DYNAMIC MODELING OF A SUSPENDED AND SHOCK ISOLATED SYSTEM IN SEISMIC LOADING

by

Emilie Murphy

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mechanical Engineering

Boise State University

August 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Emilie Murphy

Thesis Title: Dynamic Modeling of a Suspended and Shock Isolated System in Seismic Loading

Date of Final Oral Examination: 3 May 2019

The following individuals read and discussed the thesis submitted by student Emilie Murphy, and they evaluated her presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

John F. Gardner, Ph.D.	Chair, Supervisory Committee
Donald Plumlee, Ph.D.	Member, Supervisory Committee
Zhangxian Deng, Ph.D.	Member, Supervisory Committee
Jared D. Jeffrey, M.S.	Member, Supervisory Committee

The final reading approval of the thesis was granted by John F. Gardner, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

I would like to dedicate this effort to the people who have supported me most throughout my educational endeavors. These people are my parents, Dave and Debbie, sister, Katelyn, and my grandparents, Richard and Geri Eismann. Not only have they supported my education financially, but they have supported and encouraged every dream and aspiration I have ever had. Even as a little girl, I was told I could accomplish anything I set my mind to, which gave me the freedom to dream, and dream big. In the tough times when it felt as though I would not be able to accomplish my goals, they give me the support and confidence needed to persevere. I don't know where I'd be without them in my life.

ACKNOWLEDGEMENTS

First and foremost, I would like to acknowledge the support, time, and assistance of Dr. John Gardner. I had the pleasure of taking Kinematics from Dr. Gardner during the junior year of my undergraduate degree. Throughout the tasks he assigned, he would give optional extra tasks where he'd say, "If you liked this, you should try…". I pursued some of these extra tasks and discovered how much I enjoyed blending my Mechanical Engineering major and Computer Science minor. After a few months in his class, he asked if I had ever considered applying to the Mechanical Engineering department's Accelerated Masters program. Not only had I never considered a Master's degree, I had never heard of this accelerated program. It was because of his support and encouragement that I applied and was later accepted into the program. He also encouraged me to pursue a Masters of Science instead of a Masters of Engineering and didn't even bat an eye when I proposed doing an industry project unlike anything else anyone was doing in the graduate program. These moments and many more have been such important points in my life. I couldn't be more grateful and thankful for his support and guidance through it all.

I would also like to acknowledge the Mechanical Engineering department at Boise State University, specifically Dr. Don Plumlee, Katy D'Amico, Justin Larson, and Ashley Holton for their support. These people were my biggest cheerleaders during all of my adventures with internships and design projects. Their desire to provide students with everything they could need made my experience with the department incredible.

Last but not least, I would like to acknowledge a friend of mine, Charles “Chuck” Burnell. His dual degrees in Mathematics and Computer Science, coupled with his love for unusual and challenging problems made him a fantastic friend to have while working on this project. After struggling with being able to decipher earthquake data, Chuck was able to help me obtain format description files which decoded this data. He even assisted in writing the script to manipulate the earthquake data into a usable .csv format. Without his assistance, I don’t think I would have been able to meet my deadlines for the project, and for that, I am forever grateful.

ABSTRACT

A dynamic model for a suspended and shock isolated system is derived and implemented in MATLAB's Simulink software. The purpose of this implementation is to create a design tool which is modularized to be able to accommodate any configuration of a similar system in any kind of loading. The design tool is used to compute the level of acceleration experienced at specific points in space within the system in the presence of seismic events, as typified by the dynamic displacement caused by the Sumatra, Indonesia earthquake of 2007. It is determined that under this 8.4 magnitude earthquake, accelerations within the system are reduced by 64-96%, depending on direction and location, when compared with earthquake accelerations. A parameter sensitivity study is conducted to illustrate how the design tool can be used to determine the dependence of the system on its parameters for future development of the system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CG	Center of Gravity
FBD	Free Body Diagram
ft	foot/feet
lb	pound/pounds
s	second
USGS	United States Geological Survey

CHAPTER 1: INTRODUCTION

In engineering industry, there is more and more emphasis being placed on computer aided design and system modeling as technology becomes readily available and easily accessible. This enables companies to obtain an accurate picture of what a final product might look like or how a system may act, saving a significant amount of time and money on failed physical prototypes. Computer aided design tools such as 3D modeling, finite element analysis, dynamic modeling, etc. can all be utilized to create anything you can imagine in a minimal amount of time.

The design tool that this work will be showcasing is that of dynamic modeling. Dynamic modeling is the process of utilizing mathematical equations to simulate how a system will respond under various physical constraints and loading scenarios. This analysis can also be used to analyze things such as how subsystems interact with each other or accelerations at specific locations within a system, etc. This information is critical, as it can be used to determine things such as the loading conditions under which a system will fail, which can in turn be used to redesign and implement protective measures against failure.

The system being analyzed is that of a hypothetical underground bunker. This system consists of a shock isolated platform suspended by four chains within a capsule, buried a number of feet underground. The specifics of the system configuration and the parameters used in simulation will be discussed in detail later.

A system of this type was chosen due to its use by governments around the world as a means of protection and survival during extreme situations. Some of these situations include providing to a place to live safely in extreme physical situations, such as an earthquake or nuclear blast. Through the creation of a dynamic model to represent a system such as this underground bunker, questions such as survivability within the structure, Gs experienced by personnel during an event, and more can be answered.

By focusing on modularization of model subsystems and a straight forward user interface, a design tool can be created to allow for the analysis of any configuration of the system in any loading scenario, such as seismic loading. This work is also to act as an example of how modeling a system in this manner can be done quickly and play an important role in the physical simulation and subsequent analysis of both the components and personnel within the system.

Literature Review

When reviewing historical earthquake damage, underground structures have experienced a much lower rate of damage when compared to above ground structures. Because of this, underground structures with large cross sections have not been popular research topics. Some of the structures which have been studied, however, are tunnels and subways, which experienced significant damage during landmark earthquake events such as the 1995 Kobe, Japan earthquake, the 1999 Chi-Chi, Taiwan earthquake and the 1999 Kocaeli, Turkey earthquake. The response of underground structures to seismic events has been broken down into three categories [1]: axial compression and extension, longitudinal bending, and ovaling/racking as shown in Fig. 1.

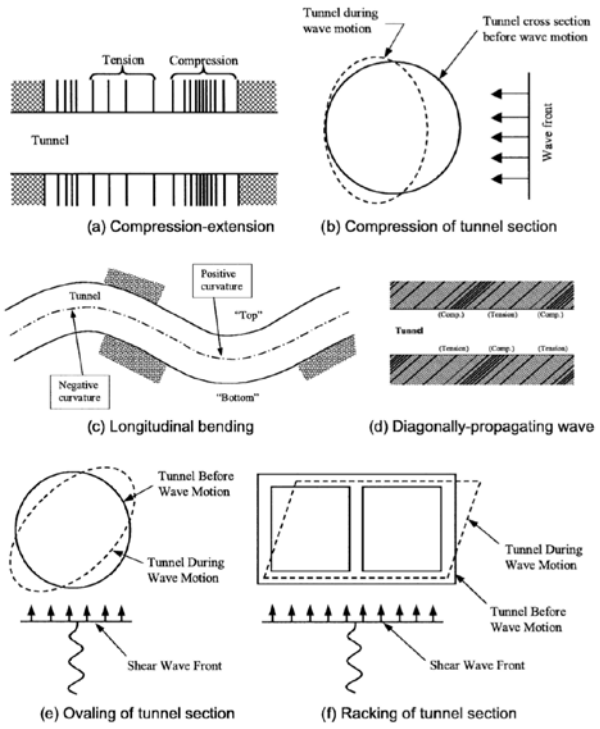


Figure 1. Deformation Modes of Tunnels During Seismic Events [2]

Large scale shake table studies have been conducted [3–5] analyzing seismic activity acting on tunnels. These studies are largely focused on tunnel structural design, the soil composition, and their interaction with the underground structure. This can help in the reader's understanding of underground structures in seismic events, but this work is more focused on what the system inside the structure experiences when systems such as shock absorbers are utilized to dampen an earthquake's effects.

Much research has been done to analyze above ground structures which utilize damping to counteract the effects of a seismic event [6,7]. In contrast, very little research has been done on shock isolation having to do with underground structures. In fact, minimal studies have been done regarding shock isolation of an underground structure through an isolation lining [8], and only one conceptual study was found regarding the shock isolation of a floor within an underground structure [9].

The system this work will be modeling is quite different from any other system found in literature review. One of the concepts that is most critical in the development of this dynamic model is the understanding of rigid body dynamics [10]. This understanding, coupled with the use of two different methods [11] to represent the rotation of a rigid body through three-dimensional space will be used to create a dynamic model to represent the system. The mathematics behind these methods will be further described in the Technical Background section.

Objective

The objective of this work is to create a design tool which is modularized to be able to accommodate any configuration of a similar system in any kind of base excitation loading. The tool can then be used to analyze how the parameters within the system affect the motion and acceleration experienced at any location within the shock isolated platform subsystem. Specifically, the desire of this analysis is to determine whether or not components and personnel within the platform subsystem will be able to survive a seismic event. Another desire of analysis is to determine how dependent the system is on its parameters to be able to create a priority list for maintenance to a current system or for development of a future system.

Description of System

The system being analyzed in this work is that of a hypothetical underground bunker. This system is made up of a shock-isolated platform suspended by four chains. These chains are directly attached to the inside shell of a cylindrical capsule with caps on each end, buried a number of feet underground. The system is therefore made up of three subsystems: chains, shock absorbers, and a platform. A visual representation of the

configuration of this system can be seen in Fig. 2. The intent of a system configuration of this type is to reduce and absorb sudden and violent motion. This is done in part through the pendulum effect resulting from the suspension of the chains and also in part from the shock absorbers.

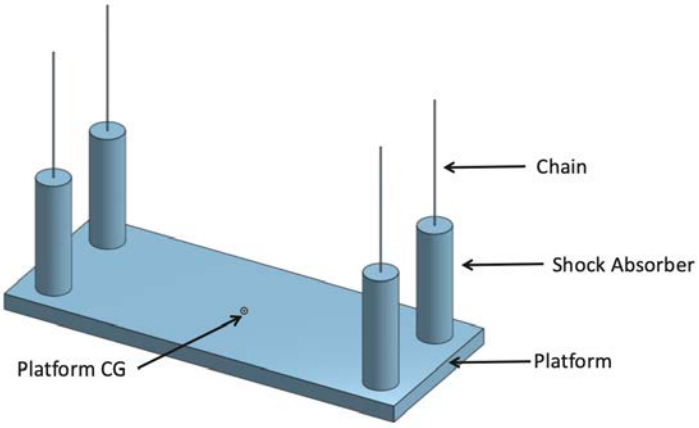


Figure 2. 3D Representation of System

Capsules which are used as the shell for underground bunkers are generally made of steel or steel rebar reinforced concrete. For our purposes, the makeup and general shape of the capsule are not considered. This is because the objective of this simulation is to analyze how the platform responds to motion of the anchor points of the chains and is not concerned with the capsule itself. The compression and ovaling described in the Literature Review section previously are not damage modes which are attempting to be mitigated with the shock absorber subsystem. Therefore, any motion experienced by the earth surrounding the capsule will be assumed to be the same motion experienced by the capsule.

The platform is the component of focus for analysis in the system. Rigidly attached to that platform are structures such as electronics racks, personnel seating, HVAC systems, living quarters, storage facilities, etc. The platform itself is assumed to

be rigid. The term “platform” will herein represent the platform and everything attached to it. With this assumption, the platform will be analyzed as a single rigid body. The dimensions of the platform are given hypothetical values of 25 ft long x 10 ft wide x 1 ft high. It is also given a hypothetical total weight of 4,000 lbs. Fig. 3 represents a platform with labeled dimensions.

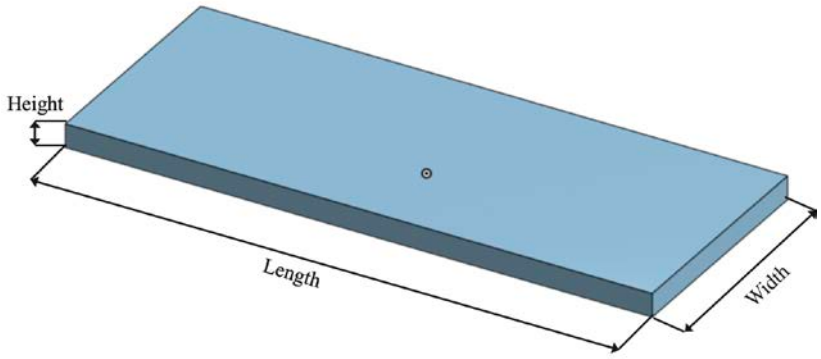


Figure 3. Labeled Platform Dimensions

Rigidly attached to the top of the platform are four shock absorbers located in the four corners of the platform. The purpose of shock absorbers in a system such as this is to counteract and damp any force acting perpendicular to the platform through the shock. The shock absorbers are considered a separate subsystem from the platform because they are altering and modifying all force being transmitted through the chains to the platform.

Shock absorbers vary widely in complexity and make up. A simple shock absorber was modeled for this hypothetical system. It is made up of a piston, oriented with the piston rod upward, a compression spring between the top of the inner cylinder volume and the top of the piston head, and a damping fluid between the bottom of the piston head and bottom of the inner cylinder volume. The configuration of this simple shock absorber can be viewed in Fig. 4. The shock absorber cylinder is given a hypothetical height of 7 ft, a diameter of 2 ft, and an assumed uniform steel cylindrical

shell thickness of 1 inch ($\frac{1}{12}$ th foot). The shock absorber piston is given a head thickness of 3 inches ($\frac{1}{4}$ th foot) and a rod length of 6.75 ft and 2 inches ($\frac{1}{6}$ th foot) in diameter. The shock absorber is given a spring constant of 700 $\frac{\text{lb}}{\text{ft}}$ and a damping coefficient of 150 $\frac{\text{lb·s}}{\text{ft}}$. Every component within the shock absorber cylinder is assumed to be made of steel.

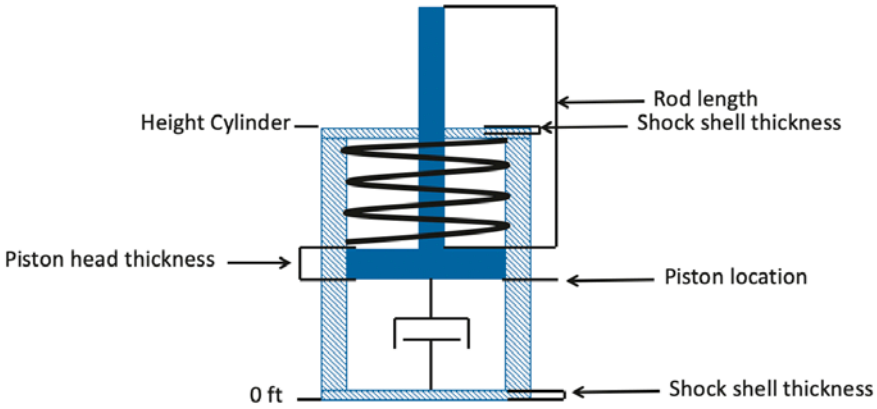


Figure 4. Labeled Shock Absorber Subsystem

Four chains are individually attached on one end to the top of each shock absorber piston rod and directly above each shock absorber to the inside of the capsule on the other end. These chains are assumed to be steel chain link. These four chains suspend the platform and shock absorber subsystems from the world around them. The chains also have the effect of making these subsystems act as a pendulum. Each chain is given a value for its unstretched length of 5 ft.

Given these initial parameters and configuration, a dynamic model can be created to represent how each subsystem interacts with each other system and how the system acts as a whole. In Chapter 2, the mathematic equations used to describe the motion of each subsystem will be derived. These equations will be the foundation for the models formulation in MATLAB's Simulink, described in Chapter 3, where the model will also

be verified. The model will then be tested using earthquake displacement data in Chapter 4. A parameter sensitivity study will also be conducted in this chapter to determine the system's dependence on its parameters, which can be used for future development of the system. A description of how the model can be applied is illustrated in Chapter 5, and the study is concluded in Chapter 6.

CHAPTER 2: TECHNICAL BACKGROUND

The study of dynamics is the study of how particles, bodies, and systems of bodies move and react when forces act upon them. It follows that a dynamic model is a computational design tool used to simulate how a particle, body, or system of bodies will move and react given the forces acting on the object over time. This is accomplished by deriving mathematical equations that represent how an object will act when excited. These mathematical equations are referred to as the model's equations of motion. The excitation of the model over time is referred to as the model's dynamic forcing function. The computational model uses initial conditions to create an initial state, then uses the forcing function to evaluate the equations of motion at each time step until the simulation time has expired.

The overall design intent behind the creation of a dynamic model is to best capture the nature of how a system acts while ensuring that all physical constraints imposed on the system are taken into account. There are many assumptions and simplifications that can be made when creating and implementing a dynamic model that, when used appropriately, represents an otherwise very complex physical system without losing the system's behavior. Simplifications can be regarding how a physical body may deflect, or even how a body moves, making the model easier to understand and implement. These simplifications can also be made in an effort to reduce the number of calculations necessary at each time step, benefiting the overall computational run-time and allowing for simulations to be run quickly and efficiently. Any time an assumption or

simplification is made, some accuracy can be lost within the model results. Therefore, assumptions must be made appropriately, when it is understood that the impact of said assumption will minimally impact the model's results.

As a three-dimensional dynamic model for the chain, shock absorber, and platform system is being designed, how the model is going to be excited needs to be considered. This work is focusing on excitation through the displacement of seismic activity, recognized as base excitation [12]. The magnitude of the force due to displacement will be calculated in the chain model and transmitted to the shock absorber model. Each shock absorber is rigidly attached to the platform and is therefore dependent on the orientation of the platform itself. The shock absorber model will take in the transmitted chain forces and has the ability to counteract and damp any forces acting perpendicular to the platform orientation. It will then transmit these modified forces to the platform model. Those chain force components not perpendicular to the platform are transmitted directly to the platform, unmodified. The platform model will then take these forces transmitted at each shock absorber attach point and will calculate any resultant dynamic motion in the form of rotation and/or displacement of the platform itself. Any motion of the platform also results in the motion of each shock absorber and chain, as they are all connected in one system.

Development of the Platform Model

The platform model is concerned with the kinetics of the platform, i.e. how the platform subsystem moves once forces have been applied. By utilizing the lumped parameter assumption, the entire platform subsystem can be treated as a single rigid body. When deriving equations of motion for this subsystem, it will be assumed that this

rigid body can undergo both translational and rotational motion, also known as general motion [13]. Any physical constraints on the amount of translation and rotation experienced by the platform subsystem will be implemented through the platform's interaction with the other subsystems within the system.

When working with rigid body dynamics in three-dimensional space, two different reference frames must be used. These two reference frames are the inertial reference frame and the body fixed reference frame. The inertial reference frame represents the "world", the origin of which can be located at any arbitrary point in space. A requirement of an inertial, or Newtonian, reference frame is that it is fixed or translating with a constant velocity. The physical earth will be used to represent the inertial reference frame because the accelerations resulting from rotation about the sun are assumed to be small and therefore negligible [13]. The body fixed reference frame represents the rigid body, the origin of which is conventionally located at the body's CG. Figure 5 further defines the two reference frames.

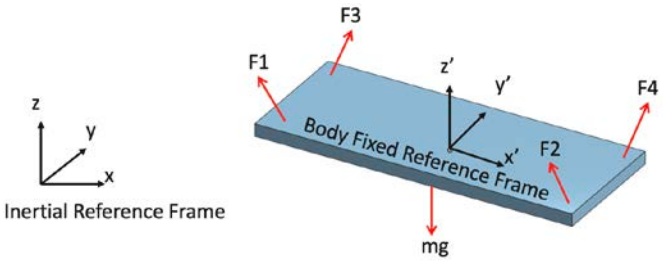


Figure 5. Platform Subsystem Coordinate Frames and FBD

Using both an inertial and body fixed reference frame is a simplification which has many benefits. Newton's Second Law is the basis for kinetics, which states that when an unbalanced force acts on a particle, the particle will accelerate in the direction of the

force with a magnitude that is proportional to the force [13]. This can be expressed mathematically with the equation:

$$\sum F = ma$$

where F is representing the forces acting on the rigid body, m represents the mass of the body, and a is the acceleration of the body. When utilizing this law, it is a requirement that accelerations be computed with respect to the inertial reference frame. However, the forces acting on the body may be easier to calculate in the body fixed reference frame. For instance, the shock absorber subsystems can only counteract and damp forces that are perpendicular to the surface of the platform, or acting in the z' axis of the body fixed reference frame as shown in Fig. 5. It is a much simpler process to calculate these forces in the body fixed frame and transform them to the inertial frame. This can be done through the use of the rotation transformation matrix.

The rotation transformation matrix is a construct of linear algebra which creates a mapping to move between the inertial and body fixed reference frames. In three-dimensional space, this is represented by a 3-by-3 orthogonal square matrix. This matrix can be computed using a number of methods. The two methods utilized in this work are the Euler angle method and the momentum method.

Rotation using the Euler Angle Method

Leonhard Euler introduced a method of representing the orientation of a rigid body with respect to a fixed coordinate system through the use of three angles. This work utilized the convention known as the roll, pitch, and yaw angles, often used in aerospace studies, to represent the three rotation angles. In this convention, roll, pitch, and yaw represent rotation about the x axis, y axis, and z axis respectively, as shown in Fig. 6.

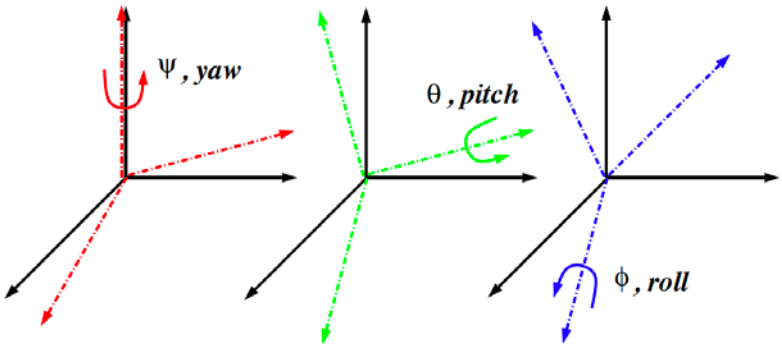


Figure 6. Visualization of Roll-Pitch-Yaw Rotations [12]

Using these angles, individual rotation matrices can be constructed to describe the rotation about each axis and then multiplied to represent the total rotation transformation matrix of the system. As derived by Ardakani and Bridges [12], these matrices are as follows:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The total rotation transformation matrix is then represented as:

$$Q = R_\psi R_\theta R_\phi$$

$$Q = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

Any vector can then be transformed from the body fixed frame to the inertial frame using the equation:

$$e_i = Q e_b$$

where e_b is the vector in the body fixed frame and e_i is the vector in the inertial frame. Transformation from the inertial to the body fixed frame can also be accomplished as follows:

$$e_b = Q^T e_i$$

Equations of Motion using the Euler Angle Method

The first step in determining how the platform moves dynamically in response to the forces acting on it is to calculate the total moment acting on the platform in the body fixed frame. It is an assumption that all the force vectors transmitted to the platform are already in the body fixed frame. Therefore, the total moment can be easily calculated in the body fixed frame as:

$$M = \sum r \times F_B$$

where r is the position vector from the platform's CG to the specific point where a shock absorber attaches to the platform and F is the corresponding force transmitted by the shock absorber at that location. Due to the assumption that each component of the platform subsystem is rigidly attached to the steel platform, it follows that the CG and the moment of inertia of the platform remain constant and unchanging in the body fixed frame. The angular acceleration, α_B, can then be solved for in the body fixed frame using Newton's Second Law for Rotation:

$$\alpha_B = {I_c}^{-1}(M - \omega_B \times H)$$

where I_c is the inertia tensor, ω_B is the angular acceleration in the body fixed frame and H is the angular momentum, which can be calculated as:

$$H = I_c \omega$$

This angular acceleration can then be transformed into the inertial frame using the previously defined relation:

$$\alpha_i = Q\alpha_B$$

From here, the angular acceleration in the inertial frame, α_i, can be integrated twice to obtain the Euler angles, ϕ, θ, and ψ. Once the Euler angles are known, the original forces in the body fixed frame can be transformed into the inertial frame using the relation:

$$F_i = QF_B$$

With all necessary quantities now in the inertial frame, Newton's Second Law can be used to calculate the acceleration, a, of the platform:

$$a = \frac{1}{m_{platform}} \sum F_i$$

This acceleration can then be integrated twice to obtain the displacement of the platform's CG in the inertial reference frame.

Rotation using the Momentum Method

Alternatively, a vector can be constructed in which all possible information necessary to represent a system is contained. This is referred to as the state vector. In this method, Baraff defines [10] the state vector, $Y(t)$, and its derivative, $\frac{d}{dt}Y(t)$, for a rigid body to be:

$$Y(t) = \begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix}$$

$$\frac{d}{dt}Y(t) = \begin{pmatrix} v(t) \\ \dot{R}(t) \\ F(t) \\ \tau(t) \end{pmatrix}$$

where $x(t)$ is position, $R(t)$ is the rotation transformation matrix, $P(t)$ is the linear momentum, $L(t)$ is the angular momentum, $v(t)$ is the velocity, $F(t)$ is the force acting on the body, and $\tau(t)$ is the torque acting on the body. $\dot{R}(t)$ is representing how the rotation matrix changes with time and can be calculated by using the equation:

$$\dot{R}(t) = \omega(t)^* R(t)$$

where $\omega(t)^*$ is a special angular velocity matrix defined as:

$$\omega(t)^* = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Equations of Motion using the Momentum Method

By utilizing this method, calculations will be done almost exclusively in the inertial reference frame. Knowing the force experienced by the platform, linear momentum can be directly solved for by integrating this force input. The velocity of the CG of the platform can then be calculated using the equation:

$$v(t) = \frac{P(t)}{m_{platform}}$$

where $m_{platform}$ is the mass of the platform. This velocity can then be integrated to directly solve for the position of the CG of the platform.

Similar to the relationship between force and linear momentum, by knowing the total moment, also referred to as torque, acting on the platform, the angular momentum can be directly solved for by integrating the moment input. The angular momentum can then be used to solve for the angular velocity, $\omega(t)$, using the relation:

$$\omega(t) = I(t)^{-1}L(t)$$

where $I(t)$ is the platform's inertia tensor in the inertial frame. As discussed previously, the platform's inertia tensor is constant and easily computed in the body fixed frame. This inertia tensor can then be converted to the inertial frame using the relation:

$$I(t)_{inertial} = R(t)I(t)_{body\ fixed}R(t)^T$$

Once these values are obtained, $\omega(t)^*$ and $\dot{R}(t)$ can be calculated and the new $R(t)$ obtained through the integration of $\dot{R}(t)$.

Development of the Shock Absorber Model

The goal of the shock absorber dynamic model is to determine how force input from the chains may be modified and then transmitted to the platform. To develop the equations of motion to represent this model, analysis must be done to determine how forces are being transmitted through the shock. This can be done in three steps: force analysis of the piston within the shock absorber, analysis of how these forces are transmitted to the shock absorber cylinder, followed by how the forces are transmitted from the cylinder to the platform. A free body diagram (FBD) of the forces acting on the shock absorber subsystem as a whole as well as on the piston and cylinder can be viewed in Fig. 7. These are two dimensional pictures to represent the three dimensional system.

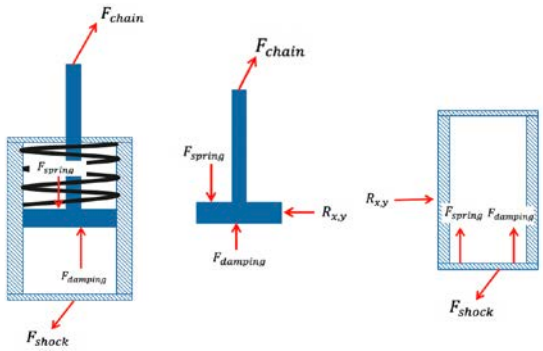


Figure 7. FBD of Shock Absorber Subsystem

An important first step in this process is to transform the input chain force acting at the top of the shock absorber piston from the inertial frame to the body fixed frame. This is due to the dependence of the shock absorbers on the orientation of the platform, represented by the body fixed frame. Forces can then be broken down into their x', y', and z' components for more straight forward analysis.

In the x' and y' directions, there is no acceleration of the shock absorber relative to the platform itself. Therefore, the two governing equations utilizing Newton's Second Law in these directions become:

$$\sum F_x = 0$$

$$\sum F_y = 0$$

In both the x' and y' cases, the only forces acting on the system are the x' and y' component of the chain force and the resultant reaction forces from the shock absorber. From the chain to the shock absorber piston, the x' and y' component of the chain force would act equal in magnitude and opposite in direction. When the forces are transmitted from the piston to the shock absorber cylinder and then again from the cylinder to the platform, they act equal in magnitude and opposite in direction. This results in the following equations of motion in the x' and y' directions:

$$F_{shock,x} = F_{chain,x'}$$

$$F_{shock,y} = F_{chain,y'}$$

When analyzing the shock absorber piston in the z' direction, there is an acceleration resulting from the motion of the piston within the shock absorber cylinder. The forces acting on the piston in this direction are the z' component of the chain force,

the spring force and the damping force. The spring and damping forces will always be opposing the motion of the piston within the cylinder. Any force resulting from gravity acting on the piston is neglected, due to the assumption that the mass of the piston is significantly smaller than that of the platform. This results in the following equation of motion for the piston:

$$F_{chain,z'} - F_{spring} - F_{damping} = m_{piston}a_{piston}$$

where m_{piston} and a_{piston} are the mass and acceleration of the piston, respectively. The spring force can be calculated as:

$$F_{spring} = k\Delta l$$

where k is the spring constant and Δl is the difference in length from the current location of the top of the piston head to location of the top of the piston head if it was resting on the bottom of the inner volume of the cylinder (i.e. the amount the spring has been compressed, assuming its free length is the equal to the maximum available length with the piston head resting on the bottom of the shock absorber cylinder). This can be best visualized through the use of Fig. 8. The damping force can be calculated as:

$$F_{damping} = cv_{piston}$$

where c is the damping coefficient and v_{piston} is the velocity of the piston. Once these quantities are calculated, the acceleration of the piston can be solved for and the velocity and location of the piston integrated from this value.

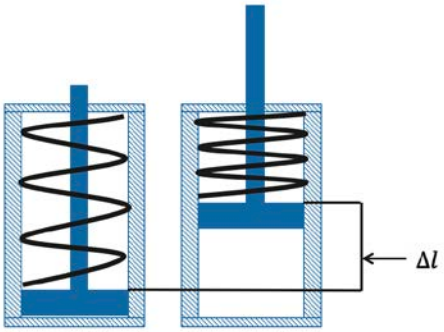


Figure 8. Visualization of Spring Length Change

When analyzing strictly the shock absorber cylinder the in the z' direction, the forces acting are the spring force, the damping force, and the reaction force from the shock absorber. The reaction force from the shock absorber piston to the cylinder will act equal in magnitude and opposite in direction to the spring and damping forces. The direction of these forces will be reversed again when the forces are transferred from the shock absorber cylinder to the platform. The shock cylinder is also not accelerating relative to the platform. Therefore, the final equation of motion for the shock absorber subsystem is the following:

$$F_{shock,z} = F_{spring} + F_{damping}$$

Development of the Chain Model

This is a case where a full model to represent how the chains move dynamically can be very complex and computationally expensive [14]. In an effort to simplify the subsystem and obtain a quick and efficient simulation, assumptions and simplifications are made to this subsystem, making it the simplest subsystem in all.

The first step in the development of the chain model is to review what is known up to this point. The locations where the chains attach to the inner surface of the capsule along with any displacement from the original locations of these points are known, as this is the forcing function exciting the simulation. It is also known where the platform will

move and/or rotate in reaction to the forces acting on it. Through the use of position vector algebra from the platform's CG to the top of the shock absorber, the locations of the tops of the shock absorber pistons where the second end of the chains attach are also a known quantity. Therefore, the current length of each chain can be assessed using a simple position vector calculation from the top of the shock absorber piston to the location of the chain attach point on the inner surface of the capsule.

The next step is to quantify the forces generated due to any displacement of the capsule attach points. By analyzing how forces are transmitted through the chains, it can be observed that either 100% of force is transmitted when the chain is in tension or zero force is transmitted when the chain is slack. This is due to the chain not being a rigid body, such as a steel rod, and not being able to compress. Unfortunately, switches such as this where the force is either "on" or "off" can be very challenging to successfully model due to the discontinuities they cause. Instead of modeling the chain as a switch, the chain can be modeled as a very stiff spring, allowing for a slight compliance.

Utilizing the above simplifications, the model to represent the chain can be obtained. If the chain is in tension and being "stretched", the force transmitted by the chain can be calculated as:

$$F_{chain} = k_{chain}(l_{current} - l_0)$$

where k_{chain} is the spring constant of the chain, $l_{current}$ is the current length of the chain calculated using the chain position vector, and l_0 is the unstretched length of the chain. If the chain is slack, then the force transmitted by the chain can be expressed as:

$$F_{chain} = 0$$

Development of the Seismic Forcing Function

Developing the seismic forcing function for this simulation is dependent on determining the displacement caused by a seismic event. The energy released during an earthquake is propagated through the crust of the earth in the form of two different waves, causing displacement. These are primary waves, “P waves”, taking the form of compression waves, and secondary waves, “S waves”, taking the form of transverse, or shear, waves [15]. This can be represented through the graphic in Fig. 9.

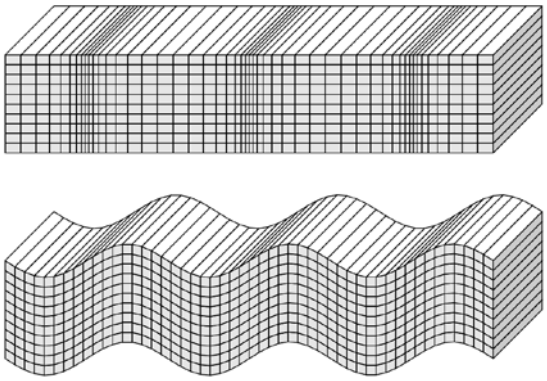


Figure 9. P and S Waves[1] [16]

The momentum equation for a seismic wave can be represented as [16]:

$$\ddot{u} = \alpha^2 \nabla\nabla \cdot u - \beta^2 \nabla \times \nabla \times u$$

where u is the displacement, $\ddot{u}$ is the acceleration, ∇ is the gradient, α is the P-wave velocity calculated as:

$$\alpha^2 = \frac{\lambda + 2\mu}{\rho}$$

and β is the S-wave velocity calculated as:

[1] P waves are represented by the top graphic and S waves are represented by the bottom graphic.

$$\beta^2 = \frac{\mu}{\rho}$$

where λ and μ are Lamé coefficients and ρ is the density. The calculations involved in these equations to model a seismic event in three dimensions can be very intensive.

Alternatively, displacement data gathered directly from seismographs can be utilized. This data can be searched for and downloaded from sources such as the U.S. Geological Survey (USGS). Three separate data files representing displacement in the east-west, north-south, and up-down directions can be manipulated and compiled together to create the total displacement experienced. These files also include necessary information such as earthquake magnitude, device orientation, and data sample rates.

Either method can be used to obtain displacement data to be used as the input for model simulation. For this work, the three dimensional earthquake displacement will be obtained by downloading data files. Specifics for this process will be discussed in the World Model Formulation section.

CHAPTER 3: SIMULINK MODEL FORMULATION

MATLAB's simulation software, Simulink [17], was chosen to be used for the formulation of a dynamic model to represent the system. There are two design goals behind the model to make it an effective and worthwhile design tool for the end user. The first goal is to focus on the modularization of each subsystem to allow for the subsystems to be arranged in any configuration. The second goal is to create an easy and straight forward user interface to allow for quick manipulation of the system parameters. This can be done through the use of initialization scripts, the details of which will be further discussed later. The formulation of each subsystem's model in Simulink will be presented. This is then followed by an overview of system operation, describing how the model components work together. All code discussed in this section can be found in Appendix A. A flowchart of how the model and its supporting functions and scripts work together is shown in Fig. 10. Orange boxes are used in this figure to represent scripts that require input from the user.

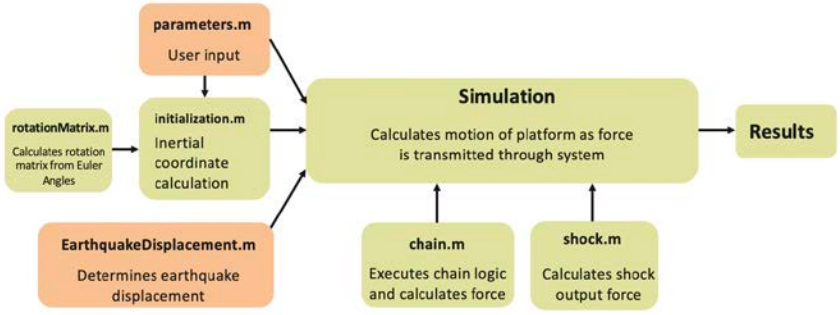


Figure 10. Model Flowchart

Model Initialization

There are many parameters which make up the system, such as the length, width, and height of the platform, or the spring and damping coefficients of the shock absorber, etc. All of these parameters will be provided by the user and need to be in one easily accessible location. This is done by declaring all the system parameters in a script called “parameters.m”. Some of the parameters entered by the user include the location of the CG and the platform attach points, where the shock absorbers are attached. These coordinates are entered in the “platform coordinate system”, with the origin in the bottom front left corner of the platform, instead of the convention with the origin located at the CG of the platform. This was a choice made by the model designer based on the assumption that these coordinates and the moment of inertia could be easily obtained from a 3D CAD model of the subsystem. Using this method, components making up the configuration of the platform subsystem can be quickly modified within the 3D CAD model and the modified parameters could be swiftly obtained and updated for model simulation.

Any other parameters used by the model during simulation that are calculated from the user given parameters are located in an additional script called “initialization.m”. System parameters calculated in this initialization script are values such as position vectors between the platform attach points and the platform’s CG in the body fixed frame. Also calculated in this script is the initial location of the platform CG in the inertial reference frame. This is done by defining a plane in three dimensional space using three of the chain attach points and calculating the normal vector to the plane, n, using the equation:

$$s = \begin{bmatrix} \text{chain attach point 1 x} \\ \text{chain attach point 1 y} \\ \text{chain attach point 1 z} \end{bmatrix}, t = \begin{bmatrix} \text{chain attach point 2 x} \\ \text{chain attach point 2 y} \\ \text{chain attach point 2 z} \end{bmatrix}, u$$

$$= \begin{bmatrix} \text{chain attach point 3 x} \\ \text{chain attach point 3 y} \\ \text{chain attach point 3 z} \end{bmatrix}$$

$$n = (t - s) \times (u - s)$$

Any initial rotations in the platform resulting from an asymmetric configuration of the system can then be calculated using the equations:

$$\phi_o = \sin\left(\frac{-n_y}{n_z}\right)$$

$$\theta_o = \sin\left(\frac{-n_x}{n_z}\right)$$

There is no configuration of the system that could cause an initial rotation about the z axis. From these rotations, the inertial CG can be calculated.

These scripts can be added to the initialization function, “InitFcn”, under the Callbacks tab of the Model Properties. A screenshot of these settings from the model can be seen in Fig. 11. The initialization functions are scripts that will be run every time the model’s run button is pressed before the simulation begins. When these scripts are run, the variables within them are loaded into the MATLAB base workspace, where they can be accessed by the model.

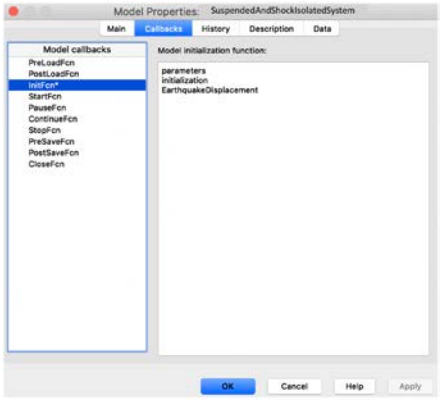


Figure 11. InitFcn Callback within Model Properties

The initialization script titles are placed within the "Model initialization function" box in the order in which they need to be run. The "EarthquakeDisplacement" script will be discussed in the World Model Formulation section later.

Platform Model Formulation

As discussed in the Development of the Platform Model section, there are two sets of equations of motion which can be used to represent the dynamic motion of the platform, resulting from the two rotation transformation matrix methods utilized in this work. There are therefore two different Simulink platform models that can be formulated. The input for each platform model will be the four force vectors in the body fixed reference frame and the four z' distances from the platform attachment point to the top of the shock absorber piston attach point from each of the four shock absorbers. The output of each model will be the location of each shock absorber piston attach point in the inertial reference frame.

Euler Angle Platform Model Formulation

An overview of the platform Simulink model utilizing the Euler angle rotation method is shown in Fig. 12. The first step of this method is to calculate the total moment

acting on the body to determine the angular acceleration of the platform in the body fixed frame. This total moment acting on the body is calculated using the equation:

$$M = r_1 \times F_1 + r_2 \times F_2 + r_3 \times F_3 + r_4 \times F_4$$

where r is the position vector from the platform's CG to the respective shock absorber piston attach points and F is the force vector for the respective points in the body fixed frame. The angular acceleration in the body fixed frame can then be calculated using the previously described equation:

$$\alpha_B = {I_c}^{-1}(M - \omega_B \times H)$$

where ω_B is the angular acceleration in the body fixed frame and H is the angular momentum of the platform. The model uses the initial condition that there is zero angular velocity to solve the first time step.

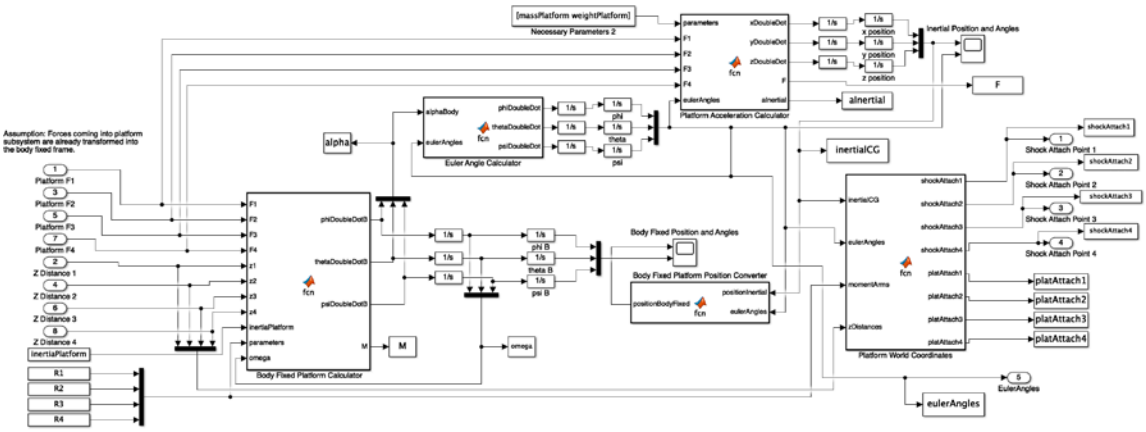


Figure 12. Euler Angle Platform Model Overview

The angular acceleration in the body fixed frame is then sent to the Euler Angle Calculator function to be transformed into the inertial frame. This is accomplished using the relation:

$$\alpha_i = R\alpha_B$$

where R is the rotation transformation matrix calculated using Euler angles. This function utilizes the initial condition that $\phi, \theta,$ and ψ are zero to solve the first time step. The inertial angular acceleration values are then integrated twice to obtain the Euler angles. A function script named “rotationMatrix.m” is utilized in this function, in which the Euler angles are passed to the function, the total rotation transformation matrix is calculated, and then the rotation transformation matrix is returned. This function is used to reduce unnecessary code duplication.

The Euler angles can now be used to solve for the translational acceleration of the platform by transforming the inputted forces from the body fixed frame to the inertial frame. Translational acceleration can be solved for using the equation:

$$a = \frac{1}{m_{platform}}(F_1 + F_2 + F_3 + F_4 - F_{weight})$$

where F_{weight} can be calculated as the platform weight vector:

$$F_{weight} = \begin{bmatrix} 0 \\ 0 \\ m_{platform}g \end{bmatrix}$$

where g is gravity. This acceleration can then be integrated twice to obtain the location of the platform’s CG in inertial space. This is accomplished by using the initial location of the platform CG, calculated by “initialization.m”, as initial conditions within the x, y, and z position integrator blocks, shown in Fig. 13.

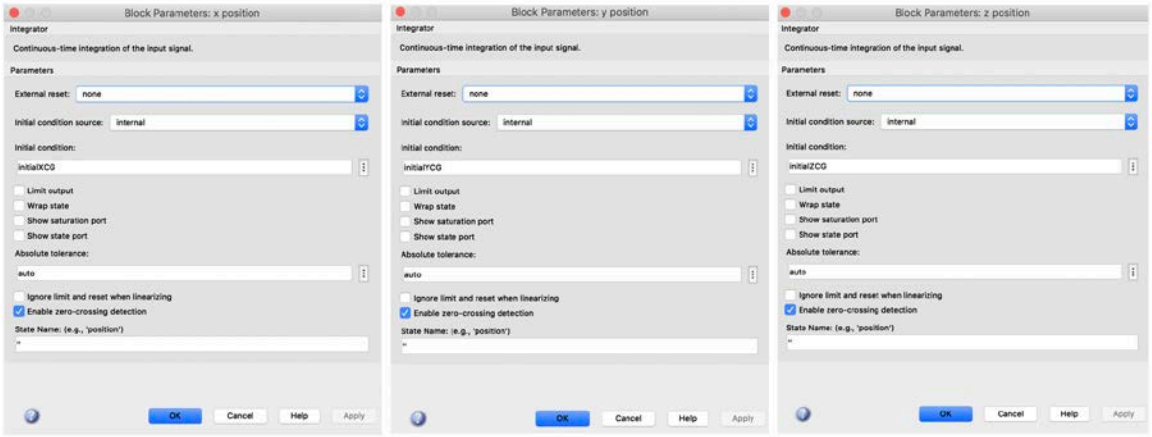


Figure 13. Initial Conditions for Position Integrator Blocks

Once the location of the platform CG is known in the inertial frame, the updated location of the shock absorber piston attach points can be calculated in the Platform World Coordinates function. This is done using simple vector algebra, adding the position vectors together with the CG of the platform. The platform attach points are calculated so they can be sent to the MATLAB base workspace in the form of timeseries data arrays. These arrays can then be used by post processing scripts, such as an animation script to assist in the visualization of the system's motion. The shock absorber piston locations and the platform's Euler angles can then be outputted for use by the rest of the system.

Derivation of State Platform Model Formulation

An overview of the platform Simulink model utilizing the derivation of state method is shown in Fig. 14. The first step of this model is to calculate the total force and total moment acting on the platform in the inertial frame within the Total Force and Moment function. This can be done using the equations:

$$F_{total} = R(F_1 + F_2 + F_3 + F_4) - F_{weight}$$

$$M = R(r_1 \times F_1 + r_2 \times F_2 + r_3 \times F_3 + r_4 \times F_4)$$

where r is the position vector from the platform's CG to the respective shock absorber piston attach points, F is the force vector for the respective points in the body fixed frame, and R is the total rotation transformation matrix.

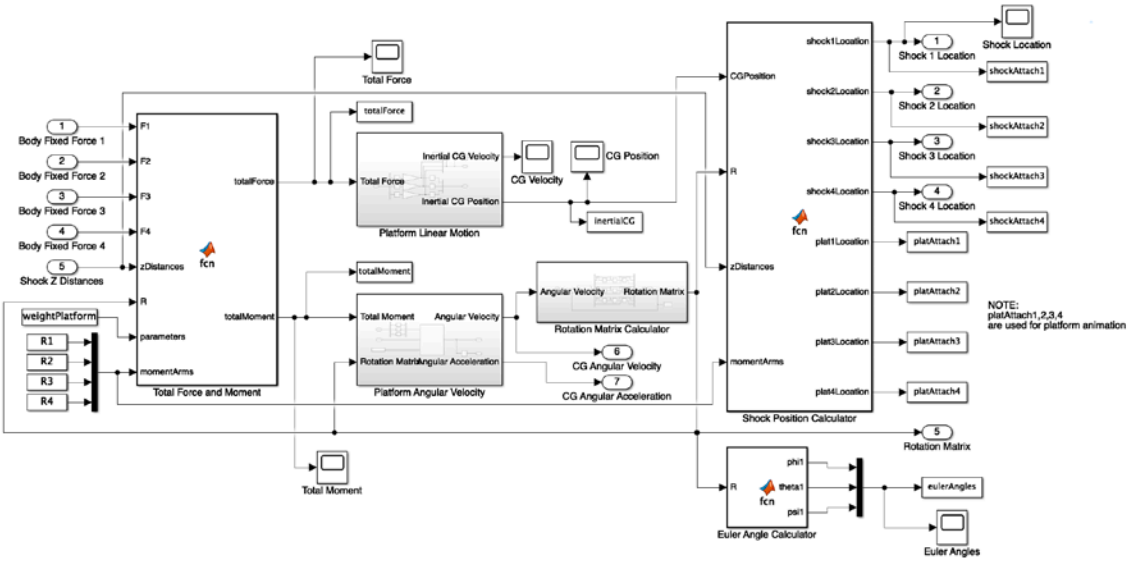


Figure 14. Momentum Method Platform Model Overview

The total moment is then passed to the Platform Angular Velocity group, along with the total rotation transformation matrix, shown in Fig. 15. The total moment is integrated to obtain the angular momentum of the platform. The platform's angular velocity can be calculated by rotating the platform's inertia tensor into the inertial frame and using the equation:

$$\omega(t) = I(t)^{-1}L(t)$$

where $L(t)$ is the angular momentum. The derivative of the angular velocity can be taken to obtain the angular acceleration.

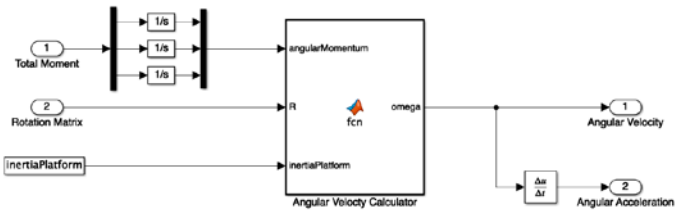


Figure 15. Momentum Method Platform Model Platform Angular Velocity Subsystem

The angular velocity is then passed to the Rotation Matrix Calculator group, shown in Fig. 16, where $\dot{R}$ can be calculated by using the equations:

$$\omega(t)^* = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$$\dot{R}(t) = \omega(t)^* R(t)$$

Each component of $\dot{R}$ can be integrated to obtain the total rotation transformation matrix, R. This updated rotation transformation matrix can then be outputted from the subsystem for use by the rest of the model.

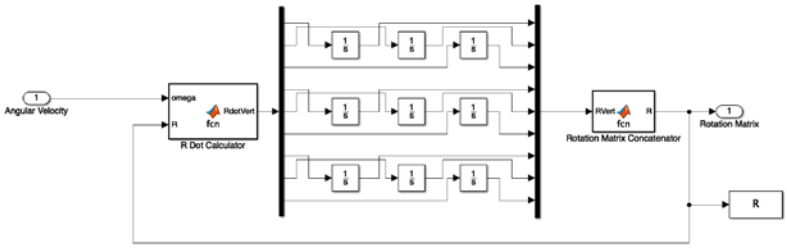


Figure 16. Momentum Method Platform Model Rotation Matrix Calculator Subsystem

The translational velocity and position of the platform's CG can be calculated within the Platform Linear Motion group, seen in Fig. 17. Here, the total force can be integrated and multiplied by $\frac{1}{m_{platform}}$ to obtain the velocity of the CG. The velocity can be integrated again to obtain the platform CG's location in the inertial frame. This is

again done by using the initial location of the platform CG, calculated by "initialization.m", as previously shown in Fig. 11.

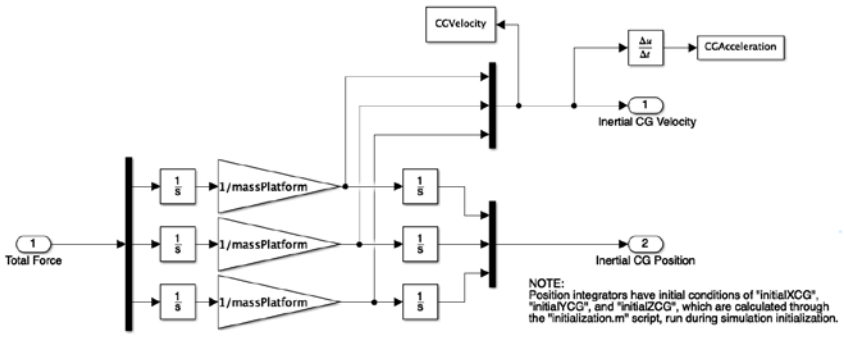


Figure 17. Momentum Method Platform Model Platform Linear Motion Subsystem

Knowing the location of the platform's CG in the inertial frame, the location of the shock absorber piston attach points can be calculated within the Shock Position Calculator function. This is done using vector algebra, as described in the Euler angle platform model method.

For comparison of Euler angle results between the two platform models, the Euler angles can be calculated from the total rotation transformation matrix. Slabaugh's method [18] was utilized in the Euler Angle Calculator function with the following equations:

$$\theta = -\sin^{-1}(R_{3,1})$$

$$\phi = \tan^{-1}2\left(\frac{R_{3,2}}{\cos\theta}, \frac{R_{3,3}}{\cos\theta}\right)$$

$$\psi = \tan^{-1}2\left(\frac{R_{2,1}}{\cos\theta}, \frac{R_{1,1}}{\cos\theta}\right)$$

Due to its relative simplicity when compared to the Euler angle platform model, the momentum method platform model is chosen as the ideal model. This model is therefore used to make up the full system model and was utilized during model testing.

Shock Absorber Model Formulation

The goal of the shock absorber Simulink model is to modify the inputted force from the chain, based on the orientation of the shock absorber, and output the resultant force to the platform. An overview of a shock absorber model is shown in Fig. 18.

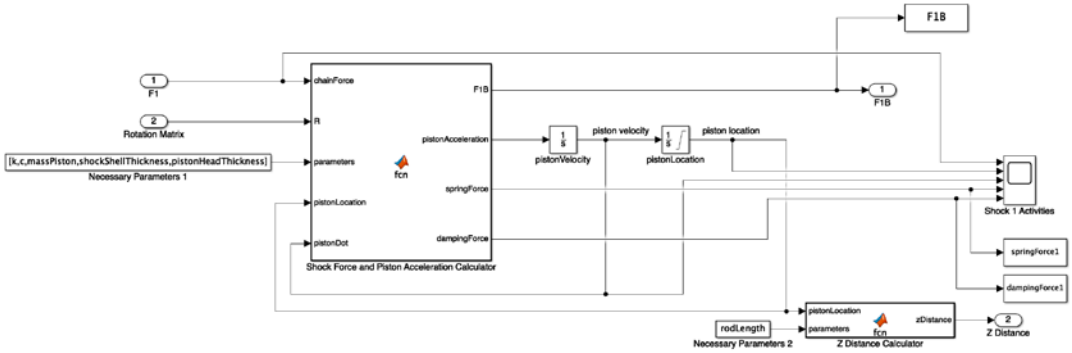


Figure 18. Shock Absorber Subsystem Model Overview

The first step of the Shock Force and Piston Acceleration Calculator function is to rotate the inputted chain force from the inertial frame to the body fixed frame. The outputted resultant force in the body fixed frame and the piston acceleration can then be calculated within the function utilizing a function script “shock.m” using the following equations:

$$a_{piston} = \frac{1}{m_{piston}}\left(F_{chain,z'} - F_{spring} - F_{damping}\right)$$

$$F_{shock} = \begin{bmatrix} F_{chain,x'} \\ F_{chain,y'} \\ F_{spring} + F_{damping} \end{bmatrix}$$

This calculated piston acceleration can then be integrated twice to solve for the current piston location within the shock absorber cylinder. Assuming the shock absorber starts the simulation with the piston head resting on the bottom of the inner volume of the shock absorber and assuming the piston cannot be at a location outside of the physical

confines of the cylinder itself, initial conditions and integration restrictions are placed on the piston location integrator block as shown in Fig. 19. Setting these initial conditions to zero forces the system to find its equilibrium position, but does not impact the dynamic response of the system.

It is also assumed that when the spring is fully compressed, it takes up the distance of about two piston head thicknesses. It is important to note that although these restrictions will keep the location data in the correct ranges, the correct physics will not be captured if the piston is in an extreme enough situation where the piston head hits either the top or bottom of the inner shock volume (i.e. the acceleration will not reach zero and a resultant impact force is not calculated). To prevent this phenomenon from occurring, if the piston location is within the last foot of the cylinder, the spring constant is multiplied by 3 to represent the nonlinearity known as spring hardening. The piston location is then sent to the Z Distance Calculator function to determine the z distance from the bottom of the shock absorber to the top of the shock piston attach point.

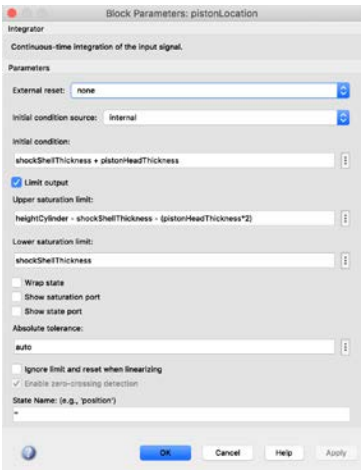


Figure 19. Shock Absorber Piston Location Integrator Block Parameters

Chain Model Formulation

The chain Simulink Model is a simple model whose overview can be seen in Fig. 20. The input to the model is the inertial location of the capsule attach point and the shock absorber piston attach point. These are passed to the Chain 1 function, where the "chain.m" function script is utilized to determine whether the chain is in tension or slack and what force is then outputted to the shock absorber. This transmitted force resulting from the chain being 'stretched' while in tension is calculated using the equation:

$$F_{chain} = k_{chain}(l_{current} - l_0)$$

if the chain is in tension. The direction cosines of the position vector are then used to break down the chain force to its x, y, and z components. If the chain is slack, the chain force becomes:

$$F_{chain} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

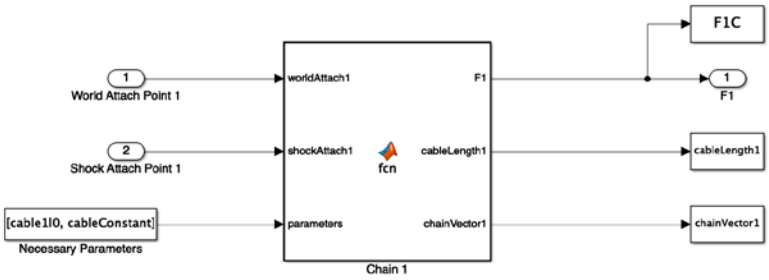


Figure 20. Chain Subsystem Model Overview

The spring constant for the chain was calculated from Young's modulus of elasticity for steel. Using the relationship $\sigma = E\varepsilon$, the spring constant can be calculated using the equation:

$$k_{chain} = E\frac{A}{L}$$

where Young's modulus of carbon steel, E, is 30 Mpsi [19], the area is the cross sectional area of a chain link, A, can be calculated:

$$A = 2(\pi r^2)$$

with a chain radius of 0.25 inches, and a length, L, of 5 feet. The spring constant is calculated to be 2,401,920 $\frac{\text{lb}}{\text{ft}}$. This spring constant is much too stiff for the simulation and causes instabilities within the system. The value was reduced till a reasonable stability was achieved in the model and a spring constant of 20,000 $\frac{\text{lb}}{\text{ft}}$ was chosen. Verification of this assumption will be discussed in the Parameter Sensitivity Study section.

World Model Formulation

The Simulink world model is where the forcing function of our simulation resides. The forcing function of this model is the displacement caused by an earthquake event. Earthquake data for this analysis was directly searched and downloaded from the Strong-Motion Virtual Data Center [20]. These files are downloaded as .smc files, which can be opened using any basic text editor. These files contain a lot of important information, such as the title and magnitude of the earthquake, as well as the sample rate in which the data was collected [21]. Three displacement files express the three dimensional earthquake displacement, where "HNN" in the file title is the North-South displacement, "HNE" is the East-West displacement, and "HNZ" is the up-down displacement. The data from these three data files was scraped and concatenated into one .csv file with each row representing the x, y, and z displacement (i.e. East-West, North-South, and up-down) for each time step.

The earthquake being used as the forcing function for this simulation is a magnitude 8.4 which occurred in Southern Sumatra, Indonesia on September 12, 2007.

The data was acquired by Caltech Tectonics Observatory and processed by USGS National Strong Motion Project from sensors located on Sikuai Island, West Sumatra. Data was collected at a sample rate of 200 samples per second with 129 seconds of the event observed. This displacement data can be loaded into the model workspace through the use of "EarthquakeDisplacement.m" as an initialization script in the initialization function callback properties of the model, shown in Fig. 11. This script loads in the .csv file created from the earthquake displacement files, converts the file from centimeters to inches, and creates data arrays for the x, y, and z, displacement data with 4 seconds of delay, where no displacement is experienced. This delay is to allow the system model to settle and find its equilibrium. It also adds 50 seconds at the end of the simulation where the displacement is set to the last value of the earthquake displacement, to allow for the model to again settle.

An overview of the World Displacement subsystem is shown in Fig. 21. The displacement vectors can be loaded from the model workspace and passed to the Earthquake Displacement Data ReadIn function. This function selects the current displacement from the total displacement vectors and outputs them to the World Attachment Location Calculator function. The displacement is equally applied to all attachment locations at each timestep. This is because the velocities of P and S waves are on the order of 14,665 $\frac{\text{ft}}{\text{s}}$ (4.47 $\frac{\text{km}}{\text{s}}$) and 8,464 $\frac{\text{ft}}{\text{s}}$ (2.58 $\frac{\text{km}}{\text{s}}$) respectively [22]. With the capsule attach points about 8-23 ft apart, this displacement will be felt 1.5-2.7 milliseconds apart. Reviewing the earthquake displacement data in Fig. 33, time between oscillation peaks is >3 seconds, or 1,000x slower than the speed of the wave through the earth. Therefore, any delay of displacement between attach points is not necessary.

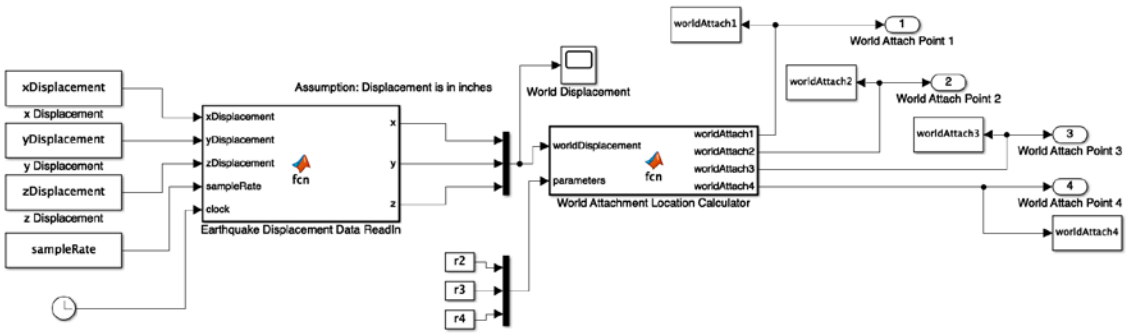


Figure 21. World Displacement Subsystem Overview

Sensor Formulation

A major focus of this model is to be able to measure the accelerations experienced at any location within the platform subsystem. This can be accomplished by using a sensor subsystem. An overview of this subsystem is shown in Fig. 22. In this subsystem, the platform CG's angular velocity and angular acceleration are passed to the Sensor Velocity and Acceleration Calculator function. Because the sensor is representing a location rigidly attached to the platform subsystem and the platform is rotating around its CG, the acceleration of a single point within this body can be calculated using the equations [13]:

$$v = \omega \times r$$

$$a = \alpha \times r + \omega \times (\omega \times r)$$

where ω is the angular velocity of the CG, α is the angular acceleration of the CG, and r is the position vector from the CG to the sensor location. The calculated acceleration is then sent to the Conversion to g's function where it is converted from $\frac{\text{ft}}{\text{s}^2}$ to g's.

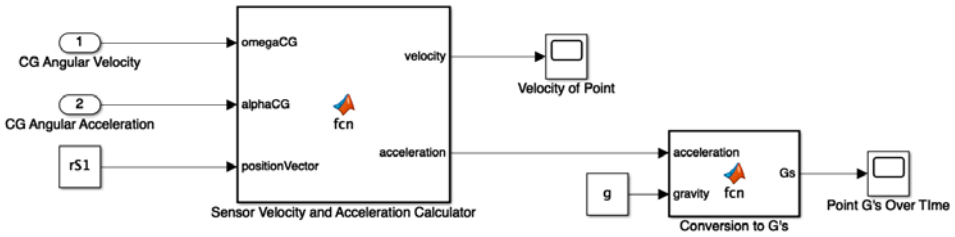


Figure 22. Sensor Subsystem Overview

System Operation

Finally, all of the subsystem components can be connected together to form a full dynamic model. The full system overview is shown in Fig. 23. By utilizing the modularized subsystems in this way, each system acts independently while adhering to the necessary physical constraints. Forces are transmitted down from the earthquake displacement through the chains, modified by the shock absorbers, to the platform. The platform then moves dynamically and the location of the system components are updated.

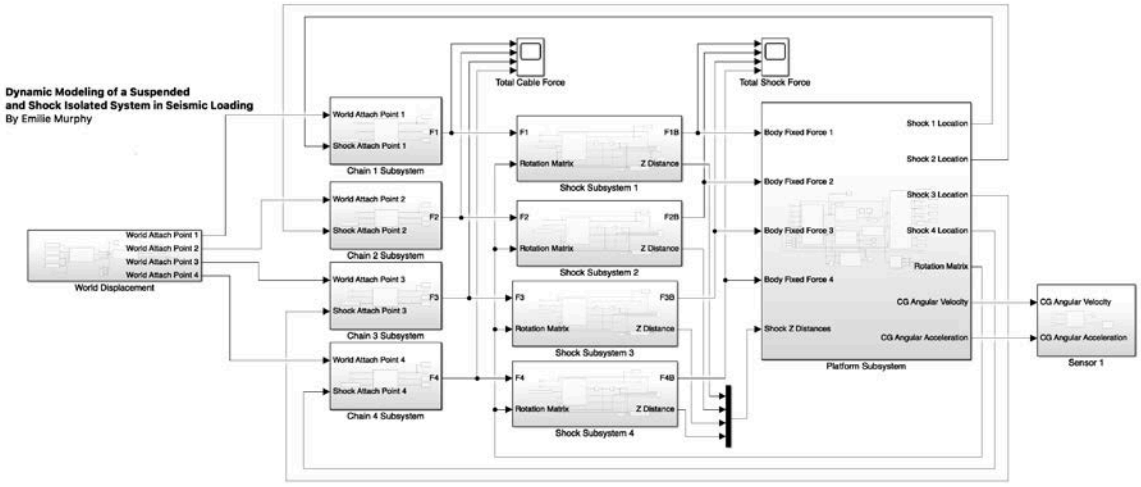


Figure 23. Full System Overview

Model Verification

Once the full model is operational, a series of simple test cases can be used to verify the results being obtained. As this is a complicated system, verifying any results

that deal with platform rotation can be very challenging. To do this, results from using the Euler angle mentod platform model will be compared with results obtained from using the momentum method platform model to verify their accuracy.

Test Case 1: Equilibrium

Set Up

Using the parameters described in the Description of System section of this work, the equilibrium state of the system can be verified. To test the equilibrium case, zero displacement will be sent to the World Attachment Location Calculator function within the World Displacement subsystem to allow for the system to find its equilibrium. This settling of the system is due to movement of the pistons within the shock absorber cylinders as the springs are compressed until they balance the weight of the platform. The amount the platform will be displaced can be calculated as follows:

$$\Delta l = \frac{F_{weight}}{k}$$

where k is the spring constant and F_{weight} is the weight of the platform. The spring constant can be calculated after analyzing that the 4 chains act in parallel and the 4 shock absorber springs act in parallel, while the chain and shock absorber springs act in series with each other. This k can be calculated as [12]:

$$k_{chain,total} = 4k_{chain}$$

$$k_{shock,total} = 4k_{shock}$$

$$k = \frac{k_{chain,total}k_{shock,total}}{k_{chain,total} + k_{shock,total}}$$

Using the system parameters, it can be expected that the platform will be displaced 1.479 ft. Being an overdamped system, it can also be expected that the platform will begin to

fall past its equilibrium point, experience a slight overshoot as it is pulled back upwards by the chains, and then settle to its equilibrium state.

Results

The expected results are corroborated by the simulation, as shown in Fig. 24. The platform's CG begins at -12.583 ft in the z direction and after 4 seconds has reached an equilibrium state of -14.062 ft.

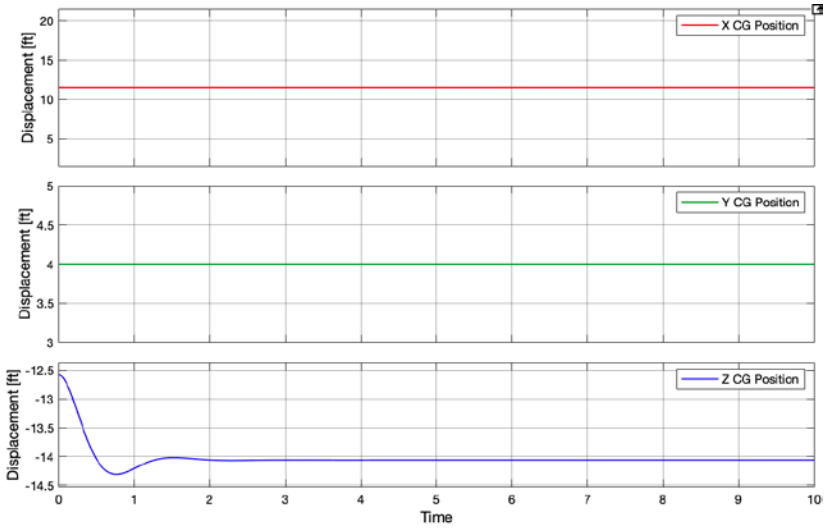


Figure 24. Equilibrium Test Case Results

Test Case 2: Measured Movement in the Z Direction

Set Up

For this case, after letting the system reach equilibrium, a step function will be used to apply a z displacement of 6 inches upward. The expected results would include the same equilibrium behavior for the first 4 seconds, followed by an overshoot from the chains sharply pulling the platform up, and then the platform reaching a new equilibrium position of -13.562 ft.

Results

The expected results are corroborated by the simulation, as shown in Fig. 25. The significant displacement at 4 seconds when the step function is applied is due to the very high spring constant of the chains. At 8 seconds, the system reaches its new equilibrium of -13.562 ft. The total forces in each chain are shown in Fig. 26. These results show that when the instantaneous displacement takes place, a large force is generated within the chains, resulting in the platform being sharply pulled upward. Once the chains have pulled the platform upward, the platform's momentum continues upward, resulting in the chains being slack and their forces going to zero for a few milliseconds. During this short time, the platform experiences freefall where it is affected by gravity alone. Once the chains are again in tension, the system again settles to its new equilibrium position.

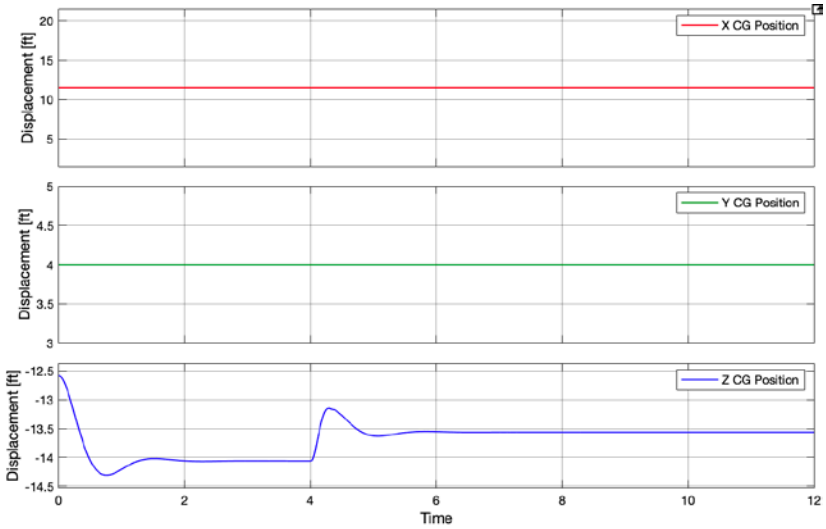


Figure 25. Measured Movement in the Z Direction Test Case Results

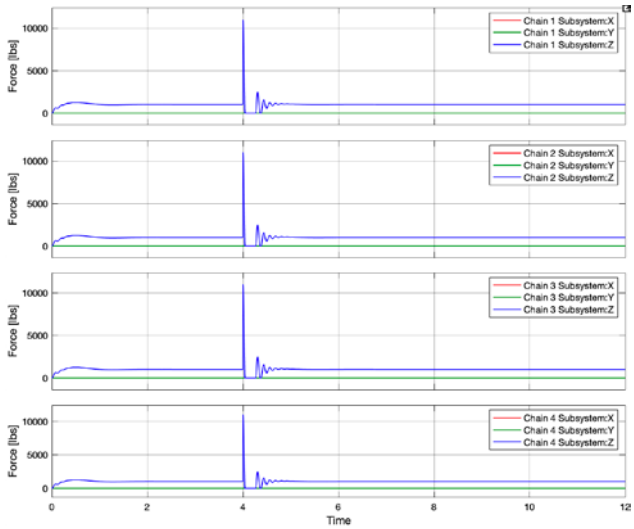


Figure 26. Measured Movement in Z Direction Chain Test Case Results

Test Case 3: Measured Movement in the X Direction

Set Up

After letting the system reach equilibrium, a step function will apply a 6 inch displacement in the positive x direction. This is expected to result in the x coordinate for the platform's CG to oscillate until reaching a new equilibrium from 11.5 to 12 ft. The x coordinate is expected to oscillate due to the platform rotations which develop from the sudden x movement. There are no mechanisms within the system to directly damp movement or forces in the x or y directions. However, the motion in the x direction is expected to be damped out over time and reach an equilibrium. This is because with each rotation about the y axis, a small component of the side loads applied to the system will act in the z' direction of the body fixed frame, which will be damped by the shock absorbers.

Results

The expected results are corroborated by the simulation, as shown in Fig. 27, utilizing both the Momentum method and Euler angle method platform model. The

location of the CG's x coordinate can be seen to oscillate, trending towards 12 ft after 30 seconds of simulation. The amplitude of these oscillations is also observed to be damped as the simulation time progresses. The Euler Angles can also be compared in Fig. 28 and although very small, rotation only about the y axis is observed.

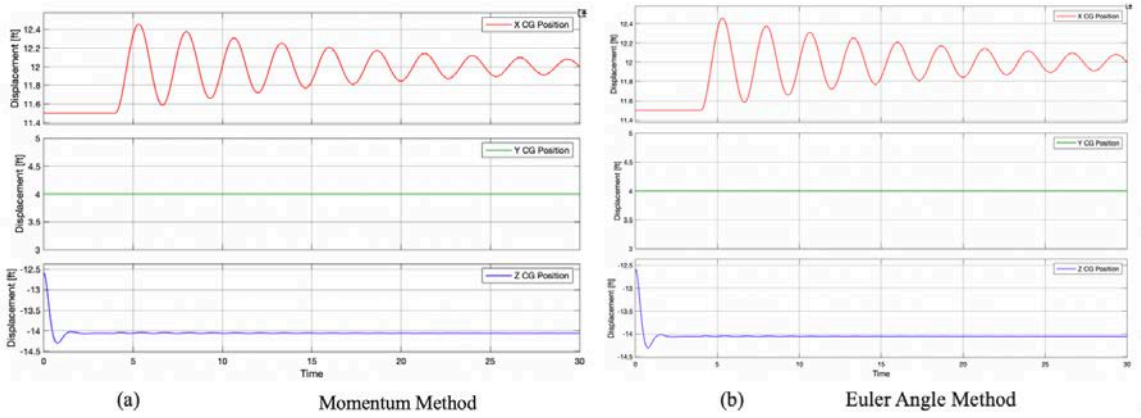


Figure 27. Measured Movement in the X Direction Test Case Results

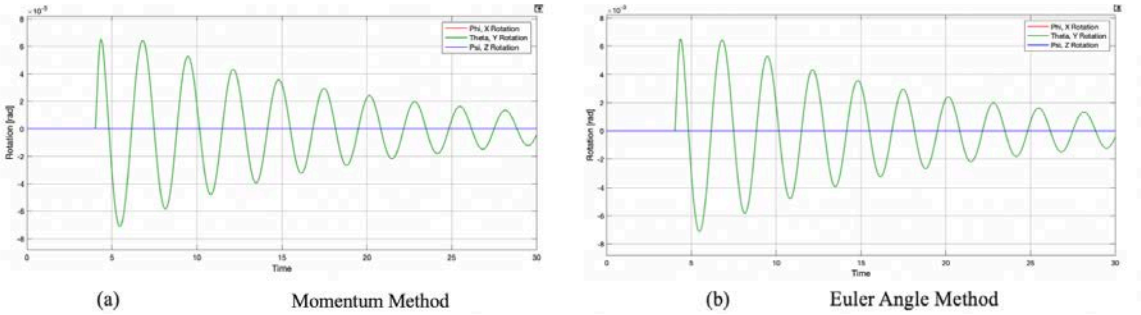


Figure 28. Measured Movement in X Direction Euler Angle Test Case Results

Test Case 4: Measured Movement in the Y Direction

Set Up

After letting the system reach equilibrium, a step function will apply a 6 inch displacement in the positive y direction. A very similar result to the measured movement in the x direction test is expected, again, due to the fact that there are no mechanisms within the system to directly damp movement or forces in the x or y directions. With this

in mind, the y coordinate is expected to oscillate due to rotation about the x axis until reaching a new equilibrium from 4 to 4.5 ft.

Results

The expected results are corroborated by the simulation, as shown in Fig. 29. As seen in the previous test case, the y coordinate oscillates, trending towards 4.5 ft after 30 seconds of simulation. The platform is also observed to be rotating about the x axis in Fig. 30, as expected.

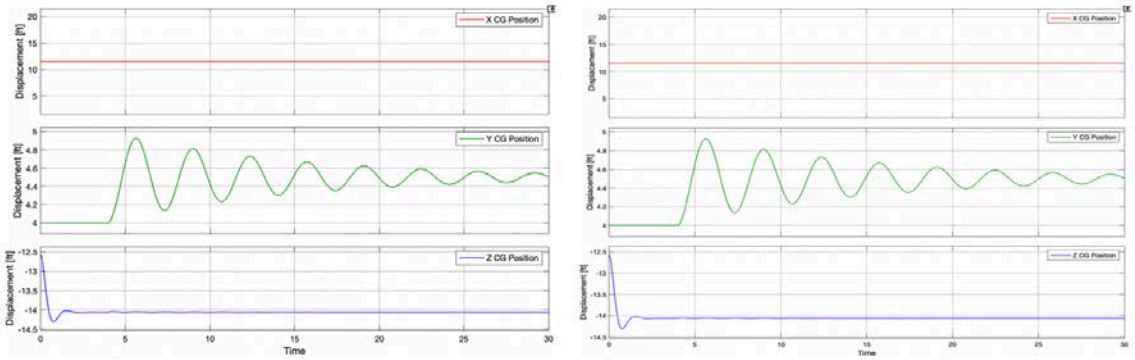


Figure 29. Measured Movement in Y Direction Test Case Results

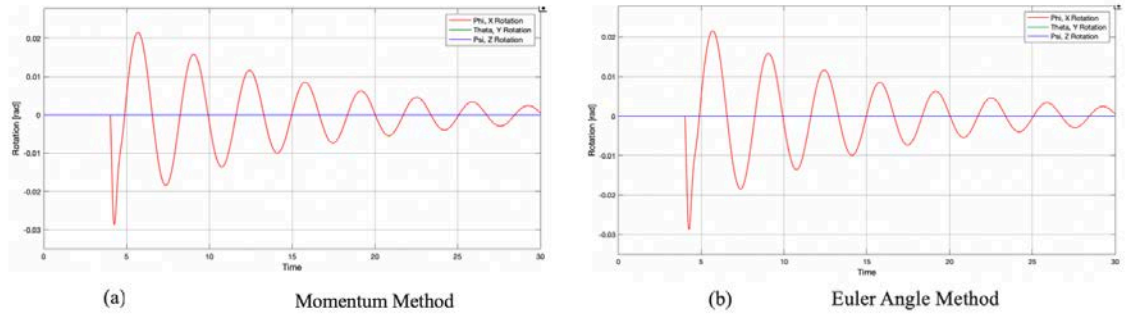


Figure 30. Measured Motion in Y Direction Euler Angle Comparison

Test Case 5: Asymmetric Configuration

Set Up

For this case, an asymmetric configuration is tested in which chains 1 and 2 are given an unstretched length of 5.25 ft, while chains 3 and 4 are kept at the previous

system parameter of 5 ft. This will cause an inherent rotation within the platform, even in its equilibrium state. This rotation angle can be calculated using geometry as follows:

$$\theta = \tan^{-1}\left(\frac{0.25\text{ ft}}{8\text{ ft}}\right)$$

This configuration results in a rotation about the y axis of 0.031 rad. The rotation matrix can then be calculated to be:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9995 & -0.0310 \\ 0 & 0.0310 & 0.9995 \end{bmatrix}$$

The position vector from the platform's attachment point 1 to the CG can be rotated from the body fixed frame to the inertial frame as follows:

$$R1_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9995 & -0.0310 \\ 0 & 0.0310 & 0.9995 \end{bmatrix}\begin{bmatrix} 11.5 \\ 4 \\ -0.5 \end{bmatrix}$$

$$R1_i = \begin{bmatrix} 11.5 \\ 4.0136 \\ -0.3758 \end{bmatrix}$$

Similarly, the initial position vector from the top of the shock absorber piston to the platform attach point can be rotated into the inertial frame as follows:

$$z1_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9995 & -0.0310 \\ 0 & 0.0310 & 0.9995 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ -7.0833 \end{bmatrix}$$

$$z1_i = \begin{bmatrix} 0 \\ 0.2195 \\ -7.0799 \end{bmatrix}$$

Using vector algebra from the first world attach point (i.e. the origin of the inertial reference frame), the new location of the platform's initial CG in the inertial frame before the platform has settled to equilibrium can be calculated to be:

$$CG = \begin{bmatrix} 0 \\ 0 \\ -5.25 \end{bmatrix} + z1_i + R1_i$$

$$CG = \begin{bmatrix} 11.5 \\ 4.2331 \\ -12.7057 \end{bmatrix}$$

The initial rotation resulting from this configuration will cause the platform's CG to oscillate about the x axis, obtaining similar results to the measured movement in the y direction case previously.

Another expected result from this configuration is that after settling to a state of equilibrium, there should be a force of 1,000 lbs acting at each chain.

Results

The expected results are corroborated by the simulation. The location of the platform CG in the inertial reference frame begins at a value of:

$$CG = \begin{bmatrix} 11.5 \\ 4.2349 \\ -12.7047 \end{bmatrix}$$

Any slight discrepancy between the expected and actual CG value can be attributed to rounding and the number of digits used in hand calculations versus computer calculations. The platform then begins to settle to its new equilibrium position and experiences an oscillation of its y coordinate, due to a slight rotation about the x axis, as shown in Fig. 31.

The chain forces acting on the system are also shown in Fig. 32. The fast oscillations during the first few seconds of simulation is attributed to the stiffness of the chains. A higher chain spring constant results in higher frequency oscillations, while a lower spring constant results in lower frequency oscillations. It can be observed that chains 1 and 2 have the same force profile, while 3 and 4 have a different force profile. Chains 1 and 2 reach an equilibrium state at ~961 lbs and chains 3 and 4 reach an equilibrium state at ~1,038 lbs. This is due to the rotation of the platform causing the

chains opposite each other to be in different amounts of tension as the rotation within the platform is damped out.

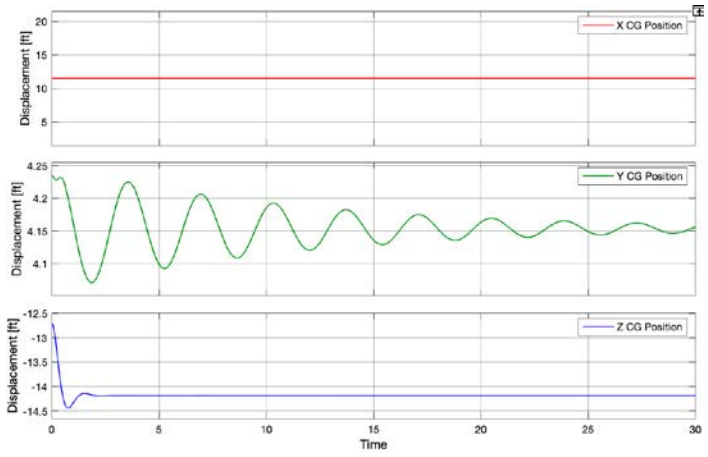


Figure 31. Asymmetric Configuration Test Case CG Results

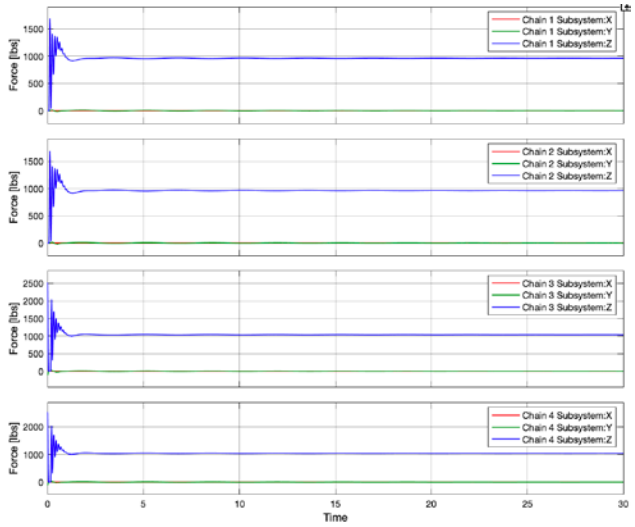


Figure 32. Asymmetric Configuration Test Case Chain Results

CHAPTER 4: MODEL EXPERIMENTATION, TESTING, AND RESULTS

In this section of the work, the model will be tested to determine how it acts dynamically from the displacement caused by an earthquake, with particular attention payed to the affects at sensor locations. A sensitivity study will then be conducted to determine how the spring and damping coefficients within the shock absorbers as well as the spring constant of the chains will be conducted to evaluate how the change of parameters affects the overall behavior of the system.

Earthquake Testing

The model is tested to analyze how it reacts under excitation from the displacement of an earthquake. The displacement data from the 8.4 magnitude earthquake occurring in Sumatra, Indonesia on September 12, 2007 is shown in Fig. 33. As described previously, there are 4 seconds of delay with zero displacement to allow the system to find equilibrium before the displacement data begins. There is then 50 seconds at the end of the data where the displacement is kept constant at the last data point given by the displacement data, to be able to observe how long it takes for the system to regain an equilibrium state.

Two sensors are placed within the platform system to monitor the accelerations at these specific points. The first sensor is placed in the middle of the length of the platform, 4 feet above the surface, near the back edge of the platform. This could represent the head of a person sitting at a table. The second sensor is placed two thirds of the way down the length of the platform, 6 feet above the surface, near the front edge of the platform. This

sensor could represent the center of a rack, containing sensitive electronics. A general idea of where these sensors are located within the system is shown in Fig. 34. It is expected that sensor 2 should experience higher G forces, as it has a larger position vector from the platform's CG.

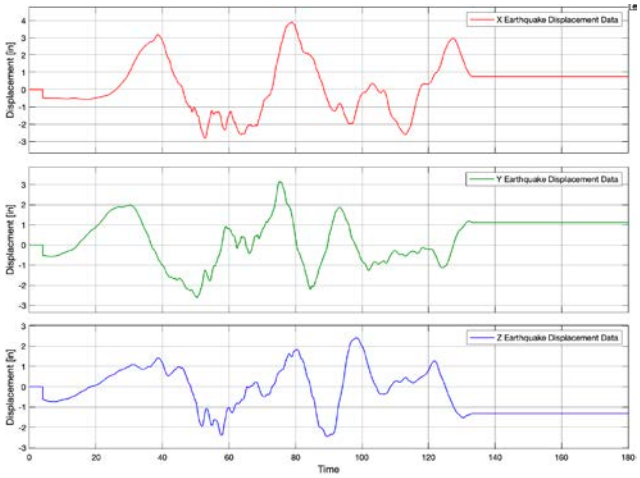


Figure 33. 2007 Sumatra Displacement Data

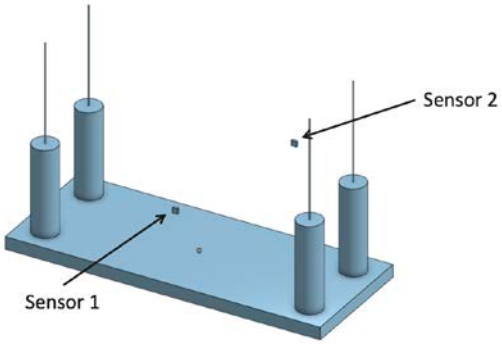


Figure 34. Sensor Locations within Platform System

The results of how the system acts dynamically as a result of the earthquake displacement forcing function is shown in Fig. 35. The platform CG experiences a total displacement range of 0.669 ft, 0.641 ft, and 0.411 ft in the x, y, and z directions respectively, after the system had reached equilibrium. Comparatively, the total earthquake displacement data range is 0.562 ft, 0.481 ft, and 0.406 ft in the x, y, and z

directions respectively. The platform therefore experiences a higher range of displacement in every direction when compared to the earthquake displacement. The system can also be observed to be near an equilibrium state 50 seconds after the earthquake has passed.

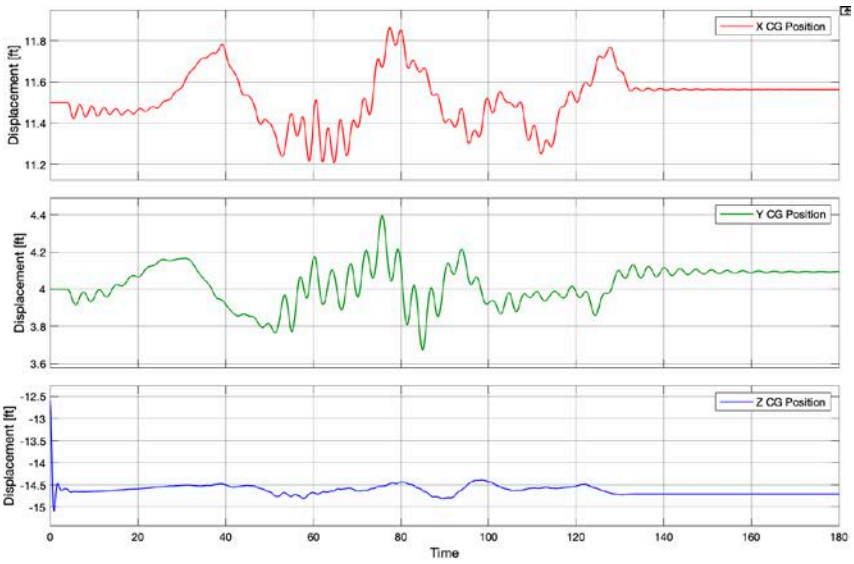


Figure 35. Dynamic System Results of Sumatra Earthquake Displacement

The accelerations experienced by the sensors can then be compared with the earthquake acceleration data. Earthquake acceleration data can also be directly downloaded from the Strong-Motion Virtual Data Center [20], as described in the World Model Formulation section previously. The sensor accelerations are shown in Fig. 36 and the earthquake acceleration data in Fig. 37. By looking closely at the displacement data in Fig. 33, a step can be seen at the instant the earthquake data is applied at 4 seconds. This is a direct artifact of the data itself and was not manipulated in an effort to maintain data integrity. With this in mind, large spikes can be observed in both figures at 4 seconds from this step in displacement data.

As expected, sensor 2 experiences higher overall accelerations compared to sensor 1 due to its position relative to the CG. However, neither sensor experiences over 1 G of acceleration.

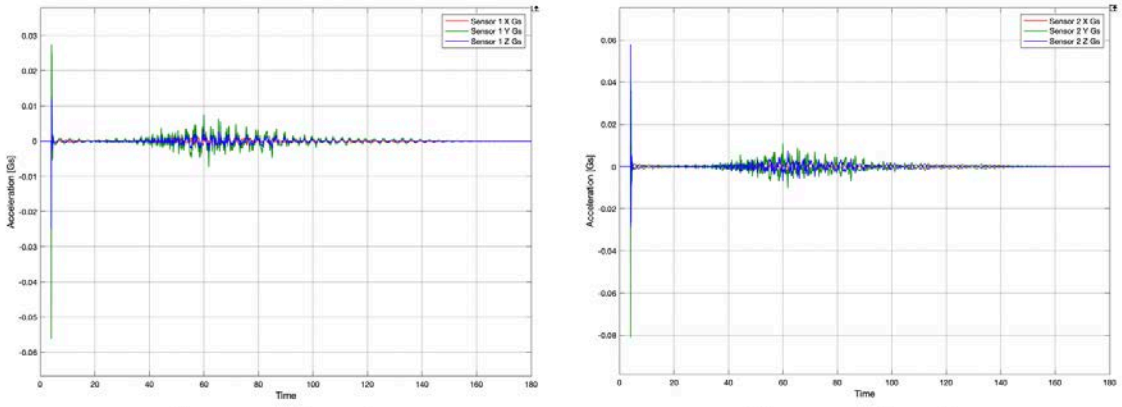


Figure 36. Sensor Acceleration Results

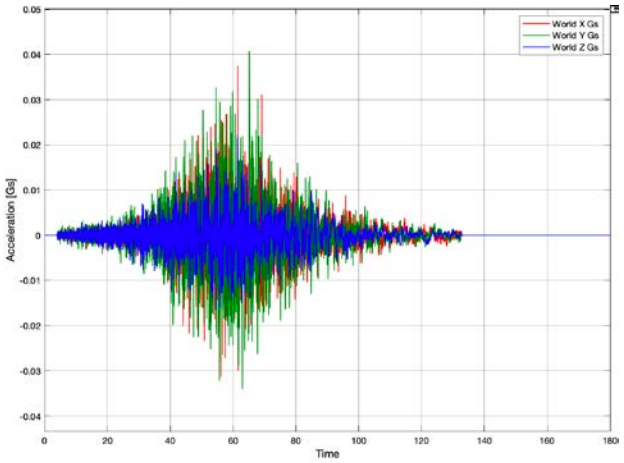


Figure 37. Data Acceleration Results

After the initial acceleration spike, beginning at 4.75 seconds, the acceleration ranges experienced by the system are in Table 1. This is a reduction of accelerations by the system of 93%, 72%, and 64% in the x, y, and z directions respectively at sensor 2. Although accelerations from the earthquake are minimal, this is a significant reduction by the system.

Table 1. Comparison of Acceleration Range Results

	X Acceleration Range (g's)	Y Acceleration Range (g's)	Z Acceleration Range (g's)
Sensor 1	0.003	0.015	0.007
Sensor 2	0.005	0.021	0.014
Acceleration Data	0.069	0.075	0.039

Parameter Sensitivity Study

A sensitivity study was conducted to determine how dependent the system is on its parameters by examining how the system reacts when parameters are changed. The parameters that will be studied are the spring constant and damping coefficient of the shock absorbers and the spring constant of the chains.

The goal of these studies is to determine which parameter values result in the least amount CG movement. This is accomplished by subtracting the maximum from the minimum CG value to compare the range of values in each direction. In the case of the z direction, only values after equilibrium are considered so that the initial settling of the platform to its equilibrium state does not affect results. For each parameter tested, only that parameter will be incrementally modified and all other parameters will be set to their default values, shown in the "parameter.m" code in Appendix A. All tests will be done using the previously described earthquake displacement forcing function to simulate the system responding to a real event.

Shock Absorber Spring Constant and Damping Coefficient Study

The original shock absorber spring constant and damping coefficient values selected for the model were 700 $\frac{\text{lb}}{\text{ft}}$ and 150 $\frac{\text{lb}\cdot\text{s}}{\text{ft}}$ respectively to obtain the desired system

characteristics. For this sensitivity study, spring constant values from 300 – 1,500 $\frac{lb}{ft}$ were used with increments every 100 $\frac{lb}{ft}$. Damping coefficient values used were from 10 – 250 $\frac{lb \cdot s}{ft}$ with increments every 20 $\frac{lb \cdot s}{ft}$. By analyzing the results shown in Fig. 38, it can be observed that system performance is more strongly impacted by variance in the damping coefficient versus the spring constant.

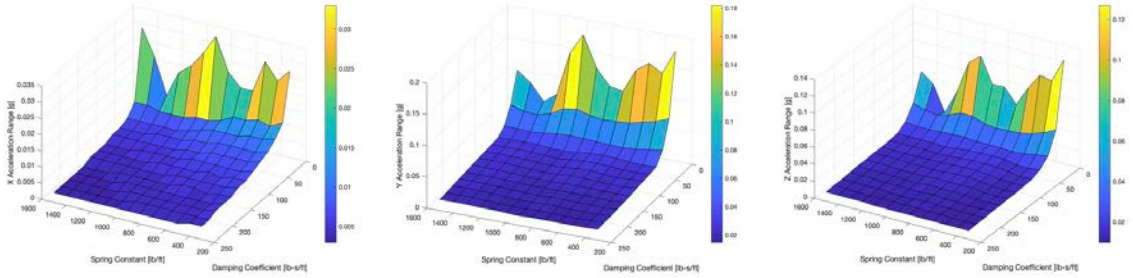


Figure 38. System Performance for Varying Spring Constant and Damping Coefficient

In order to meet the goal to minimize accelerations within the system, ideal values for the spring constant and damping coefficient can be selected as 600 $\frac{lb}{ft}$ and 125 $\frac{lb \cdot s}{ft}$ or greater, respectively. These results also verify that the original selected values of 700 $\frac{lb}{ft}$ and 150 $\frac{lb \cdot s}{ft}$ were appropriate for analysis.

Chain Spring Constant Study

The original chain spring constant value selected for the model was 20,000 $\frac{lb}{ft}$ to obtain the desired system characteristics. This parameter was selected for study to determine how its value affects the model simulation, not because its value affects the physical system. Chain spring constant values from 2,000 – 200,000 $\frac{lb}{ft}$ were used with increments every 2,000 $\frac{lb}{ft}$ in the 10,000 – 200,000 $\frac{lb}{ft}$ range and the value for the x, y, and

z range compared, as depicted in the results in Fig. 39. Through this study, it can be observed that the chain spring constant does play a role in how the system responds, but the response flattens out as the value increases. Therefore, any value of about 20,000 $\frac{\text{lb}}{\text{ft}}$ or more will be sufficient.

Although the displacement range flattens out, a high chain spring constant adds instability to the model. Figure 40 compares a chain spring constant of 2,000 $\frac{\text{lb}}{\text{ft}}$ with 200,000 $\frac{\text{lb}}{\text{ft}}$ for the scenario of no displacement in the model's equilibrium test case. It can be observed Fig. 40a that using a spring constant of 2,000 $\frac{\text{lb}}{\text{ft}}$ exhibits a higher amplitude response than the spring constant of 200,000 $\frac{\text{lb}}{\text{ft}}$, but also has a smooth response instead of the over reacting oscillation response observed in Fig. 40b.

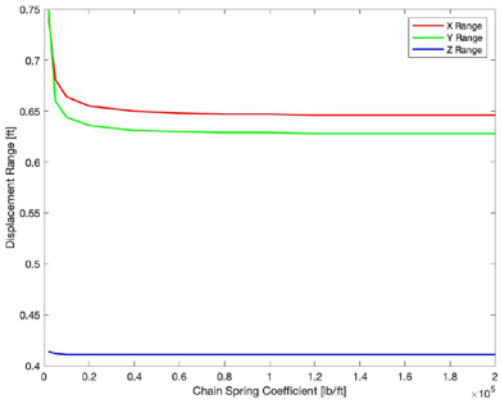
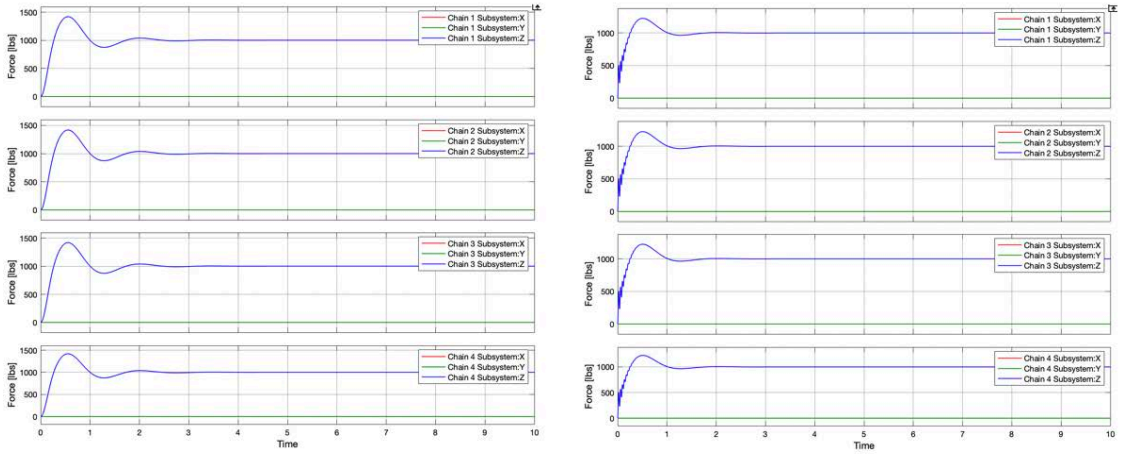


Figure 39. Chain Spring Constant Sensitivity Study Results



(a) 2,000 lb/ft (b) 200,000 lb/ft

Figure 40. Chain Instability Comparison

With these factors in mind, a chain spring constant value of 20,000 $\frac{\text{lb}}{\text{ft}}$ gives an appropriate amount of chain stiffness to the system, without introducing instabilities in the form of high frequency oscillations within the chain force, which is observed with higher chain spring constant values.

Failure Case Study

One of the biggest benefits of creating a design tool, such as this model, is having the ability to simulate failure scenarios before they take place and have the ability to see how the system will respond. The failure case that this section will be analyzing is if there was a failure in one shock absorber. Specifically, the failure mechanism will be if an orifice through which fluid passes through were to get plugged. This would be equivalent to if the spring constant was 10x higher in only one of the four shock absorbers. This scenario results are shown in Fig. 41 and Fig. 42, with the plugged shock absorber at the first platform attachment location. This was found to affect the system's ability to reach equilibrium, so the earthquake delay was changed from 4 to 10 seconds to

allow for better comparison with the previous results. The results from this scenario are tabulated in Table 2 and 3.

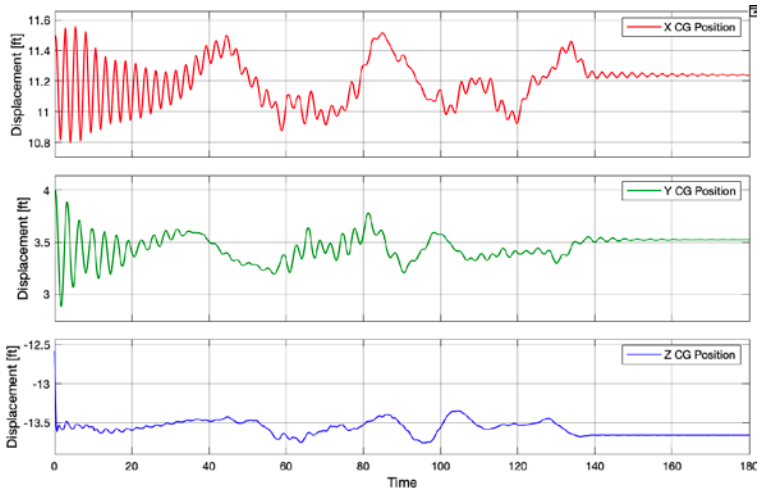
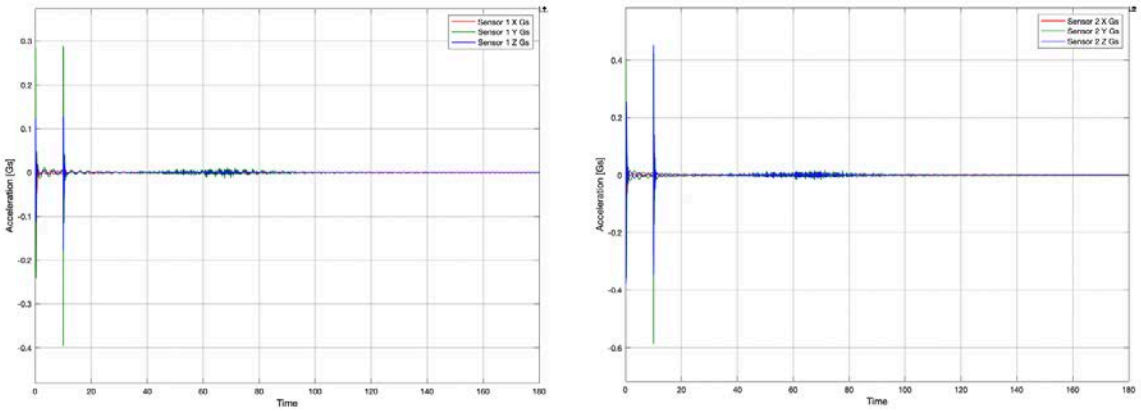


Figure 41. CG Failure Case Study Results



(a) Sensor 1 (b) Sensor 2

Figure 42. Sensor Failure Case Study Results

Table 2. Failure Case Study CG Results

Failure Case	X Range (ft)	Y Range (ft)	Z Range (ft)
Platform CG	0.762	1.120	0.403
Percent Increase from Base Case	X Range (%)	Y Range (%)	Z Range (%)
Platform CG	114	175	-1.95

Table 3. Failure Case Study Sensor Results

Failure Case	X Acceleration Range (Gs)	Y Acceleration Range (Gs)	Z Acceleration Range (Gs)
Sensor 1	0.012	0.030	0.013
Sensor 2	0.015	0.042	0.030
Percent Increase from Base Case	X Acceleration Range (%)	Y Acceleration Range (%)	Z Acceleration Range (%)
Sensor 1	400	200	186
Sensor 2	300	200	214

By reviewing the resulting figures and tables from this failure case, it can be observed that a failure of one shock absorber greatly affects the performance of the system. It can be concluded from the results of this study, that maintenance of this system is of great importance to ensure that shock absorbers are working properly. Otherwise, system response to a sudden displacement event, such as an earthquake, will increase accelerations felt within the platform subsystem by 186-400%, depending on location and direction.

CHAPTER 5: APPLICATION OF MODEL

The beauty of this dynamic model implementation is that it can be applied in countless ways. By modularizing the subsystems, they can be configured in any way the user desires. The platform can also have a CG at any location, depending on the distribution of components and personnel within the subsystem. This design tool can then be used to prioritize maintenance of an existing system or to develop a future system.

Future Model Development

There are modifications which could be made to the model to further develop the accuracy and physical representation of the system. The first modification could be the development of a more complex shock absorber model. As shown in the failure case study, shock absorber performance plays a big role in the system response to a displacement event. In reality, shock absorbers are generally more complex than the simple model represented in this work. A more complex model could represent a shock absorber with pneumatic and fluid damping components and include things such as spring and gas non-linearities. It should be noted that adding computational complexity of this type will result in the model taking more time to complete the simulation. However, running the simulation with the 2007 Sumatra, Indonesia earthquake displacement data currently takes ~23 - 48 seconds. If better accuracy is what is desired, then a longer simulation time will not be a deterrent.

Another development that could be added to increase the physical representation of the system would be to include the physics of when the platform subsystem runs into

the physical constraints of the capsule. Using the 2007 Sumatra, Indonesia earthquake displacement data, the platform subsystem only moves +/- 3.6 inches in the x and y directions, but it can be observed that during a failure case or in a stronger displacement event the effects from running into the capsule walls could come into play. However, modeling the physics as a switch, as discussed in the chain model formulation section, can result in model discontinuities. Therefore, if the user would like to incorporate these physics, it would be best done through the use of a “snubber”, where once the platform gets near the capsule wall, a strong spring force begins to push back in the opposite direction.

Future Study

A lot of work could be put into studies using this model in any desired configuration. In particular, many studies could be done into failure modes of the system. This could be done through the manipulation of the system parameters as well as through the use of many other displacement forcing functions. By using a stronger displacement event, such as a nuclear blast, the accelerations and survivability could be analyzed within the system in a much more extreme situation. Utilizing the design tool in this way will allow the user to determine how the system may fail, in any desired loading or configuration, which can be used to prioritize maintenance of an existing system or in the development of a future system.

CHAPTER 6: CONCLUSIONS

A dynamic model representing a system consisting of chains, shock absorbers, and a platform subsystem, can be derived and implemented in MATLAB's Simulink [17]. This model can be utilized as a design tool to represent any similar system configuration under any displacement forcing function by modularizing the subsystems and focusing on a straight forward user interface. In the presented model, system parameters need only to be updated in a single MATLAB script, which are immediately updated and used throughout the system.

Two different methods were used to represent the orientation of the platform subsystem in three dimensional space and were used for results comparison. These two methods are the Euler angle method and the momentum method. There are pros and cons to both methods, but the momentum method was determined to be most ideal. There are two cons associated with using the Euler angle method: Gimbal lock and the complexity of implementation. Gimbal lock [18] is a phenomenon which is known to occur in 90° rotation situations. Because it is not anticipated that the platform subsystem will be in a 90° rotation about any axis, this is not a pressing concern, but is important to be aware of. The major concern with the momentum method is the numerical drift [10] that can take place from obtaining the rotation transformation matrix through the integration of the $\dot{R}(t)$ matrix at each time step. However, because the simulated events being analyzed occur in such a short time span, the numerical drift was also not a pressing concern. The numerical drift can also be quantified by multiplying the final rotation transformation

matrix by its transform. This calculation should result in the identity matrix and any variance from that result would indicate the amount of drift which has taken place over the time of the simulation. Overall, the momentum method was determined to be most ideal because the simplicity of its implementation allowed for easier debugging of the system as a whole.

This simulation used the 8.4 magnitude displacement data from the 2007 Sumatra, Indonesia earthquake as the model's forcing function. Minimal accelerations were felt by the sensors within the system. Accelerations from the earthquake were reduced by 64 - 92% depending on direction. This significant reduction is due to the damping of the shock absorbers and by energy being dissipated through the pendulum effect created by the chains. This results in a safe and very survivable event for the personnel and equipment within the platform subsystem.

The dependence of the system on its parameters was analyzed in a parameter sensitivity study. This study found that the shock absorber damping coefficient has a much greater impact on the system's performance when compared with the shock absorber spring constant. It was also shown through the failure case study that each shock absorber needs to be working as designed. Variance between the shock absorbers was shown to increase the accelerations felt within the system by as much as 400%.

Further development and study can be accomplished utilizing this design tool to allow the user to prioritize maintenance of an existing system or to develop a future system of any similar system configuration in any displacement forcing function scenario.

REFERENCES

[1] Owen, G. N., and Scholl, R. E., 1981, *Earthquake Engineering of Large Underground Structures*, U.S. [Gov.] P[rint.] O[ff., Washington, D.C.

[2] Hashash, Y. M. A., Hook, J. J., Schmidt, B., and I-Chiang Yao, J., 2001, "Seismic Design and Analysis of Underground Structures," Tunnelling and Underground Space Technology, **16**(4), pp. 247–293.

[3] Kawamata, Y., Nakayama, M., Towhata, I., and Yasuda, S., 2016, "Dynamic Behaviors of Underground Structures in E-Defense Shaking Experiments," Soil Dynamics and Earthquake Engineering, **82**, pp. 24–39.

[4] Xu, H., Li, T., Xia, L., Zhao, J. X., and Wang, D., 2016, "Shaking Table Tests on Seismic Measures of a Model Mountain Tunnel," Tunnelling and Underground Space Technology, **60**, pp. 197–209.

[5] Chen, J., Shi, X., and Li, J., 2010, "Shaking Table Test of Utility Tunnel under Non-Uniform Earthquake Wave Excitation," Soil Dynamics and Earthquake Engineering, **30**(11), pp. 1400–1416.

[6] Roh, H., and Reinhorn, A. M., 2010, "Modeling and Seismic Response of Structures with Concrete Rocking Columns and Viscous Dampers," Engineering Structures, **32**(8), pp. 2096–2107.

[7] Saiful Islam, A. B. M., Jameel, M., and Jumaat, M. Z., 2011, "Seismic Isolation in Buildings to Be a Practical Reality: Behavior of Structure and Installation Technique," Journal of Engineering and Technology Research, **Vol. 3**(4), pp. 99–117.

[8] Chen, Z. Y., and Shen, H., 2014, "Dynamic Centrifuge Tests on Isolation Mechanism of Tunnels Subjected to Seismic Shaking," Tunnelling and Underground Space Technology, **42**, pp. 67–77.

[9] Zhou, H., and Ma, G., 2012, “Double-Layer Floor to Mitigate in-Structure Shock of Underground Structures: A Conceptual Design,” Engineering Structures, **35**, pp. 314–321.

[10] Baraff, D., 1997, “An Introduction to Physically Based Modeling: Rigid Body Simulation I - Unconstrained Rigid Body Dynamics,” p. 32.

[11] Ardakani, H. A., and Bridges, T. J., 2010, “Review of the 3-2-1 Euler Angles: A Yaw–Pitch–Roll Sequence,” p. 9.

[12] Inman, D. J., 2014, *Engineering Vibration*.

[13] Hibbeler, R. C., 2013, *Engineering Mechanics - Dynamics*.

[14] Lee, T., Leok, M., and McClamroch, N. H., 2009, “Dynamics of a 3D Elastic String Pendulum,” *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, IEEE, Shanghai, China, pp. 3347–3352.

[15] Shedlock, K. M., and Pakiser, L. C., 1994, *Earthquakes - General Interest Publication*.

[16] Shearer, P. M., 1999, *Introduction to Seismology*, Cambridge University Press, Cambridge; New York.

[17] *MATLAB and Simulink R2019a*, The MathWorks Inc., Natick, Massachusetts, United States.

[18] Slabaugh, G. G., “Computing Euler Angles from a Rotation Matrix,” p. 7.

[19] Budynas, R. G., and Nisbett, J. K., 2015, *Shigley’s Mechanical Engineering Design*, McGraw-Hill Education, New York.

[20] “COSMOS Virtual Data Center” [Online]. Available: https://strongmotioncenter.org/vdc/scripts/default.plx. [Accessed: 18-Apr-2019].

[21] 2001, “SMC-Format Data Files,” U.S. Geological Survey [Online]. Available: https://escweb.wr.usgs.gov/nsmp-data/smcfmt.html. [Accessed: 18-Apr-2019].

[22] Aster, R., 2011, “The Seismic Wave Equation,” Seismological Society of America, p. 18.

APPENDIX A

parameters.m Code

```
% User Parameters for Suspended and Shock Isolated System
% Author:   Emilie Murphy

% General Parameters:
g = 32.2; % gravity [ft/s^2]

% Platform Parameters:
weightPlatform = 4000;  % weight of platform [lbs]
massPlatform = weightPlatform/g; % mass of platform [slug]
leng = 25;              % legth of platform [ft]
width = 10;             % width of platform [ft]
height = 1;             % height of platform [ft]
CG = [leng/2, width/2, height/2]; % center of gravity of platform relative to platform coordinate system [ft]
inertiaPlatform = (massPlatform/12).*[(width^2 + height^2) 0 0; 0 (height^2 + leng^2) 0; 0 0 (width^2 + leng^2)]; % mass moment of inertia of platform about CG [lbm*ft^2]
attachPoint1 = [1 1 height]; % x y z coordinates of attach point #1 relative to platform coordinate system [ft]
attachPoint2 = [(leng-1) 1 height]; % x y z coordinates of attach point #2 relative to platform coordinate system [ft]
attachPoint3 = [1 (width-1) height]; % x y z coordinates of attach point #3 relative to platform coordinate system [ft]
attachPoint4 = [(leng-1) (width-1) height]; % x y z coordinates of attach point #4 relative to platform coordinate system [ft]
% coordiantes are assuming "platform coordinate system" where origin is in front left corner of platform

% Sensor Parameters:
sensor1 = [leng/2 3 5]; % location point where sensor is located in platform coordinate system [ft]
sensor2 = [(2/3)*leng 9 7]; % location point where sensor is located in platform coordinate system [ft]

% Cable Parameters:
cable1l0 = 5;               % initial length of cable 1 [ft]
cable2l0 = 5;               % initial length of cable 2 [ft]
cable3l0 = 5;               % initial length of cable 3 [ft]
cable4l0 = 5;               % initial length of cable 4 [ft]
cableConstant = 20000;        % spring constant of cable [lbs/ft]

% Shock Absorber Parameters:
weightPiston = 240.33436;       % weight of piston within shock [lbs]
heightCylinder = 7;             % external height of the shock cylinder [ft]
heightVolume = 6.83;            % internal height of volume of shock cylinder [ft]
rodLength = 6.75;               % length of piston rod [ft]
pistonHeadThickness = 3/12;     % thickness of piston head [ft]
shockShellThickness = 1/12;     % thickness of shock absorber casing [ft]
k = 700;                        % spring constant [lbs/ft]
c = 150;                        % damping coefficient [lb-s/ft]
```

initialization.m Code

```
% Initialization for Suspended and Shock Isolated System
% Author: Emilie Murphy

% System Parameters:
massPiston = weightPiston/g;    % mass of piston within shock [slug]

% Position Vectors Between Platform Attach Points and Platform Attach Point 1
r2 = attachPoint2' - attachPoint1'; % [rX rY rZ] [ft]
r3 = attachPoint3' - attachPoint1'; % [rX rY rZ] [ft]
r4 = attachPoint4' - attachPoint1'; % [rX rY rZ] [ft]
% Note: these are the same for world attach points
% Note: position vectors may not all be used here, but may be used in the
% simulation

% Position Vectors from Platform CG to Platform Attach Points
R1 = attachPoint1' - CG'; % [rX rY rZ] [ft]
R2 = attachPoint2' - CG'; % [rX rY rZ] [ft]
R3 = attachPoint3' - CG'; % [rX rY rZ] [ft]
R4 = attachPoint4' - CG'; % [rX rY rZ] [ft]
% Note: position vectors may not all be used here, but may be used in the
% simulation

% Sensor Position Vectors:
rS1 = sensor1' - CG'; % position vector from platform CG to sensor location [rX rY rZ] [ft]
rS2 = sensor2' - CG'; % position vector from platform CG to sensor location [rX rY rZ] [ft]

% Inertial coordinates for the bottom of each chain
chain1Point = [0 0 -cable1l0]';
chain2Point = r2 + [0 0 -cable2l0]';
chain3Point = r3 + [0 0 -cable3l0]';
% chain4Point = r4 + [0 0 -cable4l0]';

% Calculating any Initial Rotation due to Chain Inequalities:
normal = cross((chain2Point - chain1Point),(chain3Point - chain1Point));
xRotationInitial = sin((normal(2)*-1)/normal(3)); % rotation about the x axis [rad]
yRotationInitial = sin((normal(1)*-1)/normal(3)); % rotation about the y axis [rad]
clear normal chain2Point chain3Point chain4Point

% Determining Initial Platform Rotation Transformation Matrix:
R = rotationMatrix([xRotationInitial,yRotationInitial,0]);

% Calculating Body Fixed Initial Rotations:
bodyRotation = R' * [xRotationInitial yRotationInitial 0]';
xRotationInitialBody = bodyRotation(1);
yRotationInitialBody = bodyRotation(2);
zRotationInitialBody = bodyRotation(3);
clear bodyRotation

% Determining Initial Inertial Platform CG Coordinates
initialZ1 = shockShellThickness + pistonHeadThickness + rodLength; % [body fixed]
pistonDistanceAboveShock = initialZ1 - heightCylinder; % [body fixed]
initialCableToPlatAttach1 = [0 0 (- pistonDistanceAboveShock - heightCylinder)]'; % [body fixed]
initialCG = chain1Point + (R * (initialCableToPlatAttach1 - R1)); % original location of platform CG in inertial frame [x y z]'
initialXCG = initialCG(1);
initialYCG = initialCG(2);
initialZCG = initialCG(3);

clear chain1Point R plat1ToCG initialZ1 pistonDistanceAboveShock initialCableToPlatAttach1 initialCG
```

Euler Angle Platform Model Body Fixed Platform Calculator Function Code

```
function [phiDoubleDotB,thetaDoubleDotB,psiDoubleDotB,M] = fcn(F1,F2,F3,F4,z1,z2,z3,z4,inertiaPlatform,parameters,omega)

% Body Fixed Platform Calculator:
% Calculate the angular acceleration of the platform in the body fixed
% frame, given the forces and moments acting on the body.
%
% Input:
%   F1 = force vector coming from attachment point 1 = [F1x F1y F1z]' [lbs] in body fixed frame
%   F2 = force vector coming from attachment point 2 = [F2x F2y F2z]' [lbs] in body fixed frame
%   F3 = force vector coming from attachment point 3 = [F3x F3y F3z]' [lbs] in body fixed frame
%   F4 = force vector coming from attachment point 4 = [F4x F4y F4z]' [lbs] in body fixed frame
%   z1 = distance from the bottom of the shock absorber to the current shock absorber piston location [ft]
%   z2 = distance from the bottom of the shock absorber to the current shock absorber piston location [ft]
%   z3 = distance from the bottom of the shock absorber to the current shock absorber piston location [ft]
%   z4 = distance from the bottom of the shock absorber to the current shock absorber piston location [ft]
%   inertiaPlatform = inertia of the platform [lbm*ft^2] [3x3 matrix]
%   parameters = vector of system parameters passed from model workspace
%   omega = angular velocity of the platform = [omegaX omegaY omegaZ]' [rad/s] in inertial reference frame
%
% Output:
%   phiDoubleDotB = angular acceleration about the x axis of CG in body fixed frame
%   thetaDoubleDotB = anguclar acceleration about the y axis of the CG in body fixed frame
%   psiDoubleDotB = angular acceleration about the z axis of the CG in body fixed frame

% Load in System Parameters:
Ic = inertiaPlatform; % platform inertial [lbm*ft^2] [3x3 matrix]
R1 = [parameters(1) parameters(2) parameters(3)]; % platform attach point 1 [x y z]
R2 = [parameters(4) parameters(5) parameters(6)]; % platform attach point 2 [x y z]
R3 = [parameters(7) parameters(8) parameters(9)]; % platform attach point 3 [x y z]
R4 = [parameters(10) parameters(11) parameters(12)]; % platform attach point 4 [x y z]

% Create Additional Vectors:
r1 = R1 + [0 0 z1]; % position vector from CG to top of shock piston 1
r2 = R2 + [0 0 z2]; % position vector from CG to top of shock piston 2
r3 = R3 + [0 0 z3]; % position vector from CG to top of shock piston 3
r4 = R4 + [0 0 z4]; % position vector from CG to top of shock piston 4

% Sum of the Moments:
M =  cross(r1',F1) + cross(r2',F2) + cross(r3',F3) + cross(r4',F4); % total moment [lbs-ft]
H = Ic * omega; % angular momentum [(slug-m^2)/s]
angularAcceleration = Ic \ (M - cross(omega,H));
phiDoubleDotB = angularAcceleration(1); % acceleration about x axis [rad/s^2]
thetaDoubleDotB = angularAcceleration(2); % acceleration about y axis [rad/s^2]
psiDoubleDotB = angularAcceleration(3); % acceleration about z axis [rad/s^2]
```

Euler Angle Platform Model Euler Angle Calculator Function Code

```
function [phiDoubleDot,thetaDoubleDot,psiDoubleDot] = fcn(alphaBody,eulerAngles)

% Euler Angle Calculator
% Convert angular acceleration from the body fixed reference frame to the
% inertial reference frame
%
% Input:
%   alphaBody: angular acceleration of the platform CG in the body fixed frame [rad/s^2]
%   eulerAngles: Euler Angles in the inertial reference frame [rad]
%
% Output:
%   phiDoubleDot: angular acceleration about the x axis in the inertial reference frame [rad/s^2]
%   thetaDoubleDot: angular acceleration about the y axis in the inertial reference frame [rad/s^2]
%   psiDoubleDot: angular acceleration about the z axis in the inertial reference frame [rad/s^2]

% Calculate Rotation Matrix
R = rotationMatrix(eulerAngles); % total transformation matrix

% Convert Body Fixed Acceleration to Inertial Acceleration
alphaInertial = R * alphaBody; % angular acceleration in the inertial reference frame [rad/s^2]
phiDoubleDot = alphaInertial(1); % angular acceleration about the x axis in the inertial reference frame [rad/s^2]
thetaDoubleDot = alphaInertial(2); % angular acceleration about the y axis in the inertial reference frame [rad/s^2]
psiDoubleDot = alphaInertial(3); % angular acceleration about the z axis in the inertial reference frame [rad/s^2]
```

rotationMatrix.m Code

```
function R = rotationMatrix(eulerAngles)
% Rotation Matrix Function
% Calculates the total rotation transformation matrix using the method
% derived in Ardakani, H. A., and Bridges, T. J., 2010, "Review of the
% 3-2-1 Euler Angles: A Yaw–Pitch–Roll Sequence"
%
% Input:
%   eulerAngles: rotation angles [phi theta phi] [rad]
%
% Output:
%   R: rotation transformation matrix [3x3]

% Get Angles:
phi = eulerAngles(1); % [rad]
theta = eulerAngles(2); % [rad]
psi = eulerAngles(3); % [rad]

% Create Transformation Matrices
Tphi = [1 0 0; 0 cos(phi) -sin(phi); 0 sin(phi) cos(phi)]; % rotation about the x axis
Ttheta = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)]; % rotation about the y axis
Tpsi = [cos(psi) -sin(psi) 0; sin(psi) cos(psi) 0; 0 0 1]; % rotation about the z axis
R = Tpsi * Ttheta * Tphi; % total transformation matrix

end
```

Euler Angle Platform Model Platform Acceleration Calculator Function Code

```
function [xDoubleDot,yDoubleDot,zDoubleDot,F,aInertial] = fcn(parameters,F1,F2,F3,F4,eulerAngles)

% Platform Acceleration Calculator
% Calculates the acceleration experienced by the platform CG.
%
% Input:
%   F1 = force vector coming from attachment point 1 = [F1x F1y F1z]' [lbs] in body fixed frame
%   F2 = force vector coming from attachment point 2 = [F2x F2y F2z]' [lbs] in body fixed frame
%   F3 = force vector coming from attachment point 3 = [F3x F3y F3z]' [lbs] in body fixed frame
%   F4 = force vector coming from attachment point 4 = [F4x F4y F4z]' [lbs] in body fixed frame
%   eulerAngles = euler angles of platform in inertial reference frame = [phi theta psi]' [rad]
%
% Output:
%   xDoubleDot = x acceleration of platform CG in inertial reference frame
%   yDoubleDot = y acceleration of platform CG in inertial reference frame
%   zDoubleDot = z acceleration of platform CG in inertial reference frame

% System Parameters:
mass = parameters(1); % mass of the platform [slugs]
weight = parameters(2); % weight of platform [lbf]

% Create Additional Vectors
F = F1 + F2 + F3 + F4; % total force vector

% Calculate Rotation Matrix
R = rotationMatrix(eulerAngles); % total transformation matrix

% Convert Body Fixed Acceleration to Inertial Acceleration
aInertial = (1/mass) * ((R * F) - [0 0 weight]'); % acceleration in the inertial frame
xDoubleDot = aInertial(1);
yDoubleDot = aInertial(2);
zDoubleDot = aInertial(3);
```

Euler Angle Platform Model Platform World Coordinates Function Code

```
function [shockAttach1,shockAttach2,shockAttach3,shockAttach4,platAttach1,platAttach2,platAttach3,platAttach4]  = fcn(inertialCG,eulerAngles,momentArms,zDistances)

% Platform World Coordinates
% Calculating the location of the 4 attach points in the inertial reference
% frame.
%
% Input:
%   CGDisplacement: the calculated displacement of the platform's CG [3x1] [ft]
%   eulerAngles: euler angles of platform in inertial reference frame = [phi theta psi]' [rad]
%   momentArms: position vectors from the platform CG to the respective platform attach points in the body fixed frame [ft]
%   zDistances: distance from the bottom of the shock absorber to the top of the shock absorber piston attach point [ft]
%
% Output:
%   shockAttach1: location of shock attach point 1 in inertial reference frame [x y z]' [ft]
%   shockAttach2: location of shock ttach point 2 in inertial reference frame [x y z]' [ft]
%   shockAttach3: location of shock attach point 3 in inertial reference frame [x y z]' [ft]
%   shockAttach4: location of shock attach point 4 in inertial reference frame [x y z]' [ft]
%   platAttach1: location of attach point 1 in inertial reference frame [x y z]' [ft]
%   platAttach2: location of attach point 2 in inertial reference frame [x y z]' [ft]
%   platAttach3: location of attach point 3 in inertial reference frame [x y z]' [ft]
%   platAttach4: location of attach point 4 in inertial reference frame [x y z]' [ft]

% Position Vectors from Platform CG to Platform Attach Points [Body Fixed] [ft]:
R1 = [momentArms(1) momentArms(2) momentArms(3)]';
R2 = [momentArms(4) momentArms(5) momentArms(6)]';
R3 = [momentArms(7) momentArms(8) momentArms(9)]';
R4 = [momentArms(10) momentArms(11) momentArms(12)]';

% Shock Absorber Z Distances [Body Fixed] [ft]:
z1 = zDistances(1);
z2 = zDistances(2);
z3 = zDistances(3);
z4 = zDistances(4);

% Calculate Rotation Transformation Matrix:
R = rotationMatrix(eulerAngles);

% Converting from Body Fixed to Inertial Reference Frame:
inertialR1 = R * R1;
inertialR2 = R * R2;
inertialR3 = R * R3;
inertialR4 = R * R4;
z1i = R * [0 0 z1]';
z2i = R * [0 0 z2]';
z3i = R * [0 0 z3]';
z4i = R * [0 0 z4]';

% Current Location of Attach Points (Inertial Frame):
platAttach1 = inertialCG + inertialR1; % current platform attach point 1 in inertial frame [x y z]'
platAttach2 = inertialCG + inertialR2; % current platform attach point 2 in inertial frame [x y z]'
platAttach3 = inertialCG + inertialR3; % current platform attach point 3 in inertial frame [x y z]'
platAttach4 = inertialCG + inertialR4; % current platform attach point 4 in inertial frame [x y z]'
shockAttach1 = platAttach1 + z1i; % current shock attach point 1 in inertial frame [x y z]'
shockAttach2 = platAttach2 + z2i; % current shock attach point 2 in inertial frame [x y z]'
shockAttach3 = platAttach3 + z3i; % current shock attach point 3 in inertial frame [x y z]'
shockAttach4 = platAttach4 + z4i; % current shock attach point 4 in inertial frame [x y z]'
```

Derivation of State Method Platform Model Total Force and Moment Function Code

```
function [totalForce,totalMoment] = fcn(F1,F2,F3,F4,zDistances,R,parameters,momentArms)

% Total Force and Moment
% Calculates the total force and moment being experienced by the platform.
%
% Input:
%   F1: the force vector coming from the platform attach point 1, already in the body fixed frame [Fx Fy Fz]' [lbs]
%   F2: the force vector coming from the platform attach point 2, already in the body fixed frame [Fx Fy Fz]' [lbs]
%   F3: the force vector coming from the platform attach point 3, already in the body fixed frame [Fx Fy Fz]' [lbs]
%   F4: the force vector coming from the platform attach point 4, already in the body fixed frame [Fx Fy Fz]' [lbs]
%   zDistances: z distance from each of the platform attach points to the top of the shock absorber piston [ft]
%   R: rotation transformation matrix [3x3]
%   parameters: vector of system parameters
%   momentArms: vector of the moment arms from the platform CG to the platform attach points in the body fixed frame [rX rY rZ] [ft]
%
% Output:
%   totalForce: the total calculated force vector [Fx Fy Fz]' [lbs]
%   totalMoment: the total calculated moment vector [Mx My Mz]' [ft-lbs]

% System Parameters:
platformWeight = parameters(1); % total platform weight [lbs]

% Position Vectors from Platform CG to Platform Attach Points [Body Fixed]:
R1 = [momentArms(1) momentArms(2) momentArms(3)]';
R2 = [momentArms(4) momentArms(5) momentArms(6)]';
R3 = [momentArms(7) momentArms(8) momentArms(9)]';
R4 = [momentArms(10) momentArms(11) momentArms(12)]';

% Z Position Vectors from Platform Attach Points to Top of Shock Absorber:
z1 = zDistances(1);
z2 = zDistances(2);
z3 = zDistances(3);
z4 = zDistances(4);

% Converting into Position Vectors in the Inertial Reference System:
z1i = R * [0 0 z1]';
z2i = R * [0 0 z2]';
z3i = R * [0 0 z3]';
z4i = R * [0 0 z4]';

% Converting Moment Arms from Body Fixed to Inertial Reference Frame:
R1i = (R * R1) + z1i;
R2i = (R * R2) + z2i;
R3i = (R * R3) + z3i;
R4i = (R * R4) + z4i;

% Converting Forces from Body Fixed to Inertial Reference Frame:
F1i = R * F1;
F2i = R * F2;
F3i = R * F3;
F4i = R * F4;

% Calculating Total Force:
inputForce = F1i + F2i + F3i + F4i; % total input force in the inertial frame [lbs]
totalForce = inputForce - [0 0 platformWeight]'; % total force acting on platform in inertial frame

% Calculating Total Moment:
totalMoment = cross(R1i,F1i) + cross(R2i,F2i) + cross(R3i,F3i) + cross(R4i,F4i); % total moment acting on the platform in inertial frame
end
```

Derivation of State Method Platform Model Angular Velocity Calculator Function Code

```
function omega = fcn(angularMomentum,R,inertiaPlatform)

% Angular Velocity Calculator
% Calculates the angular velocity of the platform.
%
% Input:
%   angularMomentum: the angular momentum of the platform [Lx Ly Lz]' [slug-ft^2/s]
%   R: rotation transformation matrix [3x3]
%   inertiaPlatform: inertia matrix for the platform [3x3] [lb-ft-s^2]
%
% Output:
%   omega: angular velocity of the platform in the inertial reference frame [omegaX omegaY omegaZ] [rad/s]

inertiaPlatformInertial = R * inertiaPlatform * R'; % inertia matrix for platform rotated into inertial reference frame

omega = inertiaPlatformInertial \ angularMomentum; % angular velocity of platform in inertial reference frame

end
```

Derivation of State Method Platform Model R Dot Calculator Function Code

```
function RdotVert = fcn(omega,R)

% R Dot Calculator
% Calculates the R Dot matrix, representing the how the rotation
% transformation matrix changes with time.
%
% Input:
%   omega: the angular velocity of the platform [omegaX omegaY omegaZ] [rad/s]
%   R: the rotation transformation matrix [3x3]
%
% Output:
%   RdotVert: the R Dot matrix, converted from a 3x3 matrix to a 9x1 matrix

% Preallocating:
RdotVert = zeros(9,1);

% Calculating R Dot:
omegaStar = [0 -omega(3) omega(2); omega(3) 0 -omega(1); -omega(2) omega(1) 0];
Rdot = omegaStar * R;

% Converting the R Dot Matrix into a 9x1 Matrix:
RdotVert(1) = Rdot(1,1);
RdotVert(2) = Rdot(1,2);
RdotVert(3) = Rdot(1,3);
RdotVert(4) = Rdot(2,1);
RdotVert(5) = Rdot(2,2);
RdotVert(6) = Rdot(2,3);
RdotVert(7) = Rdot(3,1);
RdotVert(8) = Rdot(3,2);
RdotVert(9) = Rdot(3,3);
```

Derivation of State Method Platform Model Rotation Matrix Concatenator

Function Code

```
function R = fcn(RVert)

% Rotation Matrix Concatenator
% Converts the rotation transformation matrix from a 9x1 into a 3x3 matrix.
%
% Input:
%   RVert: the rotation transformaiton matrix in 9x1 form
%
% Output:
%   R: the rotation transformation matrix in 3x3 form

% Preallocating:
R = zeros(3,3);

% Converting from a 9x1 to a 3x3:
R(1,1) = RVert(1);
R(1,2) = RVert(2);
R(1,3) = RVert(3);
R(2,1) = RVert(4);
R(2,2) = RVert(5);
R(2,3) = RVert(6);
R(3,1) = RVert(7);
R(3,2) = RVert(8);
R(3,3) = RVert(9);
```

Derivation of State Method Platform Model Shock Position Calculator Function Code

```
function [shock1Location,shock2Location,shock3Location,shock4Location,plat1Location,plat2Location,plat3Location,plat4Location] = fcn(CGPosition,R,zDistances,momentArms)

% Shock Position Calculator
% Calculates the current position of the top of the shock piston at each
% time step.
%
% Input:
%   CGPosition: current location of platform CG in inertial coordinates [x y z]' [ft]
%   R: rotation transformation matrix [3x3]
%   zDistances: z distance from platform attachment points to the top of the shock piston [ft]
%   momentArms: vector of the the moment arms from the platform CG to the platform attach points in the body fixed frame [rX rY rZ] [ft]
%
% Output:
%   shock1Location: location of the top of shock piston 1 in the inertial reference frame [x y z]' [ft]
%   shock2Location: location of the top of shock piston 2 in the inertial reference frame [x y z]' [ft]
%   shock3Location: location of the top of shock piston 3 in the inertial reference frame [x y z]' [ft]
%   shock4Location: location of the top of shock piston 4 in the inertial reference frame [x y z]' [ft]
%   plat1Location: location of platform attach point 1 in the inertial reference frame [x y z]' [ft]
%   plat2Location: location of platform attach point 2 in the inertial reference frame [x y z]' [ft]
%   plat3Location: location of platform attach point 3 in the inertial reference frame [x y z]' [ft]
%   plat4Location: location of platform attach point 4 in the inertial reference frame [x y z]' [ft]

% Position Vectors from Platform CG to Platform Attach Points [Body Fixed]:
R1 = [momentArms(1) momentArms(2) momentArms(3)]';
R2 = [momentArms(4) momentArms(5) momentArms(6)]';
R3 = [momentArms(7) momentArms(8) momentArms(9)]';
R4 = [momentArms(10) momentArms(11) momentArms(12)]';

% Z Position Vectors to Top of Shock Absorber:
z1 = zDistances(1);
z2 = zDistances(2);
z3 = zDistances(3);
z4 = zDistances(4);

% Converting from Body Fixed to Inertial Reference Frame:
R1i = R * R1;
R2i = R * R2;
R3i = R * R3;
R4i = R * R4;
z1i = R * [0 0 z1]';
z2i = R * [0 0 z2]';
z3i = R * [0 0 z3]';
z4i = R * [0 0 z4]';

% Calculating the Platform Attach Points in the Inertial Frame:
plat1Location = CGPosition + R1i;
plat2Location = CGPosition + R2i;
plat3Location = CGPosition + R3i;
plat4Location = CGPosition + R4i;

% Calculating the Shock Attach Points in the Inertial Frame:
shock1Location = plat1Location + z1i;
shock2Location = plat2Location + z2i;
shock3Location = plat3Location + z3i;
shock4Location = plat4Location + z4i;
end
```

Derivation of State Method Platform Model Euler Angle Calculator Function Code

```
function [phi1,theta1,psi1] = fcn(R)

% Euler Angle Calculator
% Calculates the platform euler angles of rotation, given the rotation
% transformation matrix.
%
% Input:
%   R: rotation transformation matrix [3x3]
%
% Output:
%   phi1: euler angle "phi"; rotation about the x axis [rad]
%   theta1: euler angle "theta"; rotation about the y axis [rad]
%   psi1: euler angle "psi"; rotation about the z axis [rad]

% NOTE:
% Method from "Computing Euler angles from a rotation matrix" by Gregory G. Slabaugh

theta1 = -asin(R(3,1));
phi1 = atan2(R(3,2)/cos(theta1),R(3,3)/cos(theta1));
psi1 = atan2(R(2,1)/cos(theta1),R(1,1)/cos(theta1));

% NOTE:
% A second set of euler angles can be calculated, but these were not found
% to be useful.
% theta2 = pi + theta1;
% phi2 = atan2(R(3,2)/cos(theta2),R(3,3)/cos(theta2));
% psi2 = atan2(R(2,1)/cos(theta2),R(1,1)/cos(theta2));

end
```

Shock Model Shock Force and Piston Acceleration Calculator Function Code

```
function [F1B,pistonAcceleration,springForce,dampingForce] = fcn(chainForce,R,parameters,pistonLocation,pistonDot)

% Shock Force and Piston Acceleration Calculator
% Calculates force transmitted from chains to the platform through the
% shock absorber. These forces are sent to the platform in the body fixed
% reference frame.
%
% Input:
%   chainForce: force transmitted from chains [Fx Fy Fz] [lbs]
%   R: rotation transformation matrix [3x3]
%   parameters: vector of system parameters
%   pistonLocation: distance in the z axis of the body fixed frame representing the location of the piston within the shock cylinder [ft]
%   pistonDot: velocity of piston within shock cylinder [ft/s]
%
% Output:
%   F1B: force transmitted from shock to platform in the body fixed frame [Fx Fy Fz] [lbs]
%   pistonAcceleration: accleration of the piston within the shock cylinder [ft/s^2]
%   springForce: calculated spring force within the shock cylinder [lbs]
%   dampingForce: calculated damping force within the shock cylinder [lbs]

[F1B,pistonAcceleration,springForce,dampingForce] = shock(chainForce,R,parameters,pistonLocation,pistonDot);
```

shock.m Code

```
function [FB,pistonAcceleration,springForce,dampingForce] = shock(chainForce,R,parameters,pistonLocation,pistonDot)

% Shock
% Calculates the force transmitted by the shock absorber.
%
% Input:
%   chainForce: force being transmitted from the chains [Fx Fy Fz] [lbs]
%   R: rotation transformation matrix [3x3]
%   parameters: vector of system parameters
%   pistonLocation: current location of the shock absorber piston in the body fixed frame [ft]
%   pistonDot: current velocity of the piston [ft/s]
%
% Output:
%   FB: force transmitted by the shock absorber in the body fixed frame [Fx Fy Fz] [lbs]
%   pistonAcceleration: acceleration of the piston [ft/s^2]
%   springForce: reaction force of the spring within the shock cylinder [lbs]
%   dampingForce: reaction damping force within the shock cylinder [lbs]

% System Parameters:
k = parameters(1);
c = parameters(2);
massPiston = parameters(3);
shockShellThickness = parameters(4);
pistonHeadThickness = parameters(5);

% Spring Constant Update as Spring Reaches Full Compression:
if(pistonLocation > 5.4)
    k = k * 3;
end

Fc = R' * chainForce;

Fcx = Fc(1);
Fcy = Fc(2);
Fcz = Fc(3);

% Forces Acting in the X Direction
Rx = Fcx; % X reaction force is equal and opposite to the chain x force [lbs]

% Forces Acting in the Y Direction
Ry = Fcy; % Y reaction force is equal and opposite to the chain y force [lbs]

% Forces Acting in the Z Direction
springForce = k * (pistonLocation - shockShellThickness - pistonHeadThickness);
dampingForce = c * pistonDot;
Rz = springForce + dampingForce;

% Total Transmitted Force
FB = [Rx Ry Rz]';

% Calculate Piston Acceleration
pistonAcceleration = (1/massPiston) * (Fcz - springForce - dampingForce);
end
```

Shock Model Z Distance Calculator Function Code

```
function zDistance = fcn(pistonLocation,parameters)

% Z Distance Calculator
% Calculates the distance in the z direction of the body fixed from the
% platform attach point to the top of the shock piston.
%
% Input:
%   pistonLocation: distance in the z axis of the body fixed frame representing the location of the piston within the shock cylinder [ft]
%   parameters: vector of system parameters
%
% Output:
%   zDistance: distance in the z axis of the body fixed frame from the platform attach point to the top of the shock piston [ft]

% System Parameters:
rodLength = parameters(1);

zDistance = pistonLocation + rodLength;
```

Chain Model Chain 1 Function Code

```
function [F1,cableLength1,chainVector1] = fcn(worldAttach1,shockAttach1,parameters)

% Chain 1
% Calculates the chain force generated by any world and/or platform
% movement.
%
% Input:
%   worldAttach1: coordinates of world attach point 1 [x y z]' (ft)
%   shockAttach1: coordinates of shock attach point 1 [x y z]' (ft)
%   parameters: vector containing system parameters for the unstretched cable length and cable spring constant
%
% Output:
%   F1: force output at shock attach point 1 [Fx Fy Fz]' [lbs]
%   cableLength1: current cable length calculated from the cable position vector [ft]
%   chainVector1: position vector from shock attach point to the world attach point for cable 1 [x y z] [ft]

% System Parameters:
l0 = parameters(1); % unstretched cable length [ft]
k = parameters(2); % cable spring constant [lbs/ft]

% Determining Current Cable Length/Position:
chainVector1 = worldAttach1 - shockAttach1; % position vector from to shockAttach1 to worldAttach1 [x y z] [ft]
cableLength1 = norm(chainVector1); % current cable length [ft]

% Calculates Transmitted Force:
F1 = chain(chainVector1,cableLength1,l0,k); % force output at shock attach point 1 [Fx Fy Fz]' [lbs]
end
```

chain.m Code

```
function force = chain(positionVector,cableLength,l0,k)

% Chain
% Carries out chain logic to determine the force output at the attach point.
%   This function uses the current chain length to determine whether or not
%   the cable is slack. The transmitted force is then calculated to be the
%   cable spring force if the cable is being stretched, or zero if the
%   cable is slack.
%
% Input:
%   positionVector: position vector from the world attach point to the shock attach point [x y z]' [ft]
%   cableLength: current length of cable [ft]
%   l0: unstretched length of cable [ft]
%   k: cable spring constant [lbs/ft]
%
% Output:
%   force: force output at shock attach point 1 due to world/platform displacement [Fx Fy Fz]' [lbs]

if (cableLength > l0)
    % Case: Cable is being stretched.
    deltaLength = cableLength - l0; % [ft]
    Fk = k * deltaLength; % total spring force [lbs]
    % Calculate Direction Cosines
    nX = positionVector(1)/cableLength; % x direction cosine
    nY = positionVector(2)/cableLength; % y direction cosine
    nZ = positionVector(3)/cableLength; % z direction cosine
    % Calculate Transmitted Force
    force = [nX*Fk nY*Fk nZ*Fk]'; % transmitted force from chains to shock attach point [lbs]
else
    % Case: Cable is slack.
    force = [0 0 0]'; % ransmitted force from chains to shock attach point [lbs]
end

end
```

EarthquakeDisplacement.m Code

```
% Earthquake Displacement Script
% Author: Emilie Murphy
%
% Suspended and Shock Isolated System in Seismic Loading

% User parameters:
filename = 'SumatraDisplacement.csv';
timeDelay = 4; % time delay in seconds
endTime = 50; % extra time at the end of the simulation
sampleRate = 200; % number of samples per second

% Displacement Data Load In:
displacementMetric = csvread(filename,1,0); % displacement [cm]
cmToInchConversion = 1/2.54; % conversion factor for cm to inches
displacement = displacementMetric * cmToInchConversion; % displacement [in]

% Data Manipulation
extraRows = sampleRate * timeDelay;
endExtraRows = sampleRate * endTime;
displacementSize = length(displacement);
xDisplacement = zeros(extraRows+displacementSize+endExtraRows,1);
yDisplacement = zeros(extraRows+displacementSize+endExtraRows,1);
zDisplacement = zeros(extraRows+displacementSize+endExtraRows,1);

% Creation of Displacement Array
xDisplacement(extraRows+1:extraRows+displacementSize) = displacement(:,1);
yDisplacement(extraRows+1:extraRows+displacementSize) = displacement(:,2);
zDisplacement(extraRows+1:extraRows+displacementSize) = displacement(:,3);
xDisplacement(extraRows+displacementSize+1:extraRows+displacementSize+endExtraRows) = displacement(end,1);
yDisplacement(extraRows+displacementSize+1:extraRows+displacementSize+endExtraRows) = displacement(end,2);
zDisplacement(extraRows+displacementSize+1:extraRows+displacementSize+endExtraRows) = displacement(end,3);
clear displacementMetric displacement cmToInchConversion

filename = 'SumatraAcceleration.csv';
% Acceleration Data Load In:
accelerationMetric = csvread(filename,1,0); % displacement [cm/s^2]
cmToFeetConversion = 0.0328084;
acceleration = accelerationMetric * cmToFeetConversion; % displacement [ft/s^2]

% Acceleration Data Array
xAcceleration = zeros(extraRows+displacementSize+endExtraRows,1);
yAcceleration = zeros(extraRows+displacementSize+endExtraRows,1);
zAcceleration = zeros(extraRows+displacementSize+endExtraRows,1);
xAcceleration(extraRows+1:extraRows+displacementSize) = acceleration(:,1);
yAcceleration(extraRows+1:extraRows+displacementSize) = acceleration(:,2);
zAcceleration(extraRows+1:extraRows+displacementSize) = acceleration(:,3);

clear filename acclerationMetric acceleration cmToInchConversion...
    extraRows timeDelay endExtraRows displacementSize
```

World Displacement Model Earthquake Displacement Data ReadIn Function Code

```
function [x,y,z] = fcn(xDisplacement,yDisplacement,zDisplacement,sampleRate,clock)

% Earthquake Displacement Data ReadIn
% This function takes in the x, y, and z earthquake displacement vectors,
% determines the index which the simulation is currently on, and outputs
% the current x, y, and z displacement values.
%
% Input:
%   xDisplacement: the x earthquake displacement data vector [in]
%   yDisplacement: the y earthquake displacement data vector [in]
%   zDisplacement: the z earthquake displacement data vector [in]
%   sampleRate: the rate at which data samples were collected [samples/s]
%   clock: current simulation time step [s]
%
% Output:
%   x: the current x displacement value [in]
%   y: the current y displacement value [in]
%   z: the current z displacement value [in]


currentIndex = fix((clock * sampleRate) + 1);

x = xDisplacement(currentIndex);
y = yDisplacement(currentIndex);
z = zDisplacement(currentIndex);

end
```

World Displacement Model World Attachment Location Calculator Function Code

```
function [worldAttach1,worldAttach2,worldAttach3,worldAttach4] = fcn(worldDisplacement,parameters)

% World Attachment Location Calculator
% Calculates the current location of each attach point at each time step,
% inculding any displacement which might be taking place.
%
% Input:
%   worldDisplacement: displacement of the world attach points [inches]; forcing function for simulation
%   parameters: position vectors from attach point 1 to each respective attach point [x y z]' [ft]
%
% Output:
%   worldAttach1: location of world attach point 1 in the inertial reference frame [x y z]' [ft]
%   worldAttach2: location of world attach point 2 in the inertial reference frame [x y z]' [ft]
%   worldAttach3: location of world attach point 3 in the inertial reference frame [x y z]' [ft]
%   worldAttach4: location of world attach point 4 in the inertial reference frame [x y z]' [ft]

% System Parameters:
r2 = [parameters(1) parameters(2) parameters(3)]';
r3 = [parameters(4) parameters(5) parameters(6)]';
r4 = [parameters(7) parameters(8) parameters(9)]';

% Convert Displacement to Feet
displacement = worldDisplacement ./ 12; % world displacement [ft]

% Original Locations of World Attach Points (Inertial Frame)
originalWorld1 = [0 0 0]'; % [ft]
originalWorld2 = r2 + originalWorld1; % [ft]
originalWorld3 = r3 + originalWorld1; % [ft]
originalWorld4 = r4 + originalWorld1; % [ft]

% Actual Location of World Attach Points (Inertial Frame)
worldAttach1 = originalWorld1 + displacement; % coordinates of attach point 1 [x y z]'
worldAttach2 = originalWorld2 + displacement; % coordinates of attach point 2 [x y z]'
worldAttach3 = originalWorld3 + displacement; % coordinates of attach point 3 [x y z]'
worldAttach4 = originalWorld4 + displacement; % coordinates of attach point 4 [x y z]'
```

Sensor Model Sensor Velocity and Acceleration Calculator Function Code

```
function [velocity,acceleration] = fcn(omegaCG,alphaCG,positionVector)

% Sensor Velocity and Acceleration Calculator
% Calculates the velocity and acceleration of any specific point in space
% rigidly attached to the platform.
%
% Input:
%   omegaCG: angular velocity of CG of platform [omegaX,omegaY,omegaZ] [rad/s]
%   alphaCG: angular acceleration of CG of platform [alphaX alphaY alphaZ] [rad/s^2]
%   positionVector: position vector from CG of platform to specific point in space [rX rY rZ] [ft]
%
% Output:
%   velocity: velocity of specific point in space [ft/s]
%   acceleration: acceleration of specific point in space [ft/s^2]

velocity = cross(omegaCG,positionVector); % velocity of point in space [ft/s]
acceleration = cross(alphaCG,positionVector) + cross(omegaCG,cross(omegaCG,positionVector)); % acceleration of point in space [ft/s^2]

end
```

Sensor Model Conversion to Gs Function Code

```
function Gs = fcn(acceleration,gravity)

% Conversion to G's
% Converts acceleration into G value or, acceleration per gravity unit
%
% Input:
%   acceleration: acceleration of specific point in space [ft/s^2]
%   gravity: gravitational constant [ft/s^2]
%
% Output:
%   Gs: acceleration of specific point in space [Gs]

Gs = acceleration ./ gravity;
```