaptive control with accuracy and stability. This accuracy is evidenced by Monte Carlo analysis where less than 1 degree of system tracking error even when approaching 90 percent mass loss. The stability is evidenced by the phase portraits of system performance in each inertial frame, recovering from the sudden change in state even when the change was accompanied by a loss of mass.

Application of Study

Introducing DTC like those simulated here is achievable with today's technology. As the largest limiting factor historically has been processing power, the feasibility of retrofitting operational spacecraft with DTC is limited, but not necessarily impossible. Modern spacecraft should have no technical limitations prohibiting the implementation of DTC. Writing DTC into spacecraft controllers may even lower the cost and risk of the program as a whole. [40] DTC clearly exhibit a stability solution for all manner of spacecraft, warranting further study and operational adoption. The logical next step is to validate the simulation results experimentally.

REFERENCES

[1] P. Baines, "Prospects for non-offensive defenses in space," *Center for Nonproliferation Studies Occasional Paper*, no. 12, pp 31-48. Aug, 2003

[2] United State Strategic Command, "USSTRATCOM space control and space surveillance factsheet," USSTRATCOM, Offutt AFB, NE, Dec. 2012

[3] D. Kessler and B. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research*, vol. 83, no. A6, pp 2637-2646, Jun. 1978

[4] "Satellite collision leaves significant debris cloud," *Nat. Aeronautics and Space Admin. Orbital Debris Quarterly News,* vol. 13, no. 2, pp. 1-2, Apr. 2009.

[5]"International Space Station again dodges debris," *Nat. Aeronautics and Space Admin. Orbital Debris Quarterly News,* vol. 15, no. 3, pp. 1-2, Jul. 2011.

[6] S. Kan, "China's anti-satellite weapon test," U.S. Library of Congr., Congressional Research Service, Washington, D.C., Rep. RS22652, Apr. 2007.

[7] T. Kelso. (2013, March 8). *Chinese space debris may have hit Russian satellite* [Online]. Available: http://www.blogs.agi.com/agi/2013/03/08/chinese-space-debris-hits- russian-satellite/

[8] Rockwell Collins. (2011). *Case study: Rockwell Collins demonstrates damage tolerant flight controls and autonomous landing capabilities* [Online]. Available: http://www.rockwellcollins.com/sitecore/content/Data/Success_Stories/DARPA_Damage_Tolerance.aspx

[9] Rockwell Collins. (2011, August 18). *DARPA, U.S. Army and Rockwell Collins release video of successful damage tolerance control testing on shadow unmanned aircraft system.* [Online]. Available: http://www.rockwellcollins.com/sitecore/content/Data/News/2011_Cal_Yr/GS/FY11GSNR32-DTC_Shadow.aspx

[10] J.-J. Slotine and W. Li, "Nonlinear control systems design" in *Applied nonlinear control*, Upper Saddle River: Prentice Hall, 1991, ch. 6-9, pp 191-432.

[11] K. Åström and B. Wittenmark, "What is adaptive control?" in *Adatpive Control,* 2nd ed. Mineola: Dover Publications, 2008, pp. 376.

[12] G. Dumont and M. Huzmezan, "Concepts, methods and techniques in adaptive control," in *Proceedings of the American Control Conference,* Vancouver, B.C., 2002, pp. 1137-1150.

[13] N. Hovakimyan, "Robust adaptive control of multivariable nonlinear systems," Univ. of Illinois, Urbana, IL, Rep. AFRL-OSR-VA-TR-2012–0524, 2011.

[14] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th International Conference on Machine Learning,* Tuebingen, Germany, 2007, pp. 745-750.

[15] E. Liebemann et al., "Light commercial vehicles: challenges for vehicle stability control," Robert Bosch GmbH, Gerlingen, Germany, Rep. 07-0269, 2007.

[16] Man Truck & Bus. (2012). *Adaptive cruise control.* [Online] Available: http://www.mantruckandbus.com/com/en/innovation___competence/applied_safety/adaptive_cruise_control__acc_/Adaptive_Cruise_Control__ACC_.html

[17] W. Pananurak et al., “Adaptive cruise control for an intelligent vehicle,” in *Procedings of the 2008 IEEE International Conference on Robotics and Biometrics,* Bangkok, Thailand, 2008, pp. 1794-1799.

[18] W. MacKunis et al., “Adaptive satellite attitude control in the presence of inertia and CMG gimbal friction uncertainties,” *The Journal of the Astronautical Sciences,* vol. 56, no. 1, pp. 121-134, Jan. 2008.

[19] A. Tewari, “Adaptive vectored thrust deorbiting of space debris,” *Journal of Spacecraft and Rockets*, vol. 50, no. 2, pp. 394-401, Mar. 2013.

[20] Staff of the Flight Research Center, “Experience with the X-15 adaptive flight control system,” NASA, Edwards, CA, Rep. NASA TN H-618, 1971.

[21] S. Lilley, “Vicious cycle,” In *National Aeronautics and Space Administration System Failure Case Studies,* vol. 5, no. 3, pp. 1-4, 2011.

[22] A. Ulsoy and Y. Koren, “Applications of adaptive control to machine tool process control,” in *IEEE Control Systems Magazine*, vol. 9, no. 4, pp. 33-37, Jun. 1989.

[23] A. Taylor, “Adaptive attitude control for long-life space vehicles,” General Electric Company, Binghampton, NY, Rep. AIAA Paper No. 69–945, 1969.

[24] C.-H. Ih et al., “Application of adaptive control to space stations,” Jet Propulsion Laboratory, Pasadena, CA, Rep. AIAA Paper 85-1970, 1985.

[25] B. Govin et al., “Adaptive control of flexible space structures,” in *American Institute of Aeronautics and Astronautics Guidance and Control Conference*, Velizy, France, pp. 192-199, 1981.

[26] R. Kosut, “Adaptive control techniques for large space structures,” Integrated Systems Inc., Bolling AFB, Washington D.C., Rep. AFOSR-TR-89–0071, 1989.

[27] D. Schaechter, “Adaptive control for large space structures,” in *Aeronautics and Astronautics Guidance and Control Conference,* Palo Alto, CA, pp. 606-611, 1983.

[28] Y. Xu et al., “Adaptive control of space robot system with an attitude controlled base,” Carnegie Mellon University, Pittsburgh, PA, Rep. CMU-RI-TR-91-14, 1991.

[29] H. Ho, et al., “Comparative studies of three adaptive controllers,” in *ISA Transactions*, vol. 38, no. 1, pp. 43-53, Jan. 1999.

[30] L. Ehrenwald and M. Guelman, “Integrated adaptive control of space manipulators,” in *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 1, pp. 156-163, Jan. 1998.

[31] B. Wie, “Rotational maneuvers and attitude control,” in *Space Vehicle Dynamics and Control,* 2nd ed., Reston, VA, Amer. Inst. of Aeronautics and Astronautics Inc., 2008, ch. 7, pp. 423-466.

[32] E. Reeves, “Spacecraft design and sizing,” in *Space Mission Analysis and Design,* 3rd ed., Hawthorne, CA, Microcosm Press, 1999, pp. 324-325.

[33] Mathworks. (2013). *Quaternion to direction cosine matrix r2013a*, [Online] Available: http://www.mathworks.com/help/aeroblks/quaternionstodirectioncosinematrix.html

[34] B. Wie, “Rigid Body Dynamics,” in *Space Vehicle Dynamics and Control,* 2nd ed., Reston, VA, Amer. Inst. of Aeronautics and Astronautics Inc., 2008, ch. 6, pp. 328-335.

[35] D. Hoag, “Apollo guidance and navigation considerations of Apollo IMO gimbal lock,” Massachusetts Institute of Technology, Cambridge, MA, Rep. E-1344, 1963.

[36] D. Luenberger, *Introduction to dynamic systems: theory, models, and applications,* New York City, NY, John Wiley & Sons, 1979, pp. 300-307.

[37] G. Welch and G. Bishop, *An introduction to the Kalman filter*, Chapel Hill, NC, University of North Carolina, 2006, pp. 2-15.

[38] B. Wie, “Dynamic Systems Control,” in *Space Vehicle Dynamics and Control,* 2nd ed., Reston, VA, Amer. Inst. of Aeronautics and Astronautics Inc., 2008, ch. 2, pp. 137.

[39] N. Metropolis, “The beginning of the Monte Carlo method,” in *Los Alamos Science,* Special Issue*,* 1987, pp. 125-130.

[40] J. Wertz and W. Lason, “Applying the space mission analysis and design process,” in *Space Mission Analysis and Design*, 3rd ed., Hawthorne, CA, Microcosm Press, 1999, pp. 301.

BIOGRAPHY

***Scott Nakatani** received a B.S. in Physics from the United States Air Force Academy in 2007and a M.S. in Space Systems Operations from the Naval Postgraduate School in 2013. He has been a commissioned officer in the United States Air Force for 7 years. In that time, he has been a mission lead and lead range operational testing for Delta II, Delta IV and Atlas V missions from Cape Canaveral AFS, FL. Currently he is assigned to the 1 Air and Space Test Squadron, Vandenberg AFB, CA.*

Timothy Sands *received a B.S. from North Carolina State University and the Engineer's Degree from Columbia University both in Mechanical Engineering in addition to the PhD from the Naval Postgraduate School in Astronautical Engineering. He has been a military faculty member at the Naval Postgraduate School and the Air Force Institute of Technology where he still resides.*

APPENDIX A
Model Initialization Function

```
% Spacecraft Dynamics Simulation
%clear all;close all;clc;
warning off all;

% Simulation run parameters
Rate=34.3; DeltaT=1/Rate;

% Maneuver Parameters
roll=0; pitch=0; yaw=30; roll=roll*pi/180; pitch=pitch*pi/180;
yaw=yaw*pi/180;
SlewTime=5; SACengage=0;

%Constants0
Re=6371.2e3;mu=398601.2e9;    %earth radius and universal gravitation
constant

%Spacecraft orbit
h=0;                                    %orbit altitude
R=Re+h;                                 %orbit radius from center of
earth
we=0.000072921158553;                   %earth's angular velocity
rad/solar sec(Vallado)
wo=sqrt(mu/(Re+h)^3);                   %orbit angular velocity
incln=36.6001*pi/180;                   %Inclination assumed
Monterey Latitude
epsilon=12*pi/180; alphao=0;
uo=0; nuo=0;                                    %Start S/C beneath
subsolar point
beta=60; gamma=1.5;
a=0.545491852; b=0.314939867; c=0.704226952; %Assumed spacecraft
rectangular size
Area=[b*c a*c a*b];                     %projected area~m^2 in body
x,y,z directions
density=4.39e-14;
kpre=-9.9639/24/3600/180*pi*0;          %nodal precession constant
assumed zero here
wn=kpre*(Re/(Re+h))^3.5*cos(incln); %nodal precession (zero
eccentricity)
V=wo*(Re+h);
rho=asin(Re/(h+Re));                    %earth angular radius
Cd=2.5; psun=4.5E-6;            %Drag coefficient and solar pressure
constant~N/m^2
Kaero=-0.5*Cd*V^2; Psolar=2*psun; %constants for aero and solar torque
calculation
dL=[0.002 0.002 0.008];                 %predicted distance between
cp and cg
Kme=2.3390e-005;

%Spacecraft Magnetic Properties (assumed)
mresid=[0 0 0.01];     %Spacecraft residual magnetic moment
M=mresid;                       %Magnetic unit dipole vector
K=7.943e15;
%Spacecraft Inertia conditions (ACTUAL values....not the ones assumed in
the feedforward control calculation)
Ix=90; Iy=100; Iz=250 ;
Ixy=-10; Iyz=20; Ixz=-10;
Imo=[Ix -Ixy -Ixz;
-Ixy Iy -Iyz;
-Ixz -Iyz Iz];                          %Moment of inertia matrix
Iinv=inv(Imo);                  %Moment of inertia inverse goes in
dynamics block
%Spacecraft initial Euler state angles and rates
phio=0;thetao=0;psio=0; %Initial Euler Angles
phidoto=0;thetadoto=0;psidoto=0; %Initial Euler Rates

%Calculation of initial quaternion (qo) and angular momentum (Ho)
s1=sin(phio/2);s2=sin(thetao/2);s3=sin(psio/2);c1=cos(phio/2);c2=cos(theta
o/2);c3=cos(psio/2);
q1o=s1*c2*c3-c1*s2*s3;
q2o=c1*s2*c3+s1*c2*s3;          %Wie pg. 321
```

```
q3o=c1*c2*s3-s1*s2*c3;
q4o=c1*c2*c3+s1*s2*s3;
S1=sin(phio);S2=sin(thetao);S3=sin(psio);C1=cos(phio);C2=cos(thetao);
C3=cos(psio);
wxo=phidoto-psidoto*S2-wo*S3*C2;
wyo=thetadoto*C1+psidoto*C2*S1-wo*(C3*C1+S3*S2*S1);
wzo=psidoto*C2*C1-thetadoto*S1-wo*(S3*S2*C1-C3*S1);
qo=[q1o q2o q3o q4o];
Ho=Imo*[wxo wyo wzo]';
norm(Ho)*1000;

%Calculate eclipse time for comparison with EPS calculations
Te=100.87*2*V/2/pi;

%CMG Properties (in degrees)
%beta=90*pi/180;                         %Skew angle in degrees
converted to radians
beta=[90;90;-90;0]; beta=beta.*pi./180;
Gimbal0=[-30*pi/180; 90*pi/180; -30*pi/180;0]; % Initial Gimbal angles
for 0 H spin up
w_wheel=2800*(2*pi/60);                                 %Wheel speed
in RPM converted to rad/s
Iwheel=0.0614*1.3558179483314;                          % Wheel inertia
in slug-ft^2 converted (exact) to kg m^2
h_wheel=Iwheel*w_wheel;                                 % CMG Wheel
Angular Momentum

% [Fossen]'s adaptive feedforward parameters
ETA=-100; LAMBDA=0.5; uffGain=1;
%FEEDFORWARD control based on "assumed" Spacecraft Inertia
conditions
Ix=119.1259; Iy=150.6615; Iz=106.0288; % ACTUAL VALUES ARE
Ix=90; Iy=100; Iz=250 ;
Ixy=-15.7678; Iyz=22.31637; Ixz=-6.54859; % ACTUAL VALUES
ARE Ixy=-10; Iyz=20; Ixz=-10;
THETAo=[Ix Ixy Ixz Iy Iyz Iz 0 0 0];

% FEEDBACK CONTROL
%PDI Controller Gains % TUNED Well for Presence of LOWPASS
(Kp=0.5; Kd=Kp*750; Ki=0.1)
%Kp=0.5; Kd=Kp*750; Ki=0.1;

%PDI Controller Gains % TUNED WELL FOR NO-NOISE (Kp=1;
Kd=2000*Kp; Ki=5)
Kp=1; Kd=Kp*6000; Ki=Kp*75;
%PDI Controller Gains % TUNED WELL FOR NOISE (Kp=1;
Kd=Kp*3000; Ki=1)
%Kp=1; %Kd=Kp*3000; %Ki=1;

%PID Controller Gains tuned well for u ff and u fb decoupling
Kpx=20; Kdx=500; Kix=0.1;

% Noise Parameters
NoiseVariance=1e-9; BADNoiseVariance=NoiseVariance*1e3;

%Bandpass filter pole per Bong Wie 2nd Edition pg 137-138
wp=10*pi/SlewTime; % Product of pole frequency and zero frequency
establishes max phase lag
wz=2*wp;
dampZ=1; % >0 but small for good tracking..
dampP=dampZ;
% Luenberger Observer Gains
MaxI=300; lambda1=12.5 ; lambda2=50 ; lambda3= 200;
Kpo=MaxI*(lambda1*(lambda2+lambda3)+lambda2*lambda3)/10;
Kdo=MaxI*(lambda1+lambda2+lambda3)/10;
Kio=MaxI*lambda1*lambda2*lambda3/9;
% PID Observer Gains
KpoPID=30000;
KdoPID=3000;
KioPID=30;
```

APPENDIX B

Damage Tolerant Controls Monte Carlo Analysis

```
%Iterative Mass Loss Analysis                    Scott Nakatani
%This file calls the Spacecraft simulation with increasing mass loss to
%assess the ability of the adaptive controls to compensate for battle
%damage, characterized as a percent mass lost
clear all; close all; clc
StepSize=0.001;              %ENTER STEP SIZE HERE where 1 = 100%
Steps=round(0.9/StepSize);    %Calculates # of steps to take
Loss=[];  PercentLoss=0;  MeanErrorT=[];  StdDevT=[]; DeltaT=0.001;

%Actual Spacecraft Inertia conditions:
%These are separate from the simulation's feedforward assumed values
Ix=90;   Iy=100;   Iz=250 ;  Ixy=-10;  Iyz=20;  Ixz=-10;
ImoI=[Ix -Ixy -Ixz; -Ixy  Iy  -Iyz;  -Ixz -Iyz  Iz];  %Moment of inertia

for ii = 1:Steps
   %Sends % complete status to command window
   PercentCompleted=100*(ii/Steps)

   %INVERSE of Moment of inertia matrix
   Imo=(1-PercentLoss).*ImoI;
   %Accumulates values of battle damage percent
   Loss=[Loss; PercentLoss];

   %CALLS THE SIMULATION
   sim( 'IterativeSpacecraftModel.mdl' );

   %Accumulate Tracking Error & Standard Deviation
   MeanErrorT=[MeanErrorT; norm(MeanError,2)];
   StdDevT=[StdDevT; norm(StdDev,2)];
   %Iterates mass lost
   PercentLoss=StepSize*ii;
end

%Insure array dimensions match
Loss=Loss(1:max(size(StdDevT)));
MeanErrorT=Loss(1:max(size(StdDevT)));

%Plot the results of Monte Carlo analysis
figure(1);
plot(Loss*100,MeanErrorT,'linewidth',3); grid on; ylabel('Tracking Error
(Degrees)','fontsize',16);xlabel('Mass Loss (Percent)','fontsize',16);
hold on; plot(Loss*100,StdDevT,'--','linewidth',3);
legend('||\mu^o||','||\sigma^o||','fontsize',16,'location','NorthWest');
hold off
```

APPENDIX C
Phase Portrait Code

```
%Stability Phase Portrait Generator                    Scott Nakatani
%This file calls the Spacecraft simulation with increasing angular change
%to assess the stability of the adaptive control solution via creating a
%phase portrait
clear all; close all; clc
Omega=[]; OmegaAngles=[];
OmegaX=[]; OmegaY=[]; OmegaZ=[];
OmegaAnglesX=[]; OmegaAnglesY=[]; OmegaAnglesZ=[];

%Actual Spacecraft Inertia conditions:
%These are separate from the simulation's feedforward assumed values
Ix=90;  Iy=100;  Iz=250 ;  Ixy=-10;  Iyz=20;  Ixz=-10;
Imo=[Ix -Ixy -Ixz; -Ixy  Iy  -Iyz; -Ixz -Iyz  Iz];  %Moment of inertia

Imo=Imo.*(.1); %INPUT MOMENTUM LOSS HERE where 1 = 100%
mass, 0% loss
Steps=4; %Steps are in all quadrants, total steps = 2*Steps^2
StepSize=2*pi/360; %Stepsize is input in radians

for ii = 1:Steps
   %Iterates initial positive omega values & subordinate conditions
   omega0X=(ii)*(StepSize);
   omega0Y=(ii)*(StepSize);
   omega0Z=(ii)*(StepSize);
   omega0=[omega0X, omega0Y, omega0Z]';
   H0=Imo*omega0;

   for iii = 1:Steps
      %Sends % complete status to command window
      PercentCompleted=(50*(ii-1)/Steps)+(50/Steps*iii/Steps)

      %Iterates initial positive omega angles
      wAngleX=(iii)*(StepSize);
      wAngleY=(iii)*(StepSize);
      wAngleZ=(iii)*(StepSize);
      wAngle0=[wAngleX, wAngleY, wAngleZ];

      %CALLS THE SIMULATION
      sim( 'PhasePortraitSpacecraftModel.mdl' );

      %Accumulate resulting omega values and angles
      OmegaX=[OmegaX w(:,1)];
      OmegaY=[OmegaY w(:,2)];
      OmegaZ=[OmegaZ w(:,3)];
      OmegaAnglesX=[OmegaAnglesX wAngles(:,1)];
      OmegaAnglesY=[OmegaAnglesY wAngles(:,2)];
      OmegaAnglesZ=[OmegaAnglesZ wAngles(:,3)];


      %Iterates initial negative omega angles
      wAngleX=(-iii)*(StepSize);
      wAngleY=(-iii)*(StepSize);
      wAngleZ=(-iii)*(StepSize);
      wAngle0=[wAngleX, wAngleY, wAngleZ];

      %CALLS THE SIMULATION
      sim( 'PhasePortraitSpacecraftModel.mdl' );

      %Accumulate resulting omega values and angles
      OmegaX=[OmegaX w(:,1)];
      OmegaY=[OmegaY w(:,2)];
      OmegaZ=[OmegaZ w(:,3)];
      OmegaAnglesX=[OmegaAnglesX wAngles(:,1)];
      OmegaAnglesY=[OmegaAnglesY wAngles(:,2)];
      OmegaAnglesZ=[OmegaAnglesZ wAngles(:,3)];
   end
end
 for iiii = 1:Steps
   %Iterates initial negative omega values & subordinate conditions
   omega0X=(-iiii)*(StepSize);
   omega0Y=(-iiii)*(StepSize);  omega0Z=(-iiii)*(StepSize);
   omega0=[omega0X, omega0Y, omega0Z]';
   H0=Imo*omega0;

   for iiiii = 1:Steps
      %Sends % complete status to command window
      PercentCompleted=50+(50*(iiii-1)/Steps)+(50/Steps*iiiii/Steps)

      %Iterates initial positive omega angles
      wAngleX=(iiiii)*(StepSize);
      wAngleY=(iiiii)*(StepSize);
      wAngleZ=(iiiii)*(StepSize);
      wAngle0=[wAngleX, wAngleY, wAngleZ];

      %CALLS THE SIMULATION
      sim( 'PhasePortraitSpacecraftModel.mdl' );

      %Accumulate resulting omega values and angles
      OmegaX=[OmegaX w(:,1)];
      OmegaY=[OmegaY w(:,2)];
      OmegaZ=[OmegaZ w(:,3)];
      OmegaAnglesX=[OmegaAnglesX wAngles(:,1)];
      OmegaAnglesY=[OmegaAnglesY wAngles(:,2)];
      OmegaAnglesZ=[OmegaAnglesZ wAngles(:,3)];

      %Iterates initial negative omega angles
      wAngleX=(-iiiii)*(StepSize);
      wAngleY=(-iiiii)*(StepSize);
      wAngleZ=(-iiiii)*(StepSize);
      wAngle0=[wAngleX, wAngleY, wAngleZ];

      %CALLS THE SIMULATION
      sim( 'PhasePortraitSpacecraftModel.mdl' );

      %Accumulate resulting omega values and angles
      OmegaX=[OmegaX w(:,1)];
      OmegaY=[OmegaY w(:,2)];
      OmegaZ=[OmegaZ w(:,3)];
      OmegaAnglesX=[OmegaAnglesX wAngles(:,1)];
      OmegaAnglesY=[OmegaAnglesY wAngles(:,2)];
      OmegaAnglesZ=[OmegaAnglesZ wAngles(:,3)];
   end
end

%Converts omega angles & values to degrees
OmegaX=OmegaX.*360/(2*pi);
OmegaY=OmegaY.*360/(2*pi);
OmegaZ=OmegaZ.*360/(2*pi);
OmegaAnglesX=OmegaAnglesX.*360/(2*pi);
OmegaAnglesY=OmegaAnglesY.*360/(2*pi);
OmegaAnglesZ=OmegaAnglesZ.*360/(2*pi);

%Plots the phase portraits
figure(1);
plot(OmegaAnglesX,OmegaX,'linewidth',1); grid;
xlabel('Angle_x(deg)','fontsize',16);;
ylabel('\omega_x(deg/s)','fontsize',16);
figure(2);
plot(OmegaAnglesY,OmegaY,'linewidth',1); grid;
xlabel('Angle_y(deg)','fontsize',16);;
ylabel('\omega_y(deg/s)','fontsize',16);
figure(3);
plot(OmegaAnglesZ,OmegaZ,'linewidth',1); grid;
xlabel('Angle_z(deg)','fontsize',16);;
ylabel('\omega_z(deg/s)','fontsize',16);
```