SANDIA REPORT

SAND2007-1466
Unlimited Release
Printed January 2007

Advanced Robot Locomotion

Raymond H. Byrne, Jason C. Neely, Steven Buerger, John T. Feddema, David K. Novick, Scott E. Rose, Barry L. Spletzer, Beverly R. Sturgis, and David G. Wilson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2007-1466
Unlimited Release
Printed January 2007

Advanced Robot Locomotion

Raymond H. Byrne, Jason C. Neely, Steven Buerger,
John T. Feddema, David K. Novick, Scott E. Rose,
Barry L. Spletzer, Beverly R. Sturgis, and David G. Wilson

Sandia National Laboratories
Intelligent Systems, Sensors, and Controls
Department 6473
MS 1003, PO Box 5800
Albuquerque, NM 87185-1003

Abstract

This report contains the results of a research effort on advanced robot locomotion. The majority of this work focuses on walking robots. Walking robot applications include delivery of special payloads to unique locations that require human locomotion to exo-skeleton human assistance applications. A walking robot could step over obstacles and move through narrow openings that a wheeled or tracked vehicle could not overcome. It could pick up and manipulate objects in ways that a standard robot gripper could not. Most importantly, a walking robot would be able to rapidly perform these tasks through an intuitive user interface that mimics natural human motion. The largest obstacle arises in emulating stability and balance control naturally present in humans but needed for bipedal locomotion in a robot. A tracked robot is bulky and limited, but a wide wheel base assures passive stability. Human bipedal motion is so common that it is taken for granted, but bipedal motion requires active balance and stability control for which the analysis is non-trivial. This report contains an extensive literature study on the state-of-the-art of legged robotics, and it additionally provides the analysis, simulation, and hardware verification of two variants of a proto-type leg design.

Contents

1 Introduction **7**

2 Literature Survey **7**
2.1 Monopeds and Bipeds 8
2.2 Quadrupeds 9
2.3 Control Approaches 9
2.4 Opportunities for Innovation 12

3 Monoped Hopping Robot **12**
3.1 System Model 12
3.2 Hop Generation 14
3.3 Base Positioning 18

4 Full Dynamic Simulation **20**
4.1 Balance Control 21
4.2 3-Dimensional Model 25
4.2.1 Moving mass inside a box 27
4.2.2 Gymnast 27
4.2.3 Hopping robot 27

5 Power Considerations **27**

6 Summary and Conclusions **29**

References **30**

A Dynamic Equation Derivation for Balanced Monoped with $r = 0$ **33**

B Dynamic Equation Derivation for Balanced Monoped with $r > 0$ **35**

C Dynamics Derived Using Kane's Method **37**

List of Figures

1 DP architecture consisting of trajectory generator/optimizer and feedback control system. . . . 11
2 Hopping Monoped Solidworks Model . . . 13
3 Undamped Spring-Mass System . . . 13
4 System Model . . . 15
5 Hopping Robot Dynamic Model . . . 15
6 Plot of Hopping Robot Performance . . . 17
7 Balance Model . . . 19
8 State Trajectory for Center of Gravity Controller . . . 20
9 Balanced Monoped Dynamic Model . . . 21
10 Stable State Trajectory . . . 23
11 State Trajectory Phase Diagram . . . 24
12 Balance Controller Animation . . . 24
13 Stable State Trajectory . . . 25
14 State Trajectory Phase Diagram . . . 26
15 Control Torques . . . 26
16 Rotating Mass System . . . 27
17 Gymnast Model . . . 28
18 Ring Model . . . 28
19 Ragone Plot for Various Energy Sources . . . 29
20 Hopping Robot Model . . . 38

1 Introduction

While Sony and Honda robots have focused on entertainment value and human interaction, the research goal at Sandia would be to develop practical mobility platforms (from the waist down) that would have DOE and military applications. These applications include delivery of special payloads to unique locations that require human locomotion to exo-skeleton human assistance applications. A walking robot could step over obstacles and move through narrow openings that a wheeled or tracked vehicle could not overcome. It could pick up and manipulate objects in ways that a standard robot gripper could not. Most importantly, the walking robot would be able to rapidly perform these tasks through an intuitive user interface that mimics natural human motion.

Though a user interface might capture the user intent with regards to mobility, it is unlikely to accomplish the task of balance and stability control. Human beings accomplish balance through active controls that rely on feedback from organs ("sensors") in the inner ear, peripheral nerves in the soles of the feet, and visual cues. It is conceivable that equivalent attributes might be sensed on the robot and fed back to an operator who can in turn make corrections, but there exist significant stability issues with delays in closing the feedback loop. It is more plausible to achieve stability control onboard the robot and decouple it from the mobility commands provided by the user. This principle has already been explored with balancing wheeled robot systems [1]. Stabilization becomes more challenging, however, when the system must interact with the environment. For example, control methods have been developed to identify and mitigate unstable states on the Segway robotic mobility platform[2]. Unstable states may arise if the Segway vehicle interacts with an obstacle, an attached manipulator rigidly grips a fixed point in the environment, or in any other way suffers a restriction to its state space due to interaction with its environment. For legged robots in particular, the problem of balancing a running robot (dynamic stability) is far more challenging than balancing a slow moving robot between stable poses.

Another consideration that is often ignored is that of power utilization and storage. A motor-driven bipedal robot is likely to use more power given the need to actuate more degrees of freedom, and these motors will be required to apply torques for balance control even while "standing." Although several untethered bipedal robots have been developed [3, 4], no bipedal robots have been able to achieve tetherless operation for extended periods of time. The power requirements for any practical range of remote operation would necessitate a large battery pack that must be designed around a desired dynamic model for the robot. A practical walking robot design must achieve dynamic stability control over varied terrain while performing several normal human behaviors (ie. running, walking, carrying a package), and it should achieve this untethered.

This report includes the results of a literature search summarizing the current state-of-the-art in legged robots and balanced robot control. In addition, the stability and hop control for a monoped robot is developed and presented. Finally, a plausible method for powering an untethered robot is presented and discussed to motivate further work.

2 Literature Survey

A significant and rapidly increasing body of research exists on walking, running, and hopping robots. Current and recent work spans a broad range of capabilities and approaches. While not a comprehensive review, this section provides a snapshot of the scope of current activities. We focus mostly on monopeds and bipeds, but some notable quadrupeds are also described. Another section focuses on control strategies for walking robots.

2.1 Monopeds and Bipeds

Virtually all "walking" or biped robots use one of two philosophies for control of ambulation. Some of the most highly-publicized robots, including those from Honda [3] and Sony [4], use joint trajectory control and balance by keeping the center of gravity above the support polygon at all times. This approach has yielded visually impressive results when playing trajectories for known environments, but is extremely vulnerable to disturbances and environmental uncertainty. This vulnerability arises because the support polygon for humanoid forms is extremely small compared to the height of the center of gravity. This approach is also relatively energy-inefficient because significant effort is expended to overcome the intrinsic robot and drivetrain dynamics and force adherence to the desired trajectory. A result of this somewhat unnatural approach is that the gait of such robots appears to differ significantly from natural human walking. Although Honda now reports that their ASIMO robot is capable of running, as indicated by the fact that at times both feet are simultaneously (and very briefly) off the ground, this is most accurately viewed as an extremely brief transition between periods of static stability rather than full dynamic stability. Nonetheless, speeds of up to 1.66 m/sec have been achieved as of December 2005 [3]. Another group in Japan has done similar work toward hopping and running in an ASIMO-type robot [5].

A contrasting approach relies on dynamic stability, in which walking, hopping, or running consists of a state in which the robot is constantly falling and catching itself with the next step of its leg or legs. Typically, legs are designed to include deliberate compliance that enables them to store and release energy at appropriate points in the gait cycle, improving efficiency. This approach is motivated by studies of biomechanics and muscle properties, and emulates the use of passive muscle dynamics to improve efficiency in animal and human walking. Controllers for dynamically stable walking robots range widely in complexity, but are generally simpler and require less a priori knowledge of the environment than their trajectory-based counterparts. Specific limb trajectories are often driven as much by their passive dynamics as by pre-programming.

Early work in dynamically-stable walking robots was dominated by Marc Raibert's Leglab at MIT. In the early 1980's Raibert developed a series of hopping robots, first with one leg and later with two [6]. Raibert's two-legged "pogo stick" robot remains one of the few to dynamically balance in a full three dimensions, without separate support. Later Raibert's Leglab group developed an impressive early biped called Spring Flamingo that walked at speeds greater than 1.2 m/sec and was able to handle unexpected rolling terrain while continuing to balance in two dimensions (it was supported in the plane orthogonal to the ground and walking plane by a rotating boom). Raibert and the group led by his successor Gill Pratt developed series elastic actuators, force-controlled actuators that include springs in series to improve fidelity, cushion impacts and store energy [7]. They also developed "virtual model control", a form of equilibrium point control used to drive walking over rough terrain [8]. Finally, Pratts group designed and built a 12 degree of freedom biped (called M2) made to mimic human legs and walk and run independently and in three dimensions. They were able to demonstrate independent standing under battery power and taking several steps tethered in the lab, but never truly achieved stable walking.

The influence of the work of Raibert and his successors is readily seen in work ongoing in bipedal locomotion, particularly in the prevalence of mechanical compliance in robot legs. A group led by Sang-Ho Hyon at the Tokyo Institute of Technology has developed Kenken, a single-legged hopping and running robot using two hydraulic actuators. It has been demonstrated traveling at speeds up to 2.0 m/sec around a rotary track with a planar rod for stabilization. A two-legged version was in development as of 2003 [9]. A French group working with Ohio State University and the University of Michigan has developed a bipedal robot with knobs for feet and articulated knees

known as the RABBIT. Much like Spring Flamingo and Kenken, it travels about a central pivot and cannot balance in three dimensions [10]. This robot is specifically intended to run at high speeds. Another robot known as Runbot recently received publicity for breaking a bipedal robot speed record by traveling at 3.5 leg lengths / sec (it is 30 cm tall). Videos show that this appears to be a fast walking gait, not strictly a running gait. This robot is notable because it is dynamically stable and uses an extremely simple control algorithm that simply senses foot strikes on the ground and the swing of its legs. When a foot strikes the ground, the opposite leg swings forward; the knee actively bends to achieve ground clearance until the hip reaches a certain point, when the knee straightens to prepare for impact [11]. This simple control strategy stands in contrast to the advanced force-based controllers used for many robots, but it may not prove as effective when terrain is rough or unpredictable.

2.2 Quadrupeds

Most walking robots with four or more legs have significantly different goals and capabilities than most biped or walking robots. However, several particular robots have characteristics relevant to the problem under study. The BigDog robot, developed by Boston Dynamics to act as a robotic pack mule, is dynamically stabilized and demonstrates an impressive ability to traverse rough terrain. BigDog is powered by a gasoline engine that drives hydraulic actuators, and uses mechanical compliance in its joints to emulate muscle dynamics. It has been separately shown walking at up to 3.3 mph (1.45 m/s), climbing a 35 degree slope, and carrying loads of up to 120 lb. The robot is extremely robust to disturbance inputs thanks both to its dynamic stability and relatively low center of gravity and large footprint (compared to walking robots) [12].

The Scout II robot developed at McGill University is a quadruped that is able to run with a "bounding gait at speeds greater than 1 m/s despite its small size [13]. While this gait does not risk falling instability in the same sense as most bipeds, its ability to run aggressively with all legs off the ground at times is notable.

2.3 Control Approaches

The principles of legged motion can be decomposed into the mechanical dynamics, control, and attention to balance during operation. This means that the legged systems studied operate in a regime where the velocities and kinetic energies of the masses are important determinants of behavior [6].

"The exchange of energy among its various forms is also important in understanding the dynamics of legged locomotion. For example, there is a cycle of activity in running that changes the form of the stored energy several times: the body's potential energy of elevation changes to kinetic energy during falling, then to strain energy when parts of the leg deform elastically during rebound with the ground, then into kinetic energy again as the body accelerates upward, and finally back into potential energy of elevation. This sort of dynamic exchange is central to an understanding of legged locomotion" [6].

A dynamic treatment, however, does not imply an intractable treatment. Although the detailed dynamics of a legged system may indeed be complicated, control techniques that use dynamics may be simple [6]. Control systems can generally be made simpler if they are attuned to the dynamics of the mechanism they control and to the task the mechanism performs. An initial goal of this project is to identify and explore control techniques that use dynamics in simple ways to efficiently produce optimized legged locomotion. In addition, the dynamics play a vital role in the legged

systems ability to actively balance. A statically or passively balanced system avoids tipping and typically must operate in or near equilibrium. In contrast an actively balanced system my tip and accelerate for short periods of time [6].

"The control system manipulates body and leg motions to ensure that each tipping interval is brief and that each tipping motion in one direction is compensated by a tipping motion on the opposite direction. An effective base of support is maintained over time. A system that balances actively may also permit vertical acceleration, such as the bouncing that occurs when the legs deform elastically and the ballistic travel that occurs between bounces" [6].

"The ability of an actively balanced system to depart from static equilibrium relaxes the rules on how legs can be used for support. This leads to improved mobility. For example, if a legged system can tolerate tipping, then it can position the feet far from the center of mass in order to use footholds that are widely separated or erratically placed. If it can remain upright with a small base of support, then it can travel where obstructions are closely spaced or where the path of firm support is narrow. The ability to tolerate intermittent support also contributes to mobility. It allows a system to move all its legs to new footholds at one time, to jump onto or over obstacles, and to use ballistic motions for increased speed. These abilities to use narrow base and intermittent support generally increase the types of terrain a legged system can negotiate. Animals routinely exploit active balance to travel quickly on difficult terrain. Legged vehicles will have to balance actively, too, if they are to move with animal-like mobility and speed" [6].

Early work and research on machines that run and walk with active balance have been captured in [6]. Some of the most recent biped dynamics and controls research has been reported in the following references [10], [14], [15], [16], [17], [18], and [19]. Optimization techniques [20], [21], [22], [23] have become very popular in the pursuit of finding specific optimized swing leg trajectories. Several of these researchers [22], [23] have employed differential dynamic programming techniques as an efficient method for the optimized biped walking and locomotion tasks.

Optimization methods such as Dynamic Programming (DP) techniques [24] are reviewed to

1. aid in the prediction of optimal walking and running families of design trajectories/curves for specific mission requirements,

2. eventually determine the sizing and design of the biped leg lengths, upper/lower leg length ratios, and actuator torque requirements, and

3. realize the DP algorithm as an integral part of the real-time implementation optimized feed-forward component of our robust control design architecture (see Figure 1).

Our biped locomotion control system architectures will be composed of several key components, as shown in Figure 1. The architecture will consist of a trajectory generator/optimizer and a feedback control system. The trajectory generator provides a feasible feed-forward control reference trajectory. It is desirable that the reference trajectory be optimized with respect to a performance objective that reaches for the ideal dynamic behavior of the system, keeping in mind the presence of system and actuation constraints. Theoretically, if deviations (or the errors) in the states and inputs are zero, then the dynamic system will results in optimized performance (see Figure 1). However, in real-world applications, there will always be inaccuracies. The feedback control system provides stability and robustness around the reference trajectory, subject to inherent modeling uncertainty and external disturbances. This approach has the advantage that the system is tracking a feasible optimized trajectory while maintaining stable system response. The desired objectives can be commanded by the operator or preprogrammed autonomously. Typically, this higher level of input

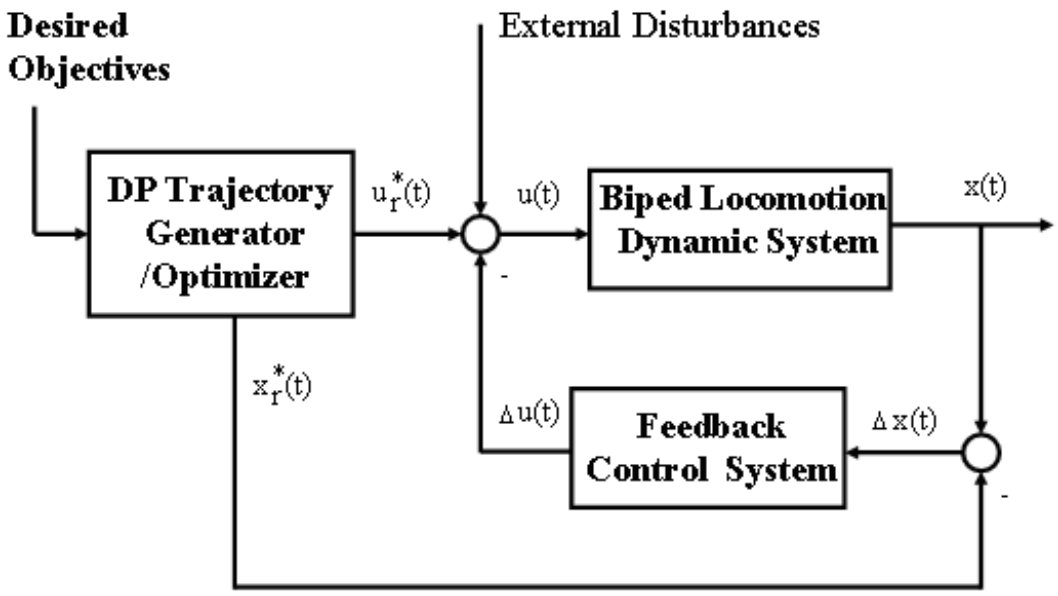

Figure 1: DP architecture consisting of trajectory generator/optimizer and feedback control system.

may consist of set points or regulation, pass through of several way-points, or tracking of specific targets. The key role of dynamic programming in the control systems design is to provide an ideal trajectory optimizer or path planner that can generate arbitrarily smooth input profiles with no initial guess for the input parameters for the biped locomotion system.

In addition, to the early literature review, several other key research efforts have been identified. The main motivation was to identify what work has been completed for leaping, landing, and run. Both contact and flight phases become important components for the control system design. For multi-segment models Requejo, McNitt-gray, and H. Flashner [25], [26] investigate the flight-phase motion for a gymnast dismount from a bar. The model is used to simulate and experimentally validate the advance understanding of flight-phase dynamics to test hypotheses regarding multi-joint control prior to landing. Alexander [27] reviews a wide range of models that have been used in biomechanics. He compares simple models with complex ones and discusses the merits of both. In [28], the control of the body orientation in flight using the internal motion of the leg is discussed. The study provided an additional degree of control to dynamically balance a legged robot during running. Wooten and Hodgins [29] describe a technique for generating transitions between simulated behaviors. By parameterizing individual basis behaviors, one can design control systems such that the exit states of one behavior leaves the simulated character in a valid initial state for the next behavior. This approach was demonstrated with four basis behaviors: leaping, tumbling, landing, and balancing. Control algorithms have been developed that allow an animator to generate a variety of dynamic behaviors by combining parameterized controls for leaping, tumbling, landing, and balancing.

The result of this preliminary literature review identified two candidate models for optimization subject to several constraints based on what particular phase the system may be in, i.e., flight-phase, landing phase, etc. The first model is a simple planar composite two-body model for which the internal body rotation can influence the final conditions of the composite two-body system.

The second model that was identified includes further complexity and is based on the 5-segment model from reference [25].

2.4 Opportunities for Innovation

Despite the high level of research activity in this area, significant improvements are needed in certain areas to achieve performance suitable for deployment for military or disaster response applications. Most research platforms continue to use balancing bars in one dimension; three-dimensional balancing has been achieved only rarely, and not robustly in bipedal robots. Few human-scale or near human-scale robots have managed to exceed about 1.5 m/sec, which is far slower than most humans are capable of running. Perhaps the area where the greatest improvement is needed is in power and energy density; few robots have shown the ability to power their own walking. Even fewer have demonstrated the ability to carry additional loads (BigDog is a notable exception, but it is not a biped). Finally, no biped of which we are aware has demonstrated a robust capability to interact physically with objects in its environment. This requires extra power, the ability to accommodate large forces coming from sources other than the ground that may be high above the center of gravity, the ability to interact stably with an unpredictable range of environment dynamics, and the ability to recover from failures (e.g. falls). All of this leaves the state-of-the-art woefully short of the capabilities needed for useful deployment, and each of these areas presents a significant opportunity to innovate.

3 Monoped Hopping Robot

Simple control algorithms were developed to enable balanced hopping of a hopping robot prototype. A solid model of the prototype is shown in Figure 2. The system consists of a wide rounded base; a flexible steel ring that acts as a spring and damper; a linear actuator that drives a mass vertically, providing hopping energy input; and a rotary actuator that moves a weighted arm, shifting the system center of gravity horizontally for balancing. A significant portion of the prototype's mass (including the mass of both actuators) is concentrated at the top of the ring, rendering the system intrinsically unstable; if released from the vertical, it falls over in the same plane in which the arm swings. In the orthogonal horizontal direction, the width of the base makes the system statically stable. To simplify controller design, control was decoupled into a hop generation controller for providing drive energy and a base positioning controller to provide balancing. Each was analyzed with independent simplified models, and controllers were then implemented on a simulated model of the full dynamics to test function.

3.1 System Model

In physics, resonance is the tendency of a system to oscillate with high amplitude when excited by energy at a certain frequency. This frequency is known as the system's natural frequency of vibration, resonant frequency or eigenfrequency. If a structure is subjected to vibration at its natural frequency, the displacements of that structure will reach a maximum. Figure 3 shows a simple undamped spring-mass system, which is assumed to move only along the vertical direction. It has only one degree of freedom (DOF), because its motion is described by a single coordinate x. When placed into motion, oscillation will take place at the natural frequency, f_n, which is a property of the system. Newton's second law is the first basis for examining the motion of the

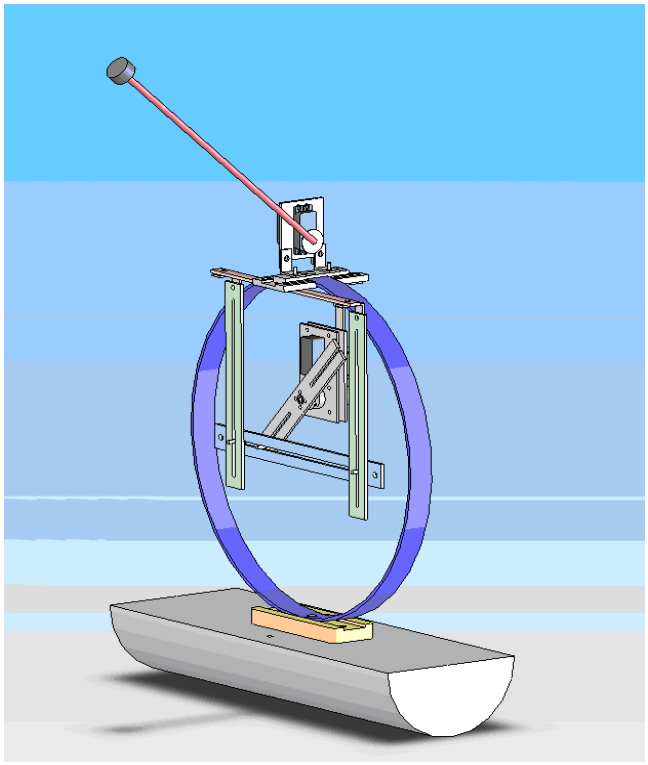

Figure 2: Hopping Monoped Solidworks Model

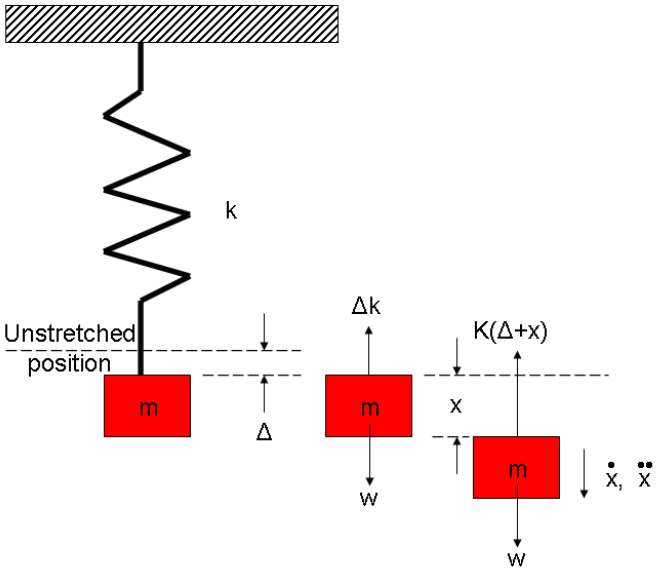

Figure 3: Undamped Spring-Mass System

system. The deformation of the spring in the static equilibrium position is Δ, and the spring force, Δk, is equal to the gravitational force, w, acting on mass m.

$$\Delta k = w = mg \tag{1}$$

By measuring the displacement x from the static equilibrium position, the forces acting on m are $\Delta k(\Delta + x)$ and w. Now apply Newton's second law of motion to the mass m:

$$\sum F = m\ddot{x} = w - k(\Delta + x) \tag{2}$$

Since $\Delta k = w$, we obtain

$$m\ddot{x} = -kx. \tag{3}$$

Defining the natural frequency as

$$w_n^2 = \frac{k}{n}, \tag{4}$$

we can rewrite the equations as

$$\ddot{x} + w_n^2 = 0. \tag{5}$$

Since this is a homogeneous second order linear differential equation, is has the following general solution

$$x = A \sin w_n t + B \cos w_n t \tag{6}$$

where A and B can be evaluated from initial conditions.

Figure 4 shows our system. The base mass is defined as $m1$, $m2$ is the mass used to excite the spring in its natural frequency, and $m3$ is the stabilizing mass. Since $m2$ and $m3$ are rigidly connected, the equation for our systems natural frequency is

$$w_n = \sqrt{\frac{k}{m2 + m3}} \tag{7}$$

Note that this approach has modeled the spring as an ideal spring. Often a damping term is associated with many spring mechanisms, or used to model contact with the ground. A damping term is added to the model in the next section.

3.2 Hop Generation

To analyze and control hop generation, balance and the movement of the weighted arm were neglected. The single-dimensional model used for this controller is shown in Figure 5. The base is modeled as a mass m_1. The flexible ring is modeled as a spring and damper with stiffness k and damping b. The fixed mass at the top of the ring is modeled as a mass m_2; this includes the mass of the swinging arm, which is assumed to be stationary for this portion of the analysis. The moving actuator mass is modeled as the mass m_3, with an actuator force F_3 applied between the two masses (positive upward on m_3 and positive downward on m_2). Contact with the ground was modeled as contact with a stiff spring k_g and damper b_g.

Several different options for driving hopping were considered. Perhaps the simplest approach is simply to oscillate m_3 open-loop, without any sensor feedback, to put energy into the system. This approach can produce hopping, and hopping height is maximized when the drive frequency matches the resonant frequency of the system (as determined by the spring stiffness k and the total above-spring mass $m_2 + m_3$). One way to do this is characterized by the following control law,

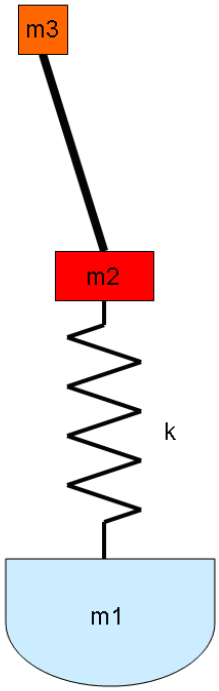

Figure 4: System Model

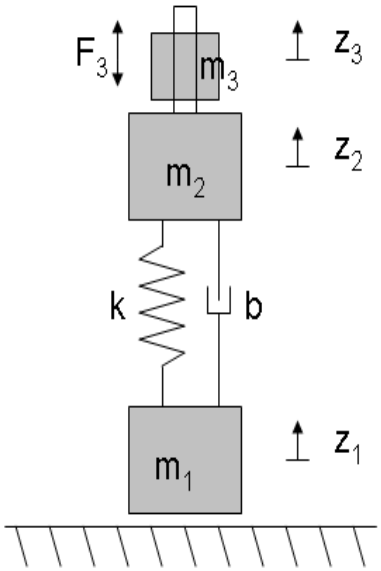

Figure 5: Hopping Robot Dynamic Model

which drives the desired mass position z_{des} (the desired difference between z_2 and z_3) in a sinusoidal trajectory, and closes a proportional-derivative (PD) control loop to determine the actuator input force from desired and measured positions and velocities:

$$z_{des} = z_{max} \sin \omega t \tag{8}$$

$$F_3 = -K_{ph}(z_3 - z_2 - z_{des}) - K_{dh}(\dot{z_3} - \dot{z_2} - \dot{z_{des}}) \tag{9}$$

In these equations, z_{max} is the maximum desired displacement amplitude separating z_3 and z_2. ω is determined by the natural frequency of the spring-mass system created by m_3, m_2, and the spring k. Depending on the mechanical parameters of the system, this approach may require oscillation at relatively high frequencies, which could lead to very high forces if m_3 is large. Additionally, when modest hopping heights are achieved, this approach is rather wasteful of actuator power, because the actuator continues to needlessly drive the system at its resonant frequency as it flies through the air, with no net effect on the system energy. Net potential energy is only imparted to the system when it contacts the ground and the spring can be compressed.

An improved approach is to coordinate the drive actuator motion with the robot's cycle of coming in and out of contact with the ground as it hops. Work done on the system is maximized if m_3 is driven upward when the spring is maximally compressed, which occurs at the bottom of the system travel [30]. Similarly, the least energy is removed from the system if m_3 is moved downward when the spring is extended and unloaded, which occurs when the robot is airborne. This approach requires sensory feedback, but a simple implementation could use a proximity sensor or limit switch to detect contact with the ground. In this case, z_{des} depends on whether or not the system is in contact with the ground. If the robot is contacting the ground ($z_1 <= 0$):

$$z_{des} = z_{max} \tag{10}$$

If the robot is airborne ($z_1 > 0$):

$$z_{des} = -z_{max} \tag{11}$$

The control law applied is then the same as in equation 9. When contact with the ground begins or ends, the desired position switches and the separation between m_3 and m_2 follows a step response, the shape of which is dictated by the control gains and the dynamics of the actuator. When the base of the robot contacts the ground, the controller rapidly moves m_3 to its maximum vertical displacement relative to m_2, compressing the spring. When it leaves the ground, m_3 is brought down relative to m_2, restoring the travel that is needed for the next hop. The result is a controller that provides a "kick each time the base contacts the ground.

Figure 6 shows the simulated hopping height of the robot with a feedforward controller and with the simple feedback controller described above. The distance between the ground and the robot base is plotted. The parameters used for this simulation were $m_1 = 0.3kg, m_2 = 1.5kg, m_3 = 0.5kg, k = 10000N/m, b = 12.25Ns/m, K_{ph} = 10000$, and $K_{dh} = 100$. The ground stiffness was assumed to be $5000000N/m$ and the ground damping $10000Ns/m$. The feedforward controller drives at the resonant frequency determined by the spring stiffness and the total mass above the spring, without knowledge of the robot's states. The feedback controller first drives at this same frequency until the robot becomes airborne; then it follows the algorithm described above. In both cases the actuator force is capped at the same level (100 N), simulating saturation.

The switching (feedback) controller reaches its asymptotic hopping height more quickly and more consistently than the feedforward controller, because movement of the masses is synchronized

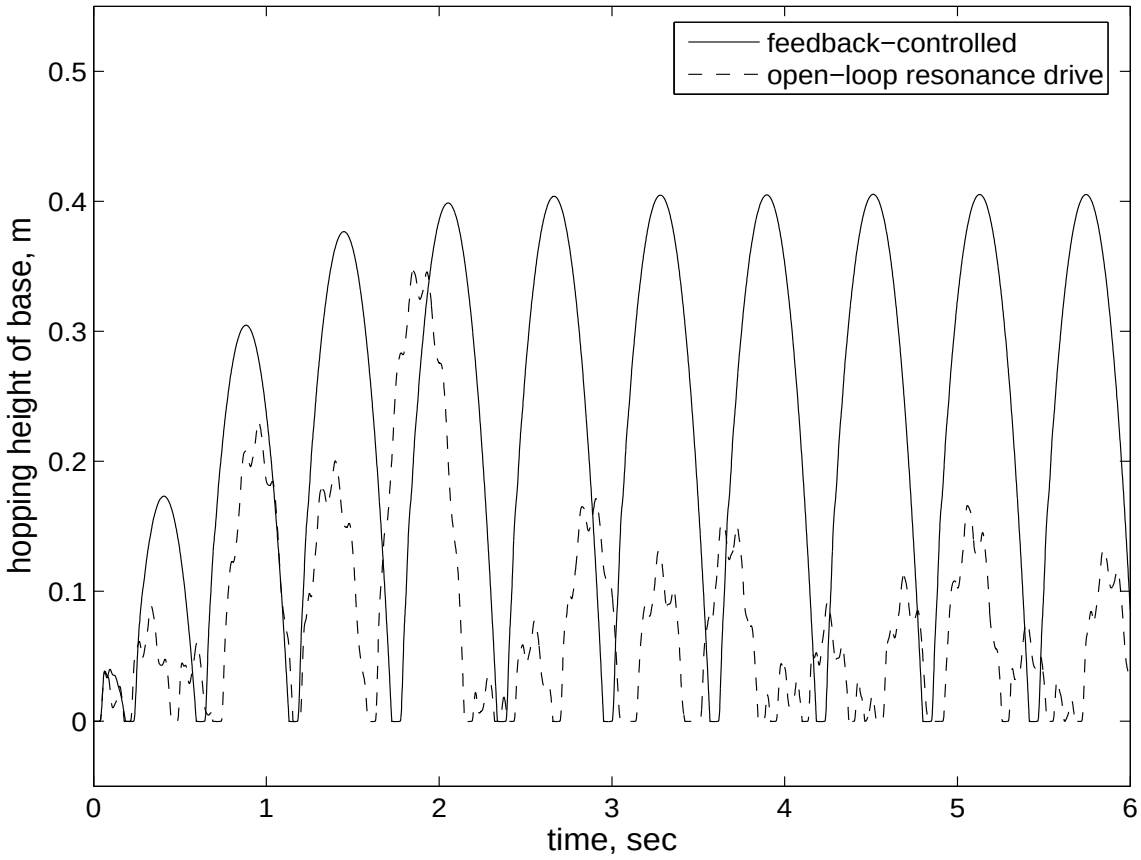

Figure 6: Plot of Hopping Robot Performance

with the hopping cycle. The feedforward trajectory also shows a significant amount of excess movement while airborne, which results from unnecessary oscillations of the upper two masses in flight. Though simple, the feedback controller is clearly superior to the feedforward resonating controller, and can be implemented with only binary contact sensors. This provides one straightforward way to generate hopping.

3.3 Base Positioning

For balancing, the simplest model may neglect the system compliance and movement of the driving mass, but must include gravity and planar motion. The model in Figure 7 shows the two-link model used to represent planar balancing. Because the robot is intended to spend a significant fraction of its duty cycle airborne, one control approach exploits the airborne time to position the links in order to facilitate upright bouncing (as opposed to tipping over). Thus when this controller is used with the hop generation controller, the goal is to position the robot for an upright (stable) bounce while airborne, and provide it with a burst of energy input while in contact with the ground to send it sufficiently high in the air to again align itself for another stable bounce. The balancing can in theory be accomplished by horizontally aligning the point on the robot of expected contact with the ground (the base) with the robots center of gravity. When in contact with the ground, the point of contact with the ground is assumed to act as a pin joint. If the center of gravity is positioned directly above the pin or very close, the torque due to gravity that acts around that pin is very small; alternatively, if the center of gravity is horizontally far from the pin joint, a significant gravitational torque acts on the robot, tipping it toward the ground. While there is always some torque due to gravity unless alignment is perfect, the robot only needs to avoid tipping for the relatively short period of contact with the ground.

The most desirable hopping controllers do not require knowledge of the location of the ground to traverse and provide stable hopping. If the balancing controller is ignorant to the ground location, it must seek to have the robot base position track the motion of the center of gravity throughout the airborne phase of the robots travel, such that at any time if it contacts the ground, it can manage contact and produce a subsequent stable hop.

While the system is airborne (i.e. in free fall), the controller should act to align the position of the base of the robot x_{foot} with the position of the center of gravity, which is determined by the centers of gravity of each of the two links. For this simplified analysis, the lower link mass m_l is assumed to include the mass of the base as well as that of the mass fixed to the top of the ring and the linearly-moving drive mass (the ring compliance is neglected as it plays no significant role in free fall). The upper link mass m_a is assumed to include only the mass of the moving rod and any mass fixed to it. The instantaneous position of the center of gravity of the two-link system can be determined by:

$$x_{cg} = \frac{m_a}{m_a + m_l} x_a + \frac{m_l}{m_a + m_l} x_l \tag{12}$$

In this expression, x_{cg} is the horizontal position of the center of gravity relative to some fixed position, and x_a and x_l are the positions of the centers of gravity of the upper and lower link, respectively, with respect to the same reference. This quantity must be compared to the position of the base x_{foot} with respect to the same fixed reference. Alternatively, the difference between these quantities can be directly computed using only the absolute angles of the links relative to the ground frame, Θ_a and Θ_l, as measured by a tilt sensor on each body.

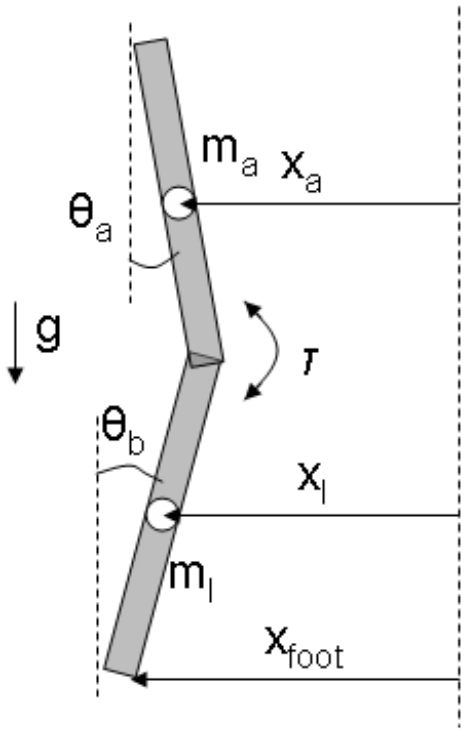

Figure 7: Balance Model

Because this difference is to be minimized, one simple controller would apply a torque between the two joints that is proportional to the error:

$$\tau_1 = K_{pb}(x_{cg} - x_{foot}) \tag{13}$$

Because there is no damping in the system, this results in large oscillations about the correct alignment. This can be corrected by including a damping term:

$$\tau_2 = K_{pb}(x_{cg} - x_{foot}) + K_{db}(\dot{x}_{cg} - \dot{x}_{foot}) \tag{14}$$

The correct signs for the gains K_{pb} and K_{db} are not obvious. For the cases simulated, in which $m_a << m_l$ and initial angles were restricted to be within 30 degrees of vertical, it was found that stable solutions resulted only when both gains were negative.

Figure 8 shows simulation results of the difference between the horizontal position of the robot base and the system center of gravity versus time for several different controlled initial conditions, for a robot in free fall. The masses used for this simulation were $m_a = 0.5kg$ and $m_l = 1.8kg$; the latter includes the mass of all parts except the moving arm. The length of the lower link was assumed to be 0.5 m with its center of gravity located halfway down. The length of the upper link was assumed to be 0.3 m. The control gains used were $K_{pb} = 10$ and $K_{db} = 2$. Note that in each case the error vanishes as the controller acts. This approach works effectively in simulation even for cases when the system has a modest amount of angular momentum; no control input could slow net system rotation, but for reasonable amounts of time the base position can continue to track the center of gravity. While this simulation alone does not demonstrate stable hopping, it does illustrate an effective method to align the center of gravity with the base, which should encourage stable repeated bounces and prevent tipping.

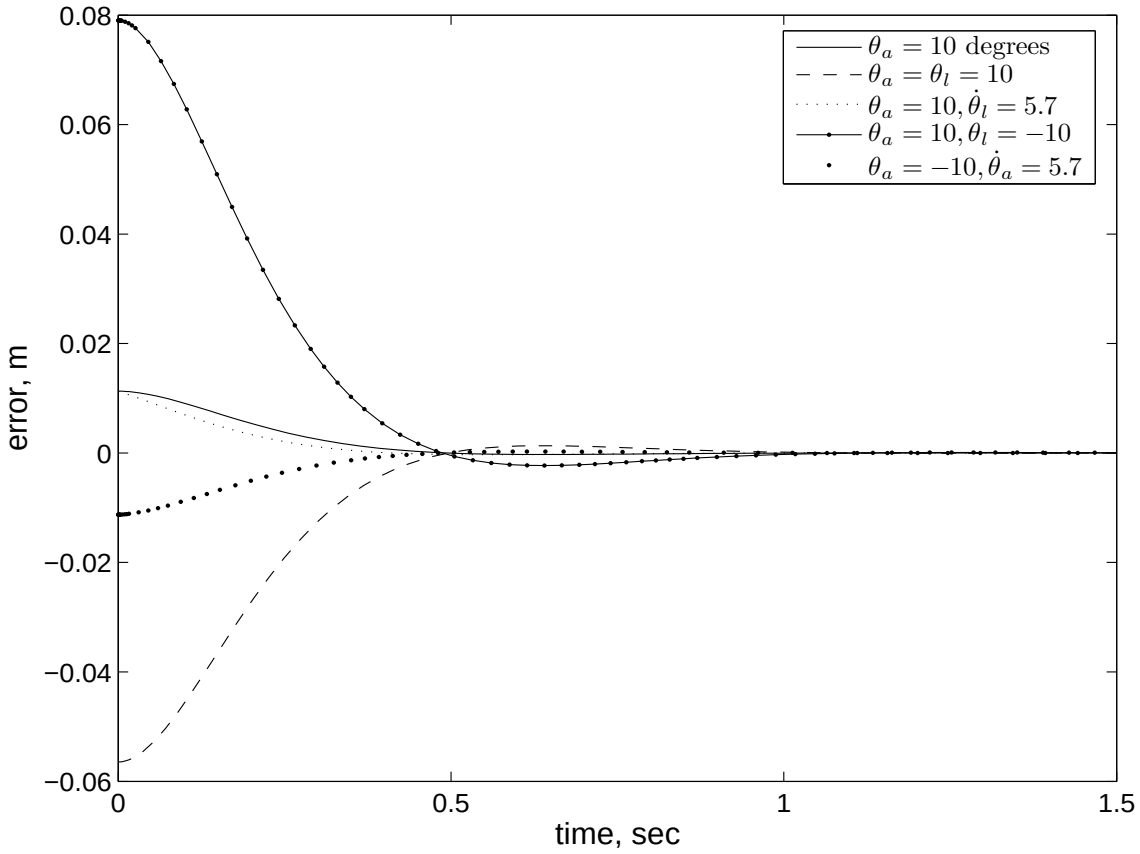

Figure 8: State Trajectory for Center of Gravity Controller

4 Full Dynamic Simulation

In addition to the independent validation of the two control approaches described above, initial simulation was performed on the full nine degree of freedom system dynamics. The hop generation controller and the base positioning controller were applied to the system model described in Section 4.2. These simulations confirm that the hop generation controller can indeed inject energy into the system and increase hopping height, and that the base positioning controller can align the center of gravity and the point of contact with the ground in ballistic flight in preparation for contact with the ground. However, sustained balanced hopping has to date not been demonstrated in the simulation for a wide range of initial conditions. Study of the simulations suggests several possible challenges that must be overcome. Impact with the ground appears to substantially disrupt the base-positioning controller, even when the controller is switched off during periods of contact. One problem is that although the center of gravity may align vertically with the base at impact, the spring and its drive actuator are not necessarily oriented vertically. This means that the spring force contributes a significant amount of angular momentum, which arm movement cannot affect. The arm rotates continually around in an attempt to position the base correctly, but the angle of the spring at impact cannot be specified by this controller. After several impacts, the angular momentum may be too large to manage, and the robot crashes.

It may be possible to construct a controller that balances the need to align the center of gravity

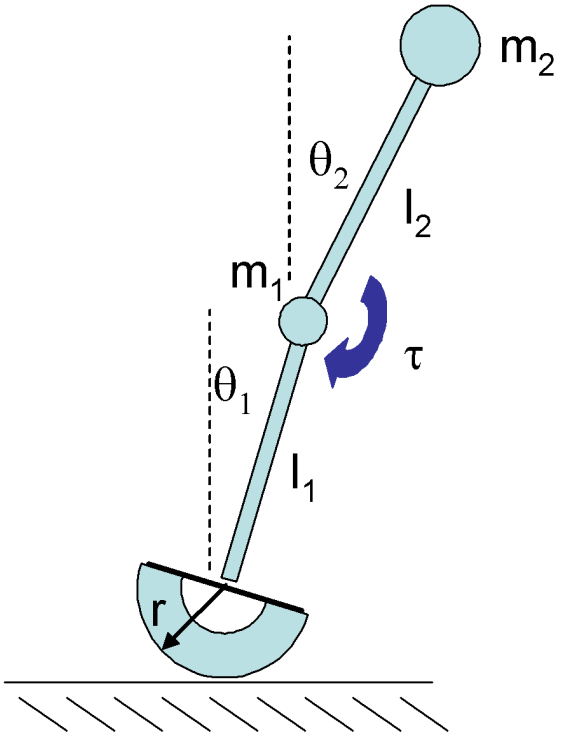

Figure 9: Balanced Monoped Dynamic Model

with the base with the need to bring the spring leg close to vertical at impact. It is unknown whether this can be done effectively without knowledge of the location or time of ground impact; perhaps some estimate of this will be needed to make control reliable. Despite the limitations of the primitive approach described here, it has shown some promise to improve balance and to provide at least two to three stable hops before crashing, as well as to generate hopping energy. While this is an inadequate solution, it is done with a minimum of sensory feedback and using two extremely simple algorithms; the approach could almost certainly be improved by using additional information.

4.1 Balance Control

The desired dynamics for this balanced monoped robot are similar to that of a double inverted pendulum. The system in Figure 9 is defined with masses m_1 and m_2, rod lengths l_1 and l_2, angles θ_1 and θ_2 defined with respect to the vertical, and foot with radius of curvature r. The double inverted pendulum state is achieved when both angles θ_1 and θ_2 are near zero. To begin, we will consider the simpler case of a leg with no foot. In other words, we consider the system with $r = 0$ whereby the bottom rod is "pinned" to the floor. For this system, the driven dynamic response is found using the Euler-Lagrange equation and is given by equations 15-18 shown below. See Appendix A for the full derivation. Balance of the system will be achieved by developing a state feedback controller to apply a control torque to the joint linking the two rods.

$$\dot{\theta_1} = \omega_1 \tag{15}$$

$$\dot{\theta_2} = \omega_2 \tag{16}$$

$$\dot{\omega_1} = \frac{-m_2 l_1 \omega_1^2 \cos\psi \sin\psi - m_2 g \cos\psi \sin\theta_2}{l_1(m_1 + m_2 \sin^2\psi)} + \frac{-m_2 l_2 \omega_2^2 \sin\psi + g(m_1+m_2)\sin\theta_1}{l_1(m_1 + m_2 \sin^2\psi)} - \frac{\tau \cos\psi}{l_1 l_2 (m_1 + m_2 \sin^2\psi)} \quad (17)$$

$$\dot{\omega_2} = \frac{m_2 l_2 \omega_2^2 \sin\psi \cos\psi - g(m_1+m_2)\sin\theta_1 \cos\psi}{l_2(m_1 + m_2 \sin^2\psi)} + \frac{(m_1+m_2)(l_1\omega_1^2 \sin\psi + g\sin\theta_2)}{l_2(m_1 + m_2 \sin^2\psi)} + \frac{\tau(m_1+m_2)}{l_2^2 m_2 (m_1 + m_2 \sin^2\psi)} \quad (18)$$

where $\psi = \theta_1 - \theta_2$. Though this system is nonlinear, the state equations 15-18 are continuously differentiable. Therefore, local linearization about an equilibrium point is possible [31]. By linearizing the system about a given equilibrium point, the dynamics of the system may be adequately described by a linear set of system equations having the form shown in 19. This allows linear state feedback control methods to be calculated that will maintain stability as long as the system remains in the neighborhood of that equilibrium point.

$$\dot{x} = Ax + B\tau \quad (19)$$

where x is the vector $x = [\theta_1\ \omega_1\ \theta_2\ \omega_2]^T$, τ is a scalar valued input torque, A is the Jacobian matrix of the system evaluated at an equilibrium point, and B is the vector sensitivity of the system to the input torque [32]. For a linearized system, A and B are calculated as shown in 20 and 21.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{d\dot{\omega}_1}{d\theta_1} & \frac{d\dot{\omega}_1}{d\omega_1} & \frac{d\dot{\omega}_1}{d\theta_2} & \frac{d\dot{\omega}_1}{d\omega_2} \\ 0 & 0 & 0 & 1 \\ \frac{d\dot{\omega}_2}{d\theta_1} & \frac{d\dot{\omega}_2}{d\omega_1} & \frac{d\dot{\omega}_2}{d\theta_2} & \frac{d\dot{\omega}_2}{d\omega_2} \end{bmatrix} \quad (20)$$

$$B = \begin{bmatrix} \frac{d\dot{\theta}_1}{d\tau} \\ \frac{d\dot{\omega}_1}{d\tau} \\ \frac{d\dot{\theta}_2}{d\tau} \\ \frac{d\dot{\omega}_2}{du\tau} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{-\cos(\theta_1-\theta_2)}{l_1 l_2 (m_1 + m_2 \sin^2(\theta_1-\theta_2))} \\ 0 \\ \frac{m_1+m_2}{l_2^2 m_2 (m_1 + m_2 \sin^2(\theta_1-\theta_2))} \end{bmatrix} \quad (21)$$

The nonlinear system described in equations 15-18 contains two plausible equilibrium points at $(\theta_1, \omega_1, \theta_2, \omega_2) = (0, 0, 0, 0)$ and $(\theta_1, \omega_1, \theta_2, \omega_2) = (0, 0, \pi, 0)$. The system is more interesting at the equilibrium point $(\theta_1, \omega_1, \theta_2, \omega_2) = (0, 0, 0, 0)$ (the origin) since it gives a double inverted pendulum; so, we linearize about this point.

If we consider the system with $m_1 = 0.2$ kg, $m_2 = 0.1$ kg, $l_1 = 0.3$ m, and $l_2 = 0.3$ m linearized about the origin, the calculated A and B matrices are given by equations 22 and 23.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 49 & 0 & -16.333 & 0 \\ 0 & 0 & 0 & 1 \\ -49 & 0 & 49 & 0 \end{bmatrix} \quad (22)$$

$$B = \begin{bmatrix} 0 \\ -55.56 \\ 0 \\ 166.67 \end{bmatrix} \tag{23}$$

The controllability matrix of this system is calculated and given by equation 24.

$$C = \begin{bmatrix} B & BA & B^2A & B^3A \end{bmatrix} = \begin{bmatrix} 0 & -55.56 & 0 & -5444 \\ -55.56 & 0 & -5444 & 0 \\ 0 & 166.67 & 0 & 10889 \\ 166.67 & 0 & 10889 & 0 \end{bmatrix} \tag{24}$$

The controllability matrix C is full rank with non-trivial eigenvalues; therefore, the system is controllable using state feedback control in the neighborhood of the origin. Ackerman's formula is used to reassign eigenvalues to the controlled system by finding a feedback gain to apply according to equation 25 [33].

$$\tau = Kx = [k_1 \ k_2 \ k_3 \ k_4][\theta_1 \ \omega_1 \ \theta_2 \ \omega_2]^T \tag{25}$$

The poles are chosen somewhat arbitrarily to be $[\lambda_1 \ \lambda_2 \ \lambda_3 \ \lambda_4] = [-5 \ -5 \ -5 \ -5]$. This pole placement results in a feedback control gain of $K = [3.93 \ 0.636 \ -0.179 \ 0.0918]$. The controlled system is simulated using MATLAB's ODE solver. The stable state trajectory for the controlled system is seen in Figure 10. The phase diagram is given in Figure 11. Note that the state approaches the origin, which is the dual inverted pendulum state, from a non-zero initial condition. The controlled system was animated in MATLAB to better illustrate its behavior. Figure 12 shows the pose of the double inverted pendulum at roughly t=0.1, t=0.50, and t=1.0 seconds.

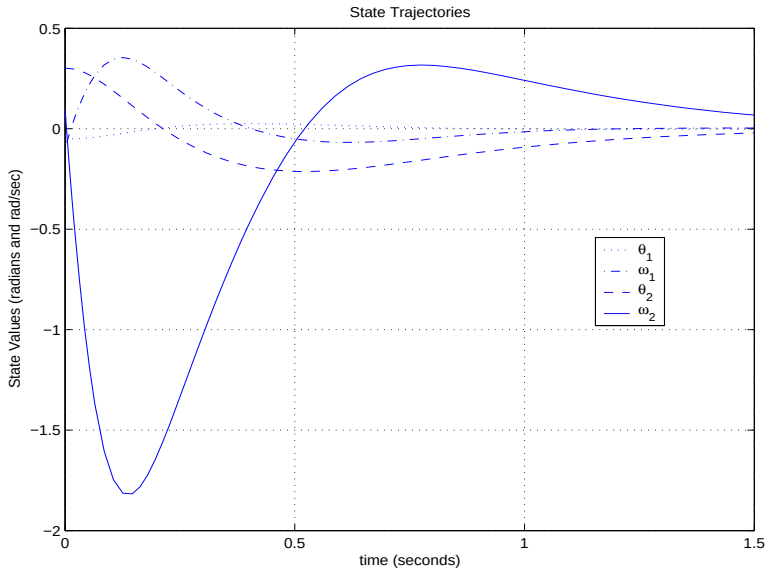

Figure 10: Stable State Trajectory

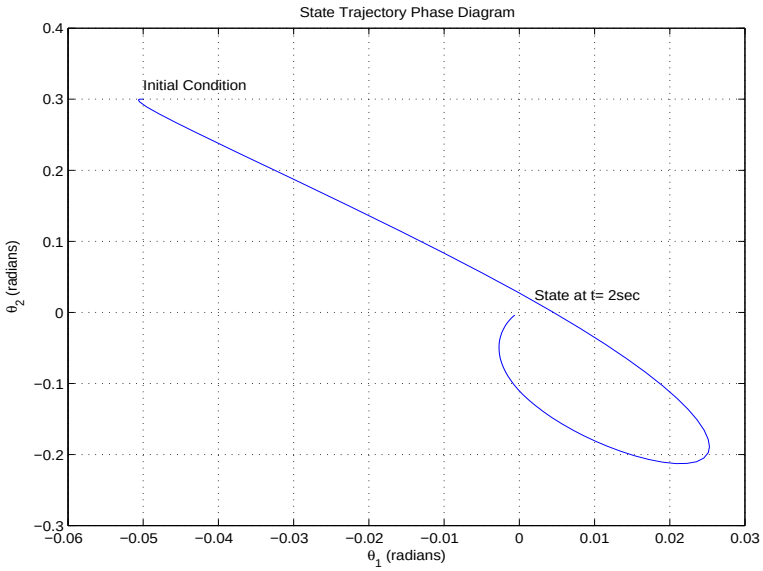

Figure 11: State Trajectory Phase Diagram

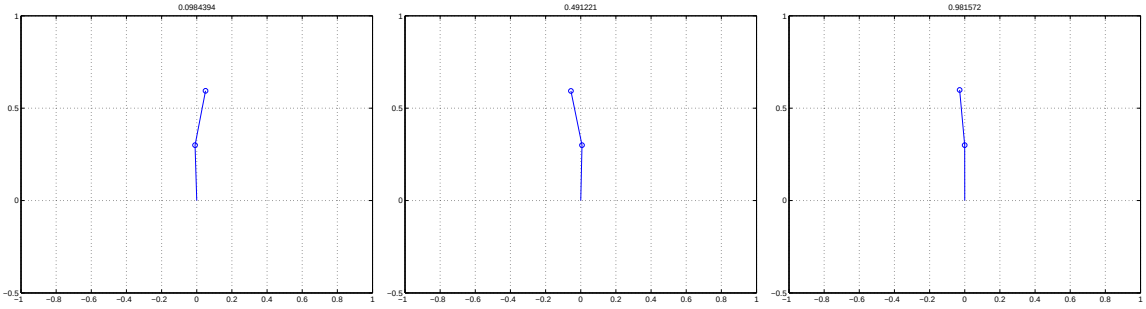

Figure 12: Balance Controller Animation

Next, we consider the analysis for the system with a foot having radius of curvature $r > 0$. For this system, the dynamic equations are given by equations 15-16 and 26-27 below. See Appendix B for the full derivation.

$$\begin{aligned}
\dot{\omega_1} &= \frac{-m_2 l_2 (l_1 \cos\psi + r\cos\theta_2)(l_1\omega_1^2 \sin\psi + g\sin\theta_2) - m_2 l_2^2 (l_1 \sin\psi - r\sin\theta_2)\omega_2^2}{\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2} \\
&\quad + \frac{l_1 l_2 \sin\theta_1 (m_1+m_2)(g + r\omega_1^2)}{\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2} - \frac{\tau(l_1\cos\psi + r\cos\theta_2)}{\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2} \qquad (26) \\
\dot{\omega_2} &= \frac{(l_1^2 + 2rl_1\cos\theta_1 + r^2)(m_1+m_2)(l_1\omega_1^2\sin\psi + g\sin\theta_2)}{\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2} \\
&\quad + \frac{(l_1\cos\psi + r\cos\theta_2)[m_2 l_2 (l_1\sin\psi - r\sin\theta_2)\omega_2^2 - l_1\sin\theta_1 (m_1+m_2)(g + r\omega_1^2)]}{\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2} \\
&\quad + \frac{\tau(l_1^2 + 2rl_1\cos\theta_1 + r^2)(m_1+m_2)}{m_2 l_2 (\Delta - m_2 l_2 (l_1\cos\psi + r\cos\theta_2)^2)} \qquad (27)
\end{aligned}$$

where $\psi = \theta_1 - \theta_2$ and $\Delta = l_2(l_1^2 + 2rl_1\cos\theta_1 + r^2)(m_1 + m_2)$

As will be illustrated by simulation, the addition of a curved foot improves the system transient response. Significant improvements only arise for somewhat large values of r. Consider again the previous example system with $m_1 = 0.2$ kg, $m_2 = 0.1$ kg, $l_1 = 0.3$ m, and $l_2 = 0.3$ m linearized about the origin. We shall simulate this system instead for $r = 0$ $(l_1 = 0.3m)$, $r = 10cm$ $(l_1 = 0.2m)$ and $r = 20cm$ $(l_1 = 0.1m)$ and compare the response. Note that we are preserving the quantity $r + l_1 = 0.3m$, the maximum height of mass m_1. As with the last system, we use Ackerman's formula to place the poles at $[\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4] = [-5\ \ -5\ \ -5\ \ -5]$. We find that the necessary gains are $K_{r=0} = [3.93\ 0.636\ 0.179\ 0.0918]$, $K_{r=10cm} = [3.80\ 0.773\ \ -0.122\ 0.138]$, and $K_{r=20cm} = [4.03\ 1.19\ 0.0500\ 0.275]$. The simulation results are given in Figures 13, 14, and 15. In particular, note that for the same pole placement, the system requires less control torque as r increases. In addition to the reduced control torque, Figure 13 shows less overshoot for ω_2 and θ_2 as r increases. Figure 14 also shows less overshoot for θ_2 for larger r.

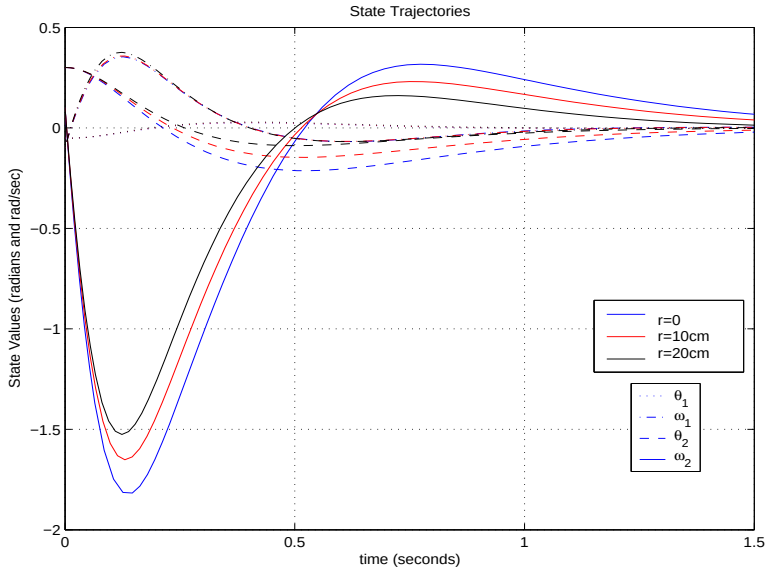

Figure 13: Stable State Trajectory

In practice, after considering the model in Figure 9, one might criticize the need for such a large r value in relation to l_1, since this would result in a very large foot. However, the model was generated to complement the analysis. In practice, the foot would not be large, but rather, just the curved portion contacting the ground would have a large radius of curvature. It is for this reason that we preserved the quantity $l_1 + r = 0.3m$ for the simulation since we did not want to consider how r would increase the height of the first link.

4.2 3-Dimensional Model

Three dynamics simulations were created to support this LDRD. Each simulation is more complex than its predecessor, and each was designed to study a certain aspect of controlling a moving body that must be oriented before landing. These models are described below in increasing order of complexity.

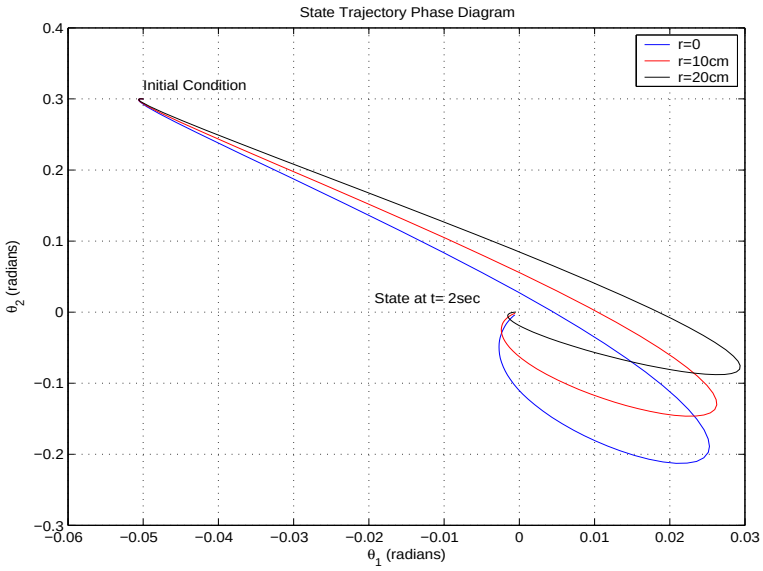

Figure 14: State Trajectory Phase Diagram

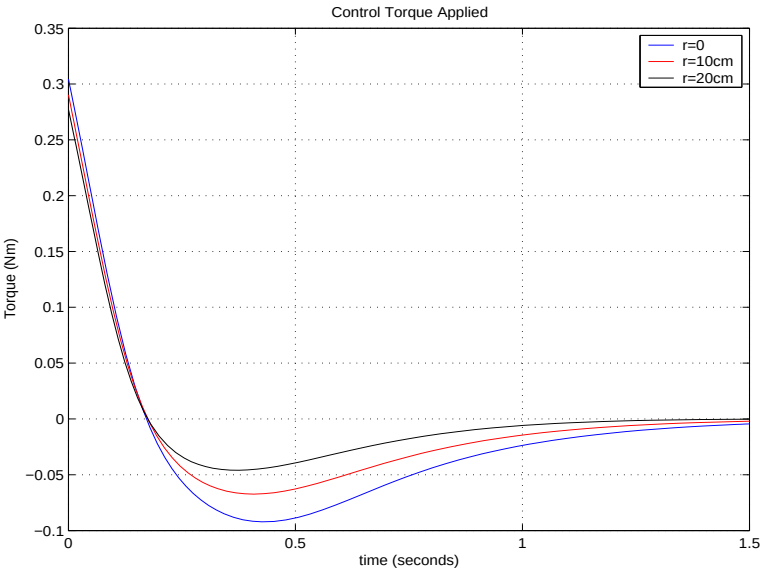

Figure 15: Control Torques

(Need Daves picture of the box and mass)

Figure 16: Rotating Mass System

4.2.1 Moving mass inside a box

The purpose of the first model was to understand how a moving mass could be used to reorient a body in flight. This model is a 7 degree-of-freedom (7 DOF) system consisting of two rigid bodies, the box and a rotating mass inside the box. The box is a 6 DOF system with three translational and three rotational degrees of freedom in the inertial frame. The seventh degree of freedom is the rotational motion of the moving mass about an axis fixed in the box. This system is shown in figure 16. No ground interaction is used in this model since the purpose of the model is solely to understand reorientation during the flight phase. The axis of rotation inside the box may be placed at any orientation with respect to the box. Once this axis is specified, the axis remains fixed with respect to the box during a simulation run. A control torque is applied to the moving mass about the fixed axis of rotation.

4.2.2 Gymnast

This model is a planar model consisting of five rigid segments and follows the gymnast simulation introduced in [26]. Since a humanoid robot is one of the configurations of interest for this LDRD, it was decided that the study in [26] would be a good starting point for a controller study. The system is shown in Figure 17. Ground interaction is not considered for the same reason as that given in the previous model. The five rigid bodies form a 7 DOF system which is constrained to remain in a vertical plane. The torso is given two translational degrees of freedom and one rotational degree of freedom in the inertial frame. The head, thigh and arm are each given one rotational degree of freedom with respect to the torso. The calf is given one rotational degree of freedom with respect to the thigh. Control torques are applied at the hip, knee, shoulder and neck joints.

4.2.3 Hopping robot

This simulation models the 9 DOF system shown in Figure 18. The goal of the simulation is to control this unstable system such that the robot lands in a stable orientation and then hops back into the air. The lightweight lower section (lander) has six degrees of freedom in the inertial frame. The heavy upper section has one translational degree of freedom with respect to the lander. This degree of freedom is the vertical position due to the deflection of the spring connecting the lander and the upper section. The motion of the linear actuators point mass with respect to the upper section adds a translational degree of freedom. The final degree of freedom is the rotational motion of the swinging rod with respect to the upper section. A control force is used to drive the linear actuator and a control torque is applied to the rotating arm. A detailed description of this model appears in Appendix C.

5 Power Considerations

An actively stable robot poses clear control challenges, but an often unconsidered problem is that of power utilization. In most tracked or wheeled robots, the main drive motors dominate the power budget. However, tracked or wheeled mobility platforms enjoy passive stability. When the

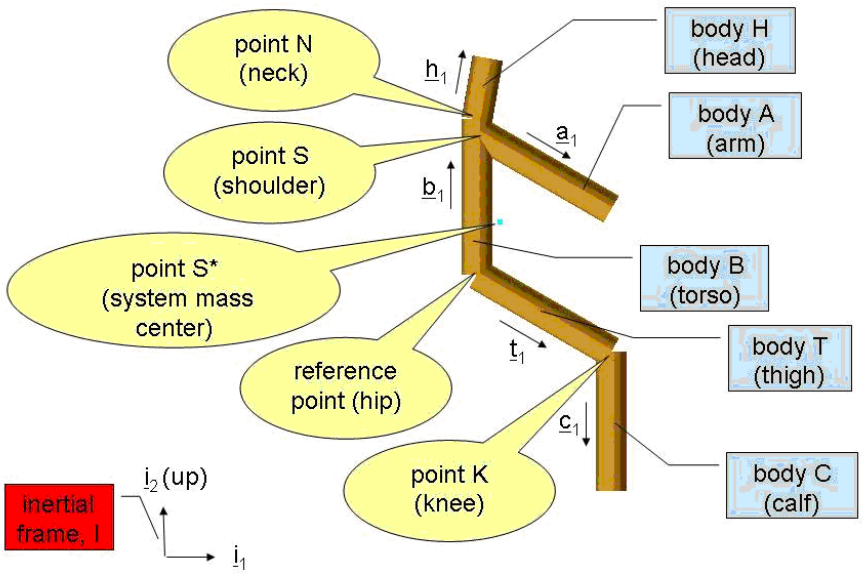

Figure 17: Gymnast Model

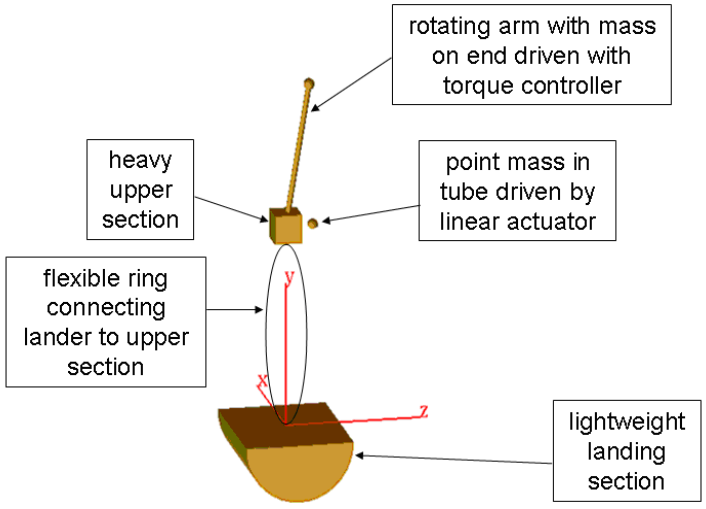

Figure 18: Ring Model

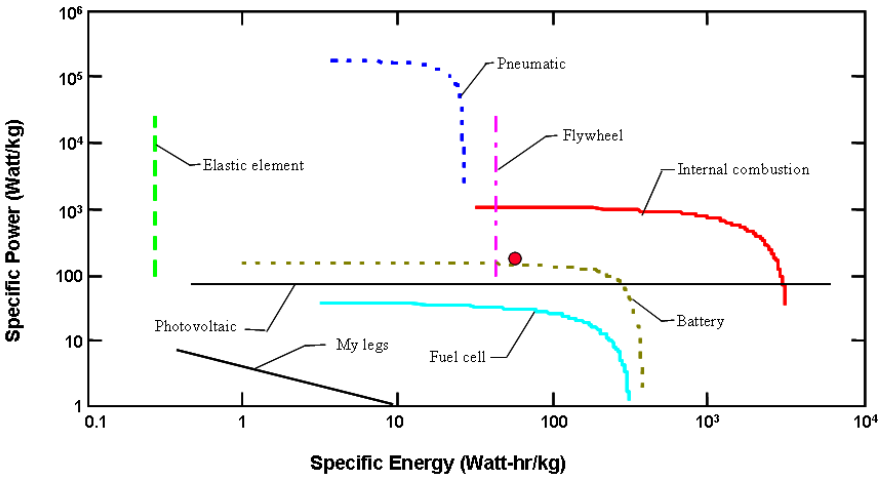

Figure 19: Ragone Plot for Various Energy Sources

passively stable robot is idle, it needs no power for its drive motors. However, for the actively stabilized motor-driven robot, even an idle, unmoving platform uses power continuously. A motor driven bipedal robot may need a dozen motors or more to maintain a pose. Even while just standing, each motor must consume current to provide the necessary torque just to stay static. A Ragone plot for various energy sources appears in Figure 19. An internal combustion engine optimizes the maximum specific energy and specific power of all energy sources shown in Figure 19. Therefore, for maximum operational life, an internal combustion engine is the best energy source given today's battery and fuel cell technologies.

6 Summary and Conclusions

This report contains the results of a research effort on advanced robot locomotion. The majority of this work focuses on walking robots. We review potential control architectures and propose an architecture based on Dynamic Programming Techniques. We also develop a system model for a balanced hopping robot and analyze two potential control strategies. The first attempts to reach a controlled resonance to hop while the second attempts to coordinate the drive actuator motion with the robot's cycle of coming in and out of contact with the ground as it hops. Two different approaches for balance control are then developed. The first uses a kinematic balance model while the second approach derives the dynamics using the Euler-Lagrange equation. Simulation results are presented for stability regions as well as system performance. Kane's method was applied to develop a nine degree of freedom system model and the detailed derivations appear in Appendix C. Power considerations were briefly discussed, and a Ragone plot of different power sources makes the argument for an internal combustion engine power source. Future research will include hardware validations and further investigations into potential energy sources.

References

[1] F. Grass, A. D'Arrigo, S. Colombi, and A. Rufer, "Joe: A mobile, inverted pendulum," *IEEE Trans. on Industrial Electronics*, vol. 49, pp. 107–114, February 2002.

[2] J. Searock, B. Browning, and M. Velosos, *Learning to Prevent Failure State for a Dynamically Balancing Robot*. PhD thesis, Carnegie Mellon University, 2005.

[3] Honda Motor Company Website. http://asimo.honda.com.

[4] Sony Corporation Website. http://www.sony.net/SonyInfo/QRIO/story /index_nf.html, 2006.

[5] S. Kajita, T. Nagasaki, K. Kaneko, K. Yokoi, and K. Tanie, "A hop towards running humanoid biped," *Proc. International Conference on Robotics and Automation*, pp. 629–635, 2004.

[6] M. H. Raibert, *Legged Robots that Balance*. Cambridge, MA: The MIT Press, 1986.

[7] D. Robinson, J. Pratt, D. Paluska, and G. Pratt, "Series elastic actuator development for a biomimetic walking robot," *Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, pp. 561–568, 1999.

[8] J. Pratt, P. Dilworth, and G. Pratt, "Virtual model control of a bipedal walking robot," *IEEE International Conference on Robotics and Automation*, pp. 193–198, 1997.

[9] S. Hyon, S. Abe, and T. Emura, "Development of a biologically-inspired robot kenkenii," *Proc. 6th Japan-France Congress on Mechatronics and 4th Asia-Europe Congress on Mechatronics*, pp. 404–409, 2003.

[10] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. C. de Wit, and J. W. Grizzle, "RABBIT: a testbed for advanced control theory," *IEEE Control Systems Magazine*, no. 5, pp. 57–79, 2003.

[11] T. Simonite, "Speedy robot legs it to break record." New Scientist.com News Service,http://www.newscientisttech.com/article/dn8957-speedy-robot-legs-it-to-break-record.html, 2006.

[12] Boston Dynamics Website. http://www.bostondynamics.com/content /sec.php?section=BigDog, 2006.

[13] S. Talebi, I. Poulakakis, E. Papadopoulos, and M. Buehler, "Quadruped robot running with a bounding gait," *International Symposium on Experimental Robotics*, 2000.

[14] C. Azevedo, P. Poignet, and B. Espiau, "Artificial locomotion control: from human to robots," *Robotics and Autonomous Sytems*, no. 47, pp. 203–223, 2004.

[15] F. Asano, Z.-W. Luo, and M. Yamakita, "Biped gait generation and control based on a unified property of passive dynamic waling," *IEEE Transactions on Robotics*, pp. 1–9, 2005.

[16] M. Spong and F. Bullo, "Controlled symmetries and passive walking," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1025–1031, July 2005.

[17] Q. Huang and Y. Hakamura, "Sensory reflex control for humanoid walking," *IEEE Transactions on Robotics*, vol. 21, pp. 977–984, Oct 2005.

[18] E. Westervelt, J. Grizzle, and D. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, pp. 42–56, Jan 2003.

[19] F. Asano and M. Hamakita, "Virtual gravity and coupling control for robotic gait synthesis," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 31, pp. 737–745, November 2001.

[20] J. Bertram and A. Ruina, "Multiple walking speed-frequency relations are predicted by constrained optimization," *Journal of Theoretical Biology*, vol. 209, pp. 445–453, 2001.

[21] Muraro, C. Chevallereau, and Y. Aoustin, "Optimal trajectories for a quadruped robot with trot, amble and curvet gaits for two energetic criteria," *Multibody Systems Dynamics*, vol. 9, pp. 39–62, 2003.

[22] J. Morimoto, G. Zeglin, and C. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, (Las Vegas, Nevada), pp. 1927–1932, Octobeer 2003.

[23] M. Stilman, C. Atkeson, J. Kuffner, and G. Zeglin, "Dynamic programming in reduced dimensional spaces: Dynamic planning for robust biped locomotion," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Barcelona, Spain), pp. 2410–2415, April 2005.

[24] R. D. R. III, D. G. Wilson, G. R. Eisler, and J. E. Hurtado, "Applied dynamic programming for optimization of dynamical systems, advances in design and control," in *Society for Industrial and Applied Mathematics, SIAM*, (Philadelphia, PA), July 2005.

[25] P. Requejo, J. McNitt-Gray, and H. K. Flashner, "An approach for developing an experimentally based model for simulating flight-phase dynamics," *Biol. Cybern.*, no. 87, pp. 289–300, 2002.

[26] P. Requejo, J. McNitt-Gray, and H. K. Flashner, "Modification of landing conditions at contact via flight phase control," *Biol. Cybern.*, no. 90, pp. 327–336, 2004.

[27] R. M. Alexander, "Modelling approaches in biomechanics," *Phil. Trans. R. Soc. Lond. B(2003)*, no. 358, pp. 1429–1435, 2003.

[28] Z. Li, "Dynamics and optimal control of a legged robot in flight phase," in *IEEE ICRA*, pp. 1816–1821, 1990.

[29] W. Wooten and J. Hodgins, "Simulating leaping, tumbling, landing, and balancing humans," in *Proceedings of the 2000 IEEE ICRA*, (San Francisco, Ca), pp. 656–662, April 2000.

[30] M. H. Raibert, "Hopping in legged systems modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, no. 3, pp. 451–463, 1984.

[31] H. K. Kahlil, *Nonlinear Systems 3rd.* Englewood Cliffs, NJ: Prentice Hall, 2002.

[32] C. Chen, *Linear System Theory and Design 3rd.* New York, NY: Oxford University Peess, 1999.

[33] P. J. Antsaklis and A. N. Michel, *Linear Systems.* New York, NY: McGraw Hill, 1997.

A Dynamic Equation Derivation for Balanced Monoped with $r = 0$

The model of the monoped is nearly identical to that of a dual inverted pendulum; see Figure 9. For this case, we assume the simpler condition of $r = 0$. In other words, there is no foot, and the bottom link is "pinned" to the floor. This derivation applies the Euler-Lagrange equation to $L = T - V$ where T is the kinetic energy of the system, and V is the potential energy of the system as a function of the system state variables. The potential energy of the system is determined simply by the heights h_1 and h_2 of each of the masses m_1 and m_2 as shown below.

$$\begin{aligned} V &= g(m_1 h_1 + m_2 h_2) \\ &= m_1 g l_1 \cos\theta_1 + m_2 g(l_1 \cos\theta_1 + l_2 \cos\theta_2) \end{aligned} \tag{28}$$

The kinetic energy of the system is determined by the absolute velocities v_1 and v_2 of the two masses.

$$\begin{aligned} T &= \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \\ &= \frac{1}{2} m_1 (\omega_1 l_1)^2 + \frac{1}{2} m_2 ((l_1 \omega_1 cos\theta_1 + l_2 \omega_2 cos\theta_2)^2 + (l_1 \omega_1 sin\theta_1 + l_2 \omega_2 sin\theta_2)^2) \\ &= \frac{1}{2} m_1 \omega_1^2 l_1^2 + \frac{1}{2} m_2 (l_1^2 \omega_1^2 + l_2^2 \omega_2^2 + 2 l_1 l_2 \omega_1 \omega_2 cos(\theta_1 - \theta_2)) \end{aligned} \tag{29}$$

The quantity $L = T - V$ is expanded in equation 30 below.

$$\begin{aligned} L &= \frac{1}{2} m_1 \omega_1^2 l_1^2 + \frac{1}{2} m_2 l_1^2 \omega_1^2 + \frac{1}{2} m_2 l_2^2 \omega_2^2 + m_2 l_1 l_2 \omega_1 \omega_2 \cos(\theta_1 - \theta_2) - \\ &\quad m_1 g l_1 \cos\theta_1 - m_2 g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \end{aligned} \tag{30}$$

The Euler-Lagrange equation is applied to equation 30 and formulated as follows.

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \omega_1}\right) - \frac{\delta L}{\delta \theta_1} = 0 \tag{31}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \omega_2}\right) - \frac{\delta L}{\delta \theta_2} = \tau \tag{32}$$

The solution to the Euler-Lagrange equation is pursued by calculating each partial derivative in equations 31 and 32.

$$\frac{\delta L}{\delta \omega_1} = l_1^2 \omega_1 (m_1 + m_2) + m_2 l_1 l_2 \omega_2 \cos\psi \tag{33}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \omega_1}\right) = l_1^2 \dot{\omega_1} (m_1 + m_2) + m_2 l_1 l_2 \dot{\omega_2} \cos\psi - m_2 l_1 l_2 \omega_2 \dot{\psi} \sin\psi \tag{34}$$

$$\frac{\delta L}{\delta \omega_2} = m_2 l_2^2 \omega_2 + m_2 l_1 l_2 \omega_1 \cos\psi \tag{35}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \omega_2}\right) = m_2 l_2^2 \dot{\omega_2} + m_2 l_1 l_2 \dot{\omega_1} \cos\psi - m_2 l_1 l_2 \omega_1 \dot{\psi} \sin\psi \tag{36}$$

where $\psi = \theta_1 - \theta_2$.

$$\frac{\delta L}{\delta \theta_1} = (m_1 + m_2) g l_1 \sin\theta_1 - m_2 l_1 l_2 \omega_1 \omega_2 \sin\psi \quad (37)$$

$$\frac{\delta L}{\delta \theta_2} = m_2 g l_2 \sin\theta_2 + m_2 l_1 l_2 \omega_1 \omega_2 \sin\psi \quad (38)$$

Substitution of equations 34 and 37 into 31, expansion of ψ, and division by l_1 gives equation 39 below.

$$l_1(m_1 + m_2)\dot{\omega_1} + m_2 l_2 \cos(\theta_1 - \theta_2)\dot{\omega_2} + m_2 l_2 \omega_2^2 \sin(\theta_1 - \theta_2) - (m_1 + m_2) g \sin\theta_1 = 0 \quad (39)$$

Similarly, substitution of equations 36 and 38 into 32, expansion of ψ, and division by $m_2 l_2$ gives equation 40 below.

$$l_1 \cos(\theta_1 - \theta_2)\dot{\omega_1} + l_2\dot{\omega_2} - l_1\omega_1^2 \sin(\theta_1 - \theta_2) - g\sin\theta_2 = \frac{\tau}{m_2 l_2} \quad (40)$$

After extensive algebraic manipulation of the simultaneous equations 39 and 40, we acquire the dynamic equations 17 and 18.

B Dynamic Equation Derivation for Balanced Monoped with $r > 0$

We now investigate the analysis for the case of $r > 0$ whereby the system includes a semi-circular rocker-foot which contacts the floor; see Figure 9. The analysis is approached in the same fashion as with the case of $r = 0$, but as we shall find, the non-zero r term complicates the analysis.

The potential energy of the system is given by equation 41 below.

$$\begin{aligned} V &= g(m_1 h_1 + m_2 h_2) \\ &= m_1 g(l_1 \cos\theta_1 + r) + m_2 g(l_1 \cos\theta_1 + l_2 \cos\theta_2 + r) \end{aligned} \tag{41}$$

The kinetic energy of the system is again determined by the velocities of the two masses; it is given in equation 42 below.

$$\begin{aligned} T &= \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \\ &= \frac{1}{2} m_1 (\omega_1^2 l_1^2 + 2r\omega_1^2 l_1 \cos\theta_1 + r^2\omega_1^2) + \frac{1}{2} m_2 [\omega_1^2 l_1^2 + \omega_2^2 l_2^2 + r^2\omega_1^2 + \\ &\quad 2l_1 l_2 \omega_1 \omega_2 \cos(\theta_1 - \theta_2) + 2r\omega_1(\omega_1 l_1 \cos\theta_1 + \omega_2 l_2 \cos\theta_2)] \end{aligned} \tag{42}$$

The quantity $L = T - V$ is expanded as shown in equation 43.

$$\begin{aligned} L &= \frac{1}{2} m_1 \omega_1^2 l_1^2 + \frac{1}{2} m_2 l_1^2 \omega_1^2 + \frac{1}{2} m_2 l_2^2 \omega_2^2 + m_2 l_1 l_2 \omega_1 \omega_2 \cos(\theta_1 - \theta_2) - \\ &\quad m_1 g l_1 \cos\theta_1 - m_2 g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 - m_1 g r - m_2 g r + \\ &\quad (m_1 + m_2) r \omega_1^2 l_1 \cos\theta_1 + \frac{1}{2}(m_1 + m_2) r^2 \omega_1^2 + m_2 r \omega_1 \omega_2 l_2 \cos\theta_2 \end{aligned} \tag{43}$$

As before, a solution to the Euler-Lagrange equation is pursued by calculating each partial derivative of equations 31 and 32.

$$\begin{aligned} \frac{\delta L}{\delta \omega_1} &= l_1^2 \omega_1 (m_1 + m_2) + m_2 l_1 l_2 \omega_2 \cos\psi + \\ &\quad 2r\omega_1 l_1 \cos\theta_1 (m_1 + m_2) + r^2 \omega_1 (m_1 + m_2) + m_2 r \omega_2 l_2 \cos\theta_2 \end{aligned} \tag{44}$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\delta L}{\delta \omega_1}\right) &= l_1^2 \dot{\omega_1} (m_1 + m_2) + m_2 l_1 l_2 \dot{\omega_2} \cos\psi - m_2 l_1 l_2 \omega_2 \dot{\psi} \sin\psi + \\ &\quad 2r\dot{\omega_1} l_1 \cos\theta_1 (m_1 + m_2) - 2r\omega_1^2 l_1 \sin\theta_1 (m_1 + m_2) + \\ &\quad (m_1 + m_2) r^2 \dot{\omega_1} + m_2 r \dot{\omega_2} l_2 \cos\theta_2 - m_2 r \omega_2^2 l_2 \sin\theta_2 \end{aligned} \tag{45}$$

$$\frac{\delta L}{\delta \omega_2} = m_2 l_2^2 \omega_2 + m_2 l_1 l_2 \omega_1 \cos\psi + m_2 r \omega_1 l_2 \cos\theta_2 \tag{46}$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\delta L}{\delta \omega_2}\right) &= m_2 l_2^2 \dot{\omega_2} + m_2 l_1 l_2 \dot{\omega_1} \cos\psi - m_2 l_1 l_2 \omega_1 \dot{\psi} \sin\psi + \\ &\quad m_2 r \dot{\omega_1} l_2 \cos\theta_2 - m_2 r \omega_1 \omega_2 l_2 \sin\theta_2 \end{aligned} \tag{47}$$

where $\psi = \theta_1 - \theta_2$.

The right sides are calculated as follows.

$$\frac{\delta L}{\delta \theta_1} = (m_1 + m_2) g l_1 \sin\theta_1 - m_2 l_1 l_2 \omega_1 \omega_2 \sin\psi - (m_1 + m_2) r \omega_1^2 l_1 \sin\theta_1 \tag{48}$$

$$\frac{\delta L}{\delta \theta_2} = m_2 g l_2 \sin\theta_2 + m_2 l_1 l_2 \omega_1 \omega_2 \sin\psi - m_2 r \omega_1 \omega_2 l_2 \sin\theta_2 \tag{49}$$

Substitution of equations 45 and 48 into 31 gives equation 50.

$$[(l_1^2 + 2rl_1\cos\theta_1 + r^2)(m_1 + m_2)]\dot{\omega_1} + [(m_2l_2)(l_1\cos(\theta_1 - \theta_2) + r\cos\theta_2)]\dot{\omega_2}$$
$$+[(m_2l_2)(l_1\sin(\theta_1 - \theta_2) - r\sin\theta_2)]\omega_2^2 - (m_1 + m_2)l_1\sin\theta_1[g + r\omega_1^2] = 0 \quad (50)$$

Similarly, substitution of equations 47 and 49 into 32, expansion of ψ, and division by m_2l_2 gives equation 51 below.

$$[l_1\cos(\theta_1 - \theta_2) + r\cos\theta_2]\dot{\omega_1} + l_2\dot{\omega_2} - l_1\omega_1^2\sin(\theta_1 - \theta_2) - g\sin\theta_2 = \frac{\tau}{m_2l_2} \quad (51)$$

C Dynamics Derived Using Kane's Method

Kane's method was used to formulate the equations of motion used in the full dynamics simulation of the hopping robot. The names of the bodies, point masses, points and reference frames used in the nine degree of freedom (9 DOF) model are shown in Figure 20. The I frame is the inertial frame. Point O is fixed in the inertial frame. The A reference frame (unit vectors $\underline{a}_1, \underline{a}_2, \underline{a}_3$) is fixed in body A, the B frame (unit vectors $\underline{b}_1, \underline{b}_2, \underline{b}_3$) is fixed in body B, etc.

The variable names of the nine degrees of freedom are:

- vaoii1: $\underline{i}_1$ component of translational velocity of A's center of mass with respect to an observer fixed in the inertial frame
- vaoii2: $\underline{i}_2$ component of translational velocity of A's center of mass with respect to an observer fixed in the inertial frame
- vaoii3: $\underline{i}_3$ component of translational velocity of A's center of mass with respect to an observer fixed in the inertial frame
- waia1: $\underline{a}_1$ component of angular velocity of A with respect to an observer fixed in the inertial frame
- waia2: $\underline{a}_2$ component of angular velocity of A with respect to an observer fixed in the inertial frame
- waia3: $\underline{a}_3$ component of angular velocity of A with respect to an observer fixed in the inertial frame
- vpba: translational velocity of point PB with respect to an observer fixed in the A frame
- wrb: angular velocity of body R with respect to an observer fixed in the B frame
- vqb: translational velocity of point Q with respect to an observer fixed in the B frame

AUTOLEV©, a symbolic manipulator based on Kanes method, was used to generate the C code simulation of the system dynamics. The equations of motion in the code take the form [COEF][xdot] = [RHS] where xdot is the column vector comprised of the time derivatives of the states. The COEF and RHS matrix elements as well as descriptions of all variables that appear in the equations of motion are provided below.

The independent control systems for the motion of point Q with respect to frame B and for the rotation of body R with respect to frame B appear in functions called by the dynamics simulation. The interaction of body A with the ground is also in a function called by the simulation. Ground interaction is modeled as a stiff spring/damper system. As body A rotates in the inertial frame, the impact point(s) are computed. Ground spring/damper forces are applied at the contact point(s).

The description of other variables used in the code is as follows:

Variable quantities:

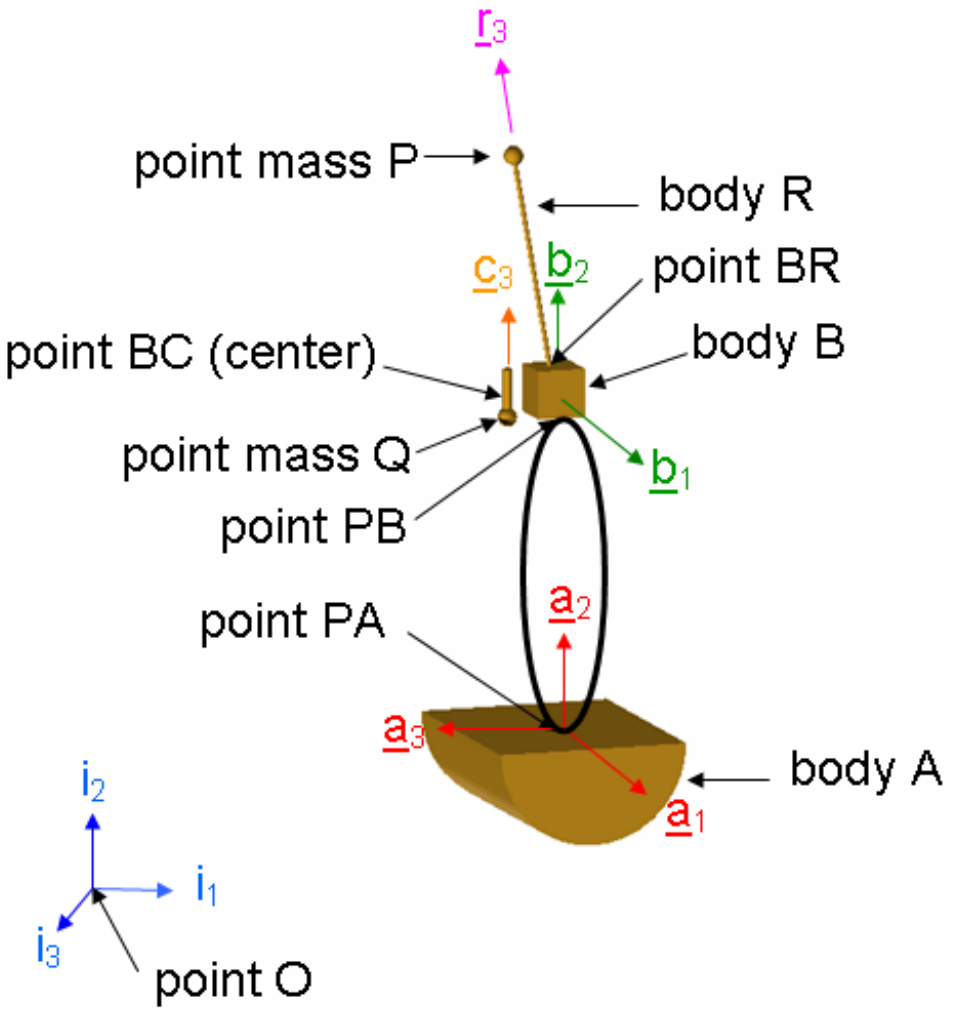

Figure 20: Hopping Robot Model

ang1ai: rotation angle 1 of a body 3 3,2,1 set orienting the A frame in the I frame

ang2ai: rotation angle 2 of a body 3 3,2,1 set orienting the A frame in the I frame

ang3ai: rotation angle 3 of a body 3 3,2,1 set orienting the A frame in the I frame

ang3rb: rotation angle 3 of a body 3 3,2,1 set orienting the R frame in the B frame

papb: $\underline{a}_2$ component of position vector from point PA to point PB

bcq: $\underline{c}_3$ component of position vector from point BC to point mass Q

Constant quantities:

qm: mass of point mass Q

pm: mass of point mass P

am: mass of body A

bm: mass of body B

rm: mass of body R

inertia matrix of body A for its mass center and axes $\underline{a}_1, \underline{a}_2, \underline{a}_3$:

$$\begin{pmatrix} ai11 & 0 & 0 \\ 0 & ai22 & 0 \\ 0 & 0 & ai33 \end{pmatrix} \tag{52}$$

inertia matrix of body B for its mass center and axes b1,b2,b3 :

$$\begin{pmatrix} bi11 & 0 & 0 \\ 0 & bi22 & 0 \\ 0 & 0 & bi33 \end{pmatrix} \tag{53}$$

inertia matrix of body R for its mass center and axes r1,r2,r3 :

$$\begin{pmatrix} ri11 & 0 & 0 \\ 0 & ri22 & 0 \\ 0 & 0 & ri33 \end{pmatrix} \tag{54}$$

ang1rb: rotation angle 1 of a body 3 3,2,1 set orienting the R frame in the B frame

ang2rb: rotation angle 2 of a body 3 3,2,1 set orienting the R frame in the B frame

ang1ba: rotation angle 1 of a body 3 3,2,1 set orienting the B frame in the A frame

ang2ba: rotation angle 2 of a body 3 3,2,1 set orienting the B frame in the A frame

ang3ba: rotation angle 3 of a body 3 3,2,1 set orienting the B frame in the A frame

ang1cb: rotation angle 1 of a body 3 3,2,1 set orienting the C frame in the B frame

ang2cb: rotation angle 2 of a body 3 3,2,1 set orienting the C frame in the B frame

ang3cb: rotation angle 3 of a body 3 3,2,1 set orienting the C frame in the B frame

paaoa1: a1 component of position vector from point PA to body A's mass center

paaoa2: a2 component of position vector from point PA to body A's mass center

paaoa3: b1 component of position vector from point PB to body B's mass center

pbbob2: b2 component of position vector from point PB to body B's mass center

pbbob3: b3 component of position vector from point PB to body B's mass center

bobcb1: b1 component of position vector from body B's mass center to point BC

bobcb2: b2 component of position vector from body B's mass center to point BC

bobcb3: b3 component of position vector from body B's mass center to point BC

bobrb1: b1 component of position vector from body B's mass center to point BR

bobrb2: b2 component of position vector from body B's mass center to point BR

bobrb3: b3 component of position vector from body B's mass center to point BR

brro: r3 component of position vector from point BR to body R's mass center

brp: r3 component of position vector from point BR to point mass P

grav: acceleration due to gravity

aleno2: half the length of body A

aradius: radius of body A

sprconab: spring constant of ring connecting points PA and PB

natlenab: natural length of ring (ring diameter) connecting points PA and PB

dmpconab: damping constant of ring connecting points PA and PB

torqonr: control torque between body B and body R

frconq: control force between body B and point mass Q

```
COEF[0][0] = -z[365];
COEF[0][1] = -z[368];
COEF[0][2] = -z[369];
```

```
COEF[0][3] = -z[370];
COEF[0][4] = -z[367];
COEF[0][5] = -z[371];
COEF[0][6] = -z[366];
COEF[0][7] = z[373];
COEF[0][8] = -z[374];
COEF[1][0] = -z[368];
COEF[1][1] = -z[376];
COEF[1][2] = -z[379];
COEF[1][3] = -z[380];
COEF[1][4] = -z[378];
COEF[1][5] = -z[381];
COEF[1][6] = -z[377];
COEF[1][7] = z[382];
COEF[1][8] = -z[383];
COEF[2][0] = -z[369];
COEF[2][1] = -z[379];
COEF[2][2] = -z[385];
COEF[2][3] = -z[388];
COEF[2][4] = -z[387];
COEF[2][5] = -z[389];
COEF[2][6] = -z[386];
COEF[2][7] = z[390];
COEF[2][8] = -z[391];
COEF[3][0] = -z[370];
COEF[3][1] = -z[380];
COEF[3][2] = -z[388];
COEF[3][3] = -z[394];
COEF[3][4] = -z[396];
COEF[3][5] = -z[399];
COEF[3][6] = -z[403];
COEF[3][7] = -z[402];
COEF[3][8] = -z[404];
COEF[4][0] = -z[367];
COEF[4][1] = -z[378];
COEF[4][2] = -z[387];
COEF[4][3] = -z[409];
COEF[4][4] = -z[407];
COEF[4][5] = -z[412];
COEF[4][6] = -z[415];
COEF[4][7] = -z[413];
COEF[4][8] = -z[416];
COEF[5][0] = -z[371];
COEF[5][1] = -z[381];
COEF[5][2] = -z[389];
COEF[5][3] = -z[428];
```

```
COEF[5][4] = -z[431];
COEF[5][5] = -z[425];
COEF[5][6] = -z[433];
COEF[5][7] = -z[432];
COEF[5][8] = -z[434];
COEF[6][0] = -z[366];
COEF[6][1] = -z[377];
COEF[6][2] = -z[386];
COEF[6][3] = -z[403];
COEF[6][4] = -z[415];
COEF[6][5] = -z[433];
COEF[6][6] = -z[438];
COEF[6][7] = z[439];
COEF[6][8] = -z[440];
COEF[7][0] = z[373];
COEF[7][1] = z[382];
COEF[7][2] = z[390];
COEF[7][3] = -z[402];
COEF[7][4] = -z[413];
COEF[7][5] = -z[432];
COEF[7][6] = z[439];
COEF[7][7] = -z[448];
COEF[7][8] = 0;
COEF[8][0] = -z[374];
COEF[8][1] = -z[383];
COEF[8][2] = -z[391];
COEF[8][3] = -z[404];
COEF[8][4] = -z[416];
COEF[8][5] = -z[434];
COEF[8][6] = -z[440];
COEF[8][7] = 0;
COEF[8][8] = -qm;

RHS[0] = -z[496];
RHS[1] = -z[497];
RHS[2] = -z[498];
RHS[3] = -z[499];
RHS[4] = -z[500];
RHS[5] = -z[501];
RHS[6] = -z[502];
RHS[7] = -z[503];
RHS[8] = -z[504];
```

where:

```
z[20] = cos(ang1ba);
z[21] = cos(ang2ba);
```

```
z[22] = z[20]*z[21];
z[23] = sin(ang2ba);
z[24] = sin(ang3ba);
z[25] = sin(ang1ba);
z[26] = cos(ang3ba);
z[27] = z[20]*z[23]*z[24] - z[25]*z[26];
z[28] = z[24]*z[25] + z[20]*z[23]*z[26];
z[29] = z[21]*z[25];
z[30] = z[20]*z[26] + z[23]*z[24]*z[25];
z[31] = z[23]*z[25]*z[26] - z[20]*z[24];
z[32] = z[21]*z[24];
z[33] = z[21]*z[26];
z[37] = -paaoa1*z[23] - paaoa3*z[22];
z[39] = paaoa1*z[29];
z[41] = paaoa3*z[29];
z[43] = paaoa1*z[32] - paaoa3*z[27];
z[47] = paaoa3*z[30];
z[49] = paaoa1*z[30];
z[51] = paaoa1*z[33] - paaoa3*z[28];
z[55] = paaoa3*z[31];
z[57] = paaoa1*z[31];
z[59] = pbbob2*z[22] - pbbob1*z[27];
z[60] = pbbob2*z[29] - pbbob1*z[30];
z[61] = -pbbob1*z[32] - pbbob2*z[23];
z[62] = pbbob1*z[28] - pbbob3*z[22];
z[63] = pbbob1*z[31] - pbbob3*z[29];
z[64] = pbbob1*z[33] + pbbob3*z[23];
z[65] = pbbob3*z[27] - pbbob2*z[28];
z[66] = pbbob3*z[30] - pbbob2*z[31];
z[67] = pbbob3*z[32] - pbbob2*z[33];
z[68] = z[37] + z[66];
z[71] = z[43] + z[63];
z[74] = z[51] + z[60];
z[77] = cos(ang1cb);
z[78] = cos(ang2cb);
z[79] = z[77]*z[78];
z[80] = sin(ang2cb);
z[81] = sin(ang3cb);
z[82] = sin(ang1cb);
z[83] = cos(ang3cb);
z[84] = z[77]*z[80]*z[81] - z[82]*z[83];
z[85] = z[81]*z[82] + z[77]*z[80]*z[83];
z[86] = z[78]*z[82];
z[87] = z[77]*z[83] + z[80]*z[81]*z[82];
z[88] = z[80]*z[82]*z[83] - z[77]*z[81];
z[89] = z[78]*z[81];
```

```
z[90] = z[78]*z[83];
z[91] = bobcb2*z[22] - bobcb1*z[27];
z[92] = bobcb2*z[29] - bobcb1*z[30];
z[93] = -bobcb1*z[32] - bobcb2*z[23];
z[94] = bobcb1*z[28] - bobcb3*z[22];
z[95] = bobcb1*z[31] - bobcb3*z[29];
z[96] = bobcb1*z[33] + bobcb3*z[23];
z[97] = bobcb3*z[27] - bobcb2*z[28];
z[98] = bobcb3*z[30] - bobcb2*z[31];
z[99] = bobcb3*z[32] - bobcb2*z[33];
z[100] = z[68] + z[98];
z[103] = z[71] + z[95];
z[106] = z[74] + z[92];
z[109] = z[29]*z[79] + z[30]*z[86] - z[31]*z[80];
z[116] = z[29]*z[84] + z[30]*z[87] + z[31]*z[89];
z[123] = z[29]*z[85] + z[30]*z[88] + z[31]*z[90];
z[130] = z[22]*z[79] + z[27]*z[86] - z[28]*z[80];
z[131] = z[22]*z[84] + z[27]*z[87] + z[28]*z[89];
z[132] = z[22]*z[85] + z[27]*z[88] + z[28]*z[90];
z[133] = z[32]*z[86] - z[23]*z[79] - z[33]*z[80];
z[134] = z[32]*z[87] + z[33]*z[89] - z[23]*z[84];
z[135] = z[32]*z[88] + z[33]*z[90] - z[23]*z[85];
z[154] = cos(ang1rb);
z[155] = cos(ang2rb);
z[156] = z[154]*z[155];
z[157] = sin(ang2rb);
z[159] = sin(ang1rb);
z[161] = z[154]*z[157];
z[164] = z[155]*z[159];
z[165] = z[157]*z[159];
z[170] = bobrb2*z[22] - bobrb1*z[27];
z[171] = bobrb2*z[29] - bobrb1*z[30];
z[172] = -bobrb1*z[32] - bobrb2*z[23];
z[173] = bobrb1*z[28] - bobrb3*z[22];
z[174] = bobrb1*z[31] - bobrb3*z[29];
z[175] = bobrb1*z[33] + bobrb3*z[23];
z[176] = bobrb3*z[27] - bobrb2*z[28];
z[177] = bobrb3*z[30] - bobrb2*z[31];
z[178] = bobrb3*z[32] - bobrb2*z[33];
z[179] = z[68] + z[177];
z[182] = z[71] + z[174];
z[185] = z[74] + z[171];
z[188] = z[29]*z[156] + z[30]*z[164] - z[31]*z[157];
z[189] = z[156]*z[179] + z[164]*z[182] - z[157]*z[185];
z[209] = z[22]*z[156] + z[27]*z[164] - z[28]*z[157];
z[212] = z[32]*z[164] - z[23]*z[156] - z[33]*z[157];
```

```
z[215] = brro*z[188];
z[216] = brro*z[209];
z[217] = brro*z[212];
z[227] = brp*z[188];
z[228] = brp*z[209];
z[229] = brp*z[212];
z[333] = bi11*z[22];
z[334] = bi11*z[29];
z[335] = bi11*z[23];
z[336] = bi22*z[27];
z[337] = bi22*z[30];
z[338] = bi22*z[32];
z[339] = bi33*z[28];
z[340] = bi33*z[31];
z[341] = bi33*z[33];
z[351] = ri11*z[188];
z[352] = ri11*z[209];
z[353] = ri11*z[212];
z[372] = brp*pm + brro*rm;
z[393] = ai11 + z[22]*z[333] + z[27]*z[336] + z[28]*z[339] + z[209]*z[352];
z[395] = z[22]*z[334] + z[27]*z[337] + z[28]*z[340] + z[209]*z[351];
z[397] = z[27]*z[338] + z[28]*z[341] + z[209]*z[353];
z[398] = z[22]*z[335];
z[400] = brp*pm;
z[401] = brro*rm;
z[406] = ai22 + z[29]*z[334] + z[30]*z[337] + z[31]*z[340] + z[188]*z[351] +
bm*(pow(z[68],2)+pow(z[71],2)+pow(z[74],2)) + qm*(pow(z[100],2)+pow(z[103],
2)+pow(z[106],2));
z[408] = z[29]*z[333] + z[30]*z[336] + z[31]*z[339] + z[188]*z[352];
z[410] = z[30]*z[338] + z[31]*z[341] + z[188]*z[353];
z[411] = z[29]*z[335];
z[414] = bm*(z[29]*z[68]+z[30]*z[71]+z[31]*z[74]) + qm*(z[29]*z[100]+z[30]*
z[103]+z[31]*z[106]);
z[416] = qm*(z[85]*z[100]+z[88]*z[103]+z[90]*z[106]);
z[417] = z[80]*z[106];
z[418] = z[84]*z[100];
z[419] = z[87]*z[103];
z[420] = z[89]*z[106];
z[421] = z[79]*z[100];
z[422] = z[86]*z[103];
z[424] = ai33 + z[23]*z[335] + z[32]*z[338] + z[33]*z[341] + z[212]*z[353];
z[426] = z[32]*z[336] + z[33]*z[339] + z[212]*z[352];
z[427] = z[23]*z[333];
z[429] = z[32]*z[337] + z[33]*z[340] + z[212]*z[351];
z[430] = z[23]*z[334];
z[436] = bm*(pow(z[29],2)+pow(z[30],2)+pow(z[31],2)) + qm*(pow(z[29],2)+
```

```
pow(z[30],2)+pow(z[31],2));
z[437] = pow(z[188],2);
z[440] = qm*(z[29]*z[85]+z[30]*z[88]+z[31]*z[90]);
z[441] = z[29]*z[84];
z[442] = z[30]*z[87];
z[443] = z[31]*z[80];
z[444] = z[31]*z[89];
z[445] = z[29]*z[79];
z[446] = z[30]*z[86];
z[448] = ri11 + pm*pow(brp,2) + rm*pow(brro,2);
z[451] = am*grav;
z[452] = bm*grav;
z[453] = grav*pm;
z[454] = grav*qm;
z[455] = grav*rm;
z[483] = z[100]*z[454];
z[484] = z[103]*z[454];
z[485] = z[106]*z[454];
z[488] = z[29]*z[454];
z[489] = z[30]*z[454];
z[490] = z[31]*z[454];
z[492] = brp*z[453];
z[493] = brro*z[455];
z[505] = pow(paaoa1,2) + pow(paaoa3,2) + pow(pbbob1,2) + pow(pbbob2,2) +
pow(pbbob3,2) + 2*paaoa3*pbbob1*z[23];
z[506] = pbbob1*z[29];
z[507] = pbbob2*z[30];
z[508] = pbbob3*z[31];
z[509] = paaoa1*pbbob1*z[22];
z[510] = paaoa1*pbbob2*z[27];
z[511] = paaoa1*pbbob3*z[28];
z[512] = paaoa3*pbbob2*z[32];
z[513] = paaoa3*pbbob3*z[33];
zilch = 0;
z[522] = bobrb1 + pbbob1;
z[523] = bobrb2 + pbbob2;
z[524] = bobrb3 + pbbob3;
z[525] = bobcb1 + pbbob1;
z[526] = bobcb2 + pbbob2;
z[527] = bobcb3 + pbbob3;
z[528] = am + bm + pm + qm + rm;
z[529] = am*paaoa1/z[528];
z[530] = am*paaoa2;
z[532] = am*paaoa3/z[528];
z[533] = (bm*pbbob1+pm*z[522]+qm*z[525]+rm*z[522])/z[528];
z[534] = (bm*pbbob2+pm*z[523]+qm*z[526]+rm*z[523])/z[528];
```

```
z[535] = (bm*pbbob3+pm*z[524]+qm*z[527]+rm*z[524])/z[528];
z[537] = (brp*pm+brro*rm)/z[528];
z[646] = qm/z[528];
z[564] = aleno2 - paaoa1;
z[655] = (bm+pm+qm+rm)/z[528];
z[5] = sin(ang3ai);
z[2] = cos(ang2ai);
z[15] = z[5]/z[2];
z[7] = cos(ang3ai);
z[16] = z[7]/z[2];
ang1aip = z[15]*waia2 + z[16]*waia3;
ang2aip = z[7]*waia2 - z[5]*waia3;
z[17] = tan(ang2ai);
z[18] = z[5]*z[17];
z[19] = z[7]*z[17];
ang3aip = waia1 + z[18]*waia2 + z[19]*waia3;
z[1] = cos(ang1ai);
z[3] = z[1]*z[2];
z[4] = sin(ang2ai);
z[6] = sin(ang1ai);
z[8] = z[1]*z[4]*z[5] - z[6]*z[7];
z[9] = z[5]*z[6] + z[1]*z[4]*z[7];
z[10] = z[2]*z[6];
z[11] = z[1]*z[7] + z[4]*z[5]*z[6];
z[12] = z[4]*z[6]*z[7] - z[1]*z[5];
z[13] = z[2]*z[5];
z[14] = z[2]*z[7];
oaoi1p = vaoii1;
oaoi2p = vaoii2;
oaoi3p = vaoii3;
papbp = vpba;
z[34] = papb - paaoa2;
z[35] = z[22]*z[3] + z[29]*z[8] - z[23]*z[9];
z[36] = z[22]*z[10] + z[29]*z[11] - z[23]*z[12];
z[38] = z[29]*z[13] - z[22]*z[4] - z[23]*z[14];
z[40] = -z[39] - z[22]*z[34];
z[42] = z[41] - z[23]*z[34];
z[44] = z[27]*z[3] + z[30]*z[8] + z[32]*z[9];
z[45] = z[27]*z[10] + z[30]*z[11] + z[32]*z[12];
z[46] = z[30]*z[13] + z[32]*z[14] - z[27]*z[4];
z[48] = z[47] + z[32]*z[34];
z[50] = -z[49] - z[27]*z[34];
z[52] = z[28]*z[3] + z[31]*z[8] + z[33]*z[9];
z[53] = z[28]*z[10] + z[31]*z[11] + z[33]*z[12];
z[54] = z[31]*z[13] + z[33]*z[14] - z[28]*z[4];
z[56] = z[55] + z[33]*z[34];
```

```
z[58] = -z[57] - z[28]*z[34];
z[69] = z[65] + z[42];
z[70] = z[67] + z[40];
z[72] = z[62] + z[48];
z[73] = z[64] + z[50];
z[75] = z[59] + z[56];
z[76] = z[61] + z[58];
z[101] = z[97] + z[69];
z[102] = z[99] + z[70];
z[104] = z[94] + z[72];
z[105] = z[96] + z[73];
z[107] = z[91] + z[75];
z[108] = z[93] + z[76];
bcqp = vqb;
z[126] = z[85]*z[36] + z[88]*z[45] + z[90]*z[53];
ang3rbp = wrb;
z[158] = sin(ang3rb);
z[160] = cos(ang3rb);
z[162] = z[161]*z[158] - z[159]*z[160];
z[163] = z[159]*z[158] + z[161]*z[160];
z[166] = z[154]*z[160] + z[165]*z[158];
z[167] = z[165]*z[160] - z[154]*z[158];
z[168] = z[155]*z[158];
z[169] = z[155]*z[160];
z[180] = z[176] + z[69];
z[181] = z[178] + z[70];
z[183] = z[173] + z[72];
z[184] = z[175] + z[73];
z[186] = z[170] + z[75];
z[187] = z[172] + z[76];
z[190] = z[156]*z[35] + z[164]*z[44] - z[157]*z[52];
z[191] = z[156]*z[36] + z[164]*z[45] - z[157]*z[53];
z[192] = z[156]*z[38] + z[164]*z[46] - z[157]*z[54];
z[193] = z[156]*z[180] + z[164]*z[183] - z[157]*z[186];
z[194] = z[156]*z[181] + z[164]*z[184] - z[157]*z[187];
z[195] = z[29]*z[162] + z[30]*z[166] + z[31]*z[168];
z[196] = z[179]*z[162] + z[182]*z[166] + z[185]*z[168];
z[197] = z[35]*z[162] + z[44]*z[166] + z[52]*z[168];
z[198] = z[36]*z[162] + z[45]*z[166] + z[53]*z[168];
z[199] = z[38]*z[162] + z[46]*z[166] + z[54]*z[168];
z[200] = z[162]*z[180] + z[166]*z[183] + z[168]*z[186];
z[201] = z[162]*z[181] + z[166]*z[184] + z[168]*z[187];
z[202] = z[29]*z[163] + z[30]*z[167] + z[31]*z[169];
z[203] = z[179]*z[163] + z[182]*z[167] + z[185]*z[169];
z[204] = z[35]*z[163] + z[44]*z[167] + z[52]*z[169];
z[205] = z[36]*z[163] + z[45]*z[167] + z[53]*z[169];
```

```
z[206] = z[38]*z[163] + z[46]*z[167] + z[54]*z[169];
z[207] = z[163]*z[180] + z[167]*z[183] + z[169]*z[186];
z[208] = z[163]*z[181] + z[167]*z[184] + z[169]*z[187];
z[210] = z[22]*z[162] + z[27]*z[166] + z[28]*z[168];
z[211] = z[22]*z[163] + z[27]*z[167] + z[28]*z[169];
z[213] = z[32]*z[166] + z[33]*z[168] - z[23]*z[162];
z[214] = z[32]*z[167] + z[33]*z[169] - z[23]*z[163];
z[218] = brro*z[195];
z[219] = brro*z[210];
z[220] = brro*z[213];
z[221] = z[189] + z[218];
z[222] = z[193] + z[219];
z[223] = z[194] + z[220];
z[224] = z[196] - z[215];
z[225] = z[200] - z[216];
z[226] = z[201] - z[217];
z[230] = brp*z[195];
z[231] = brp*z[210];
z[232] = brp*z[213];
z[233] = z[189] + z[230];
z[234] = z[193] + z[231];
z[235] = z[194] + z[232];
z[236] = z[196] - z[227];
z[237] = z[200] - z[228];
z[238] = z[201] - z[229];
z[239] = z[29]*vpba + z[68]*waia2 + z[35]*vaoii1 + z[36]*vaoii2 + z[38]*
vaoii3 + z[69]*waia1 + z[70]*waia3;
z[240] = z[30]*vpba + z[71]*waia2 + z[44]*vaoii1 + z[45]*vaoii2 + z[46]*
vaoii3 + z[72]*waia1 + z[73]*waia3;
z[241] = z[31]*vpba + z[74]*waia2 + z[52]*vaoii1 + z[53]*vaoii2 + z[54]*
vaoii3 + z[75]*waia1 + z[76]*waia3;
z[242] = -z[1]*z[4]*ang2aip - z[2]*z[6]*ang1aip;
z[243] = z[5]*z[6]*ang3aip + z[1]*z[2]*z[5]*ang2aip + z[1]*z[4]*z[7]*ang3aip -
z[1]*z[7]*ang1aip - z[4]*z[5]*z[6]*ang1aip;
z[244] = z[1]*z[5]*ang1aip + z[6]*z[7]*ang3aip + z[1]*z[2]*z[7]*ang2aip -
z[1]*z[4]*z[5]*ang3aip - z[4]*z[6]*z[7]*ang1aip;
z[245] = z[22]*z[242] + z[29]*z[243] - z[23]*z[244];
z[246] = z[1]*z[2]*ang1aip - z[4]*z[6]*ang2aip;
z[247] = z[1]*z[4]*z[5]*ang1aip + z[2]*z[5]*z[6]*ang2aip + z[4]*z[6]*z[7]*
ang3aip - z[1]*z[5]*ang3aip - z[6]*z[7]*ang1aip;
z[248] = z[5]*z[6]*ang1aip + z[1]*z[4]*z[7]*ang1aip + z[2]*z[6]*z[7]*ang2aip -
z[1]*z[7]*ang3aip - z[4]*z[5]*z[6]*ang3aip;
z[249] = z[22]*z[246] + z[29]*z[247] - z[23]*z[248];
z[250] = z[2]*z[7]*ang3aip - z[4]*z[5]*ang2aip;
z[251] = -z[2]*z[5]*ang3aip - z[4]*z[7]*ang2aip;
z[252] = z[29]*z[250] - z[22]*z[2]*ang2aip - z[23]*z[251];
```

```
z[253] = vaoii1*z[245] + vaoii2*z[249] + vaoii3*z[252] - z[22]*vpba*waia3 -
z[23]*vpba*waia1;
z[254] = z[27]*z[242] + z[30]*z[243] + z[32]*z[244];
z[255] = z[27]*z[246] + z[30]*z[247] + z[32]*z[248];
z[256] = z[30]*z[250] + z[32]*z[251] - z[27]*z[2]*ang2aip;
z[257] = z[32]*vpba*waia1 + vaoii1*z[254] + vaoii2*z[255] + vaoii3*z[256] -
z[27]*vpba*waia3;
z[258] = z[28]*z[242] + z[31]*z[243] + z[33]*z[244];
z[259] = z[28]*z[246] + z[31]*z[247] + z[33]*z[248];
z[260] = z[31]*z[250] + z[33]*z[251] - z[28]*z[2]*ang2aip;
z[261] = z[33]*vpba*waia1 + vaoii1*z[258] + vaoii2*z[259] + vaoii3*z[260] -
z[28]*vpba*waia3;
z[262] = (z[27]*waia1+z[30]*waia2+z[32]*waia3)*z[241] + z[253] - (z[28]*
waia1+z[31]*waia2+z[33]*waia3)*z[240];
z[263] = (z[28]*waia1+z[31]*waia2+z[33]*waia3)*z[239] + (z[23]*waia3-z[22]*
waia1-z[29]*waia2)*z[241] + z[257];
z[264] = z[261] - (z[27]*waia1+z[30]*waia2+z[32]*waia3)*z[239] - (z[23]*
waia3-z[22]*waia1-z[29]*waia2)*z[240];
z[265] = z[29]*vpba + z[100]*waia2 + z[35]*vaoii1 + z[36]*vaoii2 + z[38]*
vaoii3 + z[101]*waia1 + z[102]*waia3;
z[266] = z[30]*vpba + z[103]*waia2 + z[44]*vaoii1 + z[45]*vaoii2 + z[46]*
vaoii3 + z[104]*waia1 + z[105]*waia3;
z[267] = z[31]*vpba + z[106]*waia2 + z[52]*vaoii1 + z[53]*vaoii2 + z[54]*
vaoii3 + z[107]*waia1 + z[108]*waia3;
z[268] = (z[27]*waia1+z[30]*waia2+z[32]*waia3)*z[267] + z[253] - (z[28]*
waia1+z[31]*waia2+z[33]*waia3)*z[266];
z[269] = (z[28]*waia1+z[31]*waia2+z[33]*waia3)*z[265] + (z[23]*waia3-z[22]*
waia1-z[29]*waia2)*z[267] + z[257];
z[270] = z[261] - (z[27]*waia1+z[30]*waia2+z[32]*waia3)*z[265] - (z[23]*
waia3-z[22]*waia1-z[29]*waia2)*z[266];
z[271] = vqb*(z[116]*waia2+z[131]*waia1+z[134]*waia3);
z[272] = vqb*(z[109]*waia2+z[130]*waia1+z[133]*waia3);
z[273] = z[188]*vpba + z[190]*vaoii1 + z[191]*vaoii2 + z[192]*vaoii3 +
z[221]*waia2 + z[222]*waia1 + z[223]*waia3;
z[274] = z[195]*vpba + z[197]*vaoii1 + z[198]*vaoii2 + z[199]*vaoii3 +
z[224]*waia2 + z[225]*waia1 + z[226]*waia3 - brro*wrb;
z[275] = z[202]*vpba + z[203]*waia2 + z[204]*vaoii1 + z[205]*vaoii2 +
z[206]*vaoii3 + z[207]*waia1 + z[208]*waia3;
z[276] = z[156]*z[245] + z[164]*z[254] - z[157]*z[258];
z[277] = z[156]*z[249] + z[164]*z[255] - z[157]*z[259];
z[278] = z[156]*z[252] + z[164]*z[256] - z[157]*z[260];
z[279] = (z[159]*z[158]+z[161]*z[160])*wrb;
z[280] = (z[154]*z[158]-z[165]*z[160])*wrb;
z[281] = z[155]*z[160]*wrb;
z[282] = z[29]*z[279] + z[31]*z[281] - z[30]*z[280];
z[283] = z[212]*vpba;
```

```
z[284] = z[22]*z[279] + z[28]*z[281] - z[27]*z[280];
z[285] = z[283] + brro*z[284];
z[286] = z[209]*vpba;
z[287] = z[33]*z[281] - z[23]*z[279] - z[32]*z[280];
z[288] = brro*z[287] - z[286];
z[289] = waia1*z[285] + waia3*z[288] + brro*waia2*z[282] + vaoii1*z[276] +
vaoii2*z[277] + vaoii3*z[278];
z[290] = z[35]*z[279] + z[52]*z[281] + z[162]*z[245] + z[166]*z[254] +
z[168]*z[258] - z[44]*z[280];
z[291] = z[36]*z[279] + z[53]*z[281] + z[162]*z[249] + z[166]*z[255] +
z[168]*z[259] - z[45]*z[280];
z[292] = z[38]*z[279] + z[54]*z[281] + z[162]*z[252] + z[166]*z[256] +
z[168]*z[260] - z[46]*z[280];
z[293] = z[179]*z[279] + z[185]*z[281] - z[182]*z[280];
z[294] = z[32]*z[166]*vpba + z[33]*z[168]*vpba + z[180]*z[279] + z[186]*
z[281] - z[23]*z[162]*vpba - z[183]*z[280];
z[295] = z[181]*z[279] + z[187]*z[281] - z[22]*z[162]*vpba - z[27]*z[166]*
vpba - z[28]*z[168]*vpba - z[184]*z[280];
z[296] = vpba*z[282] + waia1*z[294] + waia2*z[293] + waia3*z[295] + vaoii1*
z[290] + vaoii2*z[291] + vaoii3*z[292];
z[297] = (z[159]*z[160]-z[161]*z[158])*wrb;
z[298] = (z[154]*z[160]+z[165]*z[158])*wrb;
z[299] = z[155]*z[158]*wrb;
z[300] = z[29]*z[297] - z[30]*z[298] - z[31]*z[299];
z[301] = z[179]*z[297] - z[182]*z[298] - z[185]*z[299];
z[302] = z[35]*z[297] + z[163]*z[245] + z[167]*z[254] + z[169]*z[258] -
z[44]*z[298] - z[52]*z[299];
z[303] = z[36]*z[297] + z[163]*z[249] + z[167]*z[255] + z[169]*z[259] -
z[45]*z[298] - z[53]*z[299];
z[304] = z[38]*z[297] + z[163]*z[252] + z[167]*z[256] + z[169]*z[260] -
z[46]*z[298] - z[54]*z[299];
z[305] = z[32]*z[167]*vpba + z[33]*z[169]*vpba + z[180]*z[297] - z[23]*
z[163]*vpba - z[183]*z[298] - z[186]*z[299];
z[306] = z[181]*z[297] - z[22]*z[163]*vpba - z[27]*z[167]*vpba - z[28]*
z[169]*vpba - z[184]*z[298] - z[187]*z[299];
z[307] = vpba*z[300] + waia1*z[305] + waia2*z[301] + waia3*z[306] + vaoii1*
z[302] + vaoii2*z[303] + vaoii3*z[304];
z[308] = (z[195]*waia2+z[210]*waia1+z[213]*waia3)*z[275] + z[289] - (z[202]*
waia2+z[211]*waia1+z[214]*waia3)*z[274];
z[309] = (z[202]*waia2+z[211]*waia1+z[214]*waia3)*z[273] + z[296] - (wrb+
z[188]*waia2+z[209]*waia1+z[212]*waia3)*z[275];
z[310] = (wrb+z[188]*waia2+z[209]*waia1+z[212]*waia3)*z[274] + z[307] - (
z[195]*waia2+z[210]*waia1+z[213]*waia3)*z[273];
z[311] = z[188]*vpba + z[190]*vaoii1 + z[191]*vaoii2 + z[192]*vaoii3 +
z[233]*waia2 + z[234]*waia1 + z[235]*waia3;
z[312] = z[195]*vpba + z[197]*vaoii1 + z[198]*vaoii2 + z[199]*vaoii3 +
```

```
z[236]*waia2 + z[237]*waia1 + z[238]*waia3 - brp*wrb;
z[313] = z[283] + brp*z[284];
z[314] = brp*z[287] - z[286];
z[315] = waia1*z[313] + waia3*z[314] + brp*waia2*z[282] + vaoii1*z[276] +
vaoii2*z[277] + vaoii3*z[278];
z[316] = (z[195]*waia2+z[210]*waia1+z[213]*waia3)*z[275] + z[315] - (z[202]*
waia2+z[211]*waia1+z[214]*waia3)*z[312];
z[317] = (z[202]*waia2+z[211]*waia1+z[214]*waia3)*z[311] + z[296] - (wrb+
z[188]*waia2+z[209]*waia1+z[212]*waia3)*z[275];
z[318] = (wrb+z[188]*waia2+z[209]*waia1+z[212]*waia3)*z[312] + z[307] - (
z[195]*waia2+z[210]*waia1+z[213]*waia3)*z[311];
z[319] = wrb*(z[202]*waia2+z[211]*waia1+z[214]*waia3);
z[320] = wrb*(z[195]*waia2+z[210]*waia1+z[213]*waia3);
z[321] = ai11*waia1;
z[322] = ai22*waia2;
z[323] = ai33*waia3;
z[324] = waia1*z[322] - waia2*z[321];
z[325] = waia3*z[321] - waia1*z[323];
z[326] = waia2*z[323] - waia3*z[322];
z[327] = z[22]*waia1 + z[29]*waia2 - z[23]*waia3;
z[328] = z[27]*waia1 + z[30]*waia2 + z[32]*waia3;
z[329] = z[28]*waia1 + z[31]*waia2 + z[33]*waia3;
z[330] = bi11*z[327];
z[331] = bi22*z[328];
z[332] = bi33*z[329];
z[342] = z[327]*z[331] - z[328]*z[330];
z[343] = z[329]*z[330] - z[327]*z[332];
z[344] = z[328]*z[332] - z[329]*z[331];
z[345] = wrb + z[188]*waia2 + z[209]*waia1 + z[212]*waia3;
z[346] = z[195]*waia2 + z[210]*waia1 + z[213]*waia3;
z[347] = z[202]*waia2 + z[211]*waia1 + z[214]*waia3;
z[348] = ri11*z[345];
z[349] = ri22*z[346];
z[350] = ri33*z[347];
z[354] = ri22*z[195];
z[355] = ri22*z[210];
z[356] = ri22*z[213];
z[357] = ri22*z[319];
z[358] = ri33*z[202];
z[359] = ri33*z[211];
z[360] = ri33*z[214];
z[361] = ri33*z[320];
z[362] = z[345]*z[349] - z[346]*z[348];
z[363] = z[347]*z[348] - z[345]*z[350];
z[364] = z[346]*z[350] - z[347]*z[349];
z[365] = am + bm*(pow(z[35],2)+pow(z[44],2)+pow(z[52],2)) + pm*(pow(z[190],
```

```
2)+pow(z[197],2)+pow(z[204],2)) + qm*(pow(z[35],2)+pow(z[44],2)+pow(z[52],2)) +
rm*(pow(z[190],2)+pow(z[197],2)+pow(z[204],2));
z[366] = bm*(z[29]*z[35]+z[30]*z[44]+z[31]*z[52]) + pm*(z[188]*z[190]+
z[195]*z[197]+z[202]*z[204]) + qm*(z[29]*z[35]+z[30]*z[44]+z[31]*z[52]) +
rm*(z[188]*z[190]+z[195]*z[197]+z[202]*z[204]);
z[367] = bm*(z[68]*z[35]+z[71]*z[44]+z[74]*z[52]) + pm*(z[190]*z[233]+
z[197]*z[236]+z[203]*z[204]) + qm*(z[100]*z[35]+z[103]*z[44]+z[106]*z[52]) +
rm*(z[190]*z[221]+z[197]*z[224]+z[203]*z[204]);
z[368] = bm*(z[35]*z[36]+z[44]*z[45]+z[52]*z[53]) + pm*(z[190]*z[191]+
z[197]*z[198]+z[204]*z[205]) + qm*(z[35]*z[36]+z[44]*z[45]+z[52]*z[53]) +
rm*(z[190]*z[191]+z[197]*z[198]+z[204]*z[205]);
z[369] = bm*(z[35]*z[38]+z[44]*z[46]+z[52]*z[54]) + pm*(z[190]*z[192]+
z[197]*z[199]+z[204]*z[206]) + qm*(z[35]*z[38]+z[44]*z[46]+z[52]*z[54]) +
rm*(z[190]*z[192]+z[197]*z[199]+z[204]*z[206]);
z[370] = bm*(z[35]*z[69]+z[44]*z[72]+z[52]*z[75]) + pm*(z[190]*z[234]+
z[197]*z[237]+z[204]*z[207]) + qm*(z[35]*z[101]+z[44]*z[104]+z[52]*z[107]) +
rm*(z[190]*z[222]+z[197]*z[225]+z[204]*z[207]);
z[371] = bm*(z[35]*z[70]+z[44]*z[73]+z[52]*z[76]) + pm*(z[190]*z[235]+
z[197]*z[238]+z[204]*z[208]) + qm*(z[35]*z[102]+z[44]*z[105]+z[52]*z[108]) +
rm*(z[190]*z[223]+z[197]*z[226]+z[204]*z[208]);
z[373] = z[372]*z[197];
z[374] = qm*(z[85]*z[35]+z[88]*z[44]+z[90]*z[52]);
z[375] = bm*(z[35]*z[262]+z[44]*z[263]+z[52]*z[264]) + pm*(z[190]*z[316]+
z[197]*z[317]+z[204]*z[318]) + rm*(z[190]*z[308]+z[197]*z[309]+z[204]*
z[310]) - qm*(z[80]*z[52]*z[271]+z[84]*z[35]*z[272]+z[87]*z[44]*z[272]+
z[89]*z[52]*z[272]-z[79]*z[35]*z[271]-z[86]*z[44]*z[271]-z[35]*z[268]-z[44]*
z[269]-z[52]*z[270]);
z[376] = am + bm*(pow(z[36],2)+pow(z[45],2)+pow(z[53],2)) + pm*(pow(z[191],
2)+pow(z[198],2)+pow(z[205],2)) + qm*(pow(z[36],2)+pow(z[45],2)+pow(z[53],2)) +
rm*(pow(z[191],2)+pow(z[198],2)+pow(z[205],2));
z[377] = bm*(z[29]*z[36]+z[30]*z[45]+z[31]*z[53]) + pm*(z[188]*z[191]+
z[195]*z[198]+z[202]*z[205]) + qm*(z[29]*z[36]+z[30]*z[45]+z[31]*z[53]) +
rm*(z[188]*z[191]+z[195]*z[198]+z[202]*z[205]);
z[378] = bm*(z[68]*z[36]+z[71]*z[45]+z[74]*z[53]) + pm*(z[191]*z[233]+
z[198]*z[236]+z[203]*z[205]) + qm*(z[100]*z[36]+z[103]*z[45]+z[106]*z[53]) +
rm*(z[191]*z[221]+z[198]*z[224]+z[203]*z[205]);
z[379] = bm*(z[36]*z[38]+z[45]*z[46]+z[53]*z[54]) + pm*(z[191]*z[192]+
z[198]*z[199]+z[205]*z[206]) + qm*(z[36]*z[38]+z[45]*z[46]+z[53]*z[54]) +
rm*(z[191]*z[192]+z[198]*z[199]+z[205]*z[206]);
z[380] = bm*(z[36]*z[69]+z[45]*z[72]+z[53]*z[75]) + pm*(z[191]*z[234]+
z[198]*z[237]+z[205]*z[207]) + qm*(z[36]*z[101]+z[45]*z[104]+z[53]*z[107]) +
rm*(z[191]*z[222]+z[198]*z[225]+z[205]*z[207]);
z[381] = bm*(z[36]*z[70]+z[45]*z[73]+z[53]*z[76]) + pm*(z[191]*z[235]+
z[198]*z[238]+z[205]*z[208]) + qm*(z[36]*z[102]+z[45]*z[105]+z[53]*z[108]) +
rm*(z[191]*z[223]+z[198]*z[226]+z[205]*z[208]);
z[382] = z[372]*z[198];
```

```
z[383] = qm*(z[85]*z[36]+z[88]*z[45]+z[90]*z[53]);
z[384] = bm*(z[36]*z[262]+z[45]*z[263]+z[53]*z[264]) + pm*(z[191]*z[316]+
z[198]*z[317]+z[205]*z[318]) + rm*(z[191]*z[308]+z[198]*z[309]+z[205]*
z[310]) - qm*(z[80]*z[53]*z[271]+z[84]*z[36]*z[272]+z[87]*z[45]*z[272]+
z[89]*z[53]*z[272]-z[79]*z[36]*z[271]-z[86]*z[45]*z[271]-z[36]*z[268]-z[45]*
z[269]-z[53]*z[270]);
z[385] = am + bm*(pow(z[38],2)+pow(z[46],2)+pow(z[54],2)) + pm*(pow(z[192],
2)+pow(z[199],2)+pow(z[206],2)) + qm*(pow(z[38],2)+pow(z[46],2)+pow(z[54],2)) +
rm*(pow(z[192],2)+pow(z[199],2)+pow(z[206],2));
z[386] = bm*(z[29]*z[38]+z[30]*z[46]+z[31]*z[54]) + pm*(z[188]*z[192]+
z[195]*z[199]+z[202]*z[206]) + qm*(z[29]*z[38]+z[30]*z[46]+z[31]*z[54]) +
rm*(z[188]*z[192]+z[195]*z[199]+z[202]*z[206]);
z[387] = bm*(z[68]*z[38]+z[71]*z[46]+z[74]*z[54]) + pm*(z[192]*z[233]+
z[199]*z[236]+z[203]*z[206]) + qm*(z[100]*z[38]+z[103]*z[46]+z[106]*z[54]) +
rm*(z[192]*z[221]+z[199]*z[224]+z[203]*z[206]);
z[388] = bm*(z[38]*z[69]+z[46]*z[72]+z[54]*z[75]) + pm*(z[192]*z[234]+
z[199]*z[237]+z[206]*z[207]) + qm*(z[38]*z[101]+z[46]*z[104]+z[54]*z[107]) +
rm*(z[192]*z[222]+z[199]*z[225]+z[206]*z[207]);
z[389] = bm*(z[38]*z[70]+z[46]*z[73]+z[54]*z[76]) + pm*(z[192]*z[235]+
z[199]*z[238]+z[206]*z[208]) + qm*(z[38]*z[102]+z[46]*z[105]+z[54]*z[108]) +
rm*(z[192]*z[223]+z[199]*z[226]+z[206]*z[208]);
z[390] = z[372]*z[199];
z[391] = qm*(z[85]*z[38]+z[88]*z[46]+z[90]*z[54]);
z[392] = bm*(z[38]*z[262]+z[46]*z[263]+z[54]*z[264]) + pm*(z[192]*z[316]+
z[199]*z[317]+z[206]*z[318]) + rm*(z[192]*z[308]+z[199]*z[309]+z[206]*
z[310]) - qm*(z[80]*z[54]*z[271]+z[84]*z[38]*z[272]+z[87]*z[46]*z[272]+
z[89]*z[54]*z[272]-z[79]*z[38]*z[271]-z[86]*z[46]*z[271]-z[38]*z[268]-z[46]*
z[269]-z[54]*z[270]);
z[394] = z[393] + z[210]*z[355] + z[211]*z[359] + bm*(pow(z[69],2)+pow(
z[72],2)+pow(z[75],2)) + pm*(pow(z[207],2)+pow(z[234],2)+pow(z[237],2)) +
qm*(pow(z[101],2)+pow(z[104],2)+pow(z[107],2)) + rm*(pow(z[207],2)+pow(
z[222],2)+pow(z[225],2));
z[396] = z[395] + z[210]*z[354] + z[211]*z[358] + bm*(z[68]*z[69]+z[71]*
z[72]+z[74]*z[75]) + pm*(z[203]*z[207]+z[233]*z[234]+z[236]*z[237]) + qm*(
z[100]*z[101]+z[103]*z[104]+z[106]*z[107]) + rm*(z[203]*z[207]+z[221]*
z[222]+z[224]*z[225]);
z[399] = z[397] + z[210]*z[356] + z[211]*z[360] + bm*(z[69]*z[70]+z[72]*
z[73]+z[75]*z[76]) + pm*(z[207]*z[208]+z[234]*z[235]+z[237]*z[238]) + qm*(
z[101]*z[102]+z[104]*z[105]+z[107]*z[108]) + rm*(z[207]*z[208]+z[222]*
z[223]+z[225]*z[226]) - z[398];
z[402] = z[352] - z[400]*z[237] - z[401]*z[225];
z[403] = bm*(z[29]*z[69]+z[30]*z[72]+z[31]*z[75]) + pm*(z[188]*z[234]+
z[195]*z[237]+z[202]*z[207]) + qm*(z[29]*z[101]+z[30]*z[104]+z[31]*z[107]) +
rm*(z[188]*z[222]+z[195]*z[225]+z[202]*z[207]);
z[404] = qm*(z[85]*z[101]+z[88]*z[104]+z[90]*z[107]);
z[405] = z[326] + z[22]*z[344] + z[27]*z[343] + z[28]*z[342] + z[209]*
```

```
z[364] + z[210]*z[357] + z[210]*z[363] + z[211]*z[362] + bm*(z[69]*z[262]+
z[72]*z[263]+z[75]*z[264]) + pm*(z[207]*z[318]+z[234]*z[316]+z[237]*z[317]) +
rm*(z[207]*z[310]+z[222]*z[308]+z[225]*z[309]) - z[211]*z[361] - qm*(z[80]*
z[107]*z[271]+z[84]*z[101]*z[272]+z[87]*z[104]*z[272]+z[89]*z[107]*z[272]-
z[79]*z[101]*z[271]-z[86]*z[104]*z[271]-z[101]*z[268]-z[104]*z[269]-z[107]*
z[270]);
z[407] = z[406] + z[195]*z[354] + z[202]*z[358] + pm*(pow(z[203],2)+pow(
z[233],2)+pow(z[236],2)) + rm*(pow(z[203],2)+pow(z[221],2)+pow(z[224],2));
z[409] = z[408] + z[195]*z[355] + z[202]*z[359] + bm*(z[68]*z[69]+z[71]*
z[72]+z[74]*z[75]) + pm*(z[203]*z[207]+z[233]*z[234]+z[236]*z[237]) + qm*(
z[100]*z[101]+z[103]*z[104]+z[106]*z[107]) + rm*(z[203]*z[207]+z[221]*
z[222]+z[224]*z[225]);
z[412] = z[410] + z[195]*z[356] + z[202]*z[360] + bm*(z[68]*z[70]+z[71]*
z[73]+z[74]*z[76]) + pm*(z[203]*z[208]+z[233]*z[235]+z[236]*z[238]) + qm*(
z[100]*z[102]+z[103]*z[105]+z[106]*z[108]) + rm*(z[203]*z[208]+z[221]*
z[223]+z[224]*z[226]) - z[411];
z[413] = z[351] - z[400]*z[236] - z[401]*z[224];
z[415] = z[414] + pm*(z[188]*z[233]+z[195]*z[236]+z[202]*z[203]) + rm*(
z[188]*z[221]+z[195]*z[224]+z[202]*z[203]);
z[423] = z[325] + z[29]*z[344] + z[30]*z[343] + z[31]*z[342] + z[188]*
z[364] + z[195]*z[357] + z[195]*z[363] + z[202]*z[362] + bm*(z[68]*z[262]+
z[71]*z[263]+z[74]*z[264]) + pm*(z[203]*z[318]+z[233]*z[316]+z[236]*z[317]) +
rm*(z[203]*z[310]+z[221]*z[308]+z[224]*z[309]) - z[202]*z[361] - qm*(z[417]*
z[271]+z[418]*z[272]+z[419]*z[272]+z[420]*z[272]-z[421]*z[271]-z[422]*
z[271]-z[100]*z[268]-z[103]*z[269]-z[106]*z[270]);
z[425] = z[424] + z[213]*z[356] + z[214]*z[360] + bm*(pow(z[70],2)+pow(
z[73],2)+pow(z[76],2)) + pm*(pow(z[208],2)+pow(z[235],2)+pow(z[238],2)) +
qm*(pow(z[102],2)+pow(z[105],2)+pow(z[108],2)) + rm*(pow(z[208],2)+pow(
z[223],2)+pow(z[226],2));
z[428] = z[426] + z[213]*z[355] + z[214]*z[359] + bm*(z[69]*z[70]+z[72]*
z[73]+z[75]*z[76]) + pm*(z[207]*z[208]+z[234]*z[235]+z[237]*z[238]) + qm*(
z[101]*z[102]+z[104]*z[105]+z[107]*z[108]) + rm*(z[207]*z[208]+z[222]*
z[223]+z[225]*z[226]) - z[427];
z[431] = z[429] + z[213]*z[354] + z[214]*z[358] + bm*(z[68]*z[70]+z[71]*
z[73]+z[74]*z[76]) + pm*(z[203]*z[208]+z[233]*z[235]+z[236]*z[238]) + qm*(
z[100]*z[102]+z[103]*z[105]+z[106]*z[108]) + rm*(z[203]*z[208]+z[221]*
z[223]+z[224]*z[226]) - z[430];
z[432] = z[353] - z[400]*z[238] - z[401]*z[226];
z[433] = bm*(z[29]*z[70]+z[30]*z[73]+z[31]*z[76]) + pm*(z[188]*z[235]+
z[195]*z[238]+z[202]*z[208]) + qm*(z[29]*z[102]+z[30]*z[105]+z[31]*z[108]) +
rm*(z[188]*z[223]+z[195]*z[226]+z[202]*z[208]);
z[434] = qm*(z[85]*z[102]+z[88]*z[105]+z[90]*z[108]);
z[435] = z[324] + z[32]*z[343] + z[33]*z[342] + z[212]*z[364] + z[213]*
z[357] + z[213]*z[363] + z[214]*z[362] + bm*(z[70]*z[262]+z[73]*z[263]+
z[76]*z[264]) + pm*(z[208]*z[318]+z[235]*z[316]+z[238]*z[317]) + rm*(z[208]*
z[310]+z[223]*z[308]+z[226]*z[309]) - z[23]*z[344] - z[214]*z[361] - qm*(
```

```
z[80]*z[108]*z[271]+z[84]*z[102]*z[272]+z[87]*z[105]*z[272]+z[89]*z[108]*
z[272]-z[79]*z[102]*z[271]-z[86]*z[105]*z[271]-z[102]*z[268]-z[105]*z[269]-
z[108]*z[270]);
z[438] = z[436] + pm*(z[437]+pow(z[195],2)+pow(z[202],2)) + rm*(z[437]+pow(
z[195],2)+pow(z[202],2));
z[439] = z[372]*z[195];
z[447] = bm*(z[29]*z[262]+z[30]*z[263]+z[31]*z[264]) + pm*(z[188]*z[316]+
z[195]*z[317]+z[202]*z[318]) + rm*(z[188]*z[308]+z[195]*z[309]+z[202]*
z[310]) - qm*(z[441]*z[272]+z[442]*z[272]+z[443]*z[271]+z[444]*z[272]-
z[445]*z[271]-z[446]*z[271]-z[29]*z[268]-z[30]*z[269]-z[31]*z[270]);
z[449] = z[364] - z[400]*z[317] - z[401]*z[309];
z[450] = qm*(z[85]*z[268]+z[88]*z[269]+z[90]*z[270]);
z[456] = pow(papb,2);
z[457] = pow(z[456],0.5);
z[458] = papb/z[457];
z[459] = fabs(papb);
z[460] = sprconab*z[458]*(natlenab-z[459]) - dmpconab*vpba;
z[491] = z[460] - z[488]*z[36] - z[489]*z[45] - z[490]*z[53] - z[452]*(
z[29]*z[36]+z[30]*z[45]+z[31]*z[53]) - z[453]*(z[188]*z[191]+z[195]*z[198]+
z[202]*z[205]) - z[455]*(z[188]*z[191]+z[195]*z[198]+z[202]*z[205]);
z[502] = z[491] - z[447];
xatoi11 = z[3];
xatoi12 = z[8];
xatoi13 = z[9];
xatoi21 = z[10];
xatoi22 = z[11];
xatoi23 = z[12];
xatoi31 = -z[4];
xatoi32 = z[13];
xatoi33 = z[14];
ozeta1i1 = oaoi1 + (aleno2-paaoa1)*z[3] - paaoa2*z[8] - paaoa3*z[9];
ozeta1i2 = oaoi2 + (aleno2-paaoa1)*z[10] - paaoa2*z[11] - paaoa3*z[12];
ozeta1i3 = oaoi3 - paaoa2*z[13] - paaoa3*z[14] - (aleno2-paaoa1)*z[4];
ozeta2i1 = oaoi1 - paaoa2*z[8] - paaoa3*z[9] - (aleno2+paaoa1)*z[3];
ozeta2i2 = oaoi2 - paaoa2*z[11] - paaoa3*z[12] - (aleno2+paaoa1)*z[10];
ozeta2i3 = oaoi3 + (aleno2+paaoa1)*z[4] - paaoa2*z[13] - paaoa3*z[14];
alphaqa = atan2(ozeta1i3,ozeta1i1);
salphaqa = sin(alphaqa);
calphaqa = cos(alphaqa);
phiqa = atan2((-salphaqa*xatoi12+calphaqa*xatoi32),(salphaqa*xatoi13-calphaqa*xatoi33));
if(phiqa > twopi)
  phiqa -= twopi;
if(phiqa < 0.)
  phiqa += twopi;
if(phiqa < pio2 || phiqa > pi3o2)
  phiqa += Pi;
```

```
paq1aa1 = aleno2;
paq1aa2 = aradius*cos(phiqa);
paq1aa3 = aradius*sin(phiqa);
paq2aa1 = -aleno2;
paq2aa2 = paq1aa2;
paq2aa3 = paq1aa3;
phiqadt = ang3aip;  // approximate phiqadt with A's roll angle rate
paq1aa1p = 0.;
paq1aa2p = -aradius*phiqadt*sin(phiqa);
paq1aa3p = aradius*phiqadt*cos(phiqa);
paq2aa1p = 0.;
paq2aa2p = paq1aa2p;
paq2aa3p = paq1aa3p;
oq1ai2 = oaoi2 - (paaoa1-paq1aa1)*z[10] - (paaoa2-paq1aa2)*z[11] - (paaoa3-
paq1aa3)*z[12];
oq2ai2 = oaoi2 - (paaoa1-paq2aa1)*z[10] - (paaoa2-paq2aa2)*z[11] - (paaoa3-
paq2aa3)*z[12];
z11lin = oq1ai2-natleniqain;
z12lin = oq2ai2-natleniqain;
z1lin = z12lin;
qaflag = 2;
if(z11lin < z12lin)
{
  z1lin = z11lin;
  qaflag = 1;
}
z[539] = sin(alphaqa);
z[538] = cos(alphaqa);
z[541] = -z[3]*z[539] - z[4]*z[538];
z[543] = z[13]*z[538] - z[8]*z[539];
z[545] = z[14]*z[538] - z[9]*z[539];
z[547] = z[22]*z[541] + z[29]*z[543] - z[23]*z[545];
z[549] = z[27]*z[541] + z[30]*z[543] + z[32]*z[545];
z[551] = z[28]*z[541] + z[31]*z[543] + z[33]*z[545];
z[563] = z[163]*z[547] + z[167]*z[549] + z[169]*z[551];
z[557] = z[85]*z[547] + z[88]*z[549] + z[90]*z[551];
q1asoi3p = z[533]*z[547] + z[534]*z[549] + z[535]*z[551] + z[537]*z[563] +
z[536]*z[557] + (z[529]-paq1aa1)*z[541] + (z[532]-paq1aa3)*z[545] - z[543]*(
paq1aa2-z[531]);
z[570] = ozeta1i1/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[582] = z[539]*z[570];
z[574] = z[538]*z[570];
z[590] = z[4]*z[582] - z[3]*z[574];
z[600] = -z[8]*z[574] - z[13]*z[582];
z[608] = -z[9]*z[574] - z[14]*z[582];
z[614] = z[22]*z[590] + z[29]*z[600] - z[23]*z[608];
```

```
z[568] = ozeta1i3/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[577] = z[538]*z[568];
z[585] = z[539]*z[568];
z[593] = z[3]*z[577] - z[4]*z[585];
z[602] = z[8]*z[577] + z[13]*z[585];
z[610] = z[9]*z[577] + z[14]*z[585];
z[615] = z[22]*z[593] + z[29]*z[602] - z[23]*z[610];
z[565] = z[564]*z[2];
z[571] = ozeta1i1*z[565]/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[578] = z[538]*z[571];
z[586] = z[539]*z[571];
z[594] = z[3]*z[578] - z[2]*z[538] - z[4]*z[586];
z[603] = z[8]*z[578] + z[13]*z[586];
z[611] = z[9]*z[578] + z[14]*z[586];
z[616] = z[22]*z[594] + z[29]*z[603] - z[23]*z[611];
z[566] = paaoa2*ozeta1i3/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[583] = z[539]*z[566];
z[575] = z[538]*z[566];
z[591] = z[4]*z[583] - z[3]*z[575];
z[599] = -z[539] - z[8]*z[575] - z[13]*z[583];
z[609] = -z[9]*z[575] - z[14]*z[583];
z[617] = z[22]*z[591] + z[29]*z[599] - z[23]*z[609];
z[567] = paaoa3*ozeta1i3/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[584] = z[539]*z[567];
z[576] = z[538]*z[567];
z[592] = z[4]*z[584] - z[3]*z[576];
z[601] = -z[8]*z[576] - z[13]*z[584];
z[607] = -z[539] - z[9]*z[576] - z[14]*z[584];
z[618] = z[22]*z[592] + z[29]*z[601] - z[23]*z[607];
z[569] = z[564]*ozeta1i3/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[579] = z[538]*z[569];
z[587] = z[539]*z[569];
z[595] = z[3]*z[579] - z[539] - z[4]*z[587];
z[604] = z[8]*z[579] + z[13]*z[587];
z[612] = z[9]*z[579] + z[14]*z[587];
z[619] = z[22]*z[595] + z[29]*z[604] - z[23]*z[612];
z[572] = paaoa2*ozeta1i1/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[580] = z[538]*z[572];
z[588] = z[539]*z[572];
z[596] = z[3]*z[580] - z[4]*z[588];
z[598] = z[538] + z[8]*z[580] + z[13]*z[588];
z[613] = z[9]*z[580] + z[14]*z[588];
z[620] = z[22]*z[596] + z[29]*z[598] - z[23]*z[613];
z[573] = paaoa3*ozeta1i1/(pow(ozeta1i1,2)+pow(ozeta1i3,2));
z[581] = z[538]*z[573];
z[589] = z[539]*z[573];
```

```
z[597] = z[3]*z[581] - z[4]*z[589];
z[605] = z[8]*z[581] + z[13]*z[589];
z[606] = z[538] + z[9]*z[581] + z[14]*z[589];
z[621] = z[22]*z[597] + z[29]*z[605] - z[23]*z[606];
z[622] = z[27]*z[590] + z[30]*z[600] + z[32]*z[608];
z[623] = z[27]*z[593] + z[30]*z[602] + z[32]*z[610];
z[624] = z[27]*z[594] + z[30]*z[603] + z[32]*z[611];
z[625] = z[27]*z[591] + z[30]*z[599] + z[32]*z[609];
z[626] = z[27]*z[592] + z[30]*z[601] + z[32]*z[607];
z[627] = z[27]*z[595] + z[30]*z[604] + z[32]*z[612];
z[628] = z[27]*z[596] + z[30]*z[598] + z[32]*z[613];
z[629] = z[27]*z[597] + z[30]*z[605] + z[32]*z[606];
z[630] = z[28]*z[590] + z[31]*z[600] + z[33]*z[608];
z[631] = z[28]*z[593] + z[31]*z[602] + z[33]*z[610];
z[632] = z[28]*z[594] + z[31]*z[603] + z[33]*z[611];
z[633] = z[28]*z[591] + z[31]*z[599] + z[33]*z[609];
z[634] = z[28]*z[592] + z[31]*z[601] + z[33]*z[607];
z[635] = z[28]*z[595] + z[31]*z[604] + z[33]*z[612];
z[636] = z[28]*z[596] + z[31]*z[598] + z[33]*z[613];
z[637] = z[28]*z[597] + z[31]*z[605] + z[33]*z[606];
z[647] = z[85]*z[614] + z[88]*z[622] + z[90]*z[630];
z[648] = z[85]*z[615] + z[88]*z[623] + z[90]*z[631];
z[649] = z[85]*z[616] + z[88]*z[624] + z[90]*z[632];
z[650] = z[85]*z[617] + z[88]*z[625] + z[90]*z[633];
z[651] = z[85]*z[618] + z[88]*z[626] + z[90]*z[634];
z[652] = z[85]*z[619] + z[88]*z[627] + z[90]*z[635];
z[653] = z[85]*z[620] + z[88]*z[628] + z[90]*z[636];
z[654] = z[85]*z[621] + z[88]*z[629] + z[90]*z[637];
z[638] = z[163]*z[614] + z[167]*z[622] + z[169]*z[630];
z[639] = z[163]*z[615] + z[167]*z[623] + z[169]*z[631];
z[640] = z[163]*z[616] + z[167]*z[624] + z[169]*z[632];
z[641] = z[163]*z[617] + z[167]*z[625] + z[169]*z[633];
z[642] = z[163]*z[618] + z[167]*z[626] + z[169]*z[634];
z[643] = z[163]*z[619] + z[167]*z[627] + z[169]*z[635];
z[644] = z[163]*z[620] + z[167]*z[628] + z[169]*z[636];
z[645] = z[163]*z[621] + z[167]*z[629] + z[169]*z[637];
q1asoi3pp = z[646]*z[557]*vqb + z[533]*(z[614]*vaoii3+z[615]*vaoii1+z[616]*
ang2aip+z[617]*z[243]+z[618]*z[244]+z[619]*z[242]+z[620]*z[250]+z[621]*
z[251]) + z[534]*(z[622]*vaoii3+z[623]*vaoii1+z[624]*ang2aip+z[625]*z[243]+
z[626]*z[244]+z[627]*z[242]+z[628]*z[250]+z[629]*z[251]) + z[535]*(z[630]*
vaoii3+z[631]*vaoii1+z[632]*ang2aip+z[633]*z[243]+z[634]*z[244]+z[635]*
z[242]+z[636]*z[250]+z[637]*z[251]) + z[536]*(z[647]*vaoii3+z[648]*vaoii1+
z[649]*ang2aip+z[650]*z[243]+z[651]*z[244]+z[652]*z[242]+z[653]*z[250]+
z[654]*z[251]) + (z[529]-paq1aa1)*(z[590]*vaoii3+z[593]*vaoii1+z[594]*
ang2aip+z[591]*z[243]+z[592]*z[244]+z[595]*z[242]+z[596]*z[250]+z[597]*
z[251]) + (z[532]-paq1aa3)*(z[608]*vaoii3+z[610]*vaoii1+z[611]*ang2aip+
```

```
z[606]*z[251]+z[607]*z[244]+z[609]*z[243]+z[612]*z[242]+z[613]*z[250]) -
paq1aa1p*z[541] - paq1aa3p*z[545] - z[543]*(paq1aa2p-z[655]*vpba) - (paq1aa2-
z[531])*(z[600]*vaoii3+z[602]*vaoii1+z[603]*ang2aip+z[598]*z[250]+z[599]*
z[243]+z[601]*z[244]+z[604]*z[242]+z[605]*z[251]) - z[537]*(z[549]*z[298]+
z[551]*z[299]-z[638]*vaoii3-z[639]*vaoii1-z[547]*z[297]-z[640]*ang2aip-
z[641]*z[243]-z[642]*z[244]-z[643]*z[242]-z[644]*z[250]-z[645]*z[251]);
q2asoi3p = z[533]*z[547] + z[534]*z[549] + z[535]*z[551] + z[537]*z[563] +
z[536]*z[557] + (z[529]-paq2aa1)*z[541] + (z[532]-paq2aa3)*z[545] - z[543]*(
paq2aa2-z[531]);
q2asoi3pp = z[646]*z[557]*vqb + z[533]*(z[614]*vaoii3+z[615]*vaoii1+z[616]*
ang2aip+z[617]*z[243]+z[618]*z[244]+z[619]*z[242]+z[620]*z[250]+z[621]*
z[251]) + z[534]*(z[622]*vaoii3+z[623]*vaoii1+z[624]*ang2aip+z[625]*z[243]+
z[626]*z[244]+z[627]*z[242]+z[628]*z[250]+z[629]*z[251]) + z[535]*(z[630]*
vaoii3+z[631]*vaoii1+z[632]*ang2aip+z[633]*z[243]+z[634]*z[244]+z[635]*
z[242]+z[636]*z[250]+z[637]*z[251]) + z[536]*(z[647]*vaoii3+z[648]*vaoii1+
z[649]*ang2aip+z[650]*z[243]+z[651]*z[244]+z[652]*z[242]+z[653]*z[250]+
z[654]*z[251]) + (z[529]-paq2aa1)*(z[590]*vaoii3+z[593]*vaoii1+z[594]*
ang2aip+z[591]*z[243]+z[592]*z[244]+z[595]*z[242]+z[596]*z[250]+z[597]*
z[251]) + (z[532]-paq2aa3)*(z[608]*vaoii3+z[610]*vaoii1+z[611]*ang2aip+
z[606]*z[251]+z[607]*z[244]+z[609]*z[243]+z[612]*z[242]+z[613]*z[250]) -
paq2aa1p*z[541] - paq2aa3p*z[545] - z[543]*(paq2aa2p-z[655]*vpba) - (paq2aa2-
z[531])*(z[600]*vaoii3+z[602]*vaoii1+z[603]*ang2aip+z[598]*z[250]+z[599]*
z[243]+z[601]*z[244]+z[604]*z[242]+z[605]*z[251]) - z[537]*(z[549]*z[298]+
z[551]*z[299]-z[638]*vaoii3-z[639]*vaoii1-z[547]*z[297]-z[640]*ang2aip-
z[641]*z[243]-z[642]*z[244]-z[643]*z[242]-z[644]*z[250]-z[645]*z[251]);

// call ground interaction function and both control system functions
ground();
linear_actuator();
swinging_arm();

z[461] = paaoa2 - paq1aa2;
z[462] = paq1aa3 - paaoa3;
z[463] = paaoa3 - paq1aa3;
z[464] = paq1aa1 - paaoa1;
z[465] = paaoa1 - paq1aa1;
z[466] = paq1aa2 - paaoa2;
oq1ai1 = oaoi1 - (paaoa1-paq1aa1)*z[3] - (paaoa2-paq1aa2)*z[8] - (paaoa3-
paq1aa3)*z[9];
oq1ai3 = oaoi3 + (paaoa1-paq1aa1)*z[4] - (paaoa2-paq1aa2)*z[13] - (paaoa3-
paq1aa3)*z[14];
z[467] = sprcon13iq1a*(natleniq1a-oq1ai1) - dmpcon13iq1a*(z[8]*(z[463]*
waia1+z[464]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+z[9]*(z[465]*
waia2+z[466]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)-z[3]*(z[4]*vaoii3-
z[461]*waia3-z[462]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[468] = signspr1*sprconi2iq1a*(natleniq1a-oq1ai2) - dmpconi2iq1a*signdmp1*(
```

```
z[11]*(z[463]*waia1+z[464]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+
z[12]*(z[465]*waia2+z[466]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)-
z[10]*(z[4]*vaoii3-z[461]*waia3-z[462]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[469] = sprcon13iq1a*(natleniq1a-oq1ai3) - dmpcon13iq1a*(z[13]*(z[463]*
waia1+z[464]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+z[14]*(z[465]*
waia2+z[466]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)+z[4]*(z[4]*vaoii3-
z[461]*waia3-z[462]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[470] = paaoa2 - paq2aa2;
z[471] = paq2aa3 - paaoa3;
z[472] = paaoa3 - paq2aa3;
z[473] = paq2aa1 - paaoa1;
z[474] = paaoa1 - paq2aa1;
z[475] = paq2aa2 - paaoa2;
oq2ai1 = oaoi1 - (paaoa1-paq2aa1)*z[3] - (paaoa2-paq2aa2)*z[8] - (paaoa3-
paq2aa3)*z[9];
oq2ai3 = oaoi3 + (paaoa1-paq2aa1)*z[4] - (paaoa2-paq2aa2)*z[13] - (paaoa3-
paq2aa3)*z[14];
z[476] = sprcon13iq2a*(natleniq2a-oq2ai1) - dmpcon13iq2a*(z[8]*(z[472]*
waia1+z[473]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+z[9]*(z[474]*
waia2+z[475]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)-z[3]*(z[4]*vaoii3-
z[470]*waia3-z[471]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[477] = signspr2*sprconi2iq2a*(natleniq2a-oq2ai2) - dmpconi2iq2a*signdmp2*(
z[11]*(z[472]*waia1+z[473]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+
z[12]*(z[474]*waia2+z[475]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)-
z[10]*(z[4]*vaoii3-z[470]*waia3-z[471]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[478] = sprcon13iq2a*(natleniq2a-oq2ai3) - dmpcon13iq2a*(z[13]*(z[472]*
waia1+z[473]*waia3+z[8]*vaoii1+z[11]*vaoii2+z[13]*vaoii3)+z[14]*(z[474]*
waia2+z[475]*waia1+z[9]*vaoii1+z[12]*vaoii2+z[14]*vaoii3)+z[4]*(z[4]*vaoii3-
z[470]*waia3-z[471]*waia2-z[3]*vaoii1-z[10]*vaoii2));
z[479] = z[3]*z[10]*z[468] + z[3]*z[10]*z[477] + pow(z[3],2)*z[467] + pow(
z[3],2)*z[476] + z[8]*z[11]*z[468] + z[8]*z[11]*z[477] + z[8]*z[13]*z[469] +
z[8]*z[13]*z[478] + pow(z[8],2)*z[467] + pow(z[8],2)*z[476] + z[9]*z[12]*
z[468] + z[9]*z[12]*z[477] + z[9]*z[14]*z[469] + z[9]*z[14]*z[478] + pow(
z[9],2)*z[467] + pow(z[9],2)*z[476] - z[454]*z[35]*z[36] - z[454]*z[44]*
z[45] - z[454]*z[52]*z[53] - z[452]*(z[35]*z[36]+z[44]*z[45]+z[52]*z[53]) -
z[453]*(z[190]*z[191]+z[197]*z[198]+z[204]*z[205]) - z[455]*(z[190]*z[191]+
z[197]*z[198]+z[204]*z[205]) - z[3]*z[4]*z[469] - z[3]*z[4]*z[478];
z[480] = z[3]*z[10]*z[467] + z[3]*z[10]*z[476] + z[8]*z[11]*z[467] + z[8]*
z[11]*z[476] + z[9]*z[12]*z[467] + z[9]*z[12]*z[476] + pow(z[10],2)*z[468] +
pow(z[10],2)*z[477] + z[11]*z[13]*z[469] + z[11]*z[13]*z[478] + pow(z[11],2)*
z[468] + pow(z[11],2)*z[477] + z[12]*z[14]*z[469] + z[12]*z[14]*z[478] +
pow(z[12],2)*z[468] + pow(z[12],2)*z[477] - z[451] - z[454]*pow(z[36],2) -
z[454]*pow(z[45],2) - z[454]*pow(z[53],2) - z[452]*(pow(z[36],2)+pow(z[45],
2)+pow(z[53],2)) - z[453]*(pow(z[191],2)+pow(z[198],2)+pow(z[205],2)) -
z[455]*(pow(z[191],2)+pow(z[198],2)+pow(z[205],2)) - z[4]*z[10]*z[469] -
z[4]*z[10]*z[478];
```

```
z[481] = pow(z[4],2)*z[469] + pow(z[4],2)*z[478] + z[8]*z[13]*z[467] + z[8]*
z[13]*z[476] + z[9]*z[14]*z[467] + z[9]*z[14]*z[476] + z[11]*z[13]*z[468] +
z[11]*z[13]*z[477] + z[12]*z[14]*z[468] + z[12]*z[14]*z[477] + pow(z[13],2)*
z[469] + pow(z[13],2)*z[478] + pow(z[14],2)*z[469] + pow(z[14],2)*z[478] -
z[454]*z[36]*z[38] - z[454]*z[45]*z[46] - z[454]*z[53]*z[54] - z[452]*(
z[36]*z[38]+z[45]*z[46]+z[53]*z[54]) - z[453]*(z[191]*z[192]+z[198]*z[199]+
z[205]*z[206]) - z[455]*(z[191]*z[192]+z[198]*z[199]+z[205]*z[206]) - z[3]*
z[4]*z[467] - z[3]*z[4]*z[476] - z[4]*z[10]*z[468] - z[4]*z[10]*z[477];
z[482] = z[463]*z[8]*z[467] + z[463]*z[11]*z[468] + z[463]*z[13]*z[469] +
z[466]*z[9]*z[467] + z[466]*z[12]*z[468] + z[466]*z[14]*z[469] + z[472]*
z[8]*z[476] + z[472]*z[11]*z[477] + z[472]*z[13]*z[478] + z[475]*z[9]*
z[476] + z[475]*z[12]*z[477] + z[475]*z[14]*z[478] - z[454]*z[36]*z[101] -
z[454]*z[45]*z[104] - z[454]*z[53]*z[107] - z[452]*(z[36]*z[69]+z[45]*z[72]+
z[53]*z[75]) - z[453]*(z[191]*z[234]+z[198]*z[237]+z[205]*z[207]) - z[455]*(
z[191]*z[222]+z[198]*z[225]+z[205]*z[207]);
z[486] = z[462]*z[3]*z[467] + z[462]*z[10]*z[468] + z[465]*z[9]*z[467] +
z[465]*z[12]*z[468] + z[465]*z[14]*z[469] + z[471]*z[3]*z[476] + z[471]*
z[10]*z[477] + z[474]*z[9]*z[476] + z[474]*z[12]*z[477] + z[474]*z[14]*
z[478] - z[483]*z[36] - z[484]*z[45] - z[485]*z[53] - z[452]*(z[68]*z[36]+
z[71]*z[45]+z[74]*z[53]) - z[453]*(z[191]*z[233]+z[198]*z[236]+z[203]*
z[205]) - z[455]*(z[191]*z[221]+z[198]*z[224]+z[203]*z[205]) - z[462]*z[4]*
z[469] - z[471]*z[4]*z[478];
z[487] = z[461]*z[3]*z[467] + z[461]*z[10]*z[468] + z[464]*z[8]*z[467] +
z[464]*z[11]*z[468] + z[464]*z[13]*z[469] + z[470]*z[3]*z[476] + z[470]*
z[10]*z[477] + z[473]*z[8]*z[476] + z[473]*z[11]*z[477] + z[473]*z[13]*
z[478] - z[454]*z[36]*z[102] - z[454]*z[45]*z[105] - z[454]*z[53]*z[108] -
z[452]*(z[36]*z[70]+z[45]*z[73]+z[53]*z[76]) - z[453]*(z[191]*z[235]+z[198]*
z[238]+z[205]*z[208]) - z[455]*(z[191]*z[223]+z[198]*z[226]+z[205]*z[208]) -
z[461]*z[4]*z[469] - z[470]*z[4]*z[478];
z[494] = torqonr + z[492]*z[198] + z[493]*z[198];
z[495] = frconq - z[454]*z[126];
z[496] = z[479] - z[375];
z[497] = z[480] - z[384];
z[498] = z[481] - z[392];
z[499] = z[482] - z[405];
z[500] = z[486] - z[423];
z[501] = z[487] - z[435];
z[503] = z[494] - z[449];
z[504] = z[495] - z[450];
```

Distribution

5	MS0741	Rush Robinett, 6330
1	MS1002	Philip Heermann, 6470
1	MS1003	Barry Spletzer, 6470
20	MS1003	John Feddema, 6473
10	MS1003	Ray Byrne, 6473
10	MS1003	Jason Neely, 6473
10	MS1108	David Wilson, 6332
10	MS1010	Steven Buerger, 6473
10	MS1009	David Novick, 6474
10	MS1007	Scott Rose, 6471
10	MS1010	Stephen Buerger, 6473
1	MS1007	Larry Shipers, 6471
1	MS1125	Jake Deuel, 6472
2	MS9018	Central Technical Files, 8944
2	MS0899	Technical Library, 4536