THESIS

DYNAMIC MODEL OF A SPHERICAL ROBOT FROM FIRST PRINCIPLES

Submitted by

Gregory C. Schroll

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2010

COLORADO STATE UNIVERSITY

July 13, 2010

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY GREGORY C. SCHROLL ENTITLED DYNAMIC MODEL OF A SPHERICAL ROBOT FROM FIRST PRINCIPLES BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

_____
Thomas H. Bradley

_____
Peter M. Young

_____
Advisor: David G. Alciatore

_____
Department Head: Allan T. Kirkpatrick

ABSTRACT OF THESIS

DYNAMIC MODEL OF A SPHERICAL ROBOT FROM FIRST PRINCIPLES

A prototype of a pendulum driven spherical robot has been developed during previous research and shown to exhibit unique dynamic behavior. Starting from first principles, a mathematical model for this spherical robot rolling on flat ground was developed in order to determine if this unique behavior was inherent to spherical robots in general or simply peculiar to this prototype. The complete equations of motion were found using Lagrangian methods, and numerically integrated using computer tools. A 3D simulation program was written to animate the results of integrating the equations. The dynamics apparent in the simulations were found to closely match the observed dynamics of the physical prototype.

Gregory C. Schroll
Department of Mechanical Engineering
Colorado State University
Fort Collins, CO 80523
Summer 2010

ACKNOWLEDGEMENTS

I would like to thank my thesis committee members, Thomas Bradley and Peter Young, as well as my advisor, David Alciatore, for their support and encouragement during this research. Their help while working with me to get through roadblocks along the way has been vital to my success.

I would also like to thank my good friend, Jon Jalving, whom I met during my studies at Colorado State University. He has been a great person to bounce ideas off of, and I hope to be able to work with him more in the future.

I dedicate this thesis to my long-time friend, Georgia Hoyer, who inspires me on a daily basis with her unwavering strength of mind. Her constant support has helped me mature into the person I am today.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The term *spherical robot* is used to describe two very different types of robots. Commonly, a spherical robot is a robot arm that forms a spherical coordinate system with two rotary joints and one prismatic joint. The term *spherical robot* is also used to describe mobile, “ball-like” robots that move along the ground by rolling about their outer spherical shell. This second definition is the focus of this study and any reference to the term *spherical robot* refers to the mobile, “ball-like” variety.

Spherical robots are generally comprised of an outer spherical shell and an internal propulsion mechanism. The shell may be made of multiple parts but they all move and rotate together as a single body as the robot rolls. While the mechanical design of the internal propulsion mechanism can vary greatly, the primary means of locomotion is by shifting the center of mass of the sphere. The acceleration due to gravity acting on the center of mass generates a torque on the sphere causing it to roll. By actively shifting the center of mass inside, a spherical robot can be directed to travel in a controlled manner.

The spherical shape of this class of mobile robot offers several advantages over other forms of surface-based locomotion like wheels, tracks, or legs. The sphere is a

strong shape offering rigidity at all points on its surface. It can provide a very high level of robustness with no major points of weakness, whereas wheels, tracks, or legs can be damaged, potentially disabling a robot's ability to move. The shell can also be resilient and serve as a protective barrier between the outside environment and the equipment inside. This protection can be in the form of robustness to impacts or as a physical barrier to hazardous chemicals and environmental conditions. The spherical shell can be sealed for either floating or submersion in liquid and also prevent dust from damaging mechanical components inside.

A spherical robot is by nature non-invertible further limiting the risk of becoming disabled. Most other vehicle designs are vulnerable to tipping over or becoming stuck on terrain where their means of locomotion loses contact with the ground. Also, the large diameter of the rolling surface of a spherical robot compared to its footprint allows it to travel over rough surfaces with greater rolling efficiency than vehicles with several smaller wheels.

These advantages indicate that a spherical robot would be appropriate for many different mobile robotics applications such as surveillance, reconnaissance, hazardous environment assessment, search and rescue, as well as planetary exploration. While research in the field of spherical robots has explored a large number of different concepts, few have been very successful in practice. The failure of spherical robots to perform effectively is due to a few significant disadvantages.

An issue presented by spherical robots as a mobile sensor platform is the difficulty of sensor integration when the entire outer shell of the robot rotates. Some sensors such as ambient environment sensors or omnidirectional microphones are not

greatly hindered, but directional sensors such as radar or cameras require sophisticated mounting or active stabilization. One solution includes using a transparent shell through which a camera can "see." Another solution requires mounting sensors on gimbaled platforms on either end of the normal axis of rotation of the sphere. Both of these concepts, however, introduce further problems and complexity.

While sensor integration difficulties can be addressed through detailed engineering, spherical robots have a much more significant performance limitation, that until recently has been largely unaddressed. The center-of-mass shifting principle by which virtually all spherical robots operate is severely limited by the maximum drive torque that can be generated. This torque limitation translates into a limit on the maximum inclination and highest obstacle that can be traversed, and is independent of available motor torque. Without the ability to effectively navigate rough terrain, spherical robots have not played a significant role in mobile robot applications despite having some marked advantages.

To address the limited mobility of previous spherical robot designs, a new mechanism has been invented using control moment gyroscopes for storing angular momentum effectively providing a spherical robot with a controllable torque "boost." The paper, "Design of a Spherical Vehicle with Flywheel Momentum Storage for High Torque Capabilities", documents the development of the invention including basic theory describing the planar motion of a spherical robot and the successful demonstration of a working prototype [1]. As a continuation of this prior work, this paper presents a more thorough compilation of the prior art, a thorough description of my earlier work, detailed theory of planar motion of a spherical robot with and without control moment

gyroscopes, derivation of a general 3-dimensional model of a spherical robot on flat ground, verification and simulation results of said model, as well as a discussion of future work.

Chapter 2

Prior Art

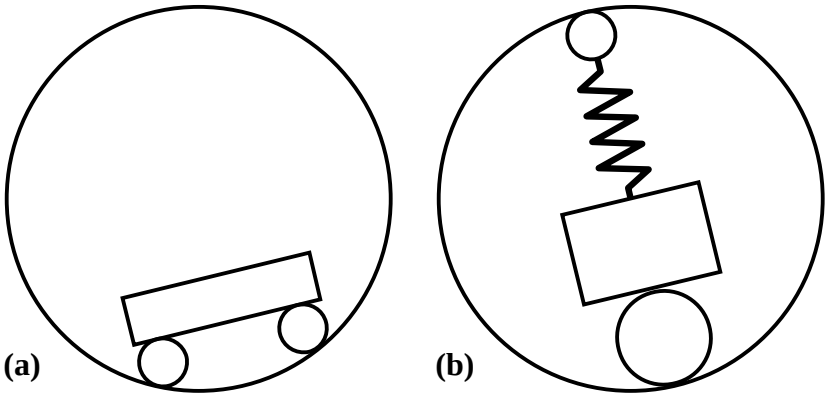
Self-propelled spheres have been in development for over 100 years. A thorough overview of the history of spherical vehicles is presented in Chapter 11 of the book, "Climbing & Walking Robots, Towards New Applications," [2]. The earliest spheres were spring driven toys in the late 1800s, followed by larger man carrying vehicles. More sophisticated mechanisms enabling steering and later electric power were developed. The current state of the art builds upon much of the prior century of work in mobile spheres. Modern spherical robots in development today generally incorporate sensors, actuators, and computer control, but propel themselves using a variety of different principles. These principles of locomotion tend to fall into the following four categories:

- Hamster Ball
- Pendulum Drive
- Multiple-Mass-Shifting
- Deformable Body

While wind-driven spheres are another significant class of spherical robot, they are propelled externally by wind rather than by an internal propulsion mechanism; Therefore, they are not of interest in this discussion.

2.1 Hamster Ball

A hamster ball style sphere is one with an inside driving unit (IDU) which transfers power directly to the inside surface of a hollow spherical shell. The IDU mechanism can take the form of an internal car or sprung central member as shown in Figure 2-1.



Internal Car Sprung Central Member

Figure 2-1: Hamster Ball Concept.

The internal car design most closely resembles a hamster inside a hamster ball. When the car starts to drive up the inside surface, the sphere will begin to roll forward. The car can have 4 wheels like a passenger vehicle, or 3 wheels with the added benefit of being perfectly constrained. Instead of wheels, the car can have treads like a tank. Since directional control of the sphere is maintained by driving the internal car in the desired direction, a variety of well known steering mechanisms can be used, such as front-wheel steering, 4-wheel steering, or differential steering.

The sprung central member design consists of a single driven wheel at the bottom, a tensioning element, and an idler wheel at the top. The tensioning element maintains contact between the driving wheel and the inside surface of the sphere. To steer, the driven wheel is rotated about its contact point with the sphere by applying a torque between the wheel and the inertia of the rest of the assembly.

The hamster ball concept has several advantages. Both the internal car and the sprung central member designs locate the majority of their mass close to the surface of the spherical shell, which helps maximize performance (see Section 2.5). They are relatively simple in design, and their drive mechanisms are straightforward to control. Because the final drive output is to small wheels compared to the diameter of the sphere, the required output torque is significantly less than with other designs discussed below.

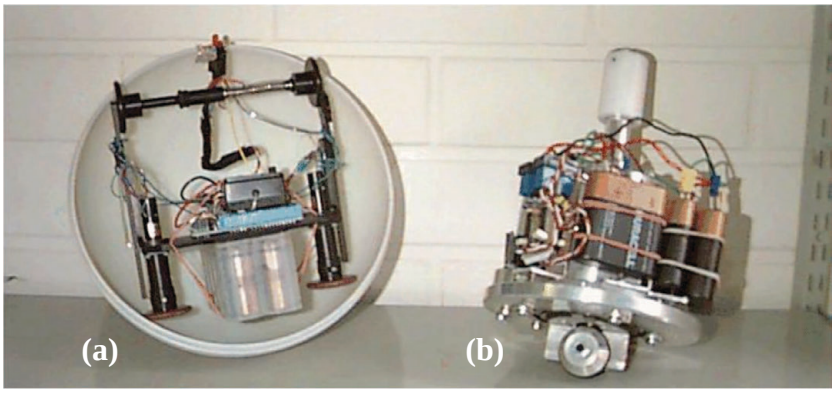
When using a sprung central member or an internal car with differential steering, the sphere is quasi-omnidirectional. The sphere can move in any direction starting from a standstill, but not instantaneously. The IDU must first be oriented to the desired direction before the sphere can move in that direction. It is also possible to build a fully omnidirectional hamster ball design using an internal car with omnidirectional wheels (see Section 2.1.2). In this case, the sphere can instantly begin moving in any direction since the internal car is omnidirectional.

One of the major disadvantages of the hamster ball concept is that the internal surface of the spherical shell is critical. The surface needs to be uniform in order for the drive wheels to operate properly. Since the IDU relies on friction between its wheels and the sphere, the wheels must maintain constant contact with the shell at all times. Impacts with obstacles can cause an internal car to lose contact and even flip over inside. A

sprung central member may require sophisticated suspension in order to maintain proper traction. The reliance on friction inside may also make it difficult to precisely monitor sphere rotation due to the risk of slippage between the IDU and the sphere.

2.1.1 Rollo Versions 1 and 2, Aalto University

The Rollo robot was developed at Aalto University (formerly Helsinki University of Technology) in Finland to explore the use of spherical robots as a mobile assistant [3]. The first Rollo prototype used the sprung central member design shown in Figure 2-1b. The second Rollo prototype used a unique design that was a combination of the internal car and sprung central member mechanisms (see Figure 2-2). These designs were rejected for the third prototype of Rollo, which utilized a pendulum mechanism (see section 2.2.4). A reason given for abandoning these hamster ball designs was the requirement for the spherical shell to be very rigid and uniform.



2nd Prototype 1st Prototype

Figure 2-2: Early Rollo Prototypes [3].

2.1.2 Spherical Autonomic Robot

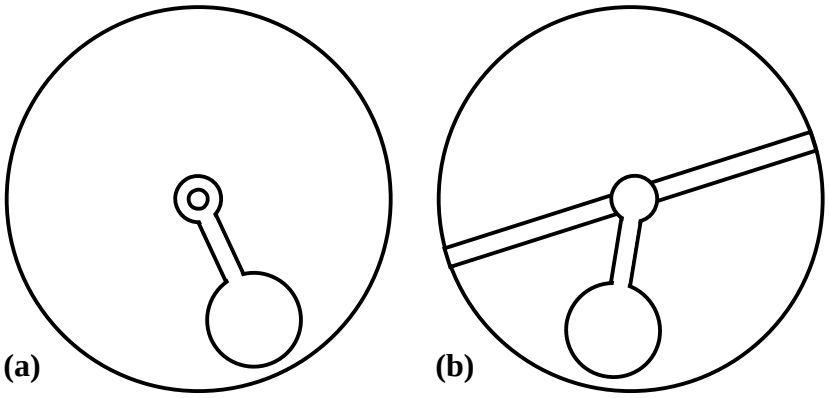
An Israeli spherical robot called the Spherical Autonomic Robot (SAR) has been developed using an internal car with 3 holonomic wheels (see Figure 2-3) [4]. The

triangular configuration of the holonomic wheels allows the internal car, as well as the sphere, to be fully omnidirectional. No additional information about the project could be found.

Figure 2-3: SAR with One Hemisphere Removed [4].

2.2 Pendulum Drive

A sphere with a pendulum drive mechanism typically has a main drive shaft fixed to the spherical shell and an offset mass (the pendulum) hanging from the drive shaft. A torque is applied between the pendulum and the drive shaft to propel the sphere forward.



Side view with pendulum rotated up for driving forward.

Front view with pendulum tilted to the side to steer. Drive shaft is tilted as a result.

Figure 2-4: Pendulum Drive Concept.

Figure 2-4a shows a side view of the pendulum drive with the pendulum tilted up in front to propel the sphere forward. Figure 2-4b shows a front view of the same mechanism. The pendulum can similarly tilt to the side, causing the main drive shaft to form an angle with the ground. When the pendulum is driven forward in this configuration, the sphere will travel in an arc. In order to locate the center of mass of the sphere as close to the outside surface as possible, the pendulum is usually not a separate weighted part, but in fact is comprised of the majority of the internal components of the sphere such as motors, batteries, and drive electronics.

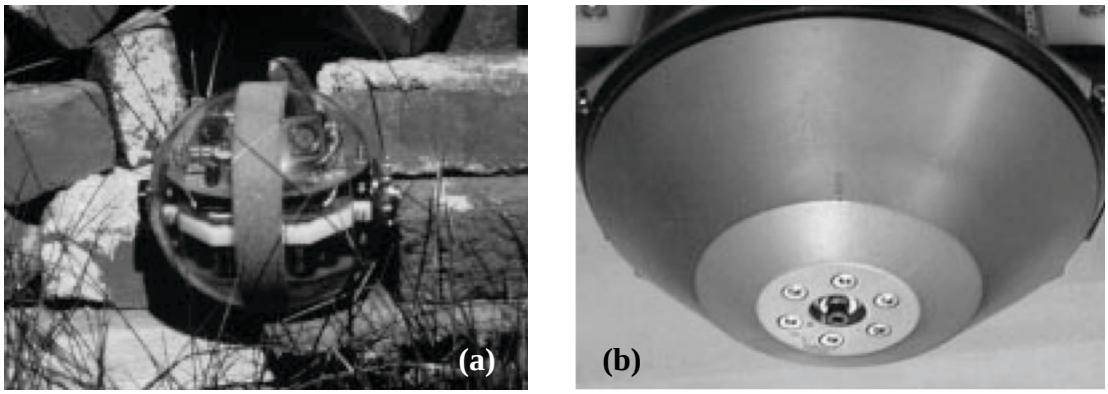
The control scheme of the pendulum mechanism is relatively straightforward and is analogous to a 4-wheeled vehicle with front wheel steering. Since the internal drive mechanism only interfaces with the spherical shell at two fixed points, the majority of the shell is dimensionally non-critical. Therefore, the shell can be flexible and act as a shock absorber. Unlike the friction connection inherent in the design of the hamster ball concept, the pendulum drive can have positive mechanical engagement allowing for accurate and precise measurement of the relative motion of the spherical shell and the pendulum.

A disadvantage of the pendulum design with respect to the other concepts is that the mechanical design tends to be challenging. The entirety of the internal structure hangs from a single shaft attached to the sphere in only two points, necessitating that these elements be quite strong in order to withstand impacts. The torque required at the main drive shaft inside is significantly higher than the torque required in the hamster ball designs. Another disadvantage is that the two points on the sphere where the main shaft attaches are very rigid and impacts at these points have the potential to be very jarring to

the mechanism inside. Also, the pendulum drive mechanism is not omnidirectional. Once stopped, a sphere with a pendulum drive must begin traveling in the same instantaneous trajectory before traveling in an arc to change direction.

2.2.1 Cyclops, Carnegie Mellon University

Cyclops, developed at Carnegie Mellon University, is a 5.5-inch diameter, 4.5-pound spherical robot intended for reconnaissance and surveillance in urban environments (see Figure 2-5a) [5]. It utilizes a pendulum mechanism for forward/backward movement, but has a unique inertial steering mechanism. The conical rotor shown in Figure 2-5b is a reaction wheel and is located at the lowest point on the pendulum. A torque motor reacts against the inertia of the reaction wheel to rotate the entire sphere in place about a nominally vertical axis.



Performing reconnaissance on location Reaction wheel for inertial steering

Figure 2-5: Cyclops robot [5].

2.2.2 Rotundus Groundbot

A Swedish company, Rotundus, has developed a product called Groundbot, which is the most deployment-ready spherical robot currently available [6]. Originally developed for space exploration, the Groundbot is now commercially available as a

sentry robot for patrolling and monitoring industrial locations. The robot utilizes a pendulum drive for both forward/backward motion as well as steering.

The unit is equipped with cameras mounted in clear domes on the sides of the robot and can be operated from a sophisticated telepresence command station. Groundbot can be controlled manually or programmed to patrol a route autonomously using GPS.

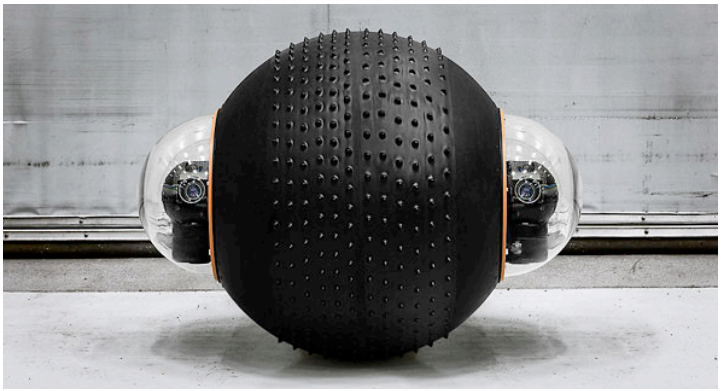
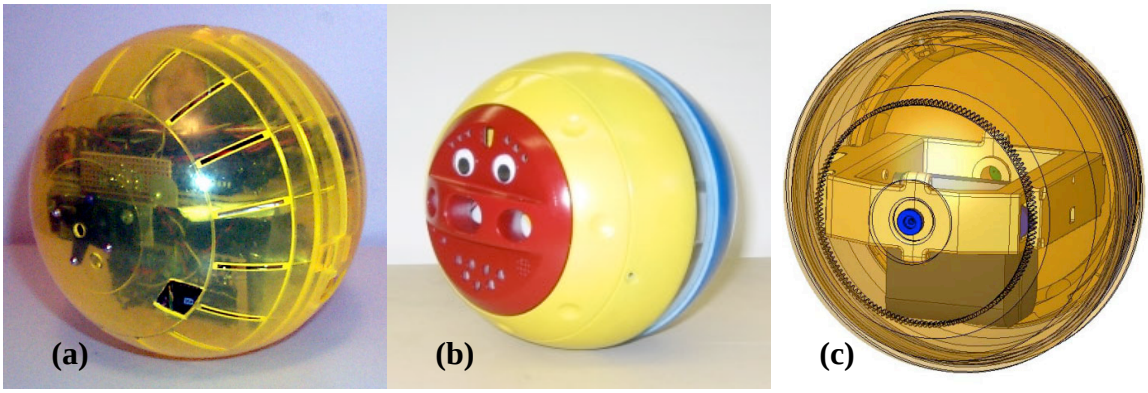


Figure 2-6: Rotundus Groundbot [6].

2.2.3 Roball, University of Sherbrook

Roball is a hamster-ball sized spherical robot developed at the University of Sherbrook, Canada [7]. Roball was designed to be a mobile, interactive toy for toddlers. The first prototype shown in Figure 2-7a demonstrated that the motion of the ball was inviting and engaging for children who played with it. A second, more advanced prototype (see Figure 2-7b) added lights and sounds which enhanced its interactivity.



1st prototype built inside a hamster ball.

2nd prototype with additional interactive features

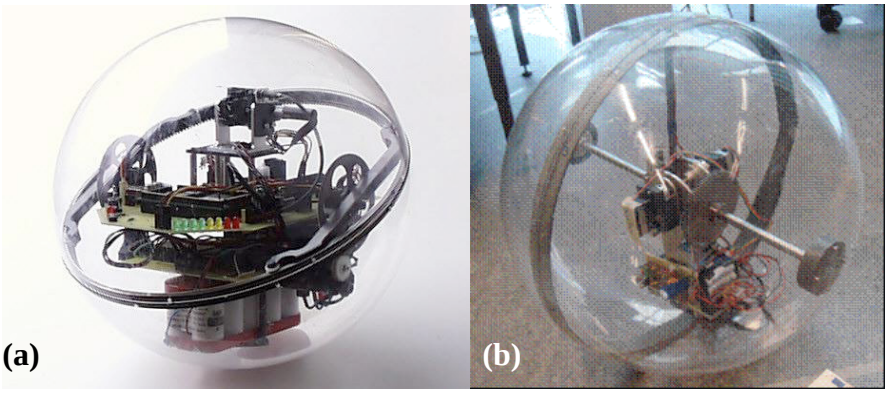
Internal CAD view showing ring gear

Figure 2-7: Roball [7].

All of the Roball prototypes made use of the pendulum mechanism for propulsion and steering. The CAD rendering shown in Figure 2-7c shows the use of a ring gear for driving the pendulum forward and backward. This mechanism is in some ways similar to the hamster ball method but has positive mechanical engagement instead of a friction interface. The ring gear design significantly lowers the output torque requirements of the internal mechanism, but also requires that the spherical outer shell be rigid.

2.2.4 Rollo Prototype 3 and GIMBall, Aalto University

While the 1st and 2nd prototypes of Rollo (see Section 2.1.1) made use of a hamster ball propulsion mechanism, prototype 3 implemented a unique pendulum mechanism. The pendulum provides the forward/backward motion of the sphere while a ring gear circumscribing the inside of the spherical shell provides the steering motion (see Figure 2-8a). When the ring gear face is parallel to the ground, the internal mechanism can reorient itself about the vertical axis making the sphere quasi-omnidirectional.



Rollo prototype 3 Gimball

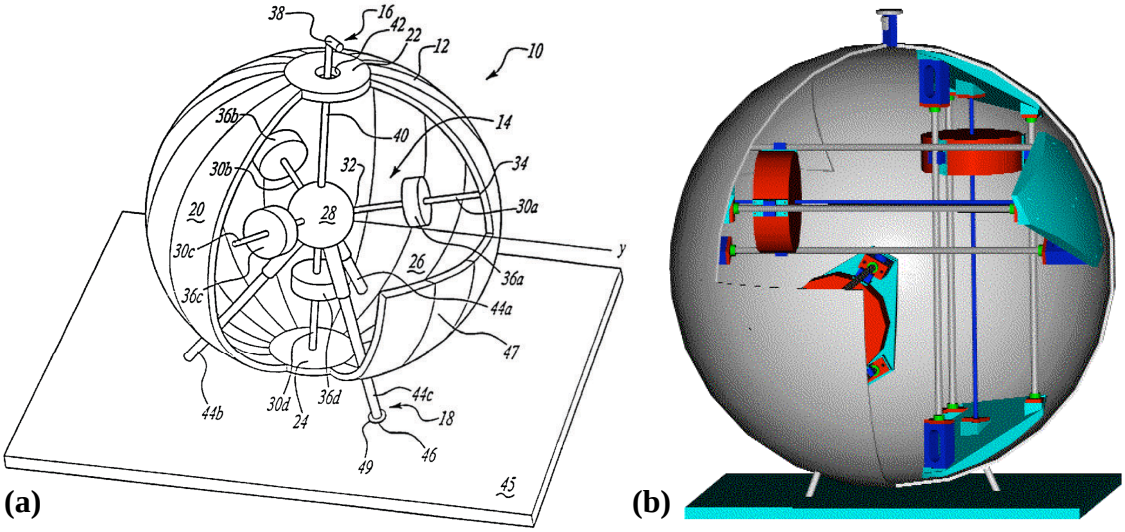
Figure 2-8: Aalto University prototypes [3],[8].

While the ring gear steering mechanism worked for the slow, methodical motion of the Rollo prototype, its use may be limited and disadvantageous for higher performance spherical robots. In order to steer in place, the mechanism requires there to be sufficient friction between the spherical shell and the ground such that the sphere remains stationary while the internal mechanism rotates. Also, the axis of the ring gear is always perpendicular to the rotating axis of the pendulum, such that when the sphere rolls, the direction of the ring gear axis constantly changes. Therefore, steering while rolling is very difficult if not impossible. The sphere must come to a stop with the ring gear face parallel to the ground before attempting to change direction.

GIMBall, shown in Figure 2-8b, is a more recent spherical robot also from Aalto University [8]. It was built to analyze and attempt to attenuate oscillatory motion inherent in spherical robot locomotion. The propulsion mechanism for GIMBall is a pendulum drive for both forward/backward motion and steering.

2.3 Multiple-Mass-Shifting

The multiple-mass-shifting concept incorporates 3 or 4 masses that can be moved independently along linear guides inside the sphere. By coordinating the motion of the masses, the location of the center of mass of the sphere can be controlled thereby enabling the sphere to move in a desired direction. Two configurations of the multiple-mass-shifting concept are shown in Figure 2-9.



Radial Design with retractable legs and camera.

Perpendicular non-intersecting design.

Figure 2-9: Multiple-Mass-Shifting [9],[11].

In the radial configuration of Figure 2-9a, 4 masses (36) move along radial spokes (30) [9]. The spokes are connected to a central hub (28) and their opposite ends are fixed to the spherical shell in an arrangement that forms a regular tetrahedron. Figure 2-9b shows a more optimized configuration where there are only 3 masses which move along non-intersecting rails [11]. The 3 rails are a set of perpendicular, non-intersecting edges of a cube as shown in Figure 2-10.

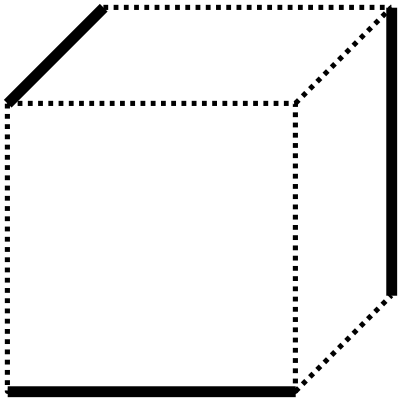


Figure 2-10: 3 perpendicular, non-intersecting edges of a cube.

The most significant feature of the multiple-mass-shifting concept is that it is fully omnidirectional. A sphere with this propulsion mechanism can instantaneously begin moving in any direction from a standstill. It can also offer very precise control of the center of mass of the sphere. There are several drawbacks to the design, however. The mechanical design tends to be very complex, and the controls necessary to coordinate the masses are complicated as well. To roll quickly in a straight line, the masses must rapidly reciprocate along their axes which may require high power actuators, and is very inefficient. Similar to the pendulum design, the locations on the sphere where the internal axle(s) mount are very rigid and impacts with obstacles may be undesirable at these points. Unlike the pendulum design which it is very unlikely to roll over these locations, the multiple mass design has these rigid areas spaced over the surface of the sphere where an impact with an obstacle it is very likely to occur. These multiple rigid locations necessitate having an overall rigid outer shell.

The radial configuration of masses has the additional drawback of requiring a central hub, which limits the performance of the mechanism. While the objective of the design is to offset the center of mass as far as possible, the central hub always prevent at

least one mass from moving past the center. The non-intersecting design avoids this issue while relying on the spherical shell as a structural exoskeleton.

2.3.1 Spherobot, Michigan State University

Omnidirectional spherical robots have seen significant development at Michigan State University, under the direction of Ranjan Mukherjee [10]. Spherobot (shown in Figure 2-9a), was the first to realize the multiple-mass-shifting propulsion mechanism with a radial configuration, and the design was patented (U.S. Patent 6,289,263) [9]. In additional to the radial masses, the design also featured retractable legs and a periscope-like camera. Since the legs and camera were fixed within the sphere, the goal of the project was to develop the trajectory planning and control theory to ensure the legs and camera were always oriented properly when the sphere reached its destination.

A newer design for Spherobot was later introduced using 3 perpendicular, non-intersecting masses as shown in Figure 2-9b [11]. This design was less complicated, requiring 1 fewer actuated mass, and had higher performance due to its ability to offset the center of mass more than the radial design.

2.3.2 August, Azad University of Qazvin and University of Tehran

August is another omnidirectional sphere based on the radial masses design (see Figure 2-11) developed at the Azad University of Qazvin in conjunction with the University of Tehran, Iran [12]. The design is such that the center of mass of the robot is located at the geometric center of the sphere when the masses are in equidistant from the center. This symmetry about the center of mass greatly simplifies the modeling and

control of the robot. A prototype has been built and successfully controlled to travel along curved trajectories.

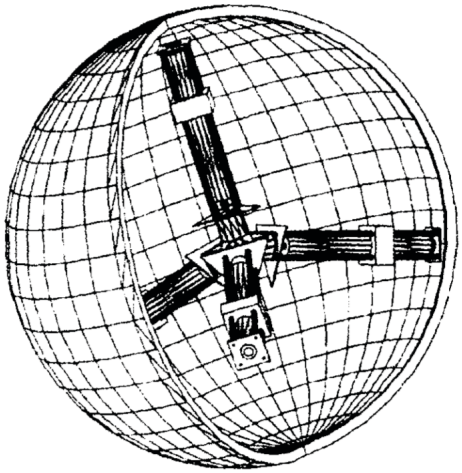


Figure 2-11: August Design [12].

2.4 Deformable Body

This class of spherical robot actively deforms its nominally spherical shape in order to propel itself. By continuously altering its shape, a deformable sphere can control the position of its center of mass relative to where the sphere contacts the ground such that the sphere rolls. Deformable spherical robots are a relatively new area of research and only a few prototypes have been demonstrated to date.

2.4.1 Koharo, Ritsumeikan University

The largest effort in this category comes from Ritsumeikan University in Japan. They have developed both a wheel and a sphere (called Koharo) with flexible outer structures and shape memory alloy (SMA) actuators [13]. By contracting and expanding the SMA wires in coordination, the outer structure of the wheel or sphere can be deformed to cause it to roll. As shown in Figure 2-12, the wheel is a flexible ring with

radial, SMA spokes. The sphere shown in Figure 2-13 is composed of several flexible rings also with radial SMA spokes.

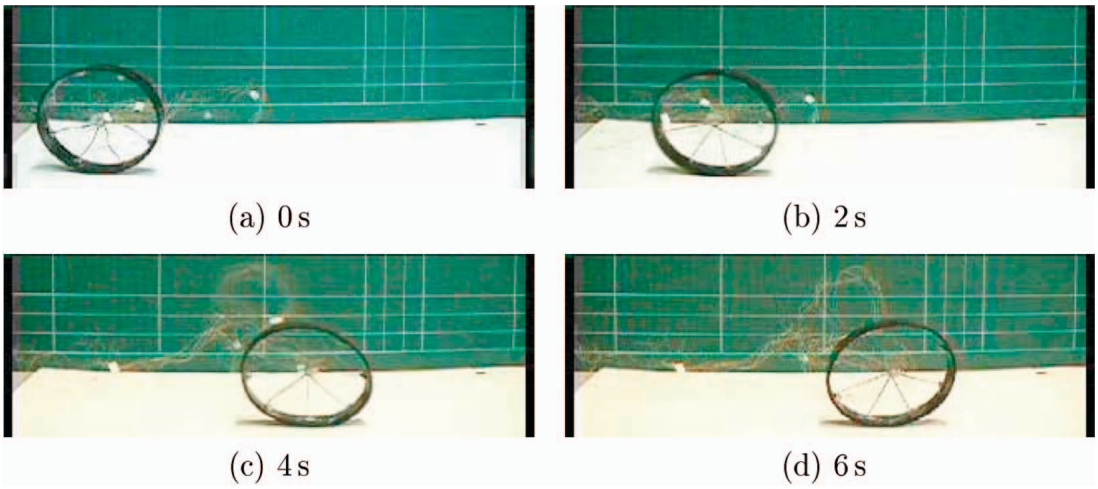


Figure 2-12: Deformable wheel rolling. Video snapshots of the wheel rolling over a period of 6 seconds [13].

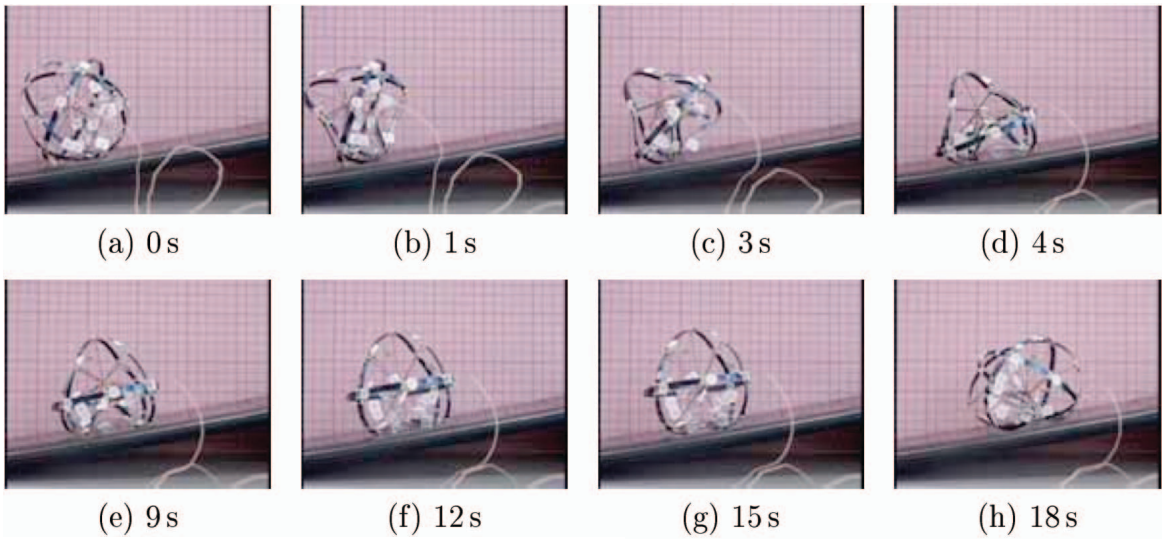


Figure 2-13: Deformable sphere rolling up an incline. Video snapshots of a the sphere rolling up an incline over a period of 18 seconds [13].

The SMA wires are contracted by passing an electric current through them to generate resistive heat. The temperature-induced change in crystalline structure causes the SMA wire to contract with significant force. Once cooled, the wire expands back to its original length. Unfortunately, since the cycle rate of the SMA actuators is very slow,

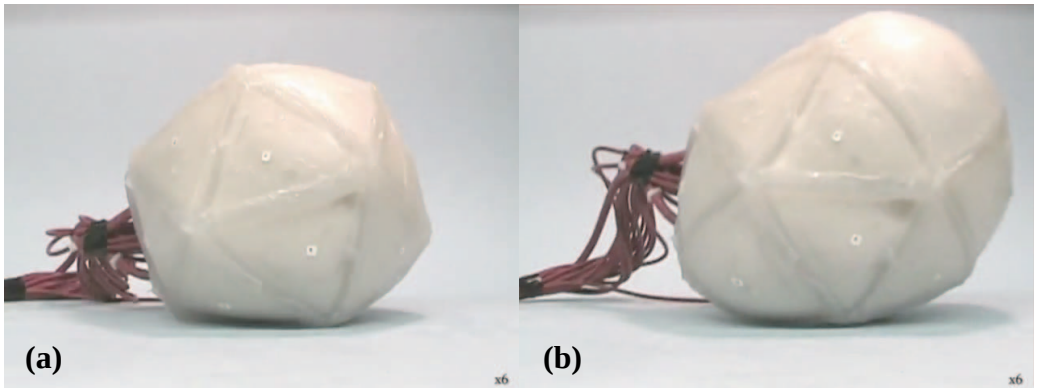
the motion of the wheel and sphere is also very slow as indicated by the time stamps on the image sequences in Figure 2-12 and Figure 2-13.

While rolling motion is prohibitively slow in the current prototypes, an effective jumping mechanic is also being developed. By deforming enough to form a concavity on the underside of the sphere, a significant amount of potential energy can be stored in the flexible outer structure. When released, the outer structure effectively "pops" back out, propelling the sphere into the air.

2.4.2 Chembot, iRobot

A different type of deformable sphere is under development by iRobot in conjunction with the University of Chicago. Funded by the DARPA Chembots program, the iRobot Chembot is a soft, inflatable, silicone sphere that uses the principle of jamming to selectively control the rigidity of individual sectors of the sphere [14]. The hollow sphere is formed from triangular sections that are actually pouches of a fine powder [15]. Normally, the sections are soft and flexible, but when a vacuum is drawn on the powder inside a section, the powder "jams" and the section hardens in its current shape.

To deform the sphere asymmetrically, all of the sections are rigidized except for one or two. When the internal space of the sphere is pressurized, the unjammed sections expand, deforming the sphere as shown in Figure 2-14. Repeating this process in a coordinated fashion enables rolling.



Chembot in its inflated, spherical form.

Left side of robot skin is jammed while right side expands.

Figure 2-14: iRobot Chembot [15].

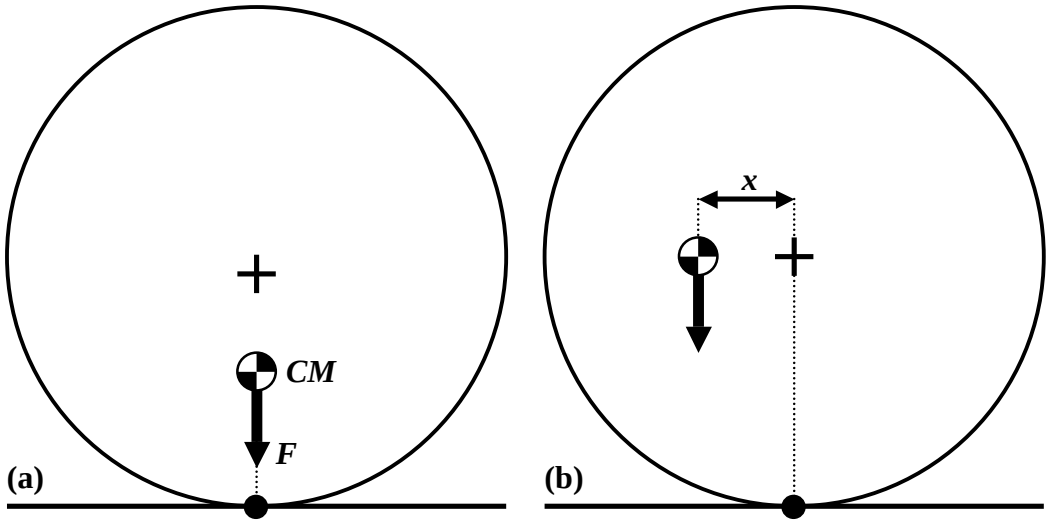
At this stage in development, the speed at which the sections can be actuated is prohibitively slow. Also the power source is external and the prototype has to be tethered during operation. The main goal of the project is not necessarily to build a deformable spherical robot, but to build a deformable robot that can squeeze through small openings.

2.5 Unifying Principle of Locomotion

The hamster ball, pendulum drive, multiple-mass-shifting, and deformable body concepts use very different mechanisms to propel a spherical robot, but in fact all operate by the same underlying principle. Ultimately, gravity causes the sphere to roll.

The acceleration due to gravity acting on the entire sphere can be simplified to a force, *F*, pulling down on the center of mass, *CM*, of the sphere. As shown in Figure 2-15a, when this force vector passes through the contact point between the sphere and the ground, the net torque on the sphere is zero. When the center of mass is displaced a horizontal distance, *x*, from the contact point as shown in Figure 2-15b, the force due to gravity creates a moment on the sphere, causing it to roll. While there are other inertial

properties that can affect the sphere's motion, this moment is the primary effect that sustains continuous motion.



Center of mass directly above contact point.

Center of mass offset a horizontal distance, *x*, from contact point.

Figure 2-15: Effect of Gravity on Sphere.

If the sphere and its center of mass were fixed in the configuration shown in Figure 2-15b and released from rest, the sphere would rock back and forth until coming to a stop (due to friction) in the configuration shown in Figure 2-15a. To enable continuous rolling motion, the mechanisms described in Sections 2.1-2.4 continuously displace the center of mass of the sphere relative to the contact point between the sphere and the ground. The moment generated by gravity is equal to the force, *F*, multiplied by the horizontal distance, *x* between the center of mass of the sphere and the contact point. Assuming the mass of the sphere cannot change, the distance, *x*, must be maximized in order to generate the largest moment.

2.6 Torque Limitation

Unfortunately, there is a practical limit to how far the center of mass can be displaced as well as significant tradeoffs between center of mass displacement, propulsion power, and system robustness. Consider the illustration in Figure 2-16.

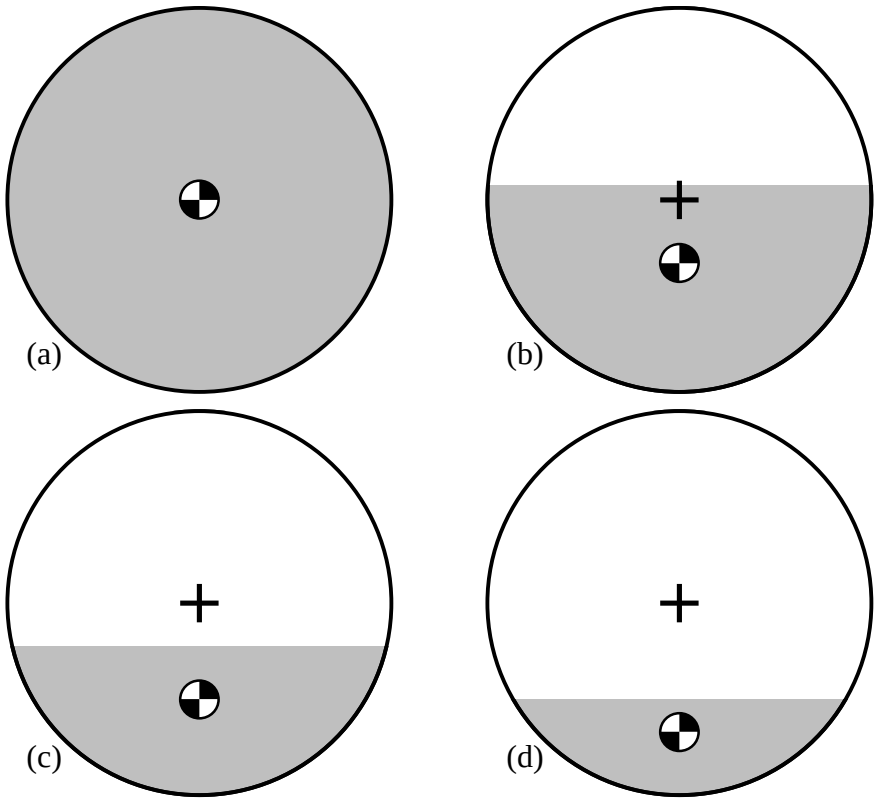


Figure 2-16: Illustration of design tradeoff. In each sphere, the center of mass is located at the geometric center of the spherical section indicated in gray. The ratio of the CM Displacement to Radius for each section: (a) 0, (b) 1/3, (c) 1/2, (d) 2/3.

For purposes of the illustration, the spherical shell is assumed to be massless, and the mass of the internal components is evenly distributed over the gray area in (a)-(d). In Figure 2-16a, the entire internal volume of the sphere can be used since the center of mass is located at the geometric center of the sphere. To displace the center of mass *R/3* (1/3 the radius of the sphere, *R*), the majority of the internal mass must be located in the bottom 56% of the internal volume. Similarly for a displacement of *R/2*, only the bottom

34% of volume can be used, and for a displacement of *2R/3*, only the bottom 16% of the volume can be used [16]. This trend indicates the tradeoff between maximizing the displacement of the center of mass and having enough room inside to fit the components necessary to displace said mass. Adding high-density ballast close to the outer shell to displace the mass further requires higher power and larger actuators inside to move said ballast.

Also, the assumption of a massless shell is unrealistic. Since the center of mass of the shell is located at the center of the sphere, it hinders the ability of the internal propulsion mechanism from displacing the overall center of mass as far as in the case of a massless shell. Therefore, it is desirable to minimize the mass of the outer spherical shell, but with the tradeoff of reduced robustness and strength of the shell. In practice, the maximum displacement of the center of mass of the sphere is close to *2R/3*, with significant sacrifices in propulsion power [2]. A higher performance design might have a limit of closer to *R/2*.

The practical limit of displacement of the center of mass translates into a limit on the maximum continuous incline and tallest obstacle that can be traversed. The maximum incline is given by,

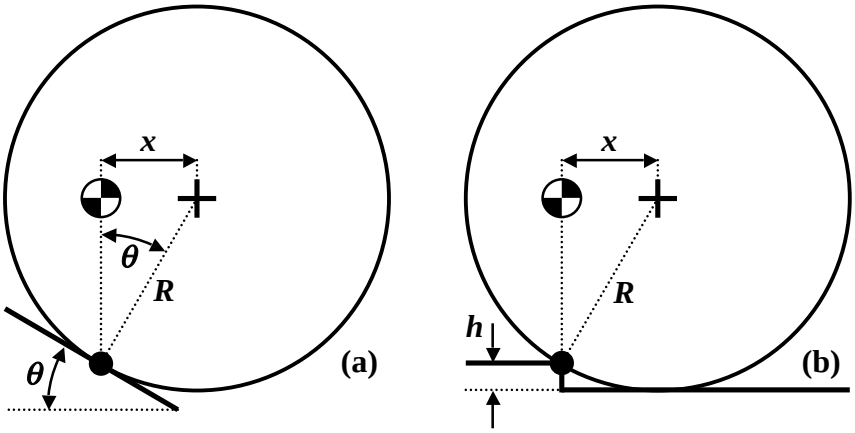
$$\theta = \sin^{-1}\frac{x}{R} \tag{2.1}$$

where θ is the inclination angle, x is the maximum displacement of the center of mass of the sphere from the center of the sphere, and R is the radius of the sphere. For a sphere with a displacement limit of $x = R/2$, the maximum continuous incline that can be traversed is 30°. This limit is reached when the center of mass of the sphere is directly

above the contact point between the sphere and the ground as shown in Figure 2-17a. The tallest step obstacle that can be climbed from rest is given by,

$$h = R - \sqrt{R^2 - x^2} \tag{2.2}$$

where h is the step height. For a sphere with a displacement limit of $x = R/2$, the maximum step that can be climbed from rest is $h = 0.14$ x R. This limit is shown in Figure 2-17b. This small step height limit is obviously very constraining.



Maximum Inclination. Maximum Step.

Figure 2-17: Mobility limits for a sphere with a max CM displacement of $x = R/2$.

The plot in Figure 2-18 shows both the maximum incline and maximum step as a function of the CM displacement capability of a spherical robot. The data points corresponding to $x = R/2$ are noted. The trend of both functions shows better mobility is achievable only at very high center of mass displacement, beyond practical limits.

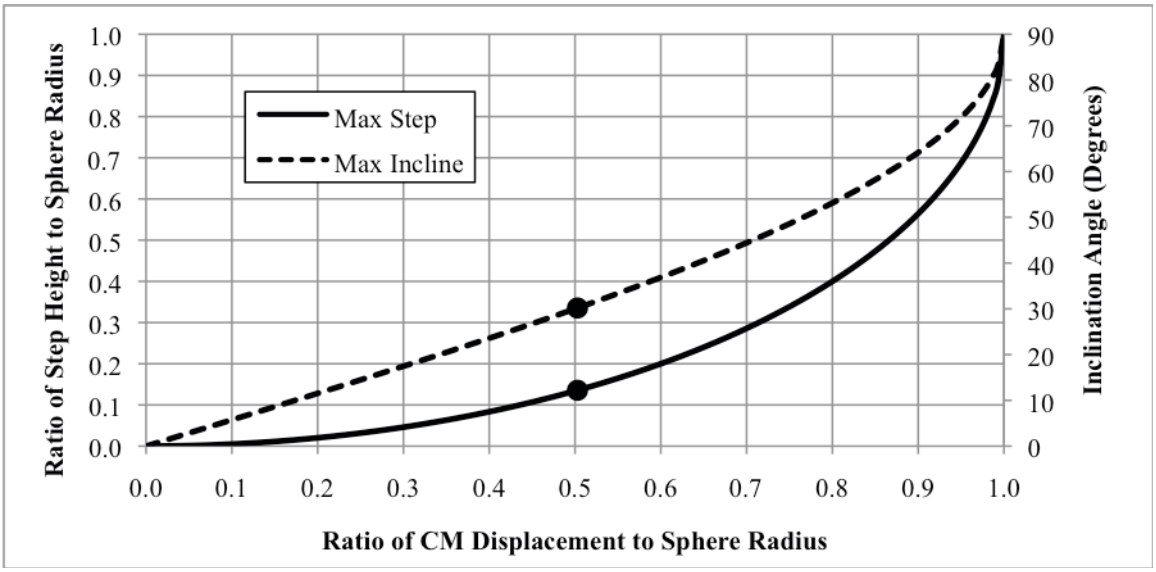


Figure 2-18: Plot of Mobility Limitations vs. Maximum CM Displacement.

As shown above, the most optimal design for a spherical robot that relies on shifting its center of mass in order to move is significantly limited in mobility regardless of the structure of the propulsion mechanism inside. In particular, being unable to overcome an obstacle taller than 1/10 the diameter of the sphere severely handicaps its utility in most applications. A solution to this problem must be found before spherical robots will see much use in the field.

Chapter 3

Preliminary Work

The mobility limitations discussed in Section 2.6 were the primary motivation for perusing research in the field of spherical robots, and a workable method to address these limitations was sought. The following is a description of the preliminary work that was done as well as the novel mechanism that was developed and tested to help overcome the inherent mobility limitations of previous spherical robot designs [1].

3.1 Angular Momentum Storage

A means of storing and dispensing angular momentum inside a spherical robot was investigated as a means to temporarily increase the maximum drive torque that can be generated. While temporary, an increase in drive torque would allow a spherical robot to ascend far steeper (yet finite) inclines, and climb over taller obstacles. The three primary mechanisms to store and dispense angular moment are as follows:

- Reaction Wheel
- Momentum-wheel
- Control Moment Gyroscope

3.1.1 Reaction Wheel

A reaction wheel is a nominally stationary flywheel with a large moment of inertia. When a torque is applied to rotate the flywheel, an equal and opposite reaction torque is applied to the structure on which it is mounted, due to the law of conservation of angular momentum. For use inside a spherical robot, a reaction wheel could be mounted such that its spin axis is parallel to the rolling axis of the sphere (see Figure 3-1). By applying a torque to the reaction wheel in the direction opposite the desired rolling direction, a reaction torque will be applied to the sphere in the desired direction. In other words, when the angular momentum of the reaction wheel is changed, the angular momentum of the rest of the sphere must change equally but in the opposite direction in order for momentum to be conserved.

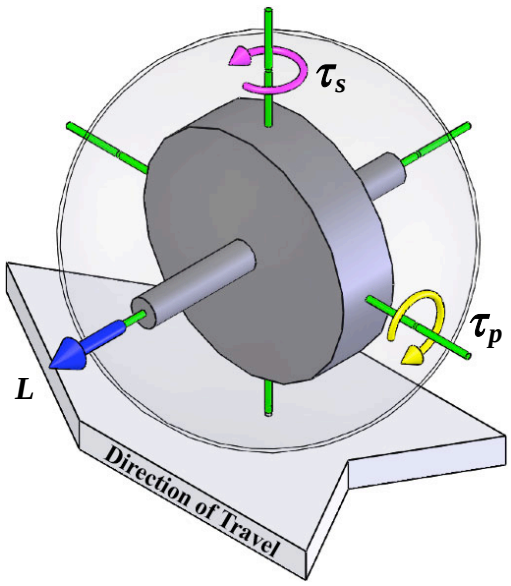


Figure 3-1: Reaction Wheel.

In Figure 3-1, $\boldsymbol{L}$ is the angular momentum of the flywheel when spinning, $\boldsymbol{\tau_s}$ is the torque on the flywheel due to turning the sphere (by some other means), and $\boldsymbol{\tau_p}$ is the precession torque induced.

While a reaction wheel has the potential to store a large amount of angular momentum, a prohibitively large and powerful motor would be required to transfer momentum to and from the wheel at a rate useful for propelling a spherical robot. Also, once spinning, the flywheel would exhibit undesirable gyroscopic precession any time the sphere changed direction when rolling. In Figure 3-1, when a torque, $\boldsymbol{\tau_s}$, is applied to the flywheel due to the sphere changing direction, a precession torque, $\boldsymbol{\tau_p}$, is induced which would cause the sphere to roll to the side. In order to avoid this problem, the reaction wheel must be slowly stopped after each use before the sphere attempts a direction change.

3.1.2 Momentum-wheel

A momentum-wheel is very similar to a reaction wheel except that it is nominally spinning at high velocity, and spun down to exchange momentum (see Figure 3-2a). The benefit to a momentum-wheel is that a much smaller motor can be used to initially spin up the flywheel over time. A brake can then be applied to dissipate its energy quickly and transfer angular momentum to the sphere.

The momentum-wheel mechanism experiences the same undesirable gyroscopic precession problem as the reaction wheel, but compounded by the fact that the flywheel must be kept spinning. To address the issue of undesirable gyroscopic effects, two counter-rotating flywheels of equal but opposite momentum can be used as shown in Figure 3-2b. When spinning, the net angular momentum of the system is zero. Therefore, tilting the spin axis of the dual flywheel assembly, as is required when steering the sphere, induces no gyroscopic precession effect on the overall assembly.

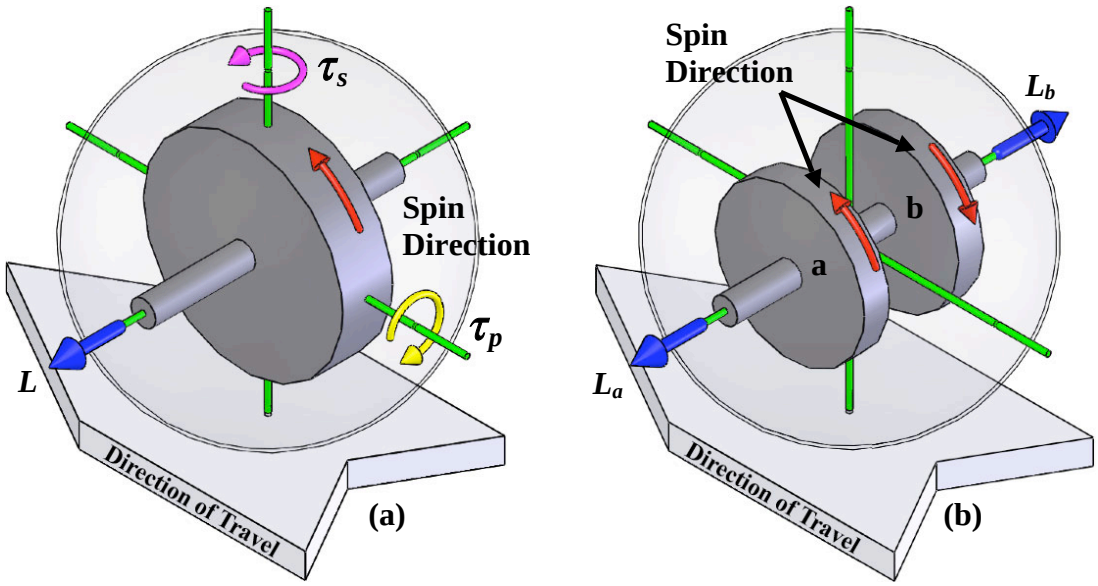


Figure 3-2: Single and dual momentum-wheel configurations.

In the single momentum-wheel configuration shown in Figure 3-2a, $\boldsymbol{L}$ is the angular momentum of the flywheel in the direction necessary for supplementing forward motion, $\boldsymbol{\tau_s}$ is the torque on the flywheel due to turning the sphere (by some other means), and $\boldsymbol{\tau_p}$ is the precession torque induced. Figure 3-2b shows a dual momentum-wheel configuration. When added together, the opposite angular momentums $\boldsymbol{L_a}$ and $\boldsymbol{L_b}$ equal a net angular momentum of zero for the two flywheels. As a unit, they behave as if no mass is spinning, and therefore do not produce precession effects on the sphere itself.

Inside the dual flywheel assembly, precession is experienced, but the precession torques generated by the flywheels are equal and opposite and cancel out. Besides addressing the issue of precession, however, the second flywheel in this configuration does not contribute to the forward motion performance of the sphere itself, since it is spinning in the opposite direction. When the momentum from the other flywheel is dispensed to help the sphere overcome obstacles, the two flywheels no longer cancel each

other out and the counter-rotating benefit is compromised until they can be equalized again.

In the dual momentum-wheel configuration described above, the momentum from the spinning flywheel is transferred to the spherical shell via torque along the spin axis of the flywheel. During this process, the kinetic energy of the flywheel is either transferred to the sphere or dissipated. To equalize the two flywheels and prepare them for use again, the lost energy must be replenished. A motor with sufficient power to spin up the flywheel quickly would again be prohibitively large.

3.1.3 Control Moment Gyroscope

With a different configuration, it is possible to utilize the angular momentum of the flywheels without disturbing their kinetic energy. By taking advantage of the gyroscopic precession deemed unfavorable in the reaction and momentum-wheel concepts above, the precession torque can be generated along the desired axis for forward motion of the sphere by changing the direction of the angular momentum while leaving its magnitude unaffected. When manipulated in this fashion, the flywheels are called control moment gyroscopes (CMGs). CMGs are a desirable alternative to reaction and momentum-wheels because they require far less power to generate the desired output torque.

With a single CMG, shown in Figure 3-3a, the applied flywheel tilting torque, $\boldsymbol{\tau_t}$, is perpendicular to the flywheel spin axis and to the desired output axis, along which the output precession torque helps drive the sphere forward. Unbalanced, this tilting torque causes the sphere to roll or tilt in an undesirable way. Two counter- rotating CMGs, however, tilted in opposite directions, will precess in the same direction and the resulting

torque will be the sum of the precession torques from the two flywheels along the desired axis for propelling the sphere (see Figure 3-3b). Essentially, when two CMGs are spinning but not being utilized, their net angular momentum is zero and the sphere can maneuver normally. When the CMGs are tilted to utilize their momentum, they maintain speed (in the absence of friction) since none of their kinetic energy is transferred to the outer shell. Therefore, tilting the flywheels requires little power and the motor(s) can be of manageable size.

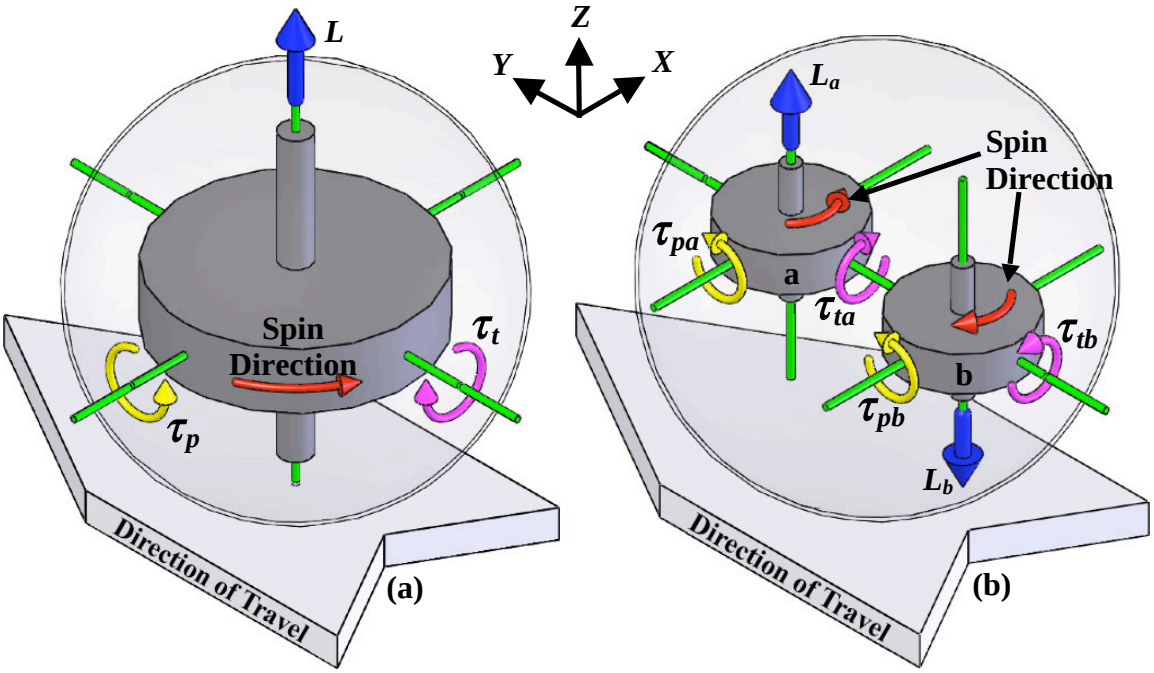


Figure 3-3: Single and Dual CMG configurations.

In the single CMG configuration shown in Figure 3-3a, $\boldsymbol{L}$ is the angular momentum of the flywheel, $\boldsymbol{\tau_t}$ is the applied tilting torque on the flywheel, and $\boldsymbol{\tau_p}$ is the precession torque produced in the direction necessary for supplementing forward motion. Figure 3-3b shows a dual CMG configurations. When added together, the angular momentums $\boldsymbol{L_a}$ and $\boldsymbol{L_b}$ equal a net angular momentum of zero for the two flywheels. The tilting torques $\boldsymbol{\tau_{ta}}$ and $\boldsymbol{\tau_{tb}}$ are applied to the flywheels in opposite directions resulting in no

net torque on the whole assembly. The precession torques $\boldsymbol{\tau_{pa}}$ and $\boldsymbol{\tau_{pb}}$ that are produced, however, add together in the direction necessary for supplementing forward motion.

3.1.4 Basic CMG Theory

The principle behind the control moment gyro is gyroscopic precession which can be explained in terms of torque and angular momentum starting with the following equations,

$$\tau = \frac{dL}{dt} \tag{3.1}$$

$$L = I\Omega \tag{3.2}$$

$$E = \frac{1}{2} I\Omega^2 \tag{3.3}$$

where $\boldsymbol{\tau}$ is torque, $\boldsymbol{L}$ is angular momentum, $\boldsymbol{I}$ is moment of inertia, $\boldsymbol{\Omega}$ is angular velocity, and $\boldsymbol{E}$ is kinetic energy. In Equation (3.1), torque produces a change in angular momentum. The angular momentum of a flywheel is a vector (magnitude and direction) and is a function of the moment of inertia of the flywheel and its angular velocity as shown in Equation (3.2). In Equation (3.3), the kinetic energy of the flywheel is a scalar and while also a function of moment of inertia and angular velocity, the direction of the angular velocity becomes irrelevant when it is squared. This fact leads to the usefulness of the control moment gyro, where a torque can be applied to change the direction of the angular momentum of a flywheel, while not changing the rotational kinetic energy.

For a physical explanation of how a control moment gyroscope works, consider the following description. In Figure 3-3a, the torque, $\boldsymbol{\tau_t}$, is attempting to tilt the flywheel about the ***Y***-axis, which would try to point the momentum vector $\boldsymbol{L}$ more towards the

positive ***X***-axis, increasing the angular momentum in the positive ***X***-direction. However, in order for there to be this increase along the ***X***-axis, there must be a torque in that direction per Equation (3.1). The gyroscope reacts in this case by producing a precession torque, $\boldsymbol{\tau_p}$, in the negative ***X***-direction, thereby canceling out the increase. Essentially, if the only torque that is applied to the flywheel is in the positive ***Y***-direction, then the angular momentum can only change in the ***Y***-direction; thus, the momentum vector tilts about the ***X***-axis, towards the positive ***Y***-axis. It is this behavior that is referred to as gyroscopic precession. This behavior can be more rigorously described mathematically.

Consider the diagram of a flywheel shown in Figure 3-4. Reference frame ***F*** is fixed to the flywheel with its origin located at the center of mass of the flywheel and its ***xyz***-axes aligned with the principle axes of the flywheel. The origin of frame ***G*** is also located at the center of mass of the flywheel and its ***Z***-axis coincides with the ***z***-axis of frame ***F***. The flywheel and frame ***F*** rotate with an angular velocity $\boldsymbol{\Omega_z}$ with respect to frame ***G*** about the ***Z***-axis of frame ***G***.

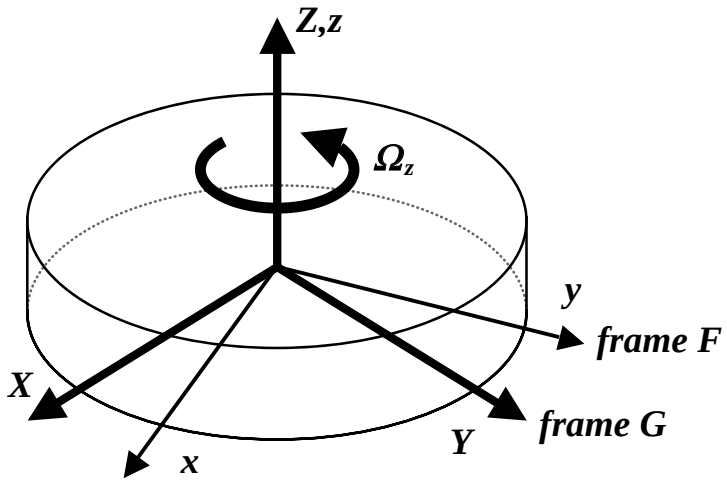


Figure 3-4: Diagram of Flywheel.

The 3D rotational equations of motion of the flywheel are [17],

$$M_X = I_{xx}\dot{\Omega}_x + I_{zz}\Omega_z\omega_Y - I_{yy}\Omega_y\omega_Z \tag{3.4}$$

$$M_Y = I_{yy}\dot{\Omega}_y + I_{xx}\Omega_x\omega_Z - I_{zz}\Omega_z\omega_X \tag{3.5}$$

$$M_Z = I_{zz}\dot{\Omega}_z + I_{yy}\Omega_y\omega_X - I_{xx}\Omega_x\omega_Y \tag{3.6}$$

where $\boldsymbol{M_X}$, $\boldsymbol{M_Y}$, and $\boldsymbol{M_Z}$ are the net moments about the ***X*-**, ***Y*-**, and ***Z*-**axes respectively, $\boldsymbol{I_{xx}}$, $\boldsymbol{I_{yy}}$, and $\boldsymbol{I_{zz}}$ are the principle moments of inertia of the flywheel about the ***x*-**, ***y*-**, and ***z*-**axes respectively, $\boldsymbol{\Omega_x}$, $\boldsymbol{\Omega_y}$, and $\boldsymbol{\Omega_z}$ are the angular velocity components of frame ***F*** with respect to frame ***G*** about the ***x*-**, ***y*-**, and ***z*-**axes respectively, and $\boldsymbol{\omega_X}$, $\boldsymbol{\omega_Y}$, and $\boldsymbol{\omega_Z}$ are the angular velocity components of frame ***G*** with respect to a global, inertial frame about the ***X*-**, ***Y*-**, and ***Z*-**axes respectively.

When frame ***G*** is stationary and the flywheel is spinning with constant angular velocity $\boldsymbol{\Omega_z}$ as shown in Figure 3-4, Equations (3.4), (3.5), and (3.6) are equal to zero, since the remaining angular velocity and acceleration terms are zero.

When an external tilting torque $\boldsymbol{\tau_t}$ is applied to the flywheel about the ***Y*-**axis, Equation (3.5) becomes,

$$M_Y = \tau_t = -I_{zz}\Omega_z\omega_X \tag{3.7}$$

Equation (3.7) indicates that the flywheel responds to this tilting torque $\boldsymbol{\tau_t}$ by rotating with angular velocity $\boldsymbol{\omega_X}$ about the positive ***X*-**axis,

$$\omega_X = -\frac{\tau_t}{I_{zz}\Omega_z} \tag{3.8}$$

If the angular velocity $\boldsymbol{\omega_X}$ about the ***X*-**axis is resisted by an opposing external torque $\boldsymbol{\tau_p}$ about the negative ***X*-**axis, Equation (3.4) becomes,

$$M_X = -\tau_p = I_{zz}\Omega_z\omega_Y \tag{3.9}$$

Equation (3.9) indicates that the flywheel ultimately responds by rotating with angular velocity $\boldsymbol{\omega_Y}$ about the ***Y***-axis,

$$\omega_Y = -\frac{\tau_p}{I_{zz}\Omega_z} \tag{3.10}$$

When the external torque $\boldsymbol{\tau_p}$ is applied to the flywheel about the negative ***X***-axis, it is opposed by an equal and opposite reaction torque $\boldsymbol{\tau_p}$ about the positive ***X***-axis. This reaction torque is the gyroscopic precession torque discussed earlier and is considered to be the output torque of the CMG.

Since $\boldsymbol{M_Z}$ remains zero, the magnitude of the flywheel angular velocity $\boldsymbol{\Omega_z}$ remains unchanged and as a result the rotational kinetic energy ***E*** and the magnitude of the angular momentum ***L*** are unaffected. Therefore, the input power, $\boldsymbol{P_{in}}$, to the CMG via the tilting torque is equal to the output power, $\boldsymbol{P_{out}}$, of the CMG via the precession torque,

$$P_{in} = \tau_t \omega_Y = -\frac{\tau_t \tau_p}{I_{zz}\Omega_z} = \tau_p \omega_X = P_{out} \tag{3.11}$$

The CMG effectively converts a torque in one direction to a torque in a perpendicular direction. It is critical to note, however, that as the gyroscope tilts, the direction of the output torque constantly changes as well.

When two CMGs are configured as in Figure 3-3b, the equal and opposite torques applied to tilt the flywheels result in no net torque on the sphere's internal assembly. Because the angular momenta of the two flywheels are in opposite directions, the tilting torques cause the CMGs to output precession torque in the same direction at the instant shown in the figure. As the flywheels tilt in opposite directions about their common ***Y***-axis, their angular momenta are no longer in parallel directions, but now have

components along the ***X***- and ***Z***-axes. The components along the ***Z***-axis remain equal and opposite and therefore cancel out. Importantly, the component of the output precession torque about the *global* ***X***-axis (the direction useful for supplementing forward motion of the sphere) goes with the cosine of the flywheel tilt angle as shown in the equation,

$$\tau_{px} = 2\tau_p \cos\theta_{tilt} \tag{3.12}$$

where $\boldsymbol{\tau_{px}}$ is the component of the precession torque about the *global* ***X***-axis and $\boldsymbol{\theta_{tilt}}$ is the angle of tilt about the ***Y***-axis where the configuration shown in Figure 3-3b is $\boldsymbol{\theta_{tilt} = 0}$, and $\dot{\theta}_{tilt} = \omega_Y$. Therefore, the CMGs are able to produce the most useful torque when aligned as shown in Figure 3-3b at zero tilt angle, and decrease with the cosine to zero useful output at 90° tilt angle.

The relationship between the output torque and the tilt rate is important since the CMGs need to produce a significant amount of torque for a useful duration of time to perform the intended maneuvers. The output torque and tilting rate are related by the angular momenta of the flywheels by substituting Equation (3.9) into Equation (3.12),

$$\tau_{px} = 2I_{zz}\Omega_z\omega_Y \cos\theta_{tilt} = 2L\omega_Y \cos\theta_{tilt} \tag{3.13}$$

Equation (3.13) indicates that the greater the angular momentum, the slower the flywheels will tilt for a given output torque, which ultimately allows them to be utilized for a longer period of time.

3.2 Prototype

While the configuration using dual, counter-rotating CMGs is the most mechanically complex, it has clear advantages over the other methods of storing and dispensing angular momentum discussed in Section 3.1. It is critical to note, however,

that the CMG mechanism does not eliminate the need for a propulsion mechanism based on shifting the center of mass of the sphere. The CMGs provide a supplementary yet *temporary* torque boost to the primary propulsion mechanism. Because the dual CMG mechanism can only generate output torque about a single axis, it is simplest to incorporate it into a spherical robot that has a primary rolling axis, such as one with a pendulum drive. An omnidirectional sphere propelled using the multiple-mass-shifting technique does not have a primary rolling axis, and would likely require a significantly more complex arrangement of CMGs to provide a full 3-axis torque output.

3.2.1 Design

A prototype was built incorporating a pendulum drive mechanism for forward/backward motion and steering, as well as a dual CMG mechanism for additional torque (see Figure 3-5). The project was a proof of concept with the primary focus on demonstrating the efficacy of the supplemental CMG mechanism. The prototype did not incorporate sensors or computer control, but instead it was deemed sufficient to rely solely on remote control to actuate the various systems.

Internal pendulum assembly. Fully assembled prototype.

Figure 3-5: Prototype with dual CMGs.

The prototype sphere was 0.46 m diameter and had a mass of 15 kg. The design was not fully optimized but was able to achieve a maximum displacement of the center of mass of x = R/3 = 0.08 m. The addition of the large flywheels in the design made it difficult to lower the center of mass of the system. Each flywheel had a mass of 2.2 kg and a moment of inertia of 10 g*m^2. When spun at a rate of 10000 rpm, the flywheels had a combined kinetic energy of 5000 J and angular momentum of 10 N*m*s. Complete detail concerning the design of the prototype can be found in the paper, "Design of a Spherical Vehicle with Flywheel Momentum Storage for High Torque Capabilities," [1].

3.2.2 Qualitative Results

Based on the specifications of the design, with a maximum CM displacement of $x = R/3$, the maximum incline that could be ascended using the pendulum alone was 20°.

Similarly, the tallest obstacle that could be tackled was 0.06*R = 1.4 cm (see Figure 2-18 for reference).

The pendulum drive prototype enhanced with dual CMGs was able to overcome obstacles of much greater height. One of the tests used a wooden apparatus to simulate a hole that was 10 cm deep (7 times taller obstacle than would be surmountable using the pendulum alone). Figure 3-6 shows the sphere climbing out of this hole with ease in about 0.5 seconds.

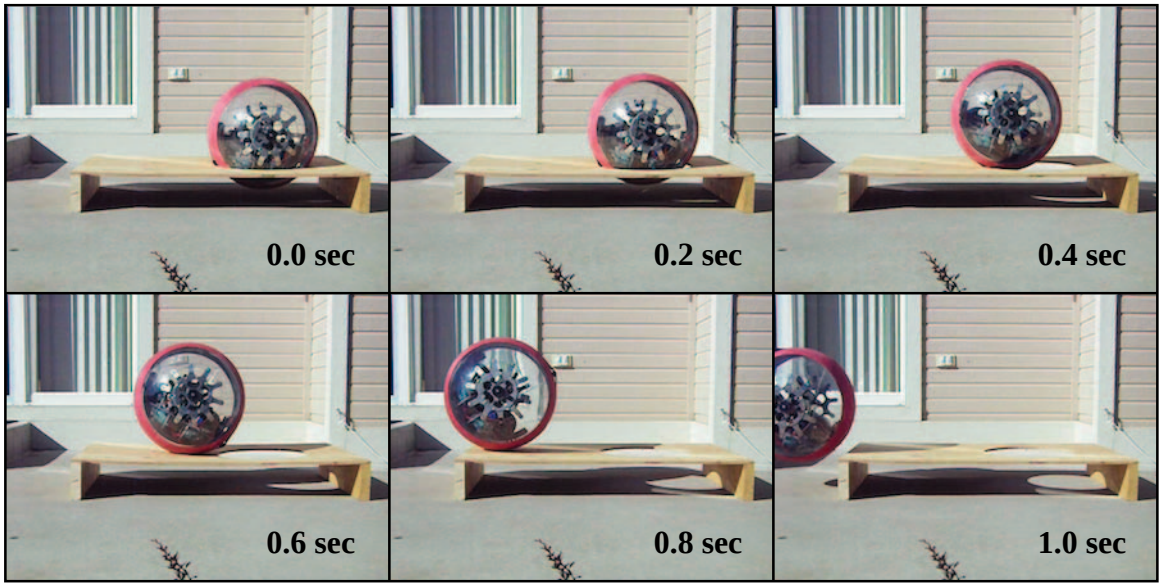


Figure 3-6: Prototype climbing out of a deep hole using CMGs. Hole is 10 cm deep. Video snapshots span a duration of 1.0 second.

Figure 3-7 shows the sphere prototype climbing up a 13 cm high step, starting from rest (9 times taller obstacle than would be surmountable using the pendulum alone). Higher obstacles were attempted and while it was clear there was sufficient torque to overcome them, the rubber tread on the spherical outer shell had inadequate traction to grip these higher obstacles. It is believed that with better traction, future prototype will be able to climb full height steps (nominally 18 cm, 13 times taller than would be surmountable using the pendulum alone). These two successful tests validated the

significant mobility enhancement afforded by the addition of CMGs inside a spherical robot.

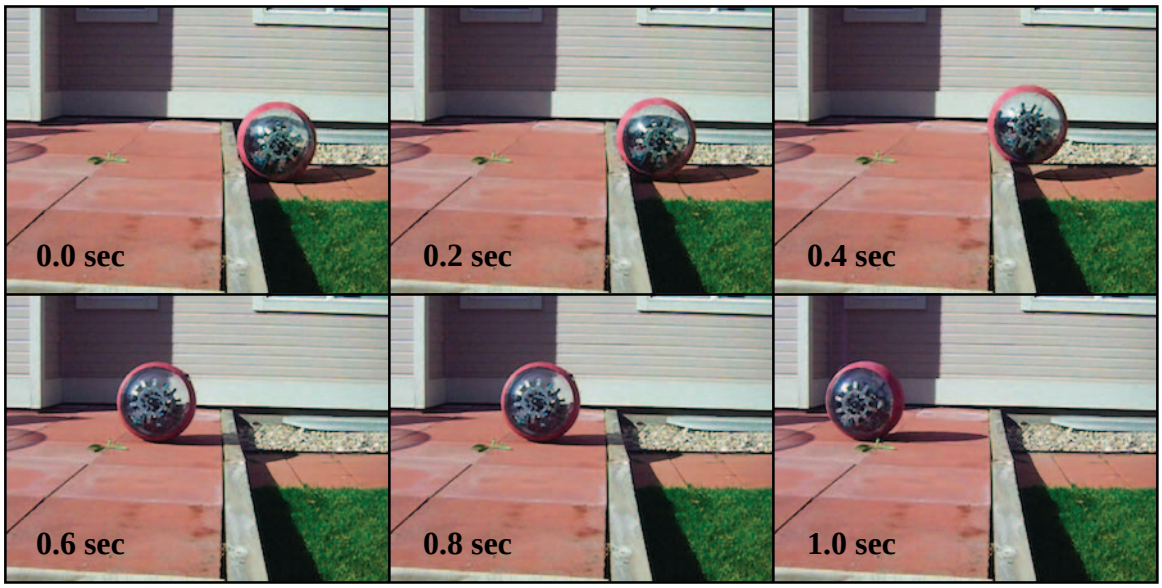


Figure 3-7: Prototype climbing up a step using CMGs. Step is 13 cm high. Video snapshots span a duration of 1.0 second.

3.3 Critical Observations

Aside from the successful CMG tests, other significant observations were made regarding the motion of the sphere when rolling using only the pendulum for propulsion (flywheels not spinning).

3.3.1 Wobble

The most apparent behavior that the spherical robot prototype displayed was a tendency to wobble or rock back and forth with little damping. For example, when the sphere was at rest with the pendulum fixed inside, bumping the sphere would cause it to oscillate back and forth about a spot on the ground. The sphere also wobbled if a constant pendulum drive torque was suddenly applied to the sphere starting from rest. In this case, it would accelerate forward while the angle between the pendulum and the

ground would oscillate. Since the pendulum was oscillating, the forward linear velocity of the sphere also appeared to oscillate as it accelerated. When traveling forward and then tilting the pendulum a fixed angle to the side to steer, the radius of the turn would oscillate as well. This oscillatory behavior was generally expected and has also been discussed in the research of others [3],[7],[8].

3.3.2 Shaft Nutation and Instability

Another behavior that was observed but not found to be discussed in the literature was the tendency of the primary drive axis to nutate when the sphere was traveling at a reasonable forward velocity. Specifically, the primary drive axis (the axis of the main drive shaft attached to the spherical shell) would incur some angular misalignment from the axis about which the sphere was actually rolling. When traveling slowly (estimated to be less than 0.5 m/s) this nutating shaft behavior, which could be initiated by a bump on the ground, would damp out quickly. When traveling at a moderate speed, the nutation would persist causing the direction of the sphere to oscillate back and forth.

When attempting to travel at high speed (estimated to be above 3 m/s) the angular misalignment between the axes would go unstable until the primary drive axis was flipping end over end. Even during a carefully controlled test on a level, smooth surface, starting with the drive axis level and the pendulum hanging straight down, an attempt at high-speed, straight-forward travel would still ultimately result in the primary drive axis flipping end over end.

3.4 Preliminary Conclusions and Future Work

The preliminary work was successful in demonstrating the use of control moment gyroscopes to effectively overcome significant mobility limitations inherent in past spherical robot designs. The mechanism that was created and reduced to practice was determined to be in fact novel, and is currently patent pending. During the rapid design, construction, and development of the prototype, many design elements and optimizations were not fully pursued and could see a significant amount of focus in future prototypes. Topics to be studied include improving the efficiency of the CMG designs, optimizing the tradeoff between CMG torque performance and pendulum performance, improving the robustness of the structural components, and studying how the performance of the CMG and pendulum system scales with varying sphere diameters.

During testing of the prototype under manual remote control, it was clear that instrumentation and computer control would be mandatory for a spherical robot to be effective in any application. The most interesting result from the project was the observation of varying oscillatory behavior of the prototype. The wobbling effects, shaft nutation, and in particular the high-speed instability that was observed indicated that sophisticated feedback control would be necessary in order to stabilize the motion of a spherical robot. The question that remained was whether all of the observed behaviors were inherent in the dynamics of a pendulum-driven spherical robot, or did some simply result from peculiarities in the prototype?

Chapter 4

Derivation of 3D Equations of Motion

The results of the preliminary work indicated a clear direction forward for the next research focus on spherical robots. The oscillatory and potentially unstable dynamics of the system need to be brought under control in order for a spherical robot to propel itself with high performance and agility while negotiating unknown terrain. To develop the necessary control algorithms and to understand what level of performance is actually achievable requires first understanding the underlying dynamics and behavior of the system. Starting from first principles, a general, 3-dimensional, mathematical model of a spherical robot rolling on flat ground was developed. The ultimate goal of this research was to see whether the observed behavior of the earlier prototype would reveal itself in simulation of the dynamic model of the sphere.

4.1 Model Description

The complex system of a spherical robot with a pendulum drive boils down to essentially two rigid bodies: the pendulum and the spherical shell (from here on referred to as simply the sphere). The center of mass of the sphere is located at its center.

Pivoting about the center of the sphere, the pendulum has its center of mass some distance offset from the center (see Figure 4-1).

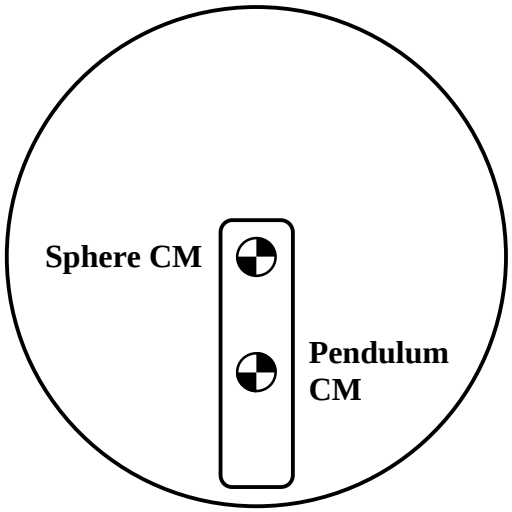


Figure 4-1: System components: spherical shell and pendulum.

The position and orientation of the sphere and pendulum can be represented using three reference frames: a global inertial frame (***O***), a sphere frame (***S***) fixed at the center (also the center of mass) of the sphere, and a pendulum frame (***P***) fixed to the pendulum at its center of mass (see Figure 4-2).

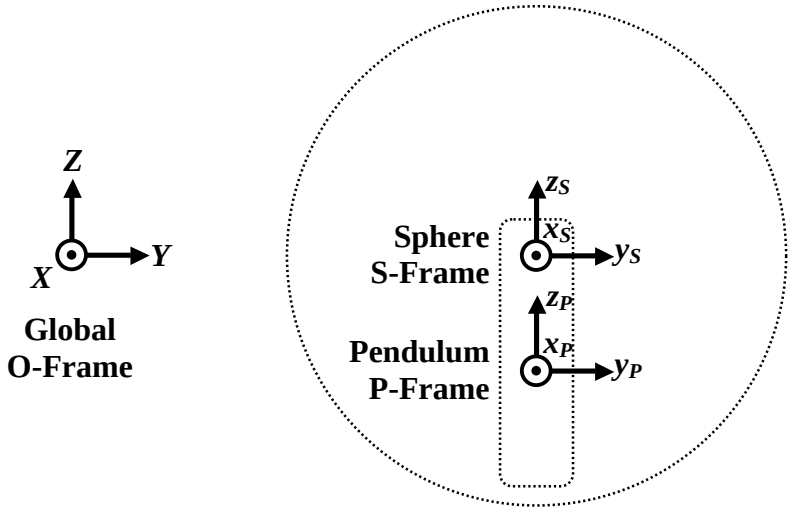


Figure 4-2: Reference frames: global, sphere, and pendulum.

Since this model is only considering motion on flat ground, the position of the sphere is represented by a displacement ***x*** along the ***X-axis*** of the global ***O*** frame and a displacement ***y*** along the ***Y-axis*** of the global ***O*** frame. To define the arbitrary rotation of the sphere (***S*** frame) relative to the ***O*** frame an appropriate set of Euler angles was used. To transform from the ***O*** frame to the ***S*** frame, the following sequence of three rotations is performed as shown in Figure 4-3a [17]:

1. A positive rotation ψ about the ***Z-axis***, resulting in the primed system.
2. A positive rotation θ about the ***y′-axis***, resulting in the double-primed system.
3. A positive rotation ϕ about the ***x″-axis***, resulting in the final unprimed system.

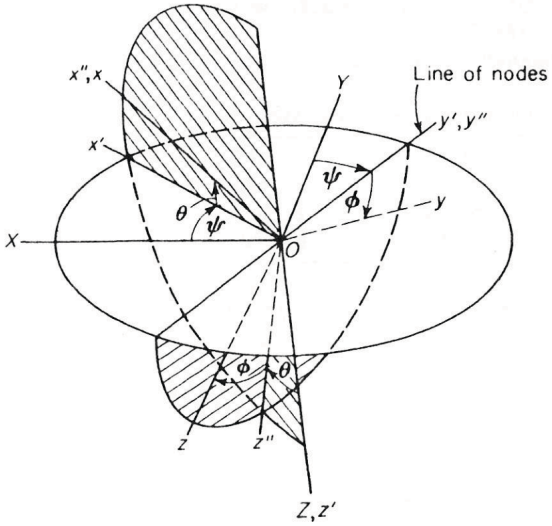
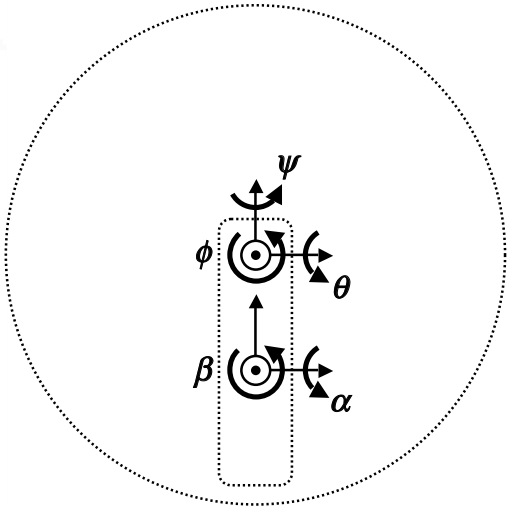


Illustration of the sequence of 3 rotations [17].



Angles associated with the orientation of the sphere and pendulum.

Figure 4-3: Euler Angles.

In Figure 4-3b, the Euler angles, ψ, θ, and ϕ, are indicated as they apply to the ***S*** frame of the sphere. An additional set of two angles, α and β, is used to define the relative rotation of the pendulum ***P*** frame relative to the ***S*** frame. Only two angles are needed, because the pendulum can only rotate in two directions relative to the sphere. The pendulum can rotate forward and backward an angle α and then tilt side to side an

angle $\boldsymbol{\beta}$. After rotating to change from the orientation of the ***S*** frame to the orientation of the ***P*** frame, the location of the ***P*** frame is a distance ***d*** along the negative $\boldsymbol{z_P}$***-axis***. These rotations and displacements are represented mathematically with the following transformation matrices,

$$D_{xy}(x,y) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

$$R_{z_S}(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

$$R_{y_S}(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

$$R_{x_S}(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

$$R_{y_P}(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

$$R_{x_P}(\beta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) & 0 \\ 0 & \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

$$ {}^{P}P_{P} = \begin{pmatrix} 0 \\ 0 \\ -d \\ 1 \end{pmatrix} \tag{4.7} $$

where D_{xy} is the translation transformation matrix from origin of the ***O*** frame to the origin of the ***S*** frame, R_{zs} is the rotation due to $\boldsymbol{\psi}$ about the $\boldsymbol{z_S}$***-axis***, R_{ys} is the rotation due to $\boldsymbol{\theta}$ about the $\boldsymbol{y_S}$***-axis***, R_{xs} is the rotation due to $\boldsymbol{\phi}$ about the $\boldsymbol{x_S}$***-axis***, R_{yp} is the rotation due to $\boldsymbol{\alpha}$ about the $\boldsymbol{y_P}$***-axis***, R_{xp} is the rotation due to $\boldsymbol{\beta}$ about the $\boldsymbol{x_P}$***-axis***, and ${}^{P}P_{P}$ is the vector from the ***S*** frame origin to the ***P*** frame origin (the pendulum CM) relative to the ***P*** frame. These transformations are important to compute the position center of mass of the pendulum relative to the global frame ***O*** as follows,

$$ {}^{O}P_{P} = D_{xy}(x,y)R_{z_S}(\psi)R_{y_S}(\theta)R_{x_S}(\phi)R_{y_P}(\alpha)R_{x_P}(\beta)\,{}^{P}P_{P} = {}^{O}\begin{pmatrix} x_P \\ y_P \\ z_P \\ 1 \end{pmatrix} \tag{4.8} $$

where ${}^{O}P_{P}$ is the vector from the ***O*** frame origin to the ***P*** frame origin relative to the ***O*** frame. *MATLAB* and the *Symbolic Toolbox* were used to symbolically compute the ${}^{O}P_{P}$ vector as well as most of the subsequent symbolic calculations. The symbolic results of most of these calculations are exceedingly long and do not simplify much; therefore, most will not be printed here.

The velocity components of the center of mass of the sphere in the ***O*** frame are simply the time derivatives of the ***x*** and ***y*** components of its position, $\dot{x}$ and $\dot{y}$. The velocity components of the center of mass of the pendulum relative to the ***O*** frame are the time derivatives of the position components given by the ${}^{O}P_{P}$ vector. A script was

written in the *AppleScript* programming language to perform this time differentiation. The script is discussed in more detail in Section 4.2.3.

The angular velocity of the sphere relative to the ***S*** frame, ${}^{S}\omega_{S}$, can be expressed as the sum of sphere Euler angle angular velocity vectors as follows [17],

$$ {}^{S}\omega_{S} = \dot{\psi} + \dot{\theta} + \dot{\phi} \tag{4.9} $$

where ${}^{S}\omega_{S}$, $\dot{\psi}$, $\dot{\theta}$, and $\dot{\phi}$ are vector quantities. The components of the angular velocity of the sphere along the x_S-, y_S-, and z_S-***axes*** of the ***S*** frame, ${}^{S}\omega_{x_S}$, ${}^{S}\omega_{y_S}$, and ${}^{S}\omega_{z_S}$ respectively, are [17],

$$ \begin{pmatrix} {}^{S}\omega_{x_S} \\ {}^{S}\omega_{y_S} \\ {}^{S}\omega_{z_S} \end{pmatrix} = \begin{pmatrix} \dot{\phi} - \dot{\psi}\sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \\ \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi \end{pmatrix} \tag{4.10} $$

The angular velocity of the pendulum relative to the sphere relative to the ***P*** frame, ${}^{P}\omega_{P}^{rel}$, can be expressed as the sum of the pendulum angle angular velocity vectors as follows [17],

$$ {}^{P}\omega_{P}^{rel} = \dot{\beta} + \dot{\alpha} \tag{4.11} $$

where ${}^{P}\omega_{P}^{rel}$, $\dot{\beta}$, and $\dot{\alpha}$ are vector quantities. Because the two relative pendulum angles were chosen using the same convention as the sphere Euler angles, the components of ${}^{P}\omega_{P}^{rel}$ can be found by substituting $\theta = \alpha$, $\phi = \beta$, and $\psi = 0$ into (4.10). The components of the relative angular velocity of the pendulum along the x_P-, y_P-, and z_P-***axes*** of the ***P*** frame, ${}^{P}\omega_{x_P}^{rel}$, ${}^{P}\omega_{y_P}^{rel}$, and ${}^{P}\omega_{z_P}^{rel}$ respectively, are,

$$\begin{pmatrix} {}^{P}\omega_{x_P}^{rel} \\ {}^{P}\omega_{y_P}^{rel} \\ {}^{P}\omega_{z_P}^{rel} \end{pmatrix} = \begin{pmatrix} \dot{\beta} \\ \dot{\alpha}\cos\beta \\ -\dot{\alpha}\sin\beta \end{pmatrix} \tag{4.12}$$

In order to find the absolute angular velocity of the pendulum relative to the sphere frame, ${}^{S}\omega_P$, the relative angular velocity of the pendulum relative to the ***P*** frame is transformed to the ***S*** frame and added to the angular velocity of the sphere relative to the ***S*** frame,

$${}^{S}\omega_P = R_{y_p}(\alpha) R_{x_p}(\beta) {}^{P}\omega_P^{rel} + {}^{S}\omega_S \tag{4.13}$$

4.2 Lagrange Equations of Motion

The Lagrangian approach was chosen to formulate the equations of motion of the sphere and pendulum system, because it relies on the kinetic and potential energies of the system which are relatively straightforward to compute given the position and velocity information derived in Section 4.1. To use the Lagrangian approach, an appropriate set of generalized coordinates, $\boldsymbol{q_i}$, were chosen which fully specify the system,

$$\begin{aligned} q_1 &= \psi \\ q_2 &= \theta \\ q_3 &= \phi \\ q_4 &= x \\ q_5 &= y \\ q_6 &= \alpha \\ q_7 &= \beta \end{aligned} \tag{4.14}$$

The general form of Lagrange's equation is as follows [17],

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \sum_{j=1}^{m} \lambda_i a_{ji} + Q_i' \tag{4.15}$$

where $\boldsymbol{L}$ is the Lagrangian function, $\boldsymbol{m}$ is the number of constraint equations, $\boldsymbol{\lambda_j}$ are the Lagrange multipliers, $\boldsymbol{a_{ji}}$ are the coefficients in front of $\dot{q}_i$ in the constraint equations, and $\boldsymbol{Q_i'}$ are the generalized non-conservative forces and moments. As shown in Equation (4.14), there are $\boldsymbol{n}$ = 7 generalized coordinates. As such, there will be 7 Lagrange equations.

To formulate the Lagrange equations several parameters of the design are also needed: the mass of the sphere, $\boldsymbol{M_S}$, the mass of the pendulum, $\boldsymbol{M_P}$, the principle moments of inertia of the sphere, $\boldsymbol{Ixx_S}$, $\boldsymbol{Iyy_S}$, and $\boldsymbol{Izz_S}$, which align with the principle $\boldsymbol{x_S}$-, $\boldsymbol{y_S}$-, and $\boldsymbol{z_S}$-***axes***, and the principle moments of inertia of the pendulum, $\boldsymbol{Ixx_P}$, $\boldsymbol{Iyy_P}$, and $\boldsymbol{Izz_P}$, which align with the principle $\boldsymbol{x_P}$-, $\boldsymbol{y_P}$-, and $\boldsymbol{z_P}$-***axes***.

4.2.1 Formulation of the Lagrangian

The Lagrangian function, $\boldsymbol{L}$, is defined as,

$$L = T - V \tag{4.16}$$

where $\boldsymbol{T}$ is the total kinetic energy of the system, and $\boldsymbol{V}$ is the total potential energy of the system. The translational and rotational kinetic energies of the sphere and pendulum were computed separately and then summed as follows,

$$T_S^{trans} = \frac{1}{2} M_S \left(\dot{x}^2 + \dot{y}^2\right) \tag{4.17}$$

$$T_P^{trans} = \frac{1}{2} M_P \left({}^{O}\dot{P}_{x_P}^2 + {}^{O}\dot{P}_{y_P}^2 + {}^{O}\dot{P}_{z_P}^2\right) \tag{4.18}$$

$$T_S^{rot} = \frac{1}{2}\left(Ixx_S \, {}^{S}\omega_{x_S}^2 + Iyy_S \, {}^{S}\omega_{y_S}^2 + Izz_S \, {}^{S}\omega_{z_S}^2\right) \tag{4.19}$$

$$T_P^{rot} = \frac{1}{2}\left(Ixx_P {}^S\omega_{x_P}^2 + Iyy_P {}^S\omega_{y_P}^2 + Izz_P {}^S\omega_{z_P}^2 \right) \tag{4.20}$$

$$T = T_S^{trans} + T_P^{trans} + T_S^{rot} + T_P^{rot} \tag{4.21}$$

where T_S^{trans} is the translational kinetic energy of the sphere, T_P^{trans} is the translational kinetic energy of the pendulum, T_S^{rot} is the rotational kinetic energy of the sphere, T_P^{rot} is the rotational kinetic energy of the pendulum, and ${}^O\dot{P}_{x_P}$, ${}^O\dot{P}_{y_P}$, and ${}^O\dot{P}_{z_P}$ are the components of the velocity of the center of mass of the pendulum along the ***X-***, ***Y-***, and ***Z-axes*** respectively. Since the model is only concerned with motion on flat ground, the sphere does not contribute to the potential energy of the system. The total potential energy is the gravitational potential energy of the pendulum as follows,

$$V = M_P g {}^O P_{z_P} \tag{4.22}$$

where ***g*** is the acceleration due to gravity, and ${}^O P_{z_P}$ is the vertical component of the position of the center of mass of the pendulum relative to the ***O*** frame.

4.2.2 No-Slip Rolling Constraint

For the mathematical model, it is assumed that the sphere rolls on the ground without slipping. This constraint couples the angular velocity of the sphere with the linear velocity of the sphere. $\dot{x}$ and $\dot{y}$ are the components of the linear velocity of the sphere along the ***X-*** and ***Y-axes*** of the ***O*** frame. In order to relate $\dot{x}$ and $\dot{y}$ to the angular velocity of the sphere, ${}^S\omega_S$ must be transformed into the ***O*** frame as follows,

$${}^O\omega_S = R_{z_S}(\psi) R_{y_S}(\theta) R_{x_S}(\phi) {}^S\omega_S \tag{4.23}$$

where ${}^{O}\omega_S$ is the angular velocity of the sphere relative to the ***O*** frame. The component of angular velocity of the sphere about the ***X-axis***, ${}^{O}\omega_{X_S}$, is coupled to $\dot{y}$, and the component of angular velocity of the sphere about the ***Y-axis***, ${}^{O}\omega_{Y_S}$, is coupled to $\dot{x}$ as follows,

$$\begin{aligned} K_1 &= \dot{x} - r\,{}^{O}\omega_{Y_S} = 0 \\ K_2 &= \dot{y} + r\,{}^{O}\omega_{X_S} = 0 \end{aligned} \tag{4.24}$$

where $\boldsymbol{K_j}$ are the constraint equations labels, and ***r*** is the radius of the sphere. Because the $\boldsymbol{K_j}$ equations contain derivatives of the generalized coordinates and cannot be integrated to relate the generalized coordinates directly, they are considered nonholonomic constraints. The $\boldsymbol{K_j}$ equations expanded and written in terms of the generalized coordinates are,

$$\begin{aligned} K_1 &= \dot{x} - \left(r\cos\psi\right)\dot{\theta} - \left(r\cos\theta\sin\psi\right)\dot{\phi} \\ K_2 &= \dot{y} - \left(r\sin\psi\right)\dot{\theta} + \left(r\cos\theta\cos\psi\right)\dot{\phi} \end{aligned} \tag{4.25}$$

The $\boldsymbol{a_{ji}}$ coefficients in front of the $\dot{q}_i$ terms in the $\boldsymbol{K_j}$ equations are,

$$\begin{aligned} a_{11} &= 0 & a_{21} &= 0 \\ a_{12} &= -r\cos\psi & a_{22} &= -r\sin\psi \\ a_{13} &= -r\cos\theta\sin\psi & a_{23} &= r\cos\theta\cos\psi \\ a_{14} &= 1 & a_{24} &= 0 \\ a_{15} &= 0 & a_{25} &= 1 \\ a_{16} &= 0 & a_{26} &= 0 \\ a_{17} &= 0 & a_{27} &= 0 \end{aligned} \tag{4.26}$$

4.2.3 Differentiation of the Lagrangian

As shown in the general form of Lagrange's equation, Equation (4.15), the partial derivatives of the Lagrangian function with respect to q_i and the $\dot{q}_i$ are required. These

partial derivatives were computed symbolically in *MATLAB* using `diff`. Unfortunately no *MATLAB* function was found to compute time derivatives, which are part of the Lagrange equation as well.

A custom script named *Differentiator* was written in the *AppleScript* programming language to compute the time derivatives. The results of the partial derivatives of the Lagrangian function with respect to $\dot{q}_i$ were fully expanded and fed into the script as a string. Computing the time derivative of these particular expressions was relatively straightforward since the terms consist of combinations of a limited set of elements. The script parses the string by first splitting up the additive terms. Each of these terms is parsed into separate product terms. The chain rule is then carried out over the finite set of possible elements. The results are then added back together to form the complete time derivative. This string was then fed back into *MATLAB*.

Automating the differentiation was necessary due to the excessive length the Lagrangian function and its derivatives. The Lagrangian function was found to have over 300 terms with combinations of the 14 q_i and $\dot{q}_i$. When computing the time derivative, the chain rule tended to cause the number of terms to balloon rapidly. After simplification, the longest time derivative was found to be 551 terms long. Working with equations of lengths of this magnitude is only feasible using computer tools. The *Differentiator* script can be found in Appendix E.

4.2.4 Generalized Non-Conservative Torques and Forces

The effect of gravity acting on the pendulum is a conservative body force and is already taken into account in the potential energy of the system in Section 4.2.1. Non-conservative torques and forces affect the system and need to be taken into account as

well. The pendulum drive torque and the pendulum tilt (steering) torque are non-conservative. Friction was also incorporated to more accurately represent the physical system. In reality, a complex combination of coulombic and velocity dependent damping affect the system. Since coulombic friction is discontinuous and difficult to model, it was left out of the model. Damping was implemented in the model to simulate viscous friction in the drive train between the pendulum and the sphere. It was also added to simulate viscous friction between the sphere and the ground when rolling as well as between the sphere and the ground when the sphere is spinning about the ***Z-axis***.

To incorporate these non-conservative torques and forces into the general form of the Lagrange equation, they were put in the generalized form as follows,

$$Q_i' = \sum_k F_k \frac{\partial x_k}{\partial q_i} + \sum_k M_k \frac{\partial \theta_k}{\partial q_i} \tag{4.27}$$

where $\boldsymbol{k}$ increments through all of the non-conservative forces and moments, $\boldsymbol{F_k}$ is a non-conservative force, $\boldsymbol{x_k}$ is a linear displacement in the direction of $\boldsymbol{F_k}$, $\boldsymbol{M_k}$ is a non-conservative moment, and $\boldsymbol{\theta_k}$ is an angular displacement about the $\boldsymbol{M_k}$ axis. The Q_i' terms for each Lagrange equation are,

$$\begin{aligned}
Q_1' &= -f_4\, {}^{O}\omega_{Z_s} \\
Q_2' &= 0 \\
Q_3' &= 0 \\
Q_4' &= -f_3\dot{x} \\
Q_5' &= -f_3\dot{y} \\
Q_6' &= M_1 - f_1\dot{\alpha} \\
Q_7' &= M_2 - f_2\dot{\beta}
\end{aligned} \tag{4.28}$$

where $\boldsymbol{f_1}$ is the damping coefficient on the relative angular velocity, $\dot{\alpha}$, between the pendulum and sphere, $\boldsymbol{f_2}$ is the damping coefficient on the relative angular velocity, $\dot{\beta}$,

between the pendulum and sphere, f_3 is the damping coefficient on the velocity components, $\dot{x}$ and $\dot{y}$, of the sphere, f_4 is the damping coefficient on the angular velocity, ${}^{O}\omega_{Z_S}$, of the sphere about the ***Z-axis***, M_1 is the drive torque between the pendulum and the sphere, and M_2 is the tilt (steering) torque between pendulum and the sphere.

4.2.5 Formulation of the Equations of Motion in the General Form

With all of the parts now computed, the equations of motion in the general form of the Lagrange equation can be formulated. For easier manipulation in MATLAB, the terms in these equations were all pulled to one side and set to zero as follows,

$$
\begin{aligned}
G_1 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right) - \frac{\partial L}{\partial q_1} - \lambda_1 a_{11} - \lambda_2 a_{21} - Q_1' = 0 \\
G_2 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_2}\right) - \frac{\partial L}{\partial q_2} - \lambda_1 a_{12} - \lambda_2 a_{22} - Q_2' = 0 \\
G_3 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_3}\right) - \frac{\partial L}{\partial q_3} - \lambda_1 a_{13} - \lambda_2 a_{23} - Q_3' = 0 \\
G_4 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_4}\right) - \frac{\partial L}{\partial q_4} - \lambda_1 a_{14} - \lambda_2 a_{24} - Q_4' = 0 \\
G_5 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_5}\right) - \frac{\partial L}{\partial q_5} - \lambda_1 a_{15} - \lambda_2 a_{25} - Q_5' = 0 \\
G_6 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_6}\right) - \frac{\partial L}{\partial q_6} - \lambda_1 a_{16} - \lambda_2 a_{26} - Q_6' = 0 \\
G_7 &: \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_7}\right) - \frac{\partial L}{\partial q_7} - \lambda_1 a_{17} - \lambda_2 a_{27} - Q_7' = 0
\end{aligned}
\tag{4.29}
$$

where λ_1 is the Lagrange multiplier associated with the first constraint equation, and λ_2 is the Lagrange multiplier associated with the second constraint equation. The Lagrange multipliers are usually related to or equal to the constraint force associated with each constraint. The entirely of equation G_1 is shown in Appendix A to demonstrate the

complexity of these equations. In addition to these 7 equations, there are the 2 constraint equations, (4.25), which are reproduced below in a different form,

$$\begin{aligned} G_8 &: a_{11}\dot{q}_1 + a_{12}\dot{q}_2 + a_{13}\dot{q}_3 + a_{14}\dot{q}_4 + a_{15}\dot{q}_5 + a_{16}\dot{q}_6 + a_{17}\dot{q}_7 = 0 \\ G_9 &: a_{21}\dot{q}_1 + a_{22}\dot{q}_2 + a_{23}\dot{q}_3 + a_{24}\dot{q}_4 + a_{25}\dot{q}_5 + a_{26}\dot{q}_6 + a_{27}\dot{q}_7 = 0 \end{aligned} \tag{4.30}$$

Since the Lagrange multipliers are additional unknowns, there are now 9 equations in 9 unknowns. The complete formulation of these equations can be found in Appendix B.

4.3 Numerical Solution to Equations of Motion

The nine differential ***G*** equations above fully describe the dynamics of the mathematical model of the sphere and pendulum system rolling on flat ground. Due to their complexity and non-linearity, no explicit analytical solution to this system of equations is even remotely possible. Using computer aided tools, approximate numerical solutions can be calculated by simultaneous numerical integration of the differential equations. In order to perform the numerical integration, the equations had to be manipulated into a different form. The 9 ***G*** equations are in fact a system of differential algebraic equations (DAEs), since they are formed from a combination of implicitly formulated algebraic and differential equations [18]. These DAEs need to be converted to a system of explicit ordinary different equations (ODEs) in order to solve them numerically.

First, the Lagrange multipliers, λ_1 and λ_2, were eliminated since their values were not of importance. Equations G_4 and G_5 were solved for λ_1 and λ_2 respectively, and then substituted into the remaining equations. These 7 new equations are now referred to as G_1', G_2', G_3', G_6', G_7', G_8', and G_9' respectively.

4.3.1 Reduction to a System of 1st Order Equations

The numerical integration tools in MATLAB only work with systems of 1st order differential equations. The seven 2nd order $\boldsymbol{G'}$ equations were reduced to a set of 14, 1st order equations by choosing and substituting in a new set of coordinates while adding 7 simple, new equations as shown in (4.31). The new set of 14 equations is now referred to as equations $\boldsymbol{H_1}$ through $\boldsymbol{H_{14}}$.

$$
\begin{array}{lll}
m_1 := q_1 & \dot{m}_1 := \dot{q}_1 & H_1 : G_1' \\
m_2 := q_2 & \dot{m}_2 := \dot{q}_2 & H_2 : G_2' \\
m_3 := q_3 & \dot{m}_3 := \dot{q}_3 & H_3 : G_3' \\
m_4 := q_4 & \dot{m}_4 := \dot{q}_4 & H_4 : G_6' \\
m_5 := q_5 & \dot{m}_5 := \dot{q}_5 & H_5 : G_7' \\
m_6 := q_6 & \dot{m}_6 := \dot{q}_6 & H_6 : G_8' \\
m_7 := q_7 & \dot{m}_7 := \dot{q}_7 & H_7 : G_9' \\
m_8 := \dot{q}_1 & \dot{m}_8 := \ddot{q}_1 & H_8 : \dot{m}_1 = m_8 \\
m_9 := \dot{q}_2 & \dot{m}_9 := \ddot{q}_2 & H_9 : \dot{m}_2 = m_9 \\
m_{10} := \dot{q}_3 & \dot{m}_{10} := \ddot{q}_3 & H_{10} : \dot{m}_3 = m_{10} \\
m_{11} := \dot{q}_4 & \dot{m}_{11} := \ddot{q}_4 & H_{11} : \dot{m}_4 = m_{11} \\
m_{12} := \dot{q}_5 & \dot{m}_{12} := \ddot{q}_5 & H_{12} : \dot{m}_5 = m_{12} \\
m_{13} := \dot{q}_6 & \dot{m}_{13} := \ddot{q}_6 & H_{13} : \dot{m}_6 = m_{13} \\
m_{14} := \dot{q}_7 & \dot{m}_{14} := \ddot{q}_7 & H_{14} : \dot{m}_7 = m_{14}
\end{array}
\tag{4.31}
$$

4.3.2 Equation Sorting

The 14 differential equations need to be “horizontally sorted” such that each equation solves for one of the 14 first order derivates, $\dot{m}_i$. They also need to be “vertically sorted” such that each subsequent equation only involves known variables or variables that have been solved for in antecedent equations. This procedure is also referred to as making the equations causal [18]. Arranged in this way, the value of each of the 14 velocity (slope) terms, $\dot{m}_i$, can be computed given a set of initial position terms,

m_i. The iterative numerical integration routine uses the slope terms to extrapolate the next position terms over a particular time step.

With 14 equations in 14 unknowns, it is useful to visualize the structure of the system graphically. Figure 4-4 shows which knowns and unknowns are present in each of the equations, and is a variation of what is called a *structure incidence matrix* [18].

EQ	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8 $\dot{m}_1$	m_9 $\dot{m}_2$	m_{10} $\dot{m}_3$	m_{11} $\dot{m}_4$	m_{12} $\dot{m}_5$	m_{13} $\dot{m}_6$	m_{14} $\dot{m}_7$	$\dot{m}_8$	$\dot{m}_9$	$\dot{m}_{10}$	$\dot{m}_{11}$	$\dot{m}_{12}$	$\dot{m}_{13}$	$\dot{m}_{14}$	#
H_1	■	■	■			■	■	■	■	■			■	■	■	■	■	■	■	■	■	
H_2	■	■	■			■	■	■	■	■			■	■	■	■	■	■	■	■	■	
H_3	■	■	■			■	■	■	■	■			■	■	■	■	■	■	■	■	■	
H_4	■	■	■			■	■	■	■	■			■	■	■	■	■	■	■	■	■	
H_5	■	■	■			■	■	■	■	■			■	■	■	■	■	■	■	■	■	
H_6	■	■							■	■	■											
H_7	■	■							■	■		■										
H_8								■														
H_9									■													
H_{10}										■												
H_{11}											■											
H_{12}												■										
H_{13}													■									
H_{14}														■								

Figure 4-4: Structure Incidence Matrix 1.

In the figure, each row corresponds to one of the 14 ***H*** equations and each column corresponds to a known or unknown variable. The middle columns have doubled up variables since they are equivalent as shown in (4.31). A shaded square indicates that a particular variable is present in a particular equation. The right hand **#** column is used to assign a new sort order.

A straightforward procedure called the Tarjan algorithm can be applied to this system to simultaneously sort the equations horizontally and vertically. The Tarjan algorithm was originally developed based on graph theory using a *structure digraph*, but can easily be applied to a structure incidence matrix as well. For the present system, the

structure incidence matrix in Figure 4-4 was used because it was found to be easier to visualize than a structure digraph. As stated above, the m_i variables are considered to be initial conditions and are therefore known. Known variables in the structure incidence matrix are indicated by an **O**, as shown in Figure 4-5.

EQ	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8 $\dot{m}_1$	m_9 $\dot{m}_2$	m_{10} $\dot{m}_3$	m_{11} $\dot{m}_4$	m_{12} $\dot{m}_5$	m_{13} $\dot{m}_6$	m_{14} $\dot{m}_7$	$\dot{m}_8$	$\dot{m}_9$	$\dot{m}_{10}$	$\dot{m}_{11}$	$\dot{m}_{12}$	$\dot{m}_{13}$	$\dot{m}_{14}$	#
H_1	**O**	**O**	**O**			**O**	**O**															
H_2	**O**	**O**	**O**			**O**	**O**															
H_3	**O**	**O**	**O**			**O**	**O**															
H_4	**O**	**O**	**O**			**O**	**O**															
H_5	**O**	**O**	**O**			**O**	**O**															
H_6	**O**	**O**																				
H_7	**O**	**O**																				
H_8																						
H_9																						
H_{10}																						
H_{11}																						
H_{12}																						
H_{13}																						
H_{14}																						

Figure 4-5: Structure Incidence Matrix 2. Known variables indicated by O.

The Tarjan algorithm can now be applied. The rules for the algorithm as applied to the present system are as follows [18]:

1. If an equation contains only 1 unknown, that unknown must be solved for using that equation. The variable is marked with an **S** in that equation, and marked with an **O** in every other equation in which it is appears. Renumber the equation in the right column with increasing numbers starting with 1.

2. If a particular unknown is only present in a single equation, that equation must be used to solve for that unknown. The variable is marked with an **S** in that equation, and all other variables in that equation are marked with an **O**.

Renumber the equation in the right column with decreasing numbers starting with the total number of equations, in this case 14.

After applying the algorithm once, we arrive at the structural incidence matrix shown in Figure 4-6.

EQ	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8 $\dot{m}_1$	m_9 $\dot{m}_2$	m_{10} $\dot{m}_3$	m_{11} $\dot{m}_4$	m_{12} $\dot{m}_5$	m_{13} $\dot{m}_6$	m_{14} $\dot{m}_7$	$\dot{m}_8$	$\dot{m}_9$	$\dot{m}_{10}$	$\dot{m}_{11}$	$\dot{m}_{12}$	$\dot{m}_{13}$	$\dot{m}_{14}$	#
H_1	O	O	O			O	O	O	O	O			O	O								
H_2	O	O	O			O	O	O	O	O			O	O								
H_3	O	O	O			O	O	O	O	O			O	O								
H_4	O	O	O			O	O	O	O	O			O	O								
H_5	O	O	O			O	O	O	O	O			O	O								
H_6	O	O							O	O	O											
H_7	O	O							O	O		O										
H_8								S														1
H_9									S													2
H_{10}										S												3
H_{11}											S											4
H_{12}												S										5
H_{13}													S									6
H_{14}														S								7

Figure 4-6: Structure Incidence Matrix 3. The result after applying the Tarjan algorithm once.

Normally the Tarjan algorithm would be applied recursively until all variables have been solved for. Unfortunately in the present system, the algorithm is now stuck, because of structural singularities and algebraic loops.

The variables in equations $\boldsymbol{H_6}$ and $\boldsymbol{H_7}$ in Figure 4-6 are now all known, but none are solved for in those equations. These equations are thus called constraint equations, and if traced back to the original formulation of the equations of motion, they are the equations defining the no-slip rolling constraint. Another algorithm called the Pantelides algorithm can be applied to eliminate the structural similarities as follows [18]:

1. If a constraint equation is found, it must be differentiated.

2. This new differentiated constraint equation is added to the existing set of equations.

3. The system now has more equations than unknowns. To equalize the number of equations and unknowns, one of the other equations that solves for a variable that is present in the constraint equation is eliminated.

When the Pantelides algorithm is applied to the present system, equations $\boldsymbol{H_6}$ and $\boldsymbol{H_7}$ are differentiated to form the additional equations $\boldsymbol{H_{15}}$ and $\boldsymbol{H_{16}}$. To equalize the number of equations and unknowns, equations $\boldsymbol{H_{11}}$ and $\boldsymbol{H_{12}}$ were chosen to be eliminated as they had the least impact on the other equations. The variables that were solved for in equations $\boldsymbol{H_{11}}$ and $\boldsymbol{H_{12}}$ are now solved for by the constraint equations $\boldsymbol{H_6}$ and $\boldsymbol{H_7}$. Equations $\boldsymbol{H_6}$ and $\boldsymbol{H_7}$ now take on the sorting numbers that were originally assigned to equations $\boldsymbol{H_{11}}$ and $\boldsymbol{H_{12}}$, and the variables in equations $\boldsymbol{H_{15}}$ and $\boldsymbol{H_{16}}$ are marked as appropriate by the Tarjan algorithm. The result of these steps is shown in Figure 4-7.

EQ	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8 $\dot{m}_1$	m_9 $\dot{m}_2$	m_{10} $\dot{m}_3$	m_{11} $\dot{m}_4$	m_{12} $\dot{m}_5$	m_{13} $\dot{m}_6$	m_{14} $\dot{m}_7$	$\dot{m}_8$	$\dot{m}_9$	$\dot{m}_{10}$	$\dot{m}_{11}$	$\dot{m}_{12}$	$\dot{m}_{13}$	$\dot{m}_{14}$	#
H_1	O	O	O			O	O	O	O	O			O	O								
H_2	O	O	O			O	O	O	O	O			O	O								
H_3	O	O	O			O	O	O	O	O			O	O								
H_4	O	O	O			O	O	O	O	O			O	O								
H_5	O	O	O			O	O	O	O	O			O	O								
H_6	O	O							O	O	S											4
H_7	O	O							O	O		S										5
H_8								S														1
H_9									S													2
H_{10}										S												3
H_{11}																						
H_{12}																						
H_{13}													S									6
H_{14}														S								7
H_{15}	O	O						O	O	O	O											
H_{16}	O	O						O	O	O		O										

Figure 4-7: Structure Incidence Matrix 4. The result after applying the Pantelides algorithm.

The Tarjan algorithm would normally be applied again at this point. Unfortunately the algorithm is still stuck due to a large algebraic loop. Each of the remaining 7 equations contains more than 1 unknown; therefore, none of the equations can be chosen to solve for a particular unknown without violating causality.

4.3.3 Reduction to system of linear algebraic equations

The remaining 7 unknowns are the variables $\dot{m}_8$ through $\dot{m}_{14}$ which are independent and not differentially related. Therefore the remaining set of 7 equations is simply a set of algebraic equations. By inspection of the equations, they were also determined to be linear in the 7 variables. The simultaneous solution to a set of linear algebraic equations is well known and can be found by a variety of different methods.

Because the finite precision inherent to numerical computation tends to cause the solution to lose accuracy or become numerically unstable when the system is ill-conditioned, minimizing the number of computations required is preferable in an effort to minimize error propagation. In that vein, Gaussian elimination was used to compute solutions to this system, because it does not require the intermediate computation of a matrix inverse. The Gaussian elimination was performed in *MATLAB* using the `/` operator (`mldivide`) and was found to run faster than when computing solutions using the matrix inverse. However, the numerical accuracy of the two methods was not rigorously investigated, and numerical instability was found to be unavoidable when the model was very ill-conditioned.

4.3.4 Numerical integration

The final order for solving the equations is as follows,

1. Solve equation $\boldsymbol{H_8}$ for $\dot{m}_1$.
2. Solve equation $\boldsymbol{H_9}$ for $\dot{m}_2$.
3. Solve equation $\boldsymbol{H_{10}}$ for $\dot{m}_3$.
4. Solve equation $\boldsymbol{H_6}$ for $\dot{m}_4$.
5. Solve equation $\boldsymbol{H_7}$ for $\dot{m}_5$.
6. Solve equation $\boldsymbol{H_{13}}$ for $\dot{m}_6$.
7. Solve equation $\boldsymbol{H_{14}}$ for $\dot{m}_7$.
8. Simultaneously solve equations $\boldsymbol{H_1}$, $\boldsymbol{H_2}$, $\boldsymbol{H_3}$, $\boldsymbol{H_4}$, $\boldsymbol{H_5}$, $\boldsymbol{H_{15}}$, and $\boldsymbol{H_{16}}$ for $\dot{m}_8$ through $\dot{m}_{14}$.

MATLAB has built in functions for numerically integrating systems of 1st order ordinary differential equations over a period of time. The function, `ode45`, was chosen which uses a variable time step and higher order approximations. A function called *numerical_integration* was written to call the ODE solver on the system of 14 equations. The function can be found in Appendix C.

Chapter 5

Simulation

The system of equations describing the motion of the sphere and pendulum can now be numerically integrated, given a set of initial conditions, to produce a time series of the position and velocity for each of the 7 generalized coordinates. 2D plots can be generated to visualize the output, but it was found to be extremely difficult to interpret the results when the motion was any more complex than the most simple of cases. To effectively visualize the motion of the sphere and pendulum a 3D animation is the by far the best tool.

Several tools exist for animating 3D objects using the data output by the ODE solver, including add-on toolboxes for *MATLAB*. These tools were unavailable at the time of this research, so a program was written using built in *MATLAB* plotting tools instead. The source code of this program, named *Simulation*, can be found in Appendix D.

5.1 Simulation Program

The simulation program utilizes *MATLAB*'s built in 3D plotting tools to generate a plot of a sphere and a pendulum, and re-plot the position of both components at each

time step. The built in commands, `sphere` and `cylinder`, produce sets of data points that define their surfaces. The transformation matrices defined in Section 4.1 are used to transform these surfaces to the position defined by the generalized coordinates at a particular time step (see Figure 5-1).

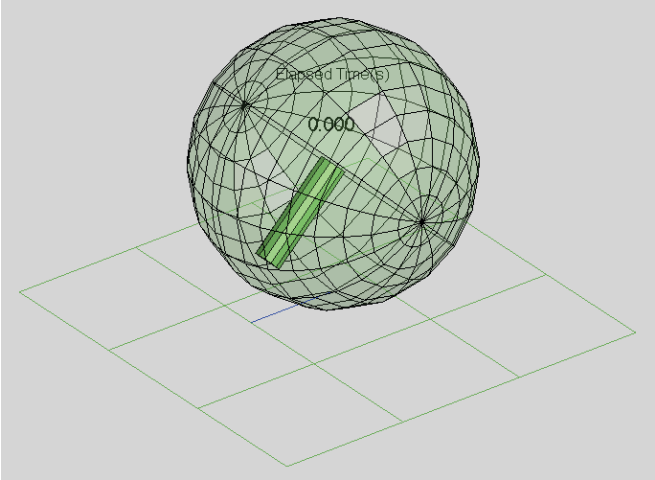


Figure 5-1: 3D plot of sphere and pendulum.

The simulation takes the time and position data output from the ODE solver, as well as a chosen frame rate. The motion data is first reprocessed using cubic spline interpolation to produce position data for every frame specified by the frame rate and total simulation duration. Each frame is sequentially plotted and saved to memory. After every frame has been produced, a video file is generated and saved to disk.

5.2 Model Parameters and Initial Conditions

When solving the equations and running the simulation, several parameters can be specified: The initial position and velocity of every degree of freedom of the system, the masses of the sphere and pendulum, the principle moments of inertia of both the sphere and pendulum, the value for the acceleration due to gravity, the drive and steering motor torques (only constants are allowed), the distance from the center of mass of the

pendulum to the center of mass of the sphere, the radius of the sphere, the four damping coefficients included in the model in Section 4.2.4, and the duration of the simulation.

The flexibility in defining the initial conditions and design parameters allows for easy experimentation with different designs and scenarios. Because the motion of the prototype is of utmost interest, the simulations in this research are for a model of that particular design.

5.3 Specific Cases

The original prototype lacked the instrumentation necessary to effectively compare its dynamics to the results of the computer model, and the effort and cost to design and construct a fully instrumented model to validate the results of the above analysis was clearly beyond the scope of this master's thesis. As an alternative, some simulation tests were carried out that could be verified mathematically or qualitatively.

5.3.1 Pendulum Period

The first test that was performed was a simple pendulum test. The initial conditions were to set the sphere and pendulum stationary with the pendulum rotated up 10° in front. The mass of the sphere was set to a very large number to simulate it being held fixed. The motor torques and damping coefficients were all set to zero. When the simulation was run, the pendulum oscillated back and forth as expected. A plot of the pendulum angle vs. time is shown in Figure 5-2.

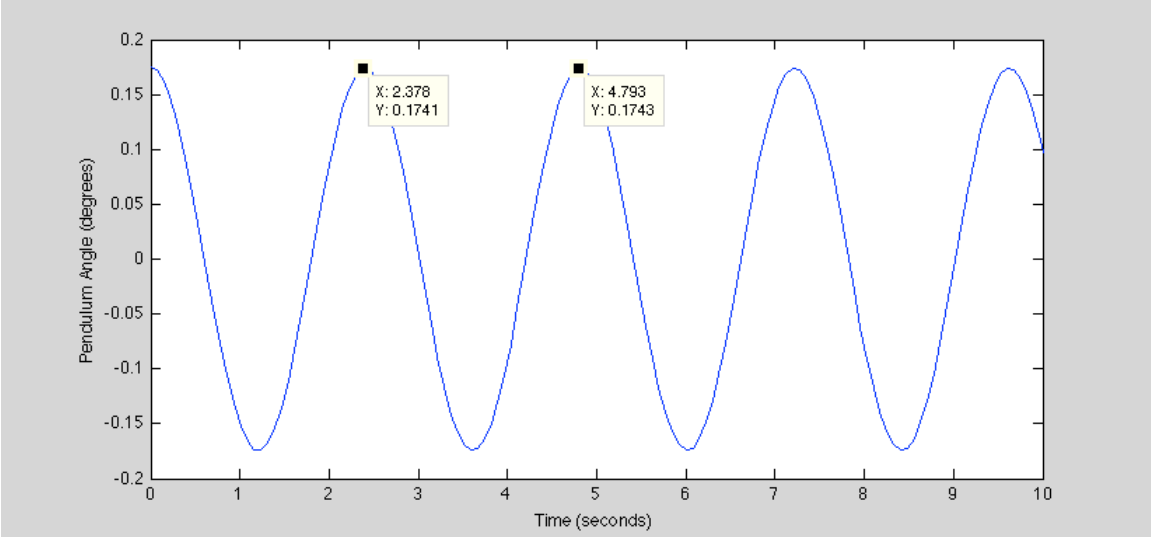


Figure 5-2: Pendulum oscillation period test. Period is 2.41 seconds.

The plot of the pendulum angle shows a period of about 2.41 seconds. The equation for the period of the physical pendulum is as follows,

$$T = 2\pi\sqrt{\frac{I_{cm} + mr^2}{mgr}} \tag{5.1}$$

where $\boldsymbol{T}$ is the oscillation period, $\boldsymbol{I_{cm}}$ is the moment of inertia of the pendulum about its center of mass, $\boldsymbol{m}$ is the mass of the pendulum, $\boldsymbol{r}$ is the distance from the center of mass of the pendulum to the rotation axis, and $\boldsymbol{g}$ is the acceleration due to gravity. The pendulum design used in the simulation had the following specifications: $\boldsymbol{I_{cm}}$ = 0.15 kg*m^2, $\boldsymbol{m}$ = 15 kg, $\boldsymbol{r}$ = 0.1 m, and $\boldsymbol{g}$ = 9.81 m/s^2. Using these same parameters in Equation (3.11), $\boldsymbol{T}$ was calculated to be 2.40 seconds, putting it in close agreement with the numerical simulation.

5.3.2 Energy Conservation

The law of conservation of energy was tested with the model. For this test the non-conservative forces and moments were again set to zero, but the mass of the sphere

was set to the same mass as the spherical shell from the prototype. The initial conditions for the simulation were set such that they would produce complex motion. Figure 5-3 is a snapshot from the simulation showing the trajectory of the sphere.

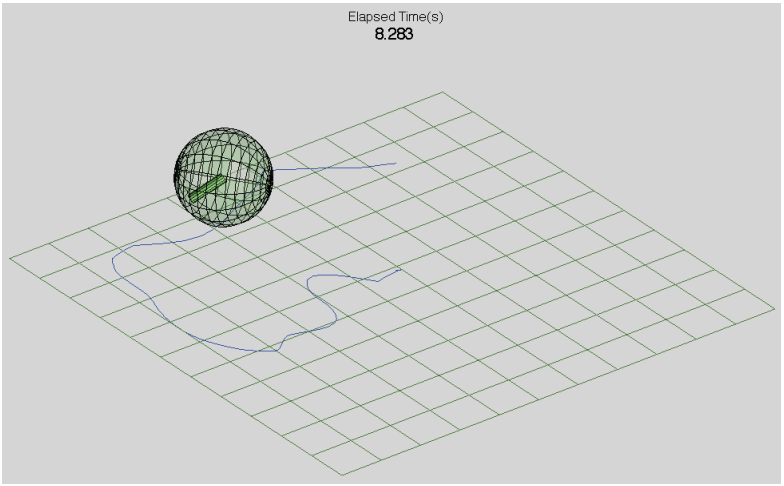


Figure 5-3: Complex motion with no damping or motor torques.

The total system energy was computed at every time step of the simulation using the equations for kinetic and potential energy shown in Section 4.2.1. A plot of the total system energy vs. time is shown in Figure 5-4. As shown, the energy remained fairly constant during the simulation. The slight variation from what was expected to be a horizontal line is attributed to error propagation in the numerical integration of the equations. During the computation, the model occasionally approaches configurations that are somewhat poorly conditioned, which also tends to increase the error in the integration.

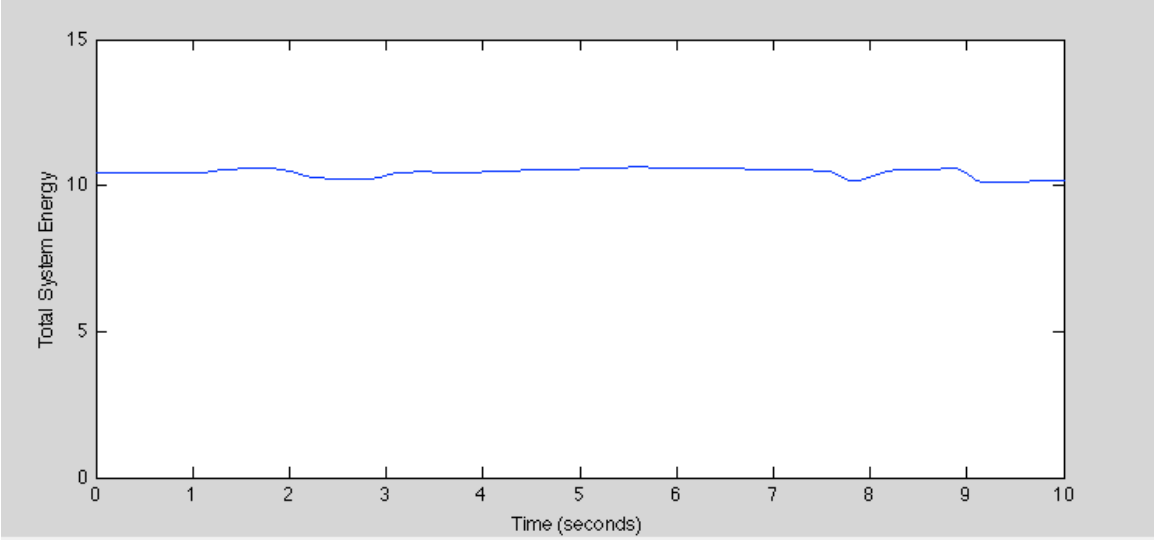


Figure 5-4: Total system energy during the simulation.

5.3.3 Reproduction of oscillatory and unstable behaviors

The tests shown above, while not completely rigorous, do demonstrate that the model behaves as expected. The topic of most interest however, was whether the oscillatory and sometimes unstable behaviors that were observed in the prototype as discussed in Section 3.3, would become apparent in the computer simulation.

The tendency for the sphere to wobble was very much expected and even indicated by the pendulum test in Section 5.3.1. To verify this behavior, the system was modeled by first fixing the pendulum within the sphere and perturbing the sphere from rest. A small initial rolling velocity simulated the sphere being perturbed. Snapshots of the simulation showing the sphere rocking back and forth about a spot on the ground are shown in Figure 5-5. The model behaved as expected.

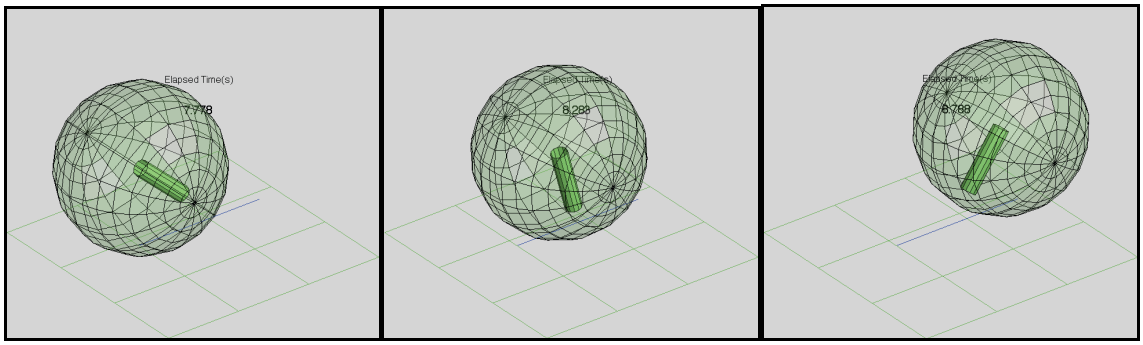


Figure 5-5: Sphere wobbling back and forth when perturbed.

When attempting to steer with the prototype it was found to wobble into and out of its curved trajectory as it traveled. A simulation was set up to test this scenario as well. By initially tilting the pendulum sideways a small amount and applying drive torque, the simulation behaved in very much the same way as the prototype (see Figure 5-6).

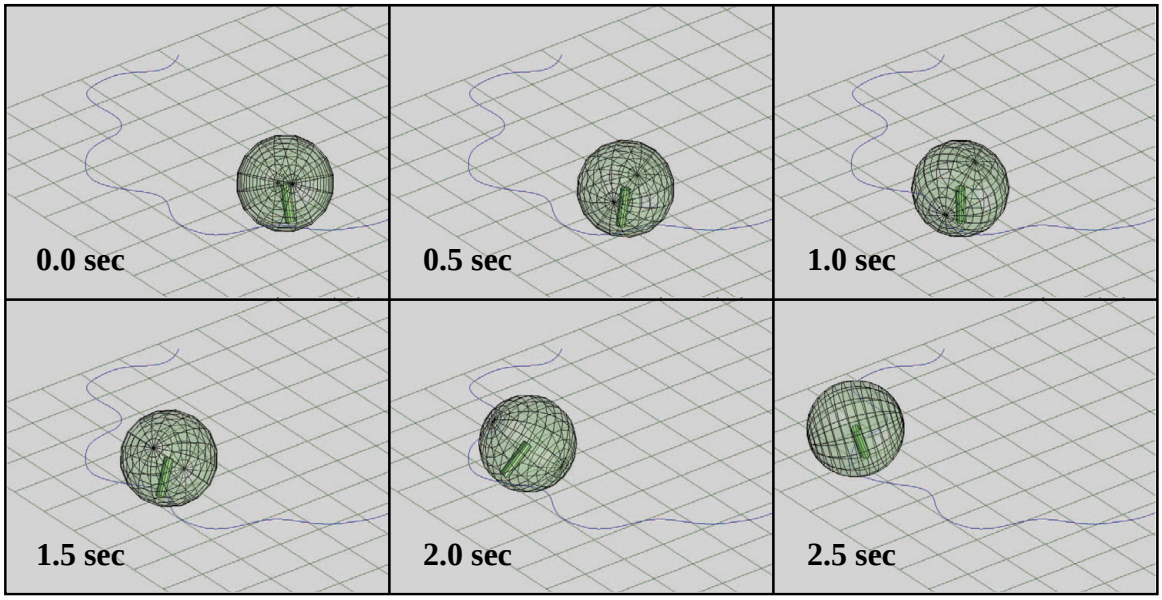


Figure 5-6: Sphere wobbling in and out of a turn.

Perhaps the most critical behavior displayed by the prototype was its shaft nutation and instability causing it to lose control. While Figure 5-6 showed wobbling while turning, upon closer inspection it also indicated that the drive shaft nutates some as

well. It appears that even though a drive torque is being applied about the primary drive axis, the sphere is rolling about an axis displaced some angle away from the primary drive axis.

When testing the sphere prototype, this nutation behavior would go unstable when the sphere was driven at a high speed (above approximately 3 m/s). Another simulation was set up to test this scenario. The initial conditions were set to have the sphere begin at rest but tilted to the side by a very small angle (1 degree). A significant drive torque was then applied to accelerate the sphere forward. As shown in Figure 5-7, the sphere travels forward for about 1 second before beginning to display the nutation behavior. Between 1.0 and 2.5 seconds its motion becomes unstable and the primary drive axis flips end over end. The results of these tests and simulations successfully confirm that the model accurately characterizes the dynamics of the physical prototype.

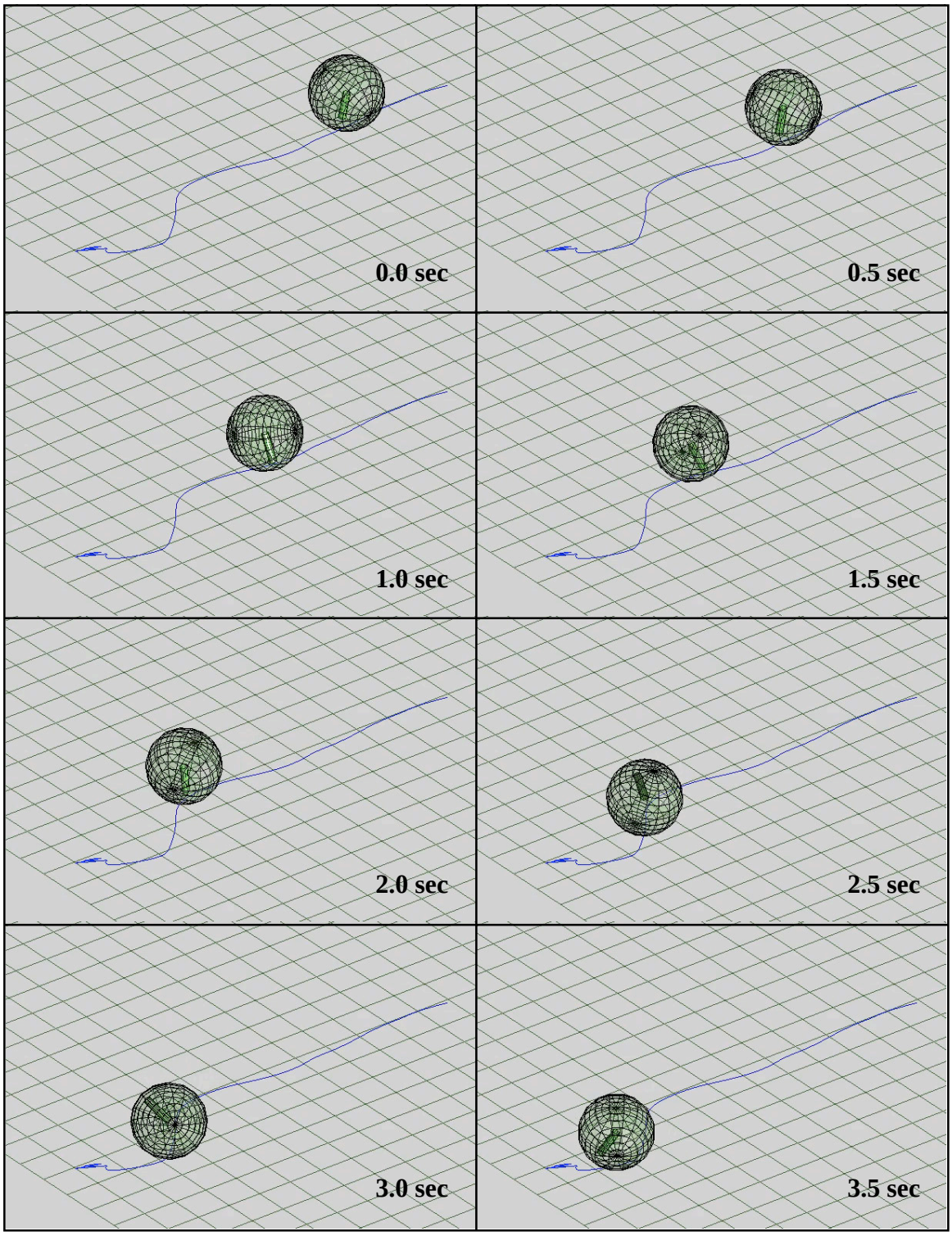


Figure 5-7: Nutation instability.

Chapter 6

Future Work

The dynamic model developed during this research only dealt with motion on flat ground. The clear next step is to expand this model to the more general case of motion over irregular terrain. Now that the mathematical machinery has been worked out, the novel CMG mechanism developed previously needs to be added to the model as well. The enhanced torque capabilities afforded by the CMGs will allow for far more agile maneuvers than was possible with previous spherical robot designs. Once the dynamics of the complete system are modeled, controls need to be developed that can effectively damp out the undesirable oscillatory and sometimes unstable behavior inherent in these types of mobile robots. The equations of motion of this simplified model were found to be very complex and non-linear, which will make the controls development exceedingly challenging.

Additional prototypes need to be built that are accurately instrumented and have sufficient processing power to allow for controls of varying complexity to be implemented. Recent advances in the smartphone industry show that tiny, energy efficient sensors and processors are in rapid development and many components are perfect for developing advanced robotics platforms.

Ultimately, all of these challenges must be overcome before spherical robots will be successful in the field.

Chapter 7

Conclusions

For more than 100 years, experimenters have been developing spherical vehicles and robots, but despite many beneficial features, limits in their mobility have prevented them from becoming much more than interesting research. In order for spherical robots to perform in the field, their mobility needs to be improved and their undesirable dynamics need to be controlled. The capability of manipulating angular momentum internally, has now been shown to significantly enhance the mobility of spherical robots, but adds complexity to the already challenging system. Development of the full equations of motion of a pendulum driven sphere rolling on flat ground is a first step in understanding the dynamics of these machines. It is hoped that more research effort will be directed toward the development of spherical robots in order to overcome the difficult control issues associated with them so that their significant advantages can be fully realized in a wide range of practical applications.

References

[1] Schroll, G. “Design of a Spherical Vehicle with Flywheel Momentum Storage for High Torque Capabilities.” B.S. Thesis, Massachusetts Institute of Technology, 2008.

[2] Ylikorpi, T. & Suomela, J. “Ball-shaped Robots.” *Climbing and Walking Robots Towards New Applications*. I-Tech Education and Publishing, Vienna, Austria, October, 2007, pp. 235-256.

[3] Halme, A.; Schönberg, T. & Wang, Y. “Motion Control of a Spherical Mobile Robot,” *4th IEEE International Workshop on Advanced Motion Control AMC’96,* Mie University, Japan, 1996, pp. 100-106.

[4] “SAR.” June 28, 2006. YouTube. Web. February 5, 2010.

[5] Chemel, B.; Mutschler, E. & Schempf, H. “Cyclops: miniature robotic reconnaissance system,” *IEEE Int. Conf. on Robotics and Automation (ICRA '99)*, May, 1999, pp. 2298-2302.

[6] Rotundus. Web. February 10, 2010. http://www.rotundus.se.

[7] Laplante, J.-F.; Masson, P. & Michaud, F. “Analytical longitudinal and lateral models of a spherical rolling robot,” Technical Report, Department of Electrical Engineering and Computer Engineering. 2007.

[8] Nagai, M. “Control System for a Spherical Robot.” M.S. Thesis, Luleå University of Technology, 2008.

[9] Mukherjee, R. “Spherical Mobile Robot.” U.S. Patent 6,289,263. September 11, 2001.

[10] Mukherjee, R.; Minor, M. & Pukrushpan, J. “Simple motion planning strategies for Spherobot: a spherical mobile robot,” *Proc. of The 38th Conference on Decision & Control*, Phoenix, AZ, December, 1999.

[11] Mukherjee, R. “Design Challenges in the Development of a Spherical Mobile Robot.” *Robotic Ball Technology Study for Planetary Surface Missions.* NASA JSC/EV George Studor. May 11, 2010.

[12] Javadi, A. H. A. & Mojabi, P. “Introducing August: A novel strategy for an omnidirectional spherical rolling robot.” Proceedings of the IEEE International Conference on Robotics & Automation, Washington, DC, 2002, pp. 3527-3533.

[13] Sugiyama, Y. & Hirai, S. “Crawling and Jumping by a Deformable Robot,” *International Journal of Robotics Research*, Vol. 25, No.5-6, pp. 603-620, 2006.

[14] Caffrey, E. “iRobot to Create Revolutionary New Robot for DARPA.” iRobot Corporation. June 17, 2008. Web. April 10, 2010. http://www.irobot.com/sp.cfm?pageid=86&id=400

[15] “iRobot's Soft Morphing Blob ‘Bot Takes Its First Steps.” October 13, 2009. YouTube. Web. December 10, 2009.

[16] Weisstein, Eric W. "Spherical Cap." Wolfram MathWorld. Web. May 10, 2010. http://mathworld.wolfram.com/SphericalCap.html

[17] Greenwood, D. *Principles of dynamics*. 2nd ed. Prentice Hall. Upper Saddle River, NJ, 1988. Print.

[18] Cellier, F. & Kofman, E. *Continuous System Simulation*. Springer. New York, NY, 2006. Print.

Appendix A

Example of Equation Complexity

The following equation (set the expression = 0) is the first Lagrange equation associated with the generalized coordinate ψ. It was symbolically simplified using the *MATLAB* `simplify` command. To read the equation use the following substitution:

`s`: ψ
`t`: θ
`p`: ϕ
`x`: x
`y`: y
`a`: α
`b`: β
`D*`: first derivative
`D2*`: second derivative

`I**s`: I_{**s}
`I**p`: I_{**p}

```
Ds*f4 + D2s*Ixxp + D2s*Ixxs - D2s*Ixxp*cos(t)^2 - D2s*Ixxs*cos(t)^2 + D2s*Iyyp*cos(t)^2 +
D2s*Iyys*cos(t)^2 + D2s*Mp*d^2 - D2p*Ixxp*sin(t) - D2p*Ixxs*sin(t) - Dp*f4*sin(t) -
Dp*Dt*Ixxp*cos(t) - Dp*Dt*Ixxs*cos(t) - Dp*Dt*Iyyp*cos(t) - Dp*Dt*Iyys*cos(t) +
Dp*Dt*Izzp*cos(t) + Dp*Dt*Izzs*cos(t) - D2s*Iyyp*cos(p)^2*cos(t)^2 -
D2s*Iyys*cos(p)^2*cos(t)^2 + D2s*Izzp*cos(p)^2*cos(t)^2 + D2s*Izzs*cos(p)^2*cos(t)^2 -
D2b*Ixxp*cos(a)*sin(t) - D2p*Mp*d^2*sin(t) + D2a*Iyyp*cos(t)*sin(p) - D2s*Mp*d^2*cos(b)^2
- D2s*Mp*d^2*cos(t)^2 + 2*D2s*Mp*d^2*cos(b)^2*cos(t)^2 + D2s*Mp*d^2*cos(p)^2*cos(t)^2 +
2*Dp*Dt*Iyyp*cos(p)^2*cos(t) + 2*Dp*Dt*Iyys*cos(p)^2*cos(t) -
2*Dp*Dt*Izzp*cos(p)^2*cos(t) - 2*Dp*Dt*Izzs*cos(p)^2*cos(t) - D2b*Mp*d^2*cos(a)*sin(t) -
D2b*Izzp*cos(p)*sin(a)*cos(t) + D2t*Iyyp*cos(p)*cos(t)*sin(p) +
D2t*Iyys*cos(p)*cos(t)*sin(p) - D2t*Izzp*cos(p)*cos(t)*sin(p) -
D2t*Izzs*cos(p)*cos(t)*sin(p) + D2p*Mp*d^2*cos(b)^2*sin(t) - Db*Dt*Ixxp*cos(a)*cos(t) +
Da*Dp*Iyyp*cos(p)*cos(t) - Dt^2*Iyyp*cos(p)*sin(p)*sin(t) -
Dt^2*Iyys*cos(p)*sin(p)*sin(t) + Dt^2*Izzp*cos(p)*sin(p)*sin(t) +
Dt^2*Izzs*cos(p)*sin(p)*sin(t) + Da*Db*Ixxp*sin(a)*sin(t) + 2*Ds*Dt*Ixxp*cos(t)*sin(t) +
2*Ds*Dt*Ixxs*cos(t)*sin(t) - 2*Ds*Dt*Iyyp*cos(t)*sin(t) - 2*Ds*Dt*Iyys*cos(t)*sin(t) -
Da*Dt*Iyyp*sin(p)*sin(t) + D2s*Mp*d^2*cos(a)^2*cos(b)^2 - 2*Dp*Dt*Mp*d^2*cos(p)^2*cos(t)
+ 2*Dp*Ds*Iyyp*cos(p)*cos(t)^2*sin(p) + 2*Dp*Ds*Iyys*cos(p)*cos(t)^2*sin(p) -
2*Dp*Ds*Izzp*cos(p)*cos(t)^2*sin(p) - 2*Dp*Ds*Izzs*cos(p)*cos(t)^2*sin(p) +
2*Ds*Dt*Iyyp*cos(p)^2*cos(t)*sin(t) + 2*Ds*Dt*Iyys*cos(p)^2*cos(t)*sin(t) -
2*Ds*Dt*Izzp*cos(p)^2*cos(t)*sin(t) - 2*Ds*Dt*Izzs*cos(p)^2*cos(t)*sin(t) -
D2b*Mp*d^2*cos(p)*sin(a)*cos(t) - D2s*Mp*d^2*cos(a)^2*cos(b)^2*cos(t)^2 -
D2t*Mp*d^2*cos(p)*cos(t)*sin(p) - D2s*Mp*d^2*cos(b)^2*cos(p)^2*cos(t)^2 +
2*Db*Ds*Mp*d^2*cos(b)*sin(b) + D2a*Mp*d^2*cos(b)^2*cos(t)*sin(p) -
Da*Db*Izzp*cos(a)*cos(p)*cos(t) + Dt^2*Mp*d^2*cos(p)*sin(p)*sin(t) +
2*Da*Db*Mp*d^2*sin(a)*sin(t) + 2*Ds*Dt*Mp*d^2*cos(t)*sin(t) +
Db*Dp*Izzp*sin(a)*cos(t)*sin(p) + Db*Dt*Izzp*cos(p)*sin(a)*sin(t) -
D2x*Mp*d*cos(p)*sin(b)*cos(s) - D2y*Mp*d*cos(p)*sin(b)*sin(s) -
D2p*Mp*d^2*cos(a)^2*cos(b)^2*sin(t) - D2a*Mp*d^2*cos(b)*sin(a)*sin(b)*sin(t) -
2*Dp*Dt*Mp*d^2*cos(a)^2*cos(b)^2*cos(t) + 2*Dp*Dt*Mp*d^2*cos(b)^2*cos(p)^2*cos(t) -
Da^2*Mp*d^2*cos(a)*cos(b)*sin(b)*sin(t) + Dt^2*Mp*d^2*cos(a)*cos(b)*sin(b)*sin(t) -
D2s*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)^2*cos(t)^2 + D2t*Mp*d^2*cos(b)^2*cos(p)*cos(t)*sin(p)
- 2*Da*Db*Mp*d^2*cos(a)*cos(p)*cos(t) - 2*Db*Dp*Mp*d^2*cos(b)*sin(b)*sin(t) +
2*Db*Dt*Mp*d^2*cos(p)*sin(a)*sin(t) - D2x*Mp*d*cos(a)*cos(b)*cos(s)*sin(p) -
D2y*Mp*d*cos(b)*sin(a)*cos(s)*cos(t) - D2y*Mp*d*cos(a)*cos(b)*sin(p)*sin(s) +
D2x*Mp*d*cos(b)*sin(a)*cos(t)*sin(s) - 2*Da*Ds*Mp*d^2*cos(a)*cos(b)^2*sin(a) -
2*Db*Ds*Mp*d^2*cos(a)^2*cos(b)*sin(b) - 2*Db*Dt*Mp*d^2*cos(a)*cos(b)^2*cos(t) -
2*Db*Dt*Mp*d^2*cos(a)*cos(p)^2*cos(t) + 2*Da*Dp*Mp*d^2*cos(b)^2*cos(p)*cos(t) -
4*Db*Ds*Mp*d^2*cos(b)*sin(b)*cos(t)^2 + D2y*Mp*d*sin(b)*cos(s)*sin(p)*sin(t) -
Dt^2*Mp*d^2*cos(b)^2*cos(p)*sin(p)*sin(t) - 2*Da*Db*Mp*d^2*cos(b)^2*sin(a)*sin(t) -
D2x*Mp*d*sin(b)*sin(p)*sin(s)*sin(t) - 4*Ds*Dt*Mp*d^2*cos(b)^2*cos(t)*sin(t) -
2*Dp*Ds*Mp*d^2*cos(p)*cos(t)^2*sin(p) - 2*Ds*Dt*Mp*d^2*cos(p)^2*cos(t)*sin(t) -
D2t*Mp*d^2*cos(a)*cos(b)*sin(b)*cos(t) + 2*Ds*Dt*Mp*d^2*cos(a)^2*cos(b)^2*cos(t)*sin(t) +
2*Dp*Ds*Mp*d^2*cos(b)^2*cos(p)*cos(t)^2*sin(p) -
2*Da*Dt*Mp*d^2*cos(a)^2*cos(b)^2*sin(p)*sin(t) +
2*Ds*Dt*Mp*d^2*cos(b)^2*cos(p)^2*cos(t)*sin(t) -
D2p*Mp*d^2*cos(a)*cos(b)^2*cos(p)*sin(a)*cos(t) +
2*D2t*Mp*d^2*cos(a)*cos(b)*cos(p)^2*sin(b)*cos(t) -
Da^2*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*cos(t) +
Dp^2*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*cos(t) -
Dt^2*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*cos(t) -
D2t*Mp*d^2*cos(a)*cos(b)^2*sin(a)*sin(p)*sin(t) +
2*Ds*Dt*Mp*d^2*cos(b)*sin(a)*sin(b)*sin(p) +
2*Dp*Dt*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)^2*cos(t) -
2*Da*Db*Mp*d^2*cos(b)*sin(b)*cos(t)*sin(p) + 2*Db*Ds*Mp*d^2*sin(a)*cos(t)*sin(p)*sin(t) -
D2y*Mp*d*cos(a)*cos(b)*cos(p)*cos(s)*sin(t) + D2x*Mp*d*cos(a)*cos(b)*cos(p)*sin(s)*sin(t)
- 2*Ds*Dt*Mp*d^2*cos(a)*cos(b)^2*cos(p)*sin(a) +
D2t*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)*cos(t)*sin(p) +
Dp^2*Mp*d^2*cos(a)*cos(b)^2*sin(a)*cos(t)*sin(p) -
Dt^2*Mp*d^2*cos(a)*cos(b)^2*sin(a)*cos(t)*sin(p) +
2*Da*Db*Mp*d^2*cos(a)*cos(b)^2*cos(p)*cos(t) -
2*Dt^2*Mp*d^2*cos(a)*cos(b)*cos(p)^2*sin(b)*sin(t) -
2*Dp*Ds*Mp*d^2*cos(a)*cos(b)*sin(b)*cos(t)^2 +
2*Da*Dp*Mp*d^2*cos(a)*cos(b)^2*sin(a)*sin(t) +
2*Db*Dp*Mp*d^2*cos(a)^2*cos(b)*sin(b)*sin(t) -
2*Db*Ds*Mp*d^2*cos(a)*cos(p)*cos(t)^2*sin(p) +
2*Db*Dp*Mp*d^2*cos(b)^2*sin(a)*cos(t)*sin(p) -
2*Db*Dt*Mp*d^2*cos(b)^2*cos(p)*sin(a)*sin(t) -
2*Da*Ds*Mp*d^2*cos(b)^2*cos(p)*cos(t)*sin(t) +
D2a*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(b)*cos(t) +
D2p*Mp*d^2*cos(b)*sin(a)*sin(b)*cos(t)*sin(p) -
D2t*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*sin(t) -
2*Da*Dp*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)*cos(t) +
```

```
4*Db*Dt*Mp*d^2*cos(a)*cos(b)^2*cos(p)^2*cos(t) +
2*Da*Ds*Mp*d^2*cos(a)*cos(b)^2*sin(a)*cos(t)^2 +
2*Db*Ds*Mp*d^2*cos(a)^2*cos(b)*sin(b)*cos(t)^2 -
Dt^2*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)*sin(p)*sin(t) +
2*Db*Ds*Mp*d^2*cos(b)*cos(p)^2*sin(b)*cos(t)^2 +
2*D2s*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(b)*cos(t)^2*sin(p) +
2*D2s*Mp*d^2*cos(a)*cos(b)^2*cos(p)*sin(a)*cos(t)*sin(t) -
2*Da*Dt*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(b)*sin(t) +
2*Da*Ds*Mp*d^2*cos(a)*cos(b)^2*cos(p)^2*sin(a)*cos(t)^2 +
2*Db*Ds*Mp*d^2*cos(a)^2*cos(b)*cos(p)^2*sin(b)*cos(t)^2 -
2*Db*Dt*Mp*d^2*cos(b)*cos(p)*sin(b)*cos(t)*sin(p) +
2*Dp*Ds*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)*cos(t)^2*sin(p) +
2*Ds*Dt*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)^2*cos(t)*sin(t) -
2*Da*Dt*Mp*d^2*cos(b)*cos(p)^2*sin(a)*sin(b)*cos(t) -
4*Ds*Dt*Mp*d^2*cos(b)*sin(a)*sin(b)*cos(t)^2*sin(p) -
4*Db*Ds*Mp*d^2*cos(b)^2*sin(a)*cos(t)*sin(p)*sin(t) +
4*Ds*Dt*Mp*d^2*cos(a)*cos(b)^2*cos(p)*sin(a)*cos(t)^2 +
4*Dp*Ds*Mp*d^2*cos(a)*cos(b)*cos(p)^2*sin(b)*cos(t)^2 -
2*D2s*Mp*d^2*cos(b)*sin(a)*sin(b)*cos(t)*sin(p)*sin(t) +
4*Db*Ds*Mp*d^2*cos(a)*cos(b)^2*cos(p)*cos(t)^2*sin(p) +
4*Da*Ds*Mp*d^2*cos(a)^2*cos(b)^2*cos(p)*cos(t)*sin(t) -
2*Da*Dt*Mp*d^2*cos(a)*cos(b)^2*cos(p)*sin(a)*cos(t)*sin(p) -
2*Db*Dt*Mp*d^2*cos(a)^2*cos(b)*cos(p)*sin(b)*cos(t)*sin(p) -
2*Da*Ds*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*cos(t)^2*sin(p) -
2*Dp*Ds*Mp*d^2*cos(a)*cos(b)^2*sin(a)*cos(t)*sin(p)*sin(t) +
2*Db*Dp*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(a)*sin(b)*cos(t) -
4*Dp*Dt*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(b)*cos(t)*sin(p) +
2*Db*Dt*Mp*d^2*cos(a)*cos(b)*sin(a)*sin(b)*sin(p)*sin(t) -
2*Da*Ds*Mp*d^2*cos(a)*cos(b)*sin(b)*cos(t)*sin(p)*sin(t) -
2*Dp*Ds*Mp*d^2*cos(b)*cos(p)*sin(a)*sin(b)*cos(t)*sin(t) -
4*Db*Ds*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(a)*sin(b)*cos(t)*sin(t) -
4*Ds*Dt*Mp*d^2*cos(a)*cos(b)*cos(p)*sin(b)*cos(t)*sin(p)*sin(t)
```

Appendix B

MATLAB: equations_of_motion.m

Note: Many lines in this program exceed the width of the page and have been truncated to fit.

```
clear;
% q1 = s: psi
% q2 = t: theta
% q3 = p: phi
% q4 = x: global x position of sphere center
% q5 = y: global y position of sphere center
% q6 = a: alpha
% q7 = b: beta

% d: distance from center of sphere to cg of pendulum
% *_dot: time derivative of variable
% *_ddot: second time derivative of variable

syms s t p x y a b d;

Dxys=[1  0  0  x;...
      0  1  0  y;...
      0  0  1  0;...
      0  0  0  1]; % translation transformation matrix from origin of inertial r

Rzs=[cos(s) -sin(s) 0  0;...
     sin(s)  cos(s) 0  0;...
     0       0      1  0;...
     0       0      0  1]; % rotation due to psi about z-axis of sphere frame

Rys=[cos(t) 0  sin(t) 0;...
     0      1  0      0;...
    -sin(t) 0  cos(t) 0;...
     0      0  0      1]; % rotation due to theta about y-axis of sphere frame

Rxs=[1  0       0      0;...
     0  cos(p) -sin(p) 0;...
     0  sin(p)  cos(p) 0;...
     0  0       0      1]; % rotation due to phi about x-axis of sphere frame

Ryp=[cos(a)  0  sin(a)  0;...
     0       1  0       0;...
    -sin(a)  0  cos(a)  0;...
     0       0  0       1]; % rotation due to alpha about y-axis of pendulum fra

Rxp=[1  0       0       0;...
     0  cos(b) -sin(b)  0;...
     0  sin(b)  cos(b)  0;...
     0  0       0       1]; % rotation due to beta about x-axis of pendulum fram

Pp_p=[0;...
      0;...
      -d;...
      1]; % vector from sphere frame origin to pendulum frame origin in pendulum

% {globalframe} = Rzs*Rys*Rxs*{sphereframe}
% {sphereframe} = Ryp*Rxp*{pendulumframe}
% {globalframe} = Rzs*Rys*Rxs*Ryp*Rxp*{pendulumframe}

syms Ds Dt Dp Dx Dy Da Db;

Wp_rel_p=[Db;...
          Da*cos(b);...
         -Da*sin(b);...
          1]; % angular velocity of pendulum frame relative to sphere frame (in

Wp_rel_s=Ryp*Rxp*Wp_rel_p; % angular velocity of pendulum frame relative to sphe
Wp_rel_s=Wp_rel_s(1:3);

Pp_o=Dxys*Rzs*Rys*Rxs*Ryp*Rxp*Pp_p; % location of pendulum frame in global frame

% Velocity of CG of pendulum in global frame (differentiation done by differenti
PDx = Dx - Db*d*cos(b)*cos(p)*sin(s) - Ds*d*cos(p)*sin(b)*cos(s) + Dp*d*sin(b)*s
PDy = Dy + Db*d*cos(b)*cos(p)*cos(s) - Dp*d*sin(b)*cos(s)*sin(p) - Ds*d*cos(p)*s
PDz = Da*d*cos(a)*cos(b)*sin(t) + Dt*d*cos(b)*sin(a)*cos(t) + Db*d*cos(b)*cos(t)

% Ms: mass of sphere
% Mp: mass of pendulum
```

```
% I**s: principle moment of inertia of sphere
% I**p: principle moment of inertia of pendulum
% g: acceleration due to gravity (~9.81 m/s)

syms Ms Mp Ixxs Iyys Izzs Ixxp Iyyp Izzp g r

Ws_s = [Dp - Ds*sin(t);...
        Dt*cos(p) + Ds*cos(t)*sin(p);...
        Ds*cos(t)*cos(p) - Dt*sin(p)]; % absolute angular velocity of sphere in

Wp_s = Ws_s + Wp_rel_s; % absolute angular velocity of pendulum in sphere frame
Wp_s = simplify(Wp_s);

Trot_s = 1/2*(Ixxs*Ws_s(1)^2 + Iyys*Ws_s(2)^2 + Izzs*Ws_s(3)^2); % rotational ki
Ttrans_s = 1/2*Ms*(Dx^2 + Dy^2); % translational kinetic energy of sphere
V_s = 0; % potential energy of sphere (constant for rolling on flat surface)

Trot_p = 1/2*(Ixxp*Wp_s(1)^2 + Iyyp*Wp_s(2)^2 + Izzp*Wp_s(3)^2); % rotational ki
Ttrans_p = 1/2*Mp*(PDx^2 + PDy^2 + PDz^2); % translational kinetic energy of pen
V_p = Mp*g*Pp_o(3); % potential energy of pendulum due to gravity


L = Trot_s + Ttrans_s + Trot_p + Ttrans_p - V_s - V_p; % Lagrangian



% no slip constraint
Ws_o = Rzs(1:3,1:3)*Rys(1:3,1:3)*Rxs(1:3,1:3)*Ws_s;
Ws_o = simplify(Ws_o);

% Dx*i + Dy*j = r(Ws_o(1)*i + Ws_o(2)*j + Ws_o(3)*k) (x) k
% ijk are unit vectors, (x) is cross product
K = [Dx; Dy; 0] - r*cross(Ws_o,[0; 0; 1]); % = [0; 0; 0]

% Coefficients in front of each qi_dot
a1(1) = diff(K(1),Ds);
a1(2) = diff(K(1),Dt);
a1(3) = diff(K(1),Dp);
a1(4) = diff(K(1),Dx);
a1(5) = diff(K(1),Dy);
a1(6) = diff(K(1),Da);
a1(7) = diff(K(1),Db);
a2(1) = diff(K(2),Ds);
a2(2) = diff(K(2),Dt);
a2(3) = diff(K(2),Dp);
a2(4) = diff(K(2),Dx);
a2(5) = diff(K(2),Dy);
a2(6) = diff(K(2),Da);
a2(7) = diff(K(2),Db);


% General Form of Lagrange Equations
% d/dt(dL/dqi_dot) - dL/dqi = Sum{j=1 to m}(vj*aji) + Qi
% L:   Lagrangian
% qi:  generalized coordinates
% m:   # constraint equations K (2 in this case)
% vj:  lagrange multipliers
% aji: coefficients in from of qi_dots in constraint eqns K
% Qi:  Generalized nonconservative forces/moments

dLdDs = diff(L,Ds);
dLdDt = diff(L,Dt);
dLdDp = diff(L,Dp);
dLdDx = diff(L,Dx);
dLdDy = diff(L,Dy);
dLdDa = diff(L,Da);
dLdDb = diff(L,Db);

dLds = diff(L,s);
dLdt = diff(L,t);
dLdp = diff(L,p);
dLdx = diff(L,x);
dLdy = diff(L,y);
```

```
dLda = diff(L,a);
dLdb = diff(L,b);

syms D2s D2t D2p D2x D2y D2a D2b;

dLdDs = expand(dLdDs);
dLdDt = expand(dLdDt);
dLdDp = expand(dLdDp);
dLdDx = expand(dLdDx);
dLdDy = expand(dLdDy);
dLdDa = expand(dLdDa);
dLdDb = expand(dLdDb);

dLds = expand(dLds);
dLdt = expand(dLdt);
dLdp = expand(dLdp);
dLdx = expand(dLdx);
dLdy = expand(dLdy);
dLda = expand(dLda);
dLdb = expand(dLdb);

% differentiation done by differentiator program
ddtdLdDs = D2s*Ixxp*sin(t)^2 + D2s*Ixxs*sin(t)^2 - D2p*Ixxp*sin(t) - D2p*Ixxs*si
ddtdLdDt = D2t*Iyyp*cos(p)^2 + D2t*Iyys*cos(p)^2 + D2t*Izzp*sin(p)^2 + D2t*Izzs*
ddtdLdDp = D2p*Ixxp + D2p*Ixxs + D2b*Ixxp*cos(a) - D2s*Ixxp*sin(t) - D2s*Ixxs*si
ddtdLdDx = D2x*Mp + D2x*Ms + Db^2*Mp*d*cos(p)*sin(b)*sin(s) + Dp^2*Mp*d*cos(p)*s
ddtdLdDy = D2y*Mp + D2y*Ms - Db^2*Mp*d*cos(p)*sin(b)*cos(s) - Dp^2*Mp*d*cos(p)*s
ddtdLdDa = D2a*Iyyp + D2t*Iyyp*cos(p) - Dp*Dt*Iyyp*sin(p) + D2s*Iyyp*cos(t)*sin(
ddtdLdDb = D2b*Ixxp*cos(a)^2 + D2b*Izzp*sin(a)^2 + D2p*Ixxp*cos(a) - Da*Dp*Ixxp*


% v1: lagrange multiplier
% v2: lagrange multiplier
% M1: drive motor torque
% M2: tilt (steering) motor torque

syms v1 v2 M1 M2 f1 f2 f3 f4

Q1 = -f4*(Ws_o(3));
%Q1 = -f4*Ds;
Q2 = 0;
Q3 = 0;
Q4 = -f3*Dx;
Q5 = -f3*Dy;
Q6 = M1 - f1*Da;    %drive motor torque and viscous damping
Q7 = M2 - f2*Db;    %steering motor torque and viscous damping

% equations of motion
% ddtdLdDs - dLds = v1*a1(1) + v2*a2(1) + Q1;
% ddtdLdDt - dLdt = v1*a1(2) + v2*a2(2) + Q2;
% ddtdLdDp - dLdp = v1*a1(3) + v2*a2(3) + Q3;
% ddtdLdDx - dLdx = v1*a1(4) + v2*a2(4) + Q4;
% ddtdLdDy - dLdy = v1*a1(5) + v2*a2(5) + Q5;
% ddtdLdDa - dLda = v1*a1(6) + v2*a2(6) + Q6;
% ddtdLdDb - dLdb = v1*a1(7) + v2*a2(7) + Q7;

% constraint equations
% a11*Ds + a12*Dt + a13*Dp + a14*Dx + a15*Dy + a16*Da + a17*Db = 0;
% a21*Ds + a22*Dt + a23*Dp + a24*Dx + a25*Dy + a26*Da + a27*Db = 0;

eqn1 = ddtdLdDs - dLds - v1*a1(1) - v2*a2(1) - Q1; % =0
eqn2 = ddtdLdDt - dLdt - v1*a1(2) - v2*a2(2) - Q2; % =0
eqn3 = ddtdLdDp - dLdp - v1*a1(3) - v2*a2(3) - Q3; % =0
eqn4 = ddtdLdDx - dLdx - v1*a1(4) - v2*a2(4) - Q4; % =0
eqn5 = ddtdLdDy - dLdy - v1*a1(5) - v2*a2(5) - Q5; % =0
eqn6 = ddtdLdDa - dLda - v1*a1(6) - v2*a2(6) - Q6; % =0
eqn7 = ddtdLdDb - dLdb - v1*a1(7) - v2*a2(7) - Q7; % =0
eqn8 = a1(1)*Ds + a1(2)*Dt + a1(3)*Dp + a1(4)*Dx + a1(5)*Dy + a1(6)*Da + a1(7)*D
eqn9 = a2(1)*Ds + a2(2)*Dt + a2(3)*Dp + a2(4)*Dx + a2(5)*Dy + a2(6)*Da + a2(7)*D

eqn10 = D2x - D2t*r*cos(s) + Ds*Dt*r*sin(s) - D2p*r*cos(t)*sin(s) - Dp*Ds*r*cos(
eqn11 = D2y - D2t*r*sin(s) - Ds*Dt*r*cos(s) + D2p*r*cos(s)*cos(t) - Dp*Ds*r*cos(
```

```
v1_eqn = solve(eqn4,'v1');
v2_eqn = solve(eqn5,'v2');

eqn1_a = subs(eqn1,{'v1','v2'},{v1_eqn,v2_eqn});
eqn2_a = subs(eqn2,{'v1','v2'},{v1_eqn,v2_eqn});
eqn3_a = subs(eqn3,{'v1','v2'},{v1_eqn,v2_eqn});
eqn4_a = eqn6;
eqn5_a = eqn7;
eqn6_a = eqn8;
eqn7_a = eqn9;
eqn8_a = eqn10;
eqn9_a = eqn11;

Dx_eqn = solve(eqn6_a,'Dx');
Dy_eqn = solve(eqn7_a,'Dy');

funcs = [eqn1_a; eqn2_a; eqn3_a; eqn4_a; eqn5_a; eqn8_a; eqn9_a];
vecs = [D2s; D2t; D2p; D2x; D2y; D2a; D2b];

J = jacobian(funcs,vecs);
J = simplify(J);

consts = funcs - J*vecs;
consts = simplify(consts);

subs(J,{'s','t','p','x','y','a','b','Ds','Dt','Dp','Dx','Dy','Da','Db','D2s','D2
subs(consts,{'s','t','p','x','y','a','b','Ds','Dt','Dp','Dx','Dy','Da','Db','D2s
subs(Dx_eqn,{'s','t','p','x','y','a','b','Ds','Dt','Dp','Dx','Dy','Da','Db','D2s
subs(Dy_eqn,{'s','t','p','x','y','a','b','Ds','Dt','Dp','Dx','Dy','Da','Db','D2s
```

Appendix C

MATLAB: numerical_integration.m

Note: Many lines in this program exceed the width of the page and have been truncated to fit.

```
function numerical_integration()
    % Initial Values
    m0(1)  = 0;% s(0)
    m0(2)  = 0;% t(0)
    m0(3)  = 0;% p(0)
    m0(4)  = 0;% x(0)
    m0(5)  = 0;% y(0)
    m0(6)  = pi/4;% a(0)
    m0(7)  = 0;% b(0)
    m0(8)  = 0;% Ds(0)
    m0(9)  = 0;% Dt(0)
    m0(10) = 0;% Dp(0)
    m0(11) = 0;% Dx(0)
    m0(12) = 0;% Dy(0)
    m0(13) = 0;% Da(0)
    m0(14) = 0;% Db(0)

    % Model Parameters
    Ms   = 1;    % Mass of Sphere (kg)
    Mp   = 15;   % Mass of Pendulum (kg)
    Ixxs = 0.25;  % Principle Moment of Interia of Sphere About x-axis (kgm^2)
    Iyys = 0.15;  % Principle Moment of Interia of Sphere About y-axis (kgm^2)
    Izzs = 0.25;  % Principle Moment of Interia of Sphere About z-axis (kgm^2)
    Ixxp = 2;    % Principle Moment of Interia of Pendulum About x-axis (kgm^2)
    Iyyp = 2;    % Principle Moment of Interia of Pendulum About y-axis (kgm^2)
    Izzp = 3;    % Principle Moment of Interia of Pendulum About z-axis (kgm^2)
    g    = 9.81; % Acceleration due to Gravity (m/s^2)
    M1   = 0;    % Drive Motor Torque (Nm)
    M2   = 0;    % Steering Motor Torque (Nm)
    d    = 0.1;  % Distance from Center of Sphere to CG of Pendulum (m)
    r    = 0.25; % Radius of Sphere (m)
    f1   = .1;    % Drive Motor damping (Nms)
    f2   = .1;    % Steering Motor damping (Nms)
    f3   = .2;   % Rolling damping (Ns)
    f4   = 10;   % Spinning damping (Nms)

    % Simulation Parameters
    tspan = [0 10];   % Simulation Time Span (s)
    fps = 24;        % Simulation Frame Rate
    Animate = true; % True/False to Save Simulation as a Video
    RecordVideo = false; % True/False to Save Simulation as a Video


    params = [Ms Mp Ixxs Iyys Izzs Ixxp Iyyp Izzp g M1 M2 d r f1 f2 f3 f4];
    [tvals,eqnvals] = ode45(@sphere_model,tspan,m0,[],params);
    if Animate, simulation(tvals, eqnvals, params, fps, tspan(2)-tspan(1),

end

% m(1): s
% m(2): t
% m(3): p
% m(4): x
% m(5): y
% m(6): a
% m(7): b
% m(8): Ds
% m(9): Dt
% m(10): Dp
% m(11): Dx
% m(12): Dy
% m(13): Da
% m(14): Db

% Dm(1): Ds
% Dm(2): Dt
% Dm(3): Dp
% Dm(4): Dx
% Dm(5): Dy
% Dm(6): Da
% Dm(7): Db
% Dm(8): D2s
% Dm(9): D2t
```

```
% Dm(10): D2p
% Dm(11): D2x
% Dm(12): D2y
% Dm(13): D2a
% Dm(14): D2b

function Dm_return = sphere_model(t,m,params)
    Ms = params(1);
    Mp = params(2);
    Ixxs = params(3);
    Iyys = params(4);
    Izzs = params(5);
    Ixxp = params(6);
    Iyyp = params(7);
    Izzp = params(8);
    g = params(9);
    M1 = params(10);
    M2 = params(11);
    d = params(12);
    r = params(13);
    f1 = params(14);
    f2 = params(15);
    f3 = params(16);
    f4 = params(17);


    Dm = zeros(14,1);

    Dm(1) = m(8);
    Dm(2) = m(9);
    Dm(3) = m(10);
    Dm(4) = r*Dm(2)*cos(m(1)) + r*Dm(3)*cos(m(2))*sin(m(1));
    Dm(5) = r*Dm(2)*sin(m(1)) - r*Dm(3)*cos(m(1))*cos(m(2));
    Dm(6) = m(13);
    Dm(7) = 0;%m(14);

    J = [Ixxp + Ixxs + Mp*d^2 - Ixxp*cos(m(2))^2 - Ixxs*cos(m(2))^2 + Iyyp*cos(m
         Iyyp*cos(m(2))*cos(m(3))*sin(m(3)) + Iyys*cos(m(2))*cos(m(3))*sin(m(3))
         Mp*d^2*cos(m(7))^2*sin(m(2)) - Ixxs*sin(m(2)) - Mp*d^2*sin(m(2)) - Ixxp
         Iyyp*cos(m(2))*sin(m(3)) + Mp*d^2*cos(m(2))*cos(m(7))^2*sin(m(3)) - Mp*
        -Ixxp*cos(m(6))*sin(m(2)) - Mp*d^2*cos(m(6))*sin(m(2)) - Izzp*cos(m(2))*
         0, -r*cos(m(1)), -r*cos(m(2))*sin(m(1)), 1, 0,0,0;...
         0, -r*sin(m(1)),r*cos(m(1))*cos(m(2)), 0, 1,0,0];

    consts = [f4*Dm(1) - f4*Dm(3)*sin(m(2)) - Ixxp*Dm(2)*Dm(3)*cos(m(2)) - Ixxs*
              Ixxp*Dm(1)*Dm(3)*cos(m(2)) + Ixxs*Dm(1)*Dm(3)*cos(m(2)) - Iyyp*Dm(
              Iyyp*Dm(1)*Dm(2)*cos(m(2)) - Ixxs*Dm(1)*Dm(2)*cos(m(2)) - Ixxp*Dm(
              f1*Dm(6) - M1 + Ixxp*Dm(3)*Dm(7)*sin(m(6)) - Iyyp*Dm(2)*Dm(3)*sin(
              f2*Dm(7) - M2 - Ixxp*Dm(3)*Dm(6)*sin(m(6)) - Mp*d^2*Dm(1)^2*cos(m(
              r*(Dm(1)*Dm(2)*sin(m(1)) - Dm(1)*Dm(3)*cos(m(1))*cos(m(2)) + Dm(2)
             -r*(Dm(1)*Dm(2)*cos(m(1)) + Dm(1)*Dm(3)*cos(m(2))*sin(m(1)) + Dm(2)


    D2qis = -J\consts;

    Dm(8) = D2qis(1);
    Dm(9) = D2qis(2);
    Dm(10) = D2qis(3);
    Dm(11) = D2qis(4);
    Dm(12) = D2qis(5);
    Dm(13) = D2qis(6);
    Dm(14) = D2qis(7);

    [t cond(J)]
    Dm_return = Dm;
end
```

Appendix D

MATLAB: simulation.m

Note: Many lines in this program exceed the width of the page and have been truncated to fit.

```
function simulation(tvals, eqnvals, params, fps, sim_time, record_video)

    global r;
    Mp = params(2);
    g = params(9);
    d = params(12);
    r = params(13);

    time_int = linspace(0,sim_time,sim_time*fps);
    s_int = spline(tvals,eqnvals(:,1),time_int);
    t_int = spline(tvals,eqnvals(:,2),time_int);
    p_int = spline(tvals,eqnvals(:,3),time_int);
    x_int = spline(tvals,eqnvals(:,4),time_int);
    y_int = spline(tvals,eqnvals(:,5),time_int);
    a_int = spline(tvals,eqnvals(:,6),time_int);
    b_int = spline(tvals,eqnvals(:,7),time_int);

    V_p = Mp*g*(d*cos(b_int).*sin(a_int).*sin(t_int) + d*sin(b_int).*cos(t_int).
    %plot(t_int,V_p);


    fig = figure('Position',[0 -50 1280 720]);
    set(fig,'Renderer','OpenGL');

    [x_s,y_s,z_s] = sphere(16);
    x_s = x_s .*r;
    y_s = y_s .*r;
    z_s = z_s .*r + r;

    [x_p,y_p,z_p] = cylinder(1,12);
    x_p = x_p .*(r/10);
    y_p = y_p .*(r/10);
    z_p = z_p .*(4*r/5) + r/5;

    [x_g,y_g] = meshgrid(-max(abs([x_int,y_int]))-r:r:max(abs([x_int,y_int]))+r,
    z_g = x_g .*0;

    ball = surf(x_s,y_s,z_s, z_s.*0);
    rotate(ball,[1 0 0],90,[0 0 r]);
    hold on;

    pendulum = surf(x_p,y_p,z_p, z_p.*0);
    set(ball,'FaceAlpha',0.2)

    ground = mesh(x_g,y_g,z_g);

    hidden off;
    %shading interp;
    lighting flat;
    camlight infinite;
    axis tight equal;
    axis off;
    plot(x_int,y_int);

    txt1 = text(0,0,max(abs([x_int,y_int]))+r,'Elapsed Time(s)');
    txt2 = text(0,0,max(abs([x_int,y_int]))+r-.1,'0.000');
    set(txt1,'HorizontalAlignment','center','FontSize',13);
    set(txt2,'HorizontalAlignment','center','FontSize',16);

    global ball_xdata_init;
    global ball_ydata_init;
    global ball_zdata_init;
    global ball_xdata;
    global ball_ydata;
    global ball_zdata;
    ball_xdata_init = get(ball, 'XData');
    ball_ydata_init = get(ball, 'YData');
    ball_zdata_init = get(ball, 'ZData') - r;
    ball_xdata = ball_xdata_init;
    ball_ydata = ball_ydata_init;
    ball_zdata = ball_zdata_init;
    set(ball,'XDataSource','ball_xdata','YDataSource','ball_ydata','ZDataSource'
```

```
    global pendulum_xdata_init;
    global pendulum_ydata_init;
    global pendulum_zdata_init;
    global pendulum_xdata;
    global pendulum_ydata;
    global pendulum_zdata;
    pendulum_xdata_init = get(pendulum, 'XData');
    pendulum_ydata_init = get(pendulum, 'YData');
    pendulum_zdata_init = get(pendulum, 'ZData') - r;
    pendulum_xdata = pendulum_xdata_init;
    pendulum_ydata = pendulum_ydata_init;
    pendulum_zdata = pendulum_zdata_init;
    set(pendulum,'XDataSource','pendulum_xdata','YDataSource','pendulum_ydata','


    move_sphere(s_int(1),t_int(1),p_int(1),x_int(1),y_int(1));
    move_pendulum(s_int(1),t_int(1),p_int(1),a_int(1),b_int(1),x_int(1),y_int(1)
    refreshdata(ball, 'caller')
    refreshdata(pendulum, 'caller')

    pause(5)

    for i=1:length(time_int)
        move_sphere(s_int(i),t_int(i),p_int(i),x_int(i),y_int(i));
        move_pendulum(s_int(i),t_int(i),p_int(i),a_int(i),b_int(i),x_int(i),y_in
        refreshdata(ball, 'caller')
        refreshdata(pendulum, 'caller')
        set(txt2,'string',num2str(time_int(i),'%6.3f'));
        drawnow;
        if record_video, MOV(i)=getframe(gcf); end;
    end
    if record_video
        movie2avi(MOV,'../../Downloads/video.avi','fps',fps,'videoname','')
        movefile('../../Downloads/video.avi','../../Downloads/Video/video.avi')
    end;
end

function move_sphere(psi,theta,phi,x_pos,y_pos)
    global ball_xdata_init;
    global ball_ydata_init;
    global ball_zdata_init;
    global ball_xdata;
    global ball_ydata;
    global ball_zdata;
    global r;

    rot_s = [cos(psi) -sin(psi) 0;sin(psi) cos(psi) 0;0 0 1];
    rot_t = [cos(theta) 0 sin(theta);0 1 0;-sin(theta) 0 cos(theta)];
    rot_p = [1 0 0;0 cos(phi) -sin(phi);0 sin(phi) cos(phi)];
    rot = (rot_s*rot_t*rot_p)';

    x = ball_xdata_init;
    y = ball_ydata_init;
    z = ball_zdata_init;

    [m,n] = size(x);
    newxyz = [x(:), y(:), z(:)];
    newxyz = newxyz*rot;
    ball_xdata = reshape(newxyz(:,1),m,n) + x_pos;
    ball_ydata = reshape(newxyz(:,2),m,n) + y_pos;
    ball_zdata = reshape(newxyz(:,3),m,n) + r;
end
function move_pendulum(psi,theta,phi,alpha,beta,x_pos,y_pos)
    global pendulum_xdata_init;
    global pendulum_ydata_init;
    global pendulum_zdata_init;
    global pendulum_xdata;
    global pendulum_ydata;
    global pendulum_zdata;
    global r;

    rot_s = [cos(psi) -sin(psi) 0;sin(psi) cos(psi) 0;0 0 1];
    rot_t = [cos(theta) 0 sin(theta);0 1 0;-sin(theta) 0 cos(theta)];
```

```
    rot_p = [1 0 0;0 cos(phi) -sin(phi);0 sin(phi) cos(phi)];
    rot_a = [cos(alpha) 0 sin(alpha);0 1 0;-sin(alpha) 0 cos(alpha)];
    rot_b = [1 0 0;0 cos(beta) -sin(beta);0 sin(beta) cos(beta)];
    rot = (rot_s*rot_t*rot_p*rot_a*rot_b)';

    x = pendulum_xdata_init;
    y = pendulum_ydata_init;
    z = pendulum_zdata_init;

    [m,n] = size(x);
    newxyz = [x(:), y(:), z(:)];
    newxyz = newxyz*rot;
    pendulum_xdata = reshape(newxyz(:,1),m,n) + x_pos;
    pendulum_ydata = reshape(newxyz(:,2),m,n) + y_pos;
    pendulum_zdata = reshape(newxyz(:,3),m,n) + r;
end
```

Appendix E

AppleScript: Differentiator.scpt

```
--input must be fully expanded
set input to "Dy - Dt*r*sin(s) + Dp*r*cos(s)*cos(t)"

set terms to {"sin(s)", "sin(t)", "sin(p)", "sin(a)", "sin(b)", "cos(s)", "cos(t)", "cos(p)", "cos(a)", "cos(b)",
"sin(s)^2", "sin(t)^2", "sin(p)^2", "sin(a)^2", "sin(b)^2", "cos(s)^2", "cos(t)^2", "cos(p)^2",
"cos(a)^2", "cos(b)^2", "sin(s)^3", "sin(t)^3", "sin(p)^3", "sin(a)^3", "sin(b)^3", "cos(s)^3",
"cos(t)^3", "cos(p)^3", "cos(a)^3", "cos(b)^3", "sin(s)^4", "sin(t)^4", "sin(p)^4", "sin(a)^4",
"sin(b)^4", "cos(s)^4", "cos(t)^4", "cos(p)^4", "cos(a)^4", "cos(b)^4", "Ds", "Dt", "Dp", "Dx", "Dy",
"Da", "Db", "x", "y"}

set diffterms to {"Ds*cos(s)", "Dt*cos(t)", "Dp*cos(p)", "Da*cos(a)", "Db*cos(b)", "-Ds*sin(s)", "-
Dt*sin(t)", "-Dp*sin(p)", "-Da*sin(a)", "-Db*sin(b)", "2*sin(s)*cos(s)*Ds", "2*sin(t)*cos(t)*Dt",
"2*sin(p)*cos(p)*Dp", "2*sin(a)*cos(a)*Da", "2*sin(b)*cos(b)*Db", "-2*sin(s)*cos(s)*Ds", "-
2*sin(t)*cos(t)*Dt", "-2*sin(p)*cos(p)*Dp", "-2*sin(a)*cos(a)*Da", "-2*sin(b)*cos(b)*Db",
"3*sin(s)^2*cos(s)*Ds", "3*sin(t)^2*cos(t)*Dt", "3*sin(p)^2*cos(p)*Dp", "3*sin(a)^2*cos(a)*Da",
"3*sin(b)^2*cos(b)*Db", "-3*sin(s)*cos(s)^2*Ds", "-3*sin(t)*cos(t)^2*Dt", "-3*sin(p)*cos(p)^2*Dp", "-
3*sin(a)*cos(a)^2*Da", "-3*sin(b)*cos(b)^2*Db", "4*sin(s)^3*cos(s)*Ds", "4*sin(t)^3*cos(t)*Dt",
"4*sin(p)^3*cos(p)*Dp", "4*sin(a)^3*cos(a)*Da", "4*sin(b)^3*cos(b)*Db", "-4*sin(s)*cos(s)^3*Ds", "-
4*sin(t)*cos(t)^3*Dt", "-4*sin(p)*cos(p)^3*Dp", "-4*sin(a)*cos(a)^3*Da", "-4*sin(b)*cos(b)^3*Db",
"D2s", "D2t", "D2p", "D2x", "D2y", "D2a", "D2b", "Dx", "Dy"}

set tdlrs to AppleScript's text item delimiters
set AppleScript's text item delimiters to {" "}
set parsedinput to every text item of input
set AppleScript's text item delimiters to tdlrs


set dLdqi to {}
set tempstring to ""

repeat with tempitem in parsedinput
	if tempitem as text = "+" then
		--do nothing
	else if tempitem as text = "-" then
		set tempstring to "-1*"
	else
		set tempstring to tempstring & tempitem
		copy tempstring to end of dLdqi
		set tempstring to ""
	end if
end repeat

set totaldiff to {}

repeat with k from 1 to count of items in dLdqi
	set test to item k in dLdqi
	set tdlrs to AppleScript's text item delimiters
	set AppleScript's text item delimiters to {"*"}
	set termsList to every text item of test
	set AppleScript's text item delimiters to tdlrs
	set difftest to {}

	repeat with i from 1 to count of items in termsList
		set currentterm to item i of termsList
		set idx to 0
		if terms contains {currentterm} then
			set idx to 1
			repeat until (item idx of terms is currentterm)
				set idx to idx + 1
			end repeat
		else
			set idx to 0
		end if
		if idx > 0 then
			set tempstring to ""
			repeat with j from 1 to count of items in termsList
```

```
                        if j = i then
                            set tempstring to tempstring & item idx of diffterms
                        else
                            set tempstring to tempstring & item j of termsList
                        end if
                        if j < (count of items in termsList) then set tempstring to tempstring & "*"
                    end repeat
                    copy tempstring to the end of difftest
                    --display dialog difftest as text
                end if
            end repeat
            set tdlrs to AppleScript's text item delimiters
            set AppleScript's text item delimiters to {" + "}
            set difftest to difftest as text
            set AppleScript's text item delimiters to tdlrs
            copy difftest to the end of totaldiff
    end repeat
    set tdlrs to AppleScript's text item delimiters
    set AppleScript's text item delimiters to {" + "}
    set totaldiff to totaldiff as text
    set AppleScript's text item delimiters to tdlrs

    totaldiff
```