DESIGNING ARTICULATED LEGS FOR RUNNING MACHINES

by

Woojin Lee

B.S. Mechanical Engineering
Massachusetts Institute of Technology
(1988)

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

May 1990

Signature of Author_____
Department of Mechanical Engineering
29 May 1990

Certified by_____
Marc Raibert
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by_____
A. A. Sonin
Chairman, Department Committee on Graduate Students

DESIGNING ARTICULATED LEGS FOR RUNNING MACHINES

by

Woojin Lee

Submitted to the Department of Mechanical Engineering
on May 31, 1990 in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Mechanical Engineering

Abstract

In previous work we built and tested a number of leg designs that are used by running robots having one, two and four legs. These designs use linear telescoping joints to change length and gas springs for axial compliance. The machine described in this thesis, the monopod, uses a rotary "ankle" joint for control of leg length and a fiberglass leafspring as the compliant element. We expect an articulated leg design to yield better performance, reliability, and simplicity relative to the telescoping leg. In this thesis we discuss the pros and cons of the monopod's articulated leg. The monopod's original articulated leg design had several mechanical advantages over the telescoping leg design such as its mechanical simplicity, lightness. However, this leg design also increased the difficulty of the control; forward velocity control was disturbed mainly due to the heel impact against the ground at low forward speed. As a possible solution to the heel impacting problem, an alternative improved leg design is proposed and implemented in which the monopod's foot is elevated on a hoof-like platform.

Although the monopod with the hoof successfully eliminated heel impact against the ground as planned, it introduced another problem; rolling of the hoof. Analysis of a simplified model of the monopod shows that rolling of the hoof is strongly influenced by two factors, the hip torque applied for body attitude control and the inertial loading due to forward acceleration. Two solutions to this problem are proposed. The hip torque can be coordinated with the downwards force applied at the hoof, and the maximum acceleration can be limited. The method of coordinating the hip torque is not successful because it disturbes the body attitude control whereas the maximum acceleration limiting method is successful in preventing the hoof from rolling.

Thesis Supervisor: Marc Raibert
Professor of Electrical Engineering and Computer Science

Acknowledgment

I would like to thank the members of the Leg Laboratory, especially Marc Raibert for continuous support, encouragement and his incredible insights, Jessica Hodgins for guidance and "A+" editting, Dave Barrett for his "piece of cake" assurance when it came to any mechanical design problems, Craig Hicks for his lovely "Elvis" voice and good times together, and Diane Dustman for her help with "administrative stuff".

Thanks also to my family for being there for me, to my O-thu-gi for waking me up every morning and having faith in me, and to my friends for late night China Town ventures and laughs. Lastly, I like to thank God for giving me everything I have, including the Brother Sun and Sister Moon who always make me smile.

Table of Contents

Chapter 1
Introduction

We daily rely on wheeled vehicles to bring us to desired places. Automobiles especially have integrated intimately into today's society, and have become a vital part of our lives. As a result, we build more and more paved roads for wheeled and tracked vehicles. However, even today, over a hundred years after the invention of automobiles, a wheeled vehicle can not reach most parts of dry land, simply because the terrain is either too soft or too rough. Rough terrain in this context includes both uneven and obstructed ground. We can try to make more area accessible by paving more roads, but an alternative solution may be legged locomotion, which we are all familiar with. In fact, much research is being conducted in the area of legged locomotion, and has produced many legged robots. However, performance of such legged machines is far more primitive than that of humans and animals, and further research is necessary.

Legged locomotion can be separated into two kinds: walking and running. We at the Leg Laboratory of Massachusett Institute of Technology are interested in legged locomotion, especially running. Over the past several years we have implemented and tested algorithms on running robots, including two and three dimensional robots with one, two, and four legs. All of these machines use linear telescoping legs. A passive air spring which is in series with a long-stroke hydraulic thrust actuator in the telescoping leg acts as an energy storage element which recovers part of the hopping energy at each landing and uses the stored energy to help power the next step. In addition to recovering hopping energy, implementation of the the linear telescoping leg also greatly simplifies the control task due to the system's dynamics being in purely polar coordinates. However, the telescoping leg has several limitations; namely its mechanical complexity and a large moment of inertia.

In contrast to the telescoping leg we have employed for the running robots, humans and animals have articulated legs with rotary joints. These articulated legs possess remarkable mechanical characteristics which the telescoping leg lacks, such as mechanical

simplicity, better ruggedness, a low moment of inertia and small friction at the joints. In order to obtain such desirable characteristics for the leg design of running machines, an articulated leg design was proposed as an alternative to the linear telescoping leg. This thesis describes the development of an articulated leg design. The articulated leg has a rotary ankle joint and stores hopping energy in a fiberglass leaf spring foot. The articulated leg has low moment of inertia, less unsprung mass, and was easier to build as planned. The articulated leg was tested on the monopod, a one-legged planar running machine (see figures 1-1 and 1-2), and the results showed that it performed quite well.

However this new design introduced several problems. A strong coupling exists between the vertical and the horizontal motions, particularly when the foot is tipped steeply downward, which is due to the circular motion of the ankle with respect to the toe planted on the ground. Such coupling substantially disturbed the forward velocity control when the machine was running at relatively low forward velocity. The angle of the foot and the ground was reduced in an attempt to minimize such coupling, but this in turn resulted in impact of the heel against the ground during the stance phase.

We have improved the design of the articulated leg by adding a hoof. The hoof eliminated the problem of heel impact by elevating the foot above the ground (see figure1-3). It also minimized the vertical and horizontal motion coupling of the running machine by allowing us to choose the foot angle so that the motion of the foot with respect to the horizontal was symmetrical. However, the hoof introduced a new problem, rolling of the hoof. We have investigated the causes of hoof rolling. We concluded that the sweeping force induced by the hip torque applied for body attitude control and the inertial loading due to forward acceleration are the main causes for rolling of the hoof. We tested several solutions to the problem, and developed a control strategy which successfully prevented rolling of the hoof.

1.1 Background

A good leg design is crucial for a legged system, and determines its overall performance. Much research has been done investigating both biomechanic and robotic legged systems, and leg design for such systems is always a major issue. Studies of human and animal movements show remarkable evolution of their anatomy for achieving

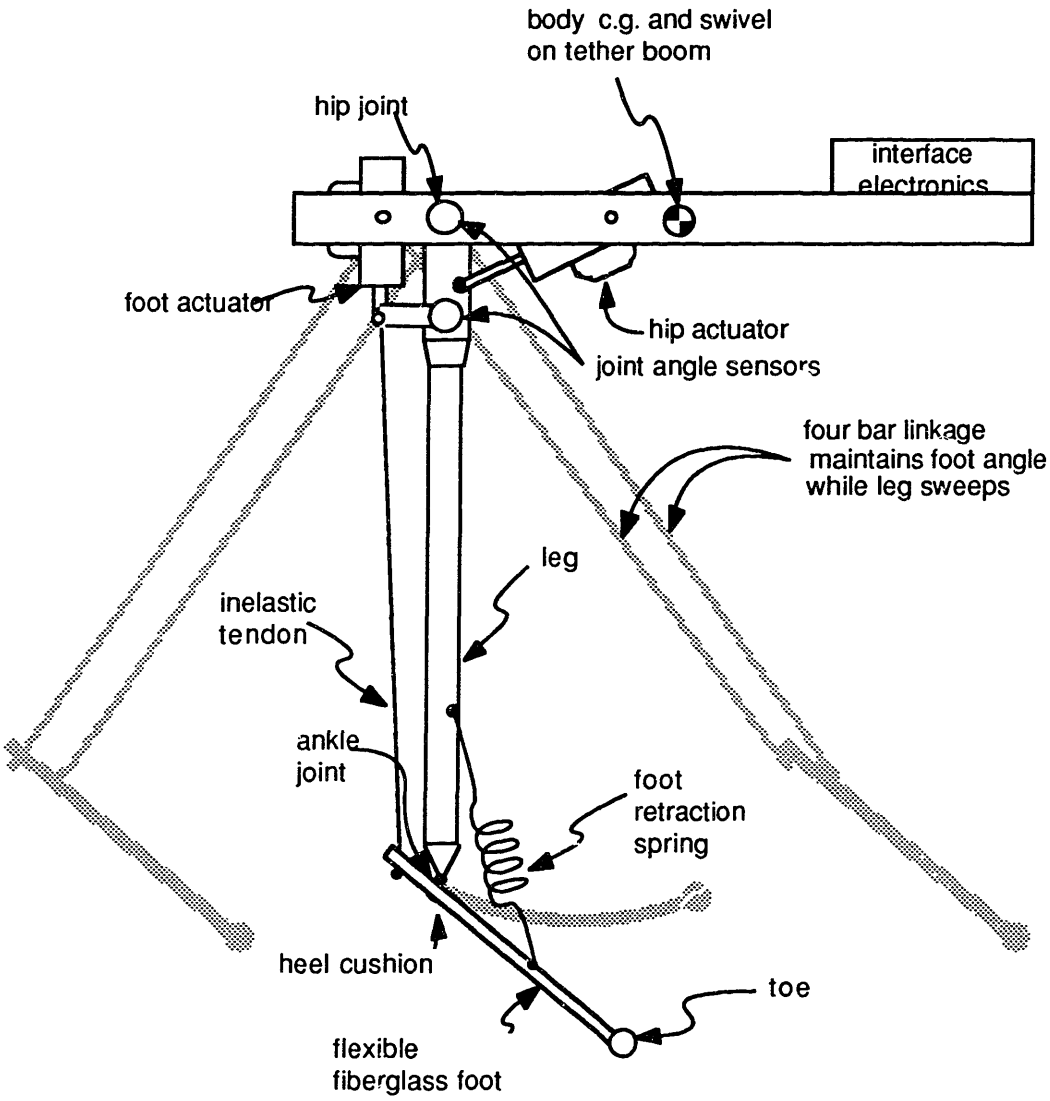


Figure 1-1: Diagram of monopod with articulated leg. The foot is a leafspring that deflects during hopping. The ankle is actuated through an inelastic tendon and hydraulic actuator mounted at the hip. A retraction spring attached to the foot maintains tension in the tendon. The linkage makes the foot angle with respect to the body nearly independent of the hip angle. Potentiometers measure the two joint positions and foot deflection. The unsprung mass is 0.063 kg and moment of inertia of the leg about the hip is 0.097 kg-m^2.

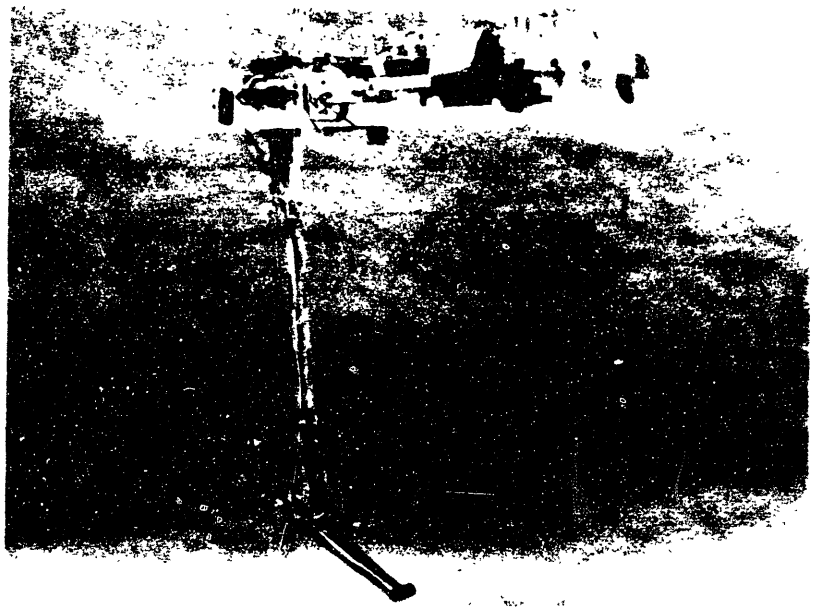


Figure 1-2: Photograph of the monopod. A long beam attached to the body allows mounting of weights to adjust location of the center of gravity and body moment of inertia. Metal tubing on frame carries hydraulic fluid which is fed through swivels to actuators. The aluminum arm and potentiometer on the foot are mounted to measure foot deflection.

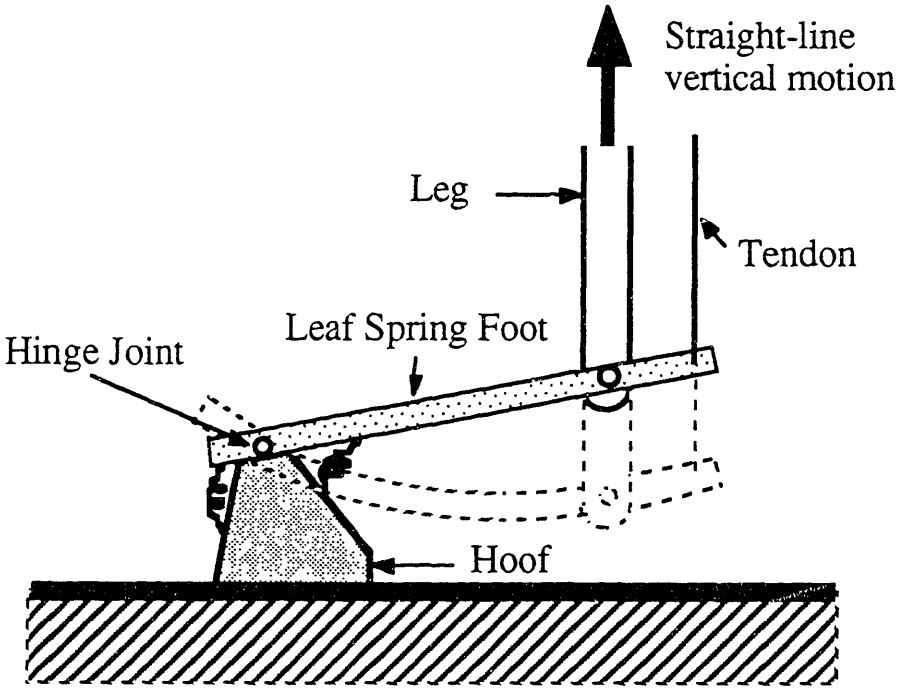


Figure 1-3: Drawing of alternative articulated leg design. This design places the toe on a hoof-like platform that elevates it with respect to the ground. It reduces the foot angle without causing the ankle to collide with the ground, and thus minimizes the coupling between the vertical motion and the horizontal motion.

energy efficiency and structural stability. Such studies of humans and animals provide an excellent guide on how legged machines should be built and controlled.

Biomechanics

In the area of biomechanical study of humans and animals, Milton Hildebrand (1960) shows how a certain animal's physical structure has evolved to achieve high running speed. For instance, Hildebrand explains how the horse and the cheetah add several inches to their stride length, thus increasing their running speed, by swiveling the shoulder blades. In addition, he shows how the legs of the cheetah are adapted for power and speed. Placement of the teres major muscle in the leg is analogous to the gear ratio in conventional mechanical system; in the cheetah the small distance between the muscle insertion and the joint it moves yields a high rate of oscillation. A higher rate of oscillation coupled with a longer leg yields a faster stride.

R. McNeill Alexander (1988) extensively studied the mechanical properties and mechanisms of human and animal movement, especially their elastic mechanisms. Alexander discussed the mechanical property of the tendons of animals and humans, and he showed how the elastic properties of such tendons reduce the work the muscles have to do in hopping or running. For instance, according to Alexander, the hoofed mammals achieve even more economical running by employing remarkably short muscle fibres, especially in the plantaris and interosseous and in similarly placed muscles in the foreleg. (This conclusion is based on the assumption that less energy is required to activate short muscle fibres than longer fibres.) In addition, the galloping animal is found to utilize the elastic structures in the back, making running much more energy efficient than the non-galloping animal. In addition to Hildebrand and Alexander many other researchers have also been studying the biomechanics and locomotion of humans and animals.

Robotics

Much of the research on legged locomotion machines has been based on statically stable walking --- or rather crawling --- movements. In statically stable walking, a solid base of support must be present at all times. This may consist of three legs forming a

tripod of support within which the center of mass of the body must lie. A statically stable walking machine using this technique must have at least four legs: three to form a basis of support while the other moves forward.

An example of a six legged walking machine is the Ohio State University's Adaptive Suspension Vehicle (ASV) (Waldron *et al.* 1984) which can carry a driver and attain a forward speed of up to 5 miles per hour. Energy effieciency and simplicity of control were the main issues in designing the leg mechanism for the ASV. As a result, planar pantograph mechanisms were chosen for the legs. The pantograph mechanism decouples the horizontal and the vertical forces and motions and eliminates unnecessary potential energy loss. However, because the ASV was designed to walk in a statically stable fashion, the legs were designed to be rigid with minimum compliance.

Another good example of a statically stable walking machine is a quadruped vehicle by Hirose (1980), the PV II (Perambulating Vehicle Mark II). Hirose studied the importance of the mechanical aspects of a walking vehicle, which have been neglected in recent computer oriented studies. He concluded that in the advent of a practical walking vehicle an insect type leg mechanism with a sprawling wide-track configuration is superior to mammal-like upright legs for the following reasons: 1) Long legs can be used to keep the body's center of gravity lower and maintain high stability. 2) Long legs enable the walking vehicle to walk faster and is adaptive to comparatively large unevenness of the ground. 3) The locomotional energy efficiency of the insect-type leg is not inferior to the mammal type, contrary to our general anticipation. The mammal-type leg is advantageous only when the machine is standing still. Hirose then chose the "3-dimensional cartesian coordinate pantographic mechanism" abbreviated as PANTOMEC since it decouples the x, y and z direction motions and forces as discussed above, resulting in good energy efficiency and controllability while maintaining the advantages of the insect-type leg configuration. As in the ASV, the PV II was designed to walk in a statically stable fashion and thus the legs were designed to be rigid.

In contrast, we are studying dynamically stable running robots in which the legged system is not statically supported by a tripod of legs. In running the compliance of the leg is essential for storing energy and cushioning the impact with the ground. The legs of animals also deform substantially under load. Compliant legs improve locomotion efficiency by recycling part of the kinetic energy from stride to stride and reducing the maximum structural loading. Such factors are especially important in running.

1.2 Mechanism for Elastic Storage

The role of elastic storage elements in running systems is crucial. The efficiency as well as the mechanical characteristics of the elastic storage help determine the performance of the running creature, whether it is an animal or a machine. Therefore, before describing the specific designs we have studied, it seems most appropriate to turn to a brief discussion of energy storage in elastic materials (H. B. Brown 1989).

Many different materials have been examined for possible use as elastic storage elements. Figure 2-1 shows the ratio of storable elastic energy to the mass of the material for several spring materials. Each material has the springiness which is necessary for an elastic storage element of a running system, but each has drawbacks as well.

Steel is used in numerous applications as a springy material. Due to its isotropic property, steel can easily be formed into desired shapes such as coils. However, as shown in figure 1-4, steel has relatively poor energy to mass ratio, about 140 J/kg. Fiberglass has roughly six times the energy capacity of steel, but because its fiber orientation is crucial, it is not so easily fashioned into a spring. Fiberglass is most readily used in bending as a leafspring or other beam shape.

Rubber and animal tendons have substantially higher energy capacities than steel, about 5000 J/kg compared to 140 J/kg for steel. The value for animal tendon is based on Alexander's work with dogs (Alexander,1974). Because these materials can undergo large elastic strains they can provide usable deflections in pure tension. The 10% strain of the Achilles tendon of an animal is compatible with the short lever arm to which the tendon is attached behind the ankle joint. For example, Alexander's data for the dog indicate that its Achilles tendon is linked to the foot so that it undergoes about one-fourth the deflection of the toe. No material usable in human-made machines has been found with equivalent elastic properties. Rubber is closest to the tendon in its mechanical nature but its strain is about 50 times as much as animal tendon, so it cannot be used directly in a leg design. Vulcanization of rubber increases the stiffness of the material by forming crosslinks between the rubber strands, but the elasticity degrades substantially.

Gas compressed in a container with high specific strength has a very high energy capacity, about 240,000 J/kg. A usable gas spring requires a cylinder and piston or comparable hardware, however, which will weigh many orders of magnitude more than the

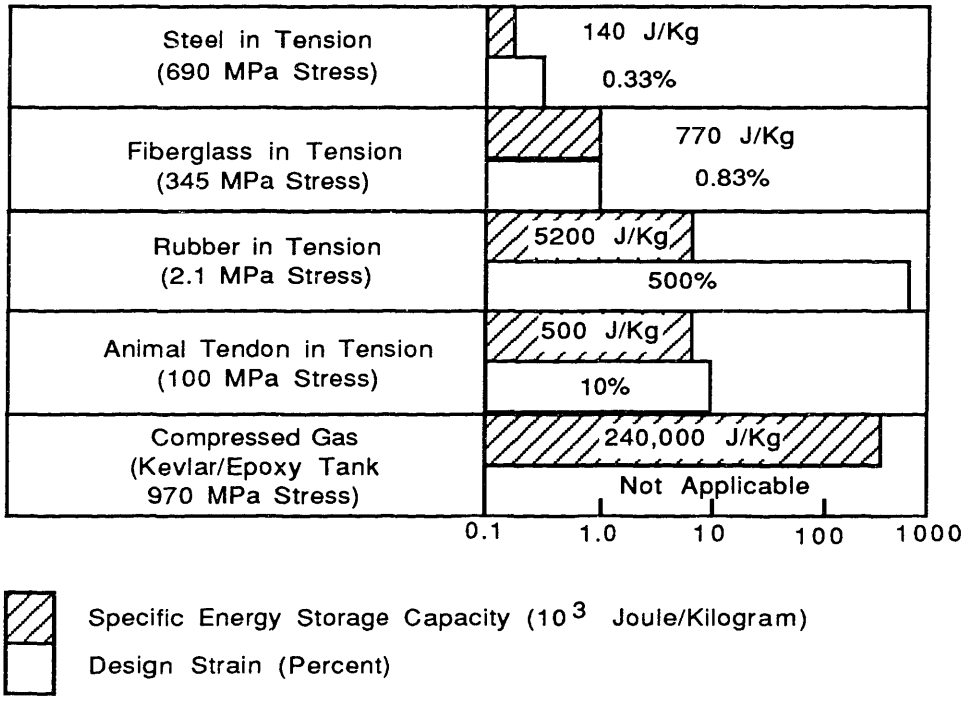


Figure 1-4: Strain energy per unit mass for various materials. The higher the value, the less mass of material needed for a given energy storage function. The strain, or relative elongation of the material, affects the design of the spring. (Brown 1986)

gas itself. Frictional and thermodynamic losses can be substantial. Still, gas springs may be used effectively if they are compatible with the overall design.

Such gas spring can be effectively implemented to a telescoping joint type leg mechanism. Fiberglass also has a desirable energy to mass ratio, but its anisometric property is a drawback since it can only be used effectively in bending. However, the bending motion of the fiberglass can be effectively implemented as linear motion of a rotary joint leg mechanism. As an alternative to the gas spring of a telescoping leg, the fiberglass leaf spring is therefore chosen as the elastic energy storage element for the articulated leg.

Chapter 2
Experimental Apparatus: Monopod

The monopod consists of a body on which the interface electronics are mounted, a leg which is connected to the body by a pivot joint at the hip, a fiberglass leafspring foot and a hoof connected to the toe by a hinge joint. (See figure 2-1.) The monopod was designed and built by Ben Brown and Marc Raibert. Originally the machine ran on the toe of the foot. The machine has since been modified by the addition of the hoof.

This machine is constrained to operate in the plane by a tether mechanism that permits forward and vertical translation and pitch rotation. We call this mechanism the planarizer. This is mounted over a treadmill so that the monopod may run freely. The foot is actuated about a rotary ankle joint by a linear hydraulic actuator which pulls on the foot through an inelastic tendon. This actuator is located at the hip to minimize the rotational inertia of the leg. A second hydraulic actuator drives the swing motion of the hip. The hydraulic valves controlling the actuators are flow control valves Model 30 by MOOG. The original machine is designed to run on its toe, which is located directly below the center of mass of the system when the leg is vertical. The present design has a hoof which elevates the toe from the ground and prevents the heel from striking the ground during stance. The hip is offset from the center of mass by a distance roughly equal to the offset of the ankle with respect to the toe, so the leg is nominally vertical when the hip is centered. The four-bar linkage formed by the leg, heel lever, body, and tendon keeps the orientation of the foot with respect to the body nearly constant as the leg swings fore and aft. Appendix A gives the physical parameters for the machine. Appendix B describes the kinematics of the machine in detail.

The number of sensors used for control of the monopod is small. Rotary potentiometers measure the angles of the hip with respect to the body and of the ankle with respect to the leg. A rotary potentiometer connected to the ankle and the toe measures deflections of the foot and contact with the ground. Optical encoders on the planarizer measure the horizontal and vertical positions of the hip and the angle of the body.

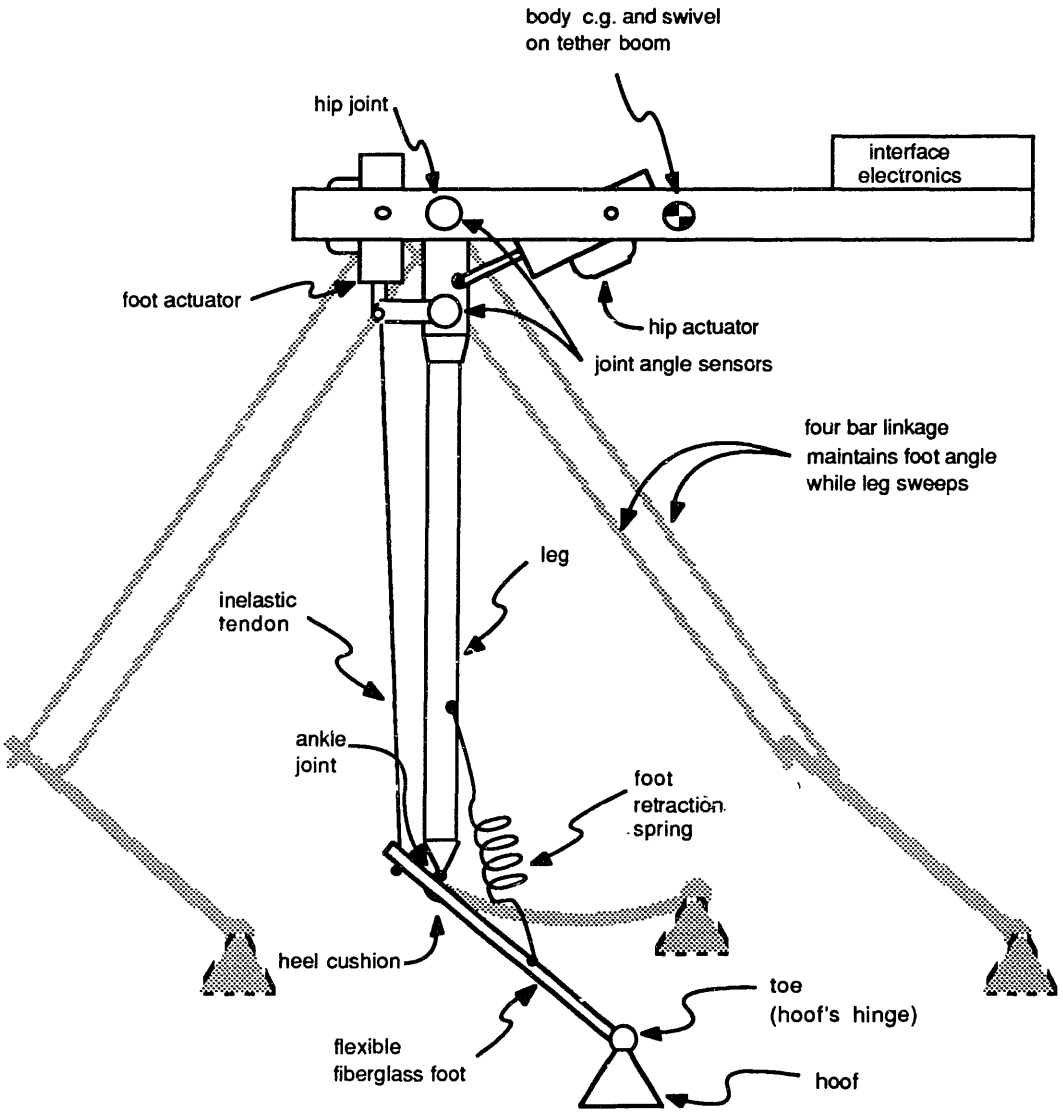


Figure 2-1: Diagram of monopod with articulated leg. The foot is a leafspring that deflects during hopping. The ankle is actuated through an inelastic tendon and hydraulic actuator mounted at the hip. A retraction spring attached to the foot maintains tension in the tendon. The linkage makes the foot angle with respect to the body nearly independent of the hip angle. Potentiometers measure the two joint positions and foot deflection. The unsprung mass is 0.063 kg and moment of inertia of the leg about the hip is 0.097 kg-m^2. The original monopod did not have a hoof. It was added later. The hoof elevates the toe from the ground to prevent the heel from hitting the ground.

A tachometer measures the velocity of the treadmill. Appendix C contains further information on the design of the planarizer.

Hopping height is determined by the amount of energy injected into the springy foot through the inelastic tendon during stance. Shortening the tendon excites the spring-mass system formed by the springy foot and the body. The bouncing height and spring energy on successive passive bounces indicate an efficiency of the monopod's springy foot of about 0.67.

The monopod is powered by a hydraulic pump with a supply pressure of approximately 1500 psi. The monopod is connected by an umblical cable containing two hydraulic lines (one supply and one return), cables for the electrical power, cables for interface to the computer controlling the monopod and cables for the interface to the encoder support electronics. The computer controlling the monopod is a DEC VAX-11/785[+] . A device driver built into a modified version of the UNIX[❖] operating system kernel is used for the control program . The device driver reads the sensors from the monopod and calculates the values of appropriate variables using the kinematics of the system. The control program then outputs the control signals to the hydraulic valves. The control program runs every 4 ms. For further information on the interface electronics, refer to Raibert *et al.* (1985) and Chepponis (1987).

2.1 Primary Control Algorithm for monopod

All of the running machines built at the Leg Laboratory have used similar control algorithms. In each case the control task is decomposed into three parts as shown in figure 2-2 (Raibert, 1986). The first part stabilizes the forward running motion, the second part is responsible for the bouncing of the machine, and the third part maintains the angle of the body at the desired value. The three part decomposition technique relies on limited coupling among the hopping height, the body attitude and the forward speed controls. The cyclic state machine shown in figure 2-3 is used and the control system activates the appropriate servo accordingly.

[+] VAX is a trademark of Digital Equipment Corporation
[❖] UNIX is a trademark of AT&T Bell Laboratories

Hopping Height

Hopping is basically a resonant bouncing motion of a mass-spring system. The mechanical losses in each bounce and the energy supplied by the thrust actuator determine the hopping height. In the monopod, the thrust actuator maintains the hopping cycle by changing the setpoint for the angle of the lever at the rear of the foot (θ_{heel}). This lengthens the zero point of the foot spring, thus supplying the energy lost mechanically . The desired hopping height is thus transformed to the amount of thrust by the control system with a linear servo of the form

$$t = -k_p(\theta_{heel} - \theta_{heelsp}) - k_v\dot{\theta}_{heel} \tag{2.1}$$

where

t	is the signal to the hydraulic servo valve,
θ_{heelsp}	is the setpoint for the heel angle,
$\dot{\theta}_{heel}$	is the time derivative of θ_{heel}, and
k_p, k_v	are position and velocity gains.

Body Attitude

Body attitude is maintained at a desired angle by applying a torque at the hip with the hip actuator when the machine is in stance phase. The vertical loading on the foot induces enough friction between the toe and the ground that the hip torque can be used to correct body attitude without causing the foot to slip. The linear servo which corrects the body attitude is

$$\tau = k_p(\phi_d - \phi) + k_v\dot{\phi} \tag{2.2}$$

where

τ	is the hip torque signal sent to the hydraulic hip actuator,
ϕ	is the body angle, ϕ_d is the desired body angle,
$\dot{\phi}$	is the body rotation rate, and
k_p, k_v	are proportional-derivative servo gains.

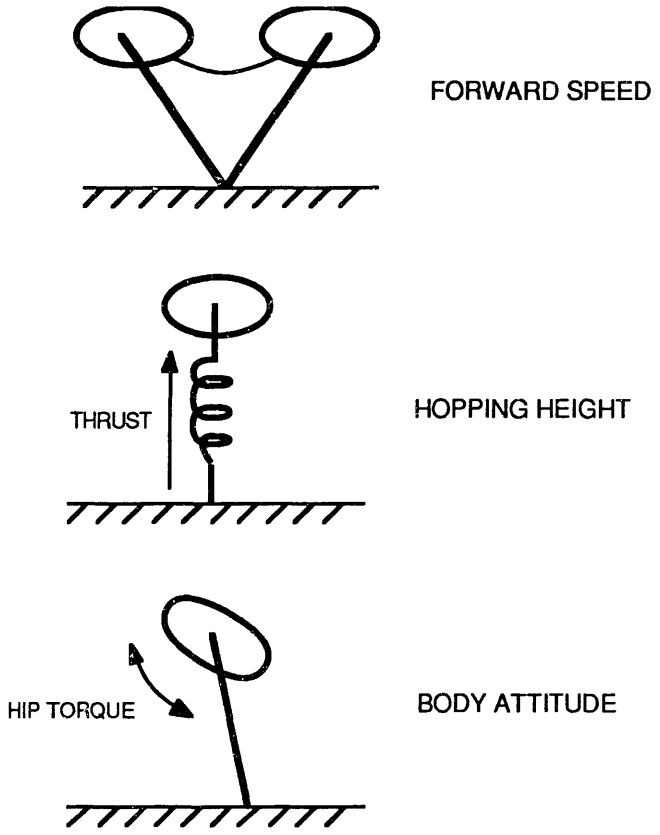


Figure 2-2: The control system for the running machines consists of three parts; one controls the forward running speed, the second controls the hopping height and the third controls the body attitude.

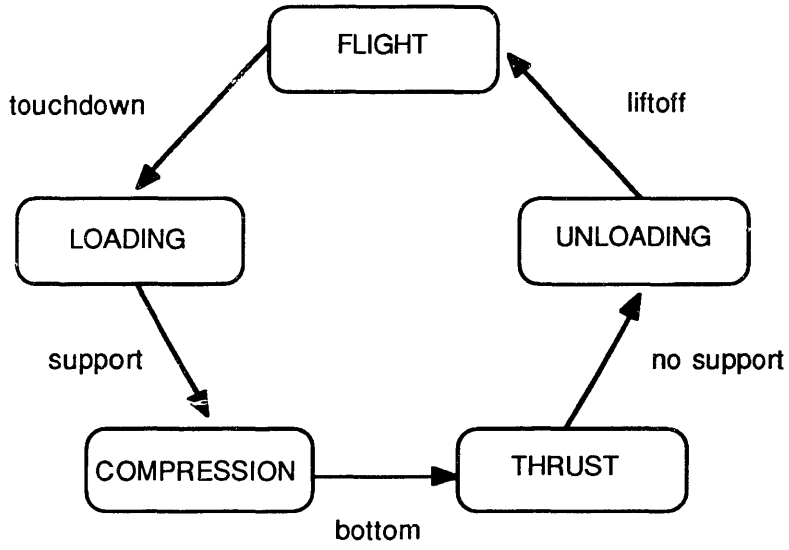


Figure 2-3: Cyclic state machine for one-legged hopping machine. Sensor events trigger the state transitions.

Forward Running Speed

The running machine places its foot during the flight phase to control the net acceleration it experiences during the following stance phase. (Figure 2-4) The placement of the foot which causes the next liftoff forward velocity to be equal to the previous touchdown forward velocity is called the *neutral point.*. The net acceleration of the machine during the stride is determined by the deviation of the foot placement from the neutral point. The experiments showed that there exists roughly a linear relationship between the net acceleration during the stride and the deviation of the foot placement from the neutral point. Thus, the control system displaces the foot from the neutral point proportional to the difference between the desired speed and the actual speed:

$$x_{fh,d} = \frac{\dot{x} T_s}{2} + k_{\dot{x}} (\dot{x} - \dot{x}_d) \tag{2.3}$$

where

$x_{fh,d}$ is the forward displacement of the foot from the projection of the center of gravity,

$\dot{x}$ is the forward velocity,

$\dot{x}_d$ is the desired forward velocity,

T_s is the predicted duration of the next support period, and

$k_{\dot{x}}$ is a velocity correction gain.

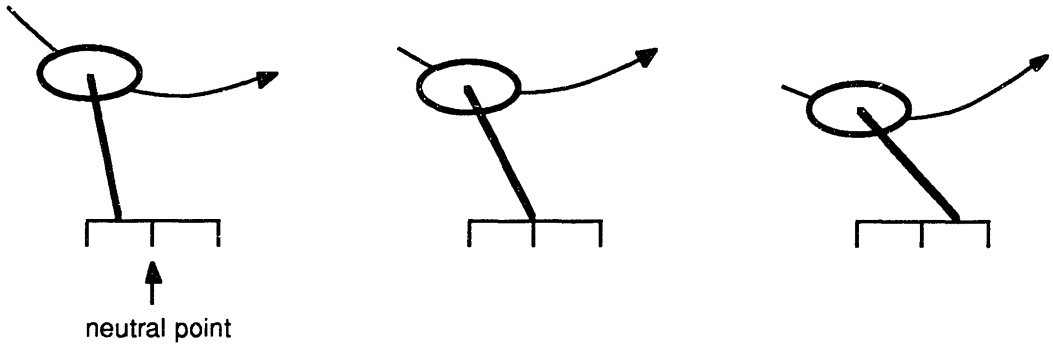


Figure 2-4: The running machine places its foot during the flight phase to control the forward speed. The placement of the foot which causes the liftoff forward velocity to be equal to the touchdown forward velocity is called the neutral point. The net acceleration of the machine during the stride is determined by the deviation of the foot placement from the neutral point. If the machine places the foot closer to the hip than the neutral point, it will accelerate, and if the machine places the foot further from the hip than the neutral point, it will decelerate.

Chapter 3
Problem Definition

In this chapter, the pros and cons of the telescoping leg and the monopod's articulated leg are studied. As discussed previously, the compressed gas has a very high energy capacity, and the telescoping leg mechanism, in which the gas spring and the thrust actuator are in series, is used for the running robots. The telescoping however poses some mechanical disadvantages, and thus the articulated leg with rotary joint ankle was investigated and implemented on the monopod. The articulated leg eliminated the problems posed by the telescoping leg, but it introduces new problems.

3.1 Telescoping Legs

The telescoping leg shown in figure 3-1 uses compressed gas as the mechanism for elastic energy storage. It has a long-stroke hydraulic actuator that operates in series with a passive air spring. One function of the air spring is to recover part of the hopping energy during landing and to return it during the following upward acceleration. This improves the efficiency of locomotion. A second function is to provide a cushion for the system, reducing the maximum impact force that the machine experiences on each landing. A third function of the air spring is to simplify control since the desired body motion is largely the passive oscillation of the body rebounding on the springy leg. In other words, the control system's task was to excite and modulate the vertical hopping oscillation, but the trajectory was determined by the passive oscillation. In addition, the legged robot using telescoping legs (figure 3-2) has an advantage over other designs because its dynamics are in pure polar coordinates. The unsprung mass is relatively small, and there is no kinematic coupling between the movement of the rotary hip and the linear movement of the telescoping leg. Such a design leads to simplicity in analyzing and controlling the system. Also the ability to lengthen and retract substantially proved to be important since the idle leg in swing phase must remain clear of the ground while the other leg is in stance. The idle leg must be able to shorten to less than the shortest length seen by the support legs during stance. This design has been used successfully in experiments with

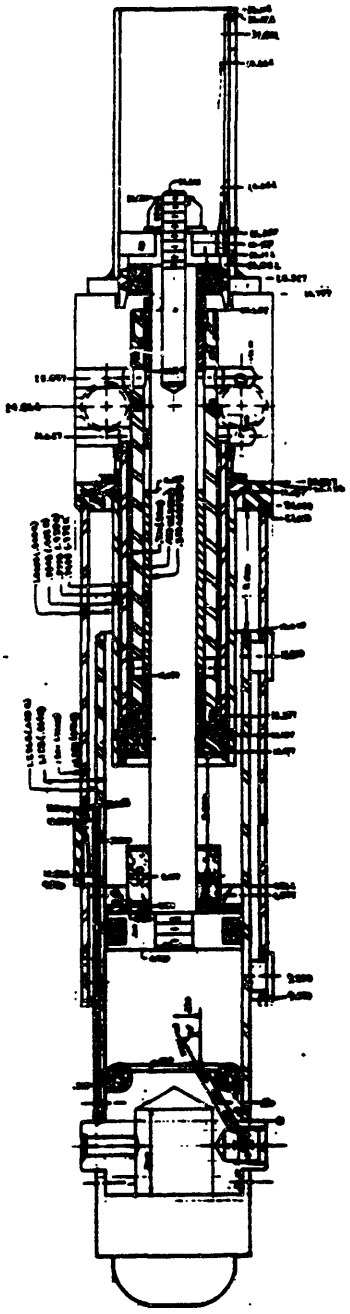


Figure 3-1: Engineering drawing of the hydraulic-pneumatic leg, showing details of the hydraulic actuator, air spring and position sensors. Leg is shown compacted in the axial direction. A long-stroke hydraulic actuator provides controlled axial thrust and rapid retraction. An air chamber near the foot provides the spring. To reduce friction in the hydraulic actuator, all high-pressure seals are clearance seals (0.025mm), with O-ring seals used to contain low-pressure leakage oil at the rods. Space between concentric cylinders provides paths for control and leakage flow to the lower end of the hydraulic actuator. The hydraulic actuator is servoed with a conventional high-bandwidth flow-control servo valve. The air cylinder forms the lower part of the leg and slides inside plastic guide buttons mounted in the upper leg tube. The foot includes a pneumatic check valve that allows makeup flow to the air spring, but prevents out-flow when the air spring is compressed. The hydraulic actuator has a 0.23m travel and the air spring has a 0.10 m travel. At 17.5 MPa (2500 psi) hydraulic pressure, maximum thrust is about 950 N and maximum speed is about 2 m/s. The unsprung mass is 0.24 kg and moment of inertia of the leg about the hip is 0.13 kg-m^2. This leg design was used in a planar biped (Hodgins, Koechling, and Raibert 1986), and in a quadruped running machine (Raibert, Chepponis and Brown 1986).

a one-legged hopper, a planar biped, a quadruped, and a 3-D biped, but still it has several limitations:

- The leg is heavy.
- The seal leakage and friction during compression degrade the resilience of the air spring.
- The sliding joint is mechanically complex, bulky, and subject to wear and looseness.
- Measuring leg length requires a long, specially made sensor.
- Wires to the foot must go through slack cables that are vulnerable to a variety of hazards.
- The moment of inertia of the leg is substantially larger than desired.

These limitations have motivated us to explore articulated legs that use only rotary joints.

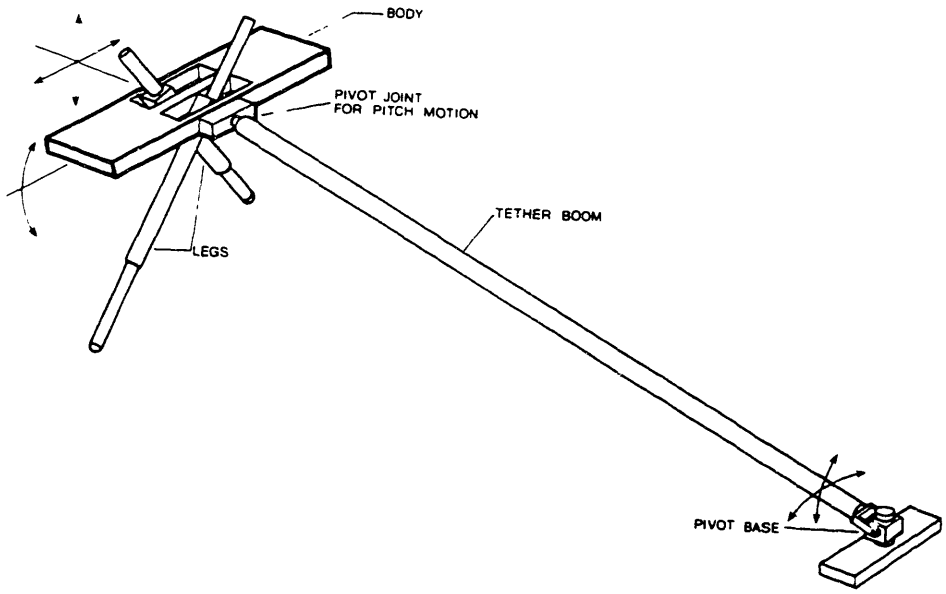


Figure 3-2: Diagram of planar biped with the telescoping legs used for experiments.

3.2 Articulated Leg

As a possible solution to the limitations of the linear telescoping leg, as well as to explore more animal-like leg design, the articulated leg shown in figure 3-3 was designed and tested . A suitable tendon material for such a design has not been found; therefore the

fiberglass leafspring is used as the mechanism for elastic energy storage. Energy is stored in the bending of the foot. This design provides compliance mainly in the radial direction and not in the tangential direction, so the functionality of the articulated leg is very similar to the telescoping leg we have discussed.
The articulated leg has several advantages over telescoping legs:

- It is mechanically simple and sturdy.
- It is light.
- The design combines the structural and elastic functions into a single unit minimizing mass. Because of the distributed nature of the leafspring, the effective unsprung mass is small, resulting in low impact forces during running and small energy losses.
- It requires simpler sensors such as rotary potentiometer for control. The telescoping leg on the other hand requires a long, specially made linear sensor.

To test the springy-foot concept the one-legged machine, the monopod was built. The hip actuator controls either the body attitude or the foot placement depending on whether the foot is on the ground or not. The thrust actuator in series with the foot spring supplies extra energy to the fiberglass leafspring foot by shifting the set point through an inelastic tendon.

One advantage of the leafspring foot is that its unsprung mass is very low compared to that of the telescoping legs on previous machine. Unsprung mass is the mass whose kinetic energy is lost at touchdown. The bouncing efficiency of a machine, that is the fraction of the circulating energy recovered from one hopping cycle to the next, is limited by the unsprung mass, as given by

$$\eta_{max} = \left(1 - \frac{M_{un}}{M_{sys}}\right)^2 \qquad (3.1)$$

where

η_{max} is the maximum theoretical bouncing efficiency,
M_{sys} is the mass of the whole system, and
M_{un} is the unsprung mass.

This equation accounts for the losses in kinetic energy of the system that occur at touchdown and lift-off. It ignores, of course, the energy needed for swinging the leg and for the various control functions, and frictional losses. Based on this equation and an unsprung mass ratio of 0.008 for the monopod, we would expect a negligible loss (1.6%) in bounc-

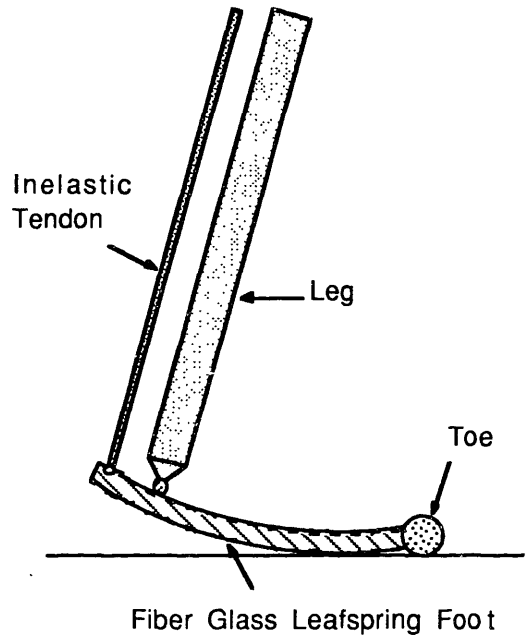


Figure 3-3: Diagram of the articulated leg. The articulated foot consists of the fiberglass leafspring foot connected to the leg by a hinge joint and the inelastic tendon which thrusts by changing the setpoint of the foot spring. The design provides compliance mainly in the radial direction and not in the tangential direction with respect to the center of the mass.

ing efficiency due to foot impacts. The passive bouncing of the monopod indicates an efficiency of about 0.67. Energy lost due to the friction at the planarizing mechanism as well as the impact at landing is probably responsible for why the bouncing efficiency is not higher.

However, this design has one major drawback. There is a strong coupling between the sweeping motion of the hip and the thrusting motion of the ankle, particularly when the foot is tipped steeply downward. As illustrated in figure 3-4, this phenomena is due to the circular motion of the ankle with respect to the toe planted on the ground. The misalignment of the ground reaction force through the center of gravity of the system increases with larger foot angles α, causing a forward pitching moment. Such coupling resulted in substantial disturbance to the forward velocity control in the original design of the monopod. In order to minimize such coupling, either the length of the foot can be increased or the angle between the foot and the ground can be reduced. However, neither alternative is acceptable. Increasing the length of the foot increases the compliance and therefore the distance the foot deflects. Reducing the angle between the foot and the ground causes the heel to hit the ground during stance, which disturbs the body attitude as

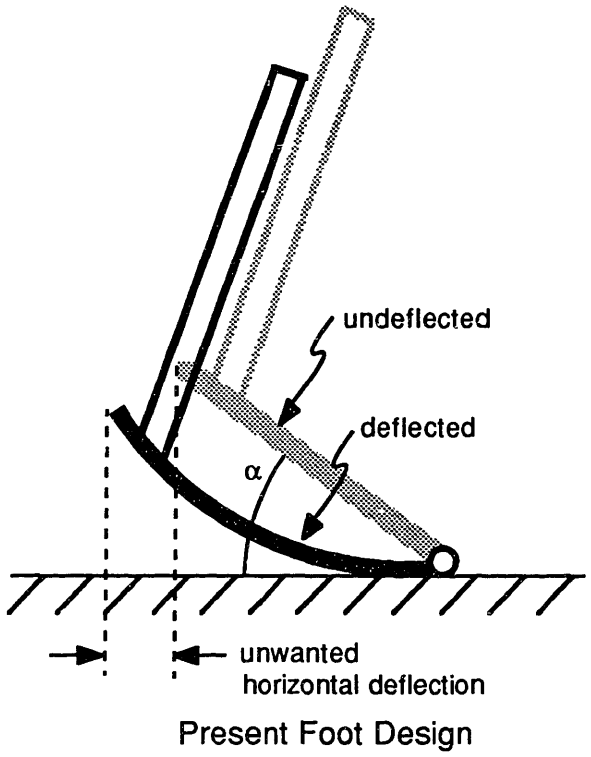


Figure 3-4: Deflection of leafspring foot introduces an undesirable horizontal motion of the toe with respect to the ankle.

shown in figure 3-5. In addition to the pure deflection of the foot leafspring, the back drive of the thrust actuator also contributed in heel impact against the ground. Increasing the actuator hydraulic pressure from 1500 psi to 3000 psi eliminated the back drive of the thrust actuator as well as the heel impact against the ground. However the increased pressure resulted in severe oscillations in thrust and body pitch. Therefore, the monopod performed poorly running in place or running with low forward velocity.

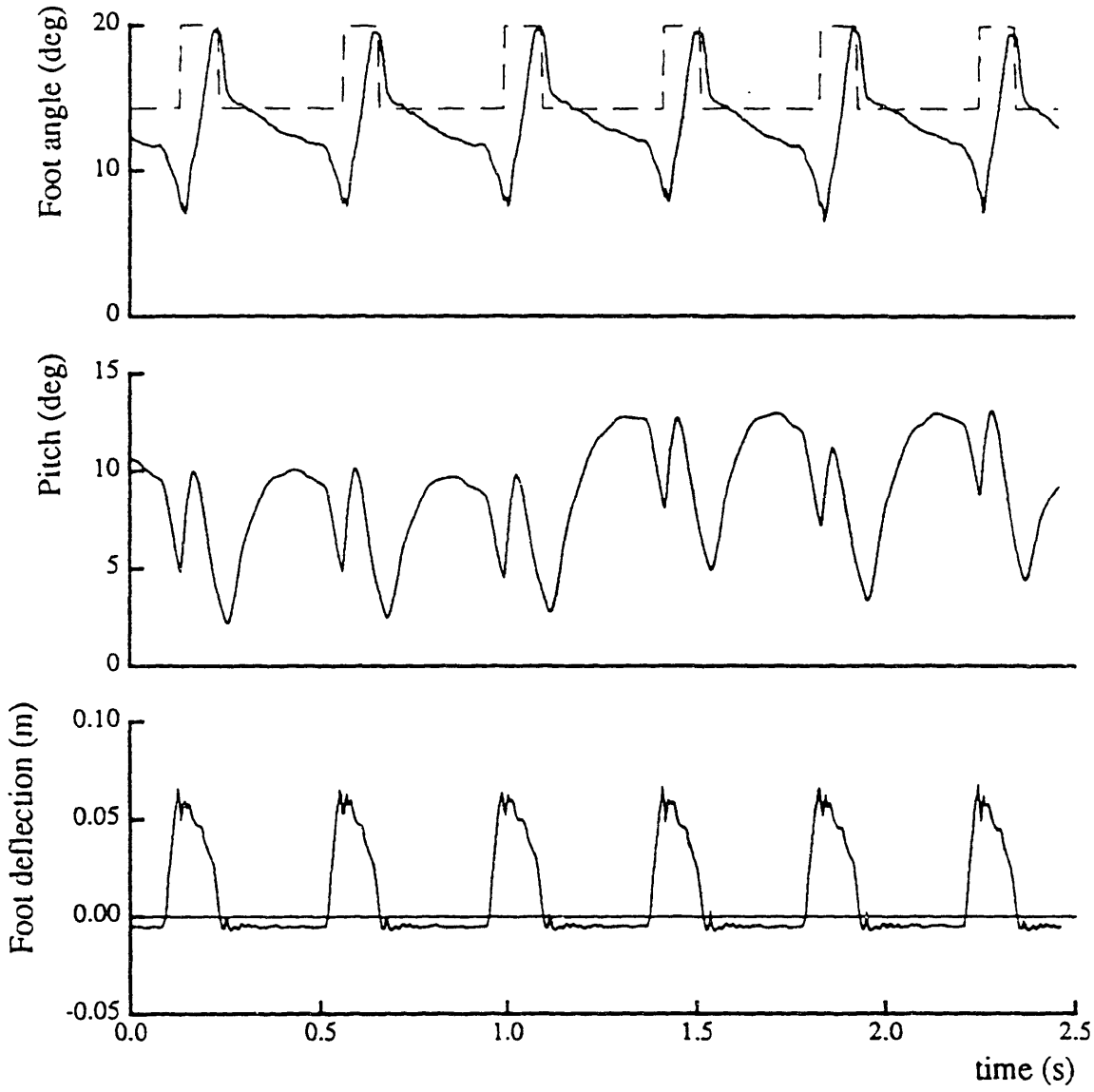


Figure 3-5: Running data for the monopod with the heel hitting the floor, 6 cycles. The disturbance in foot deflection just after the peak indicates the impact of the heel on the ground. The impact of the heel on the ground causes the positive spikes in pitch velocity which appear as substantial forward pitching moment in the data. Graph of foot angle and setpoint (broken line) shows that the ground force is driving the foot away from the set-point when foot deflection is large, due to inadequate actuator hydraulic pressure. Upward step in setpoint is where thrust begins.

Chapter 4
Alternative Design for Articulated Leg: Monopod with Hoof

As a possible solution to the problem of the strong coupling between the sweeping motion of the hip and the thrusting motion of the ankle, we propose an alternative foot design. (see figure 4-1.) This design places the toe on a hoof-like platform that elevates it with respect to the ground. One solution to the coupling problem is to minimize the foot angle α. The proposed design reduces the foot angle without causing the ankle to collide with the ground. Although the motion of the ankle about the toe is still along a circular arc, it is now symmetrical with respect to the horizontal line passing through the toe joint. Therefore it induces less horizontal motion of ankle.

This design was inspired by the mechanism of a horse's hoof (figure 4-2). Impact of the foot against the ground bends the fetlock joint and stretches an elastic ligament. The fetlock snaps back when the foot leaves the ground. Because a suitable artificial tendon-like material for such a design has not been found, the fiberglass leafspring is used in the new design as the energy storage element.

4.1 Construction of the Articulated Leg with a Hoof

The alternative foot design with a hoof-like structure was built. The toe of the original articulated leg was replaced by a hinge joint and the hoof (figure 4-3) was attached. The angle of the hoof at rest was adjusted by changing the tension of the rubber springs. We set the spring so that the hoof was horizontal at landing. The stiffness of the rubber springs were kept as low as possible since they would induce torque about the hoof's hinge as the foot is deflected during stance phase. The angle of the hoof does not have to be actively controlled because the angle of the foot with respect to the body is kept constant by the four-bar linkage formed by the leg, heel lever, body, and tendon.

The fiberglass leafspring foot design for the original monopod is also used for the monopod with a hoof. The fiberglass sheets are laminated together, but the delamination of

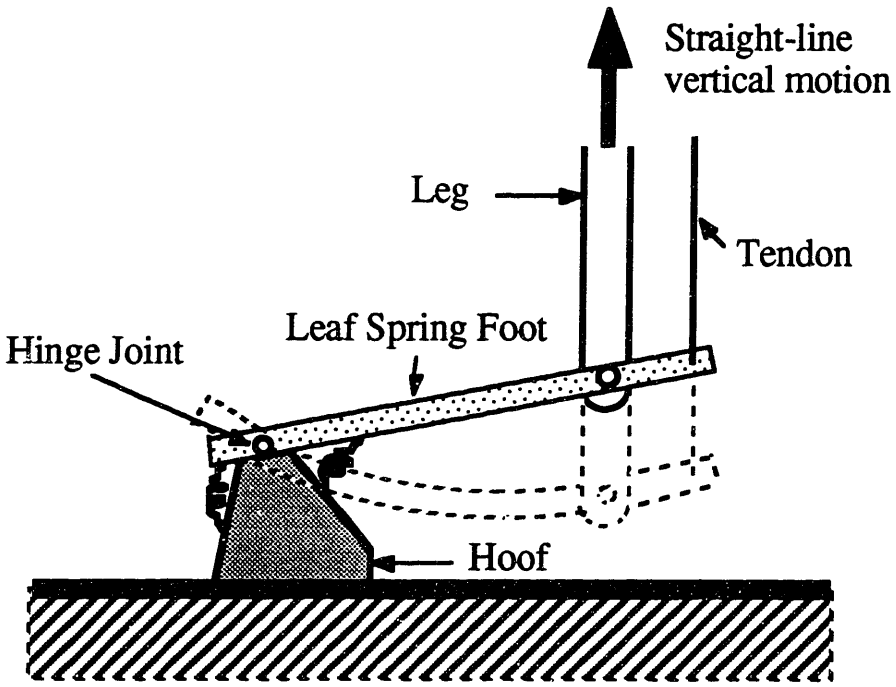


Figure 4-1: Drawing of alternative articulated leg design. This design places the toe on a hoof-like platform that elevates it with respect to the ground. It reduces the foot angle without causing the ankle to collide with the ground, and thus minimizes the coupling between the vertical motion and the horizontal motion.

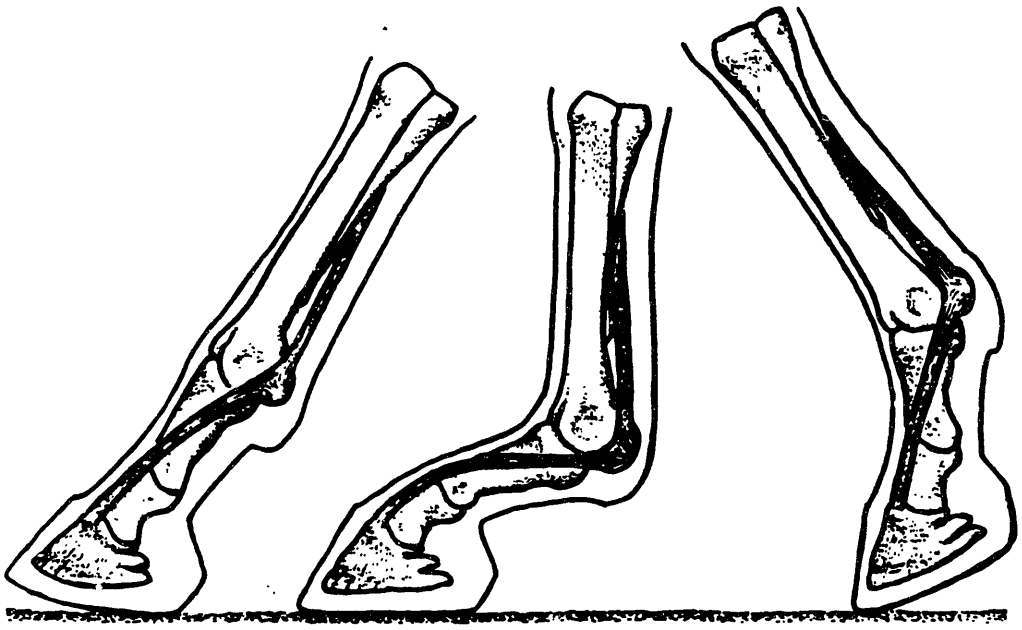


Figure 4-2: The basic idea of the alternative leg design was inspired by the mechanism of a horse's hoof. The hoof elevates the toe so the joint can move lower without touching the ground. Figure reprinted from Hildebrand (1960).

the fiberglass sheets was the most frequent source of the failure. Using a thick single piece of fiberglass sheet might work better. Also, avoidance of any spot of stress concentration is critical. The foot is designed to be triangular, with the most acute angle being the toe, to evenly distribute the stress along the leafspring and therefore make the spring as efficient as possible. However, the middle section of the triangular leafspring was the most vulnerable point judging from the crack propagations of the failed feet. The middle section was strengthened by altering the shape of the foot slightly so that it was wider than before. The fine surface finishing of the fiberglass leafspring was important in preventing the crack propagation.

The hoof must be light because the weight of the foot increases the effective unsprung mass of the system. The lightness of the hoof is also crucial in ensuring the correct angle of the hoof at touchdown because the tensions of the rubber springs which keep the hoof in the desired angle are adjusted as to be as low as possible. The hoof was first made of wood, but that design failed after several runs. The present hoof design is reinforced by making the hinge section from high strength plastic. The hoof has a layer of rubber padding on the bottom for cushioning.

The addition of the hoof to the original foot design required no additional sensors. However, the control system was modified to prevent the hoof from rolling.

4.2 Monopod with Hoof Experiment

Figure 4-4 shows data from a running experiment with the monopod and the hoof. The monopod ran at an average forward speed of 0.08 m/sec. The monopod ran comfortably up to an average forward speed of 1 m/sec, and it also ran in place and backward. Even with the hoof the monopod looked awkward running backwards. However, without the hoof this task was nearly impossible because the heel made contact with the ground. Figure 4-4 shows the single peak in foot deflection during each bounce, the absence of a substantial peak in pitch angle, and the absence of an abrupt change in the vertical position of the center of gravity. All of these signs indicate that there were no heel impacts with the ground. The body's pitch angle error was less than 7 deg, typically with a nose-up posture.

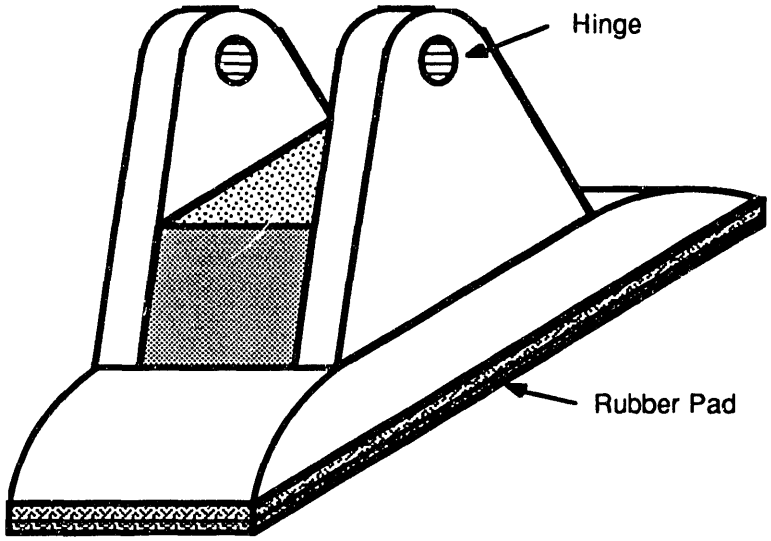


Figure 4-3: Diagram of the hoof. The low mass of the hoof is an important factor in the design of the hoof. A light weight hoof is necessary to minimize the effective unsprung mass of the system and to ensure the correct hoof angle at landing. Most of the hoof is made of wood for lightness. The hinge section is made of high strength plastic due to the high stress concentration. The rubber padding on the bottom provides cushioning.

Implementation of the hoof-like structure eliminated the problem of heel impact against the ground, and therefore also eliminated the cause of substantial disturbances in pitch. The hoof allowed the monopod to run in place. However, the implementation of the hoof-like structure introduced a new problem in the control of the monopod: the rolling of the hoof. The rest of this thesis concentrates on the analysis and control of the monopod with a hoof.

4.3 Monopod with Hoof Simulation

To develope and analyze the control system for the monopod with a hoof, a computer simulation was developed. The monopod is modelled, and the equations of motion are derived. The simulation code was written in C using the variable step size fourth order Runge-Kutta integration method.

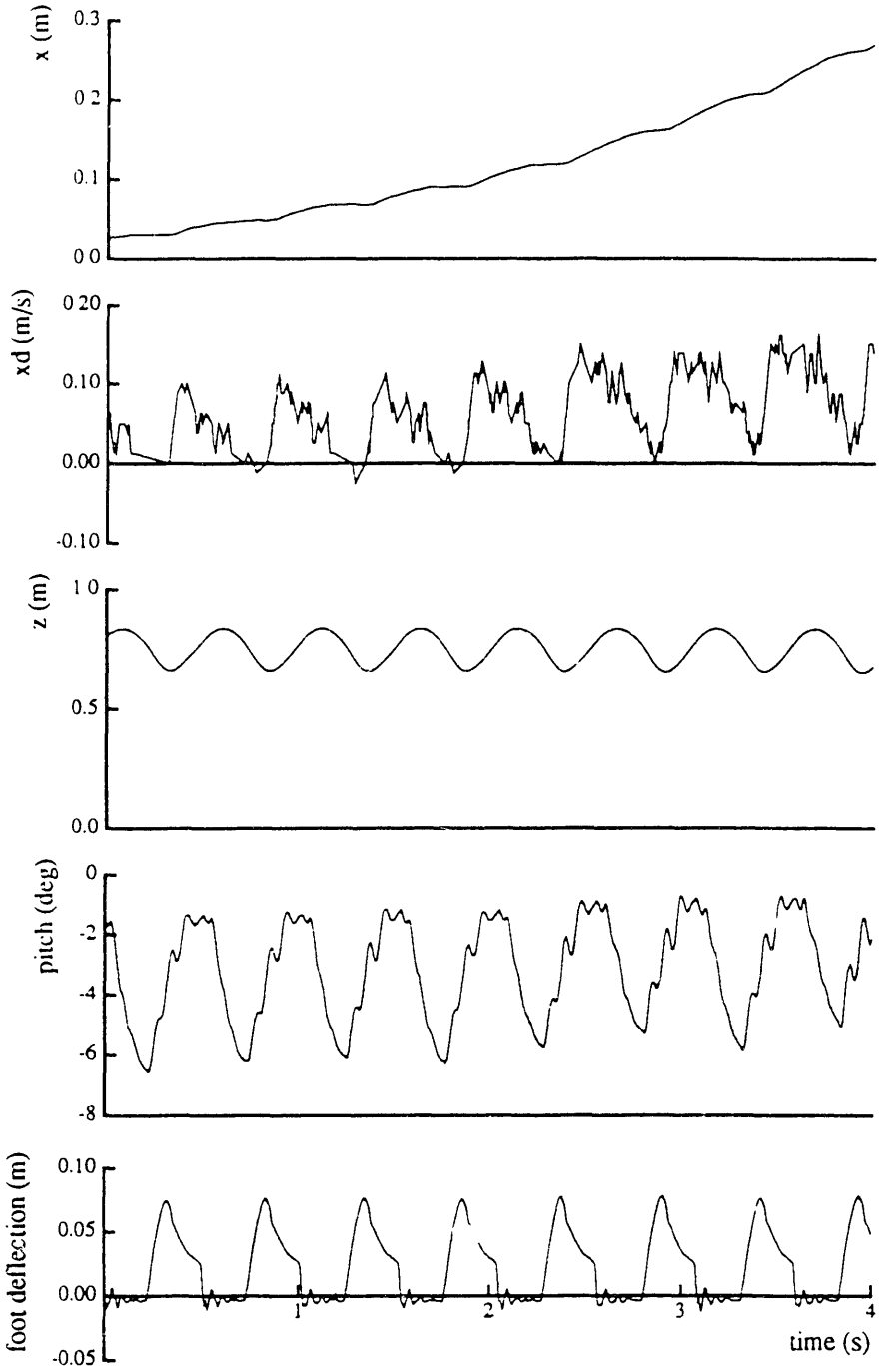


Figure 4-4: Running slowly, at approximately 0.08 m/sec. The single peaks in foot deflection indicate the absence of heel impacts with the ground. The absence of substantial single peaks in pitch angle and the abrupt change in vertical position of center of gravity also indicate no heel impacts with ground. The decrease of forward velocity during the flight phase is due to friction in the planarizer. (Data file M.235.10.)

4.3.1 Model

The model for the monopod with the hoof is shown in figure 4-5. It consists of four main parts; the body, the leg, the foot and the hoof. They are modelled as rigid bodies with uniformly distributed mass, and they are coupled by frictionless joints.

The inelastic tendon and the leafspring foot of the hoofed monopod is modelled as a torsional linear spring of stiffness K_{ankle} and a torsional damper of damping coefficient B_{ankle} in parallel. The spring and the damper are attached at the ankle joint between the leg and the foot. The thrust actuator of the monopod is modelled as a position source which varies the setpoint of the torsional spring of the ankle. This is a reasonable assumption since the hydraulic valve for the thrust actuator of the monopod is a flow control valve and the thrust actuator is in series with the spring. Note that a position source rigidly coupled with a mass without a spring inbetween is impossible since that model would require infinite acceleration. For the same reason, the hip actuator of the monopod is modelled as a torque source with a linear servo applying equal and opposite torque to both the body and the leg. A torsional linear spring of stiffness K_{toe} and a torsional damper of damping coefficient B_{toe} are attached in parallel to the foot and the hoof. The ground was modelled as a spring-damper system in both the horizontal and vertical directions. This spring-damper system has the same effect on the model as the rubber pad at the bottom of the hoof. Only the bottom two corners of the hoof are modelled as the contact points with the ground. All the physical parameters and kinematics of the model are taken directly from the monopod.

With this model, the equations of motion of the monopod are derived using Lagrangian dynamics. The generalized coordinates are horizontal and vertical positions of the center of the body X_{body}, Y_{body}, the angle of the body θ_{body}, the angle of the leg θ_{leg}, the angle of the foot θ_{foot} and the angle of the hoof θ_{hoof}. The equations of motion for the model of the monopod are included in Appendix D.

4.3.2 Dynamic Computer Simulation

The framework of the animated computer simulation code for the monopod is based on the simulation of a two dimensional, one-legged hopper written by Prof. Chris Atkeson. The routine which is specific to the monopod consists of the determination of the state and output variables, the control of the system, an integration scheme for the equations of

motion using the variable-step Runge-Kutta method. The simulation animates the behavior of the monopod using the calculations of the state variables. The variable-step size Runge-Kutta integration method procedures and the matrix inversion procedure are taken from the *Numerical Recipes in C: The Art of Scientific Computing* by William H. Press, et al. (1988). The simulation code is given in Appendix F.

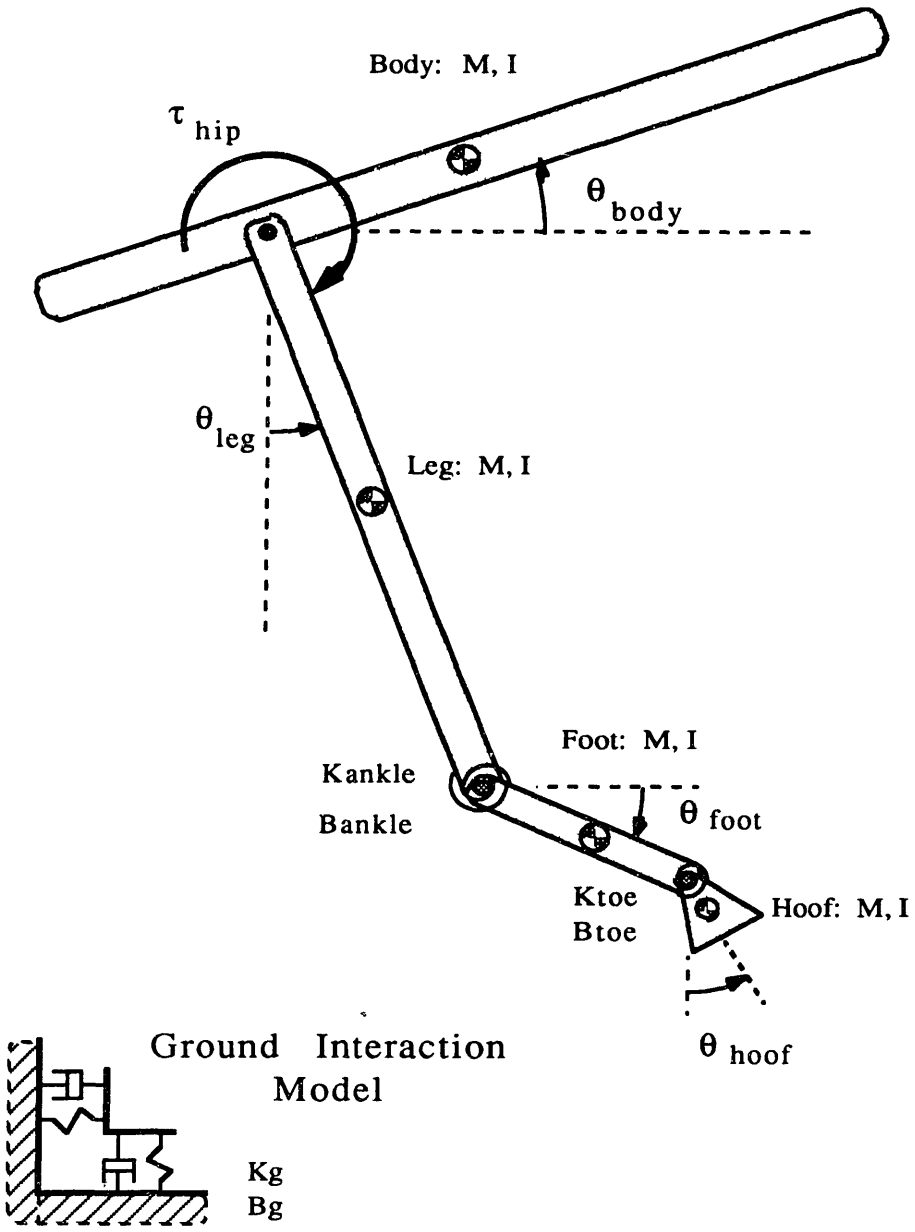


Figure 4-5: The model for the monopod with the hoof. The model consists of four main parts; the body, the leg, the foot and the hoof, and they are modelled as rigid bodies with uniformly distributed mass, coupled to each other by frictionless hinge joints. The inelastic tendon and the leafspring foot is modelled as a torsional spring and damper in parallel attached to the ankle, between the leg and the rigid foot. The thrust is modelled as a change in setpoint of the ankle spring (position source), and the hip actuator is modelled as the torque source. The toe spring is modelled as a torsional spring and damper in parallel attached at the toe between the foot and the hoof. The ground and the rubber pad of the hoof are modelled as a spring-damper system in both the horizontal and vertical direction. Only bottom two corners of the hoof are modelled as the contact points with the ground.

Chapter 5
Rolling of Hoof

Rolling of the hoof became a major problem in control of the monopod. The rolling of the hoof motivated us to examine several aspects of the running machine including its mechanical design and dynamics. In this chapter, we will first discuss when and why the rolling of the hoof occurs, and then we examine possible solution to the problem.

5.1 Rolling of Hoof: Force Analysis

In the original foot design, the toe is the single point of contact with the ground, and the horizontal component of the force applied at the toe is much smaller than the friction force between the toe and the ground in normal running. However, with the addition of the hoof, although the friction force between the hoof and the ground is still much larger than the horizontal force, in many cases the horizontal component of the force applied at the hoof is large enough to induce torque causing rolling of the hoof.

Conceptually, the problem with the rolling of the hoof is similar to the problem of slipping of the toe. The downwards force f_z and horizontal force f_x applied at the hinge of the hoof induce a resultant force f_r (figure 5-1). If f_r passes above the bottom corner of the hoof, it will induce a torque and cause the hoof to roll. In order to avoid the rolling of the hoof the control system must satisfy the condition

$$f_x < f_z \tan\beta \tag{5.1}$$

where

f_x	is the horizontal force applied at the hinge of the hoof during stance,
f_z	is the horizontal force applied at the hinge of the hoof during stance, and
β	is bisected angle of the hoof at the hinged corner.

The horizontal force f_x satisfying the above condition is much smaller than the friction force between the toe of the original foot and the ground. The hoof can roll much more easily than the original toe could slip. The torque applied at the hip to correct the body attitude is one of the main sources of the horizontal force f_x.

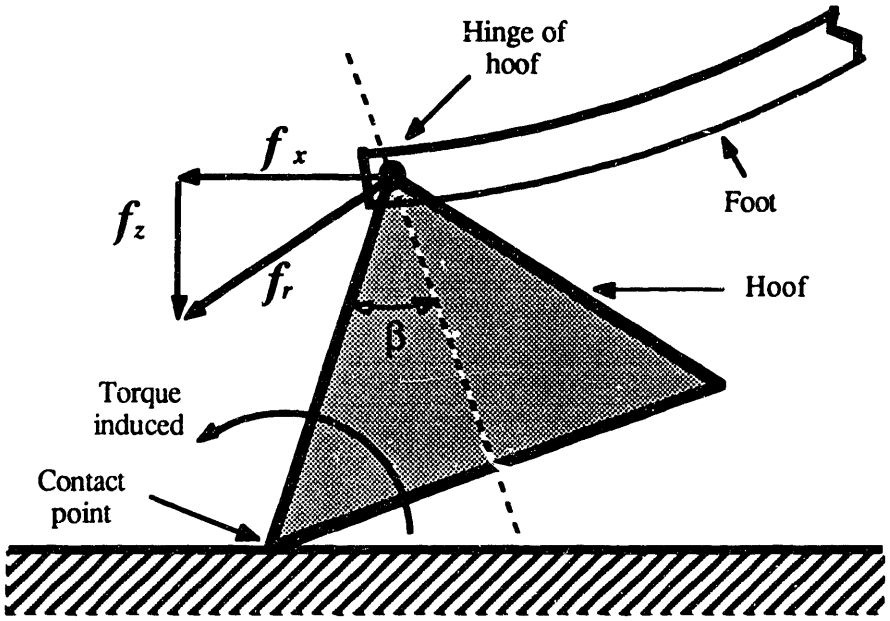


Figure 5-1: Free body diagram of the forces applied at the hinge of the hoof. The horizontal force f_x and the downwards force f_z induce the resultant force f_r. The resultant force vector f_r has to pass below the bottom corner of the hoof. Otherwise, the torque induced about the bottom corner of the hoof will result in rolling of the hoof.

the hip torque allowed for the attitude control for the monopod with the hoof is limited compared to the original design. The possibility of coordinating the hip torque used for attitude control with the vertical loading of the hoof to avoid rolling of the hoof is investigated below. As the forward velocity of the monopod increases the problem of rolling of the hoof becomes serious, because the inertial loading applied at the hinge of the hoof during stance increases. The effect of the inertial loading on hoof's rolling is also investigated below.

There are two causes for the rolling of the hoof: the mechanical design of the hoof and the control system for the monopod.

5.2 Rolling of the Hoof: Mechanical Design

One obvious factor in the mechanical design is the length of the hoof. Elongating the hoof will increase the angle β and the allowable ratio of horizontal and vertical forces. The effect of varying the length of the hoof on the performance of the monopod is clear; the longer hoof will be more immune to the rolling of the hoof, but it will increase the weight of the hoof. In addition, there probably exists a limit on the maximum length of the hoof since a longer hoof is more likely to strike the ground during flight phase. A cost function could be derived and the optimum parameters for the design of the hoof could be calculated, but this approach is not pursued further here.

Chattering

A less obvious factor in the mechanical design of the hoof is the stiffness of the rubber padding at the bottom of the hoof. In the original foot design, the toe is the single point of contact with the ground. For the monopod with a hoof, the forward and leading edges of the hoof make contact with the ground. Each of these edges of the hoof experiences a foot spring force roughly half of that which the toe of the original design experienced. The ratio of the pad spring force and the foot spring force thus increases roughly by factor of two. According to Alexander (1988), this increased ratio makes chattering more likely to occur, and in fact chattering does occur in the design with a hoof. The chattering was easily detected by the rattling noise made every time the monopod landed on the ground.

Alexander makes three assumptions:

i) The mass of the foot, which is comparable to the hoof in this case, is small compared to the mass of the body.

ii) The stiffness of the paw pad (hoof pad in this case) is large compared to the stiffness of the leg spring (foot spring).

iii) The peak force of the leg spring is large compared to the body weight.

All assumptions are valid for the monopod. The chattering causes the hoof to lose contact with the ground. Chattering is undesirable not only because the foot is likely to shift position but because it also reduces the vertical force momentarily and increases the possibility that the hoof will roll. Initially, we tried softening the stiffness of the rubber padding of the hoof to preventing the chattering, but such softening made the rolling of the hoof easier since it reduces the effective bisected angle of the hoof at the hinged corner.

According to Alexander, the chattering can be avoided by either bring the foot (hoof in this case) down slowly to reduce the pad spring force or by having a paw pad (hoof pad) with nonlinear spring characteristics. Initially the effective stiffness will be low and the pad spring force will be small early in the step. The high stiffness later in the stance phase will keep the toe or hoof from bottoming against the ground. We applied this idea and made the rubber pad at the bottom of the hoof of two layers of different stiffness: the inner layer is very soft and the outermost layer is relatively stiff. This design successfully eliminated the chattering.

5.3 Rolling of the Hoof: Control System Design

The main sources of the horizontal forces applied at the hinge of the hoof are the hip torque used for body attitude correction and the inertial loading due to the changes in momentum of the body. Therefore, the method of coordinating the hip torque with the downwards force at the hoof and the method of limiting the inertial loading at the hoof are proposed. First the hip torque coordination technique is tested, and then the inertial loading limitation technique is tested, based on both the computer simulation and the actual experiment. Due to its lengthy discussion, the inertial loading limitation technique is presented in next chapter.

5.3.1 Hip Torque Coordination

Initially, we studied coordinating the hip torque used for attitude control with the vertical loading of the hoof. We tested the method of coordinating the hip torque on the computer simulation of the monopod and then on the actual machine. The simulation was somewhat successful. However, this method was not successful on the actual monopod for following reasons.

The differential pressure in the hip actuator piston is measured using pressure transducers, and the resultant sweeping force applied at the hoof is derived. The deflection of the foot spring is assumed to induce force mostly in the downwards direction. This downwards force is then converted to the vertical force applied at the hoof. Total downwards force and horizontal force components are then computed. Refer to the free-body diagram in figure 5-2. The force induced at the ankle by the hip torque is

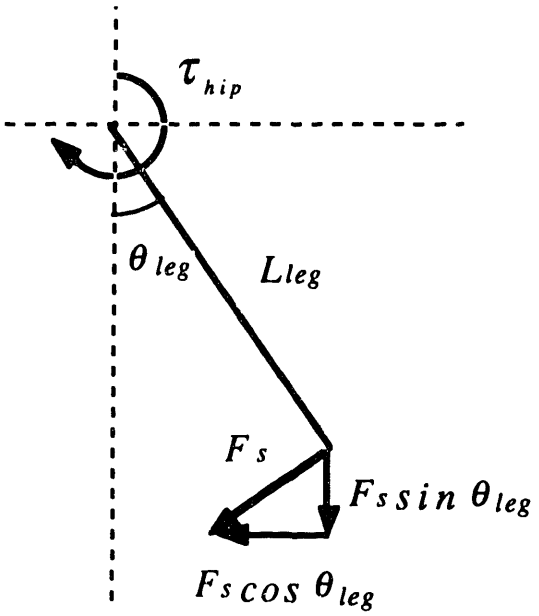


Figure 5-2: Free body diagram of the forces induced at the ankle by the hip torque, τ_{hip}, during the stance phase. F_S is the induced sweeping force at the ankle.

$$F_S = \frac{\tau_{hip}}{L_{leg}}. \tag{5.2}$$

where

F_S is the sweeping force induced at the ankle by the hip torque,
τ_{hip} is the torque applied at the hip, and
L_{leg} is the length of the leg.

The small ratio of the foot length and the leg length makes the force seen at the hoof close to F_{ankle}. Therefore assuming tha the nominal foot angle is horizontal, the the vertical and horizontal components of the resultant force are approximately

$$F_h \cong F_S \cos\theta_{leg}, F_d \cong F_S \sin\theta_{leg} + K_f \delta_f \tag{5.3}$$

where

F_h is the horizontal component of the force applied at the hoof's hinge,
F_d is the downwards component of the force applied at the hoof's hinge,
K_f is the average stiffness of the foot spring, and

θ_f is the angle of the leg with respect to horizontal.

The control system continuously monitors the values of the F_h and F_d throughout the stance phase, and whenever the F_h becomes greater than F_d, the signal sent to the hydraulic valve for the hip actuator is modified to reduce F_h. This adjustment to the hip torque will disturb the body attitude.

The figure 5-3 shows the angle of the hoof during a stance phase of the simulation. Figure 5-3-a) is the angle of the hoof with the hip torque coordination implemented, and figure 5-3-b) is the angle of the hoof without the hip torque coordination scheme. This hip torque coordination scheme successfully prevented the complete rolling of the hoof. However, because of the natural tendency of the monopod to nose down when it is running, the hip torque applied for the body attitude correction results mostly in pushing the leg backwards during the stance phase. As the hip torque increases the horizontal force about the hoof, F_h, and the hip torque coordination control scheme tries to reduce the F_h by interrupting the attitude correction. This action leads to the body attitude disturbance, as shown in the state 1 of the figure 5-4. The disturbance in body attitude correction causes the upward velocity of the center of mass of the body to be much smaller than the upward velocity of the hip joint, and the bounce of the system as a wholeis lower (state 2 of figure 5-4). The shorter flight duration allows less time for the monopod to swing its leg forward for the next touchdown, and if the leg fails to reach the desired position the forward velocity will increase on the next stride. The forward swinging of the leg causes the monopod to nose down even more during the flight phase as shown in the state 3 of figure 5-4. The following stride results in a worse situation than the previous stride, and the monopod eventually falls down. Although this phenomenon was evident in the animated computer simulation of the monopod, the hip torque coordination control scheme performed well enough to maintain the monopod's hopping motion without falling down. The average body attitude of the monopod during the simulated run was still an excessive nose-down posture which would be unacceptable for the actual machine. The range of leg angle with respect to the body attitude is limited for the acutal machine whereas it is not so for the simulated model.

As expected, the phenomenon explained above dominated the performance of the monopod in the experiment with the physical machine, and the hip torque coordination scheme was not successful. We first considered searching for a more sophisticated control strategy, but soon found that the hip torque coordination techinique was not promising, at

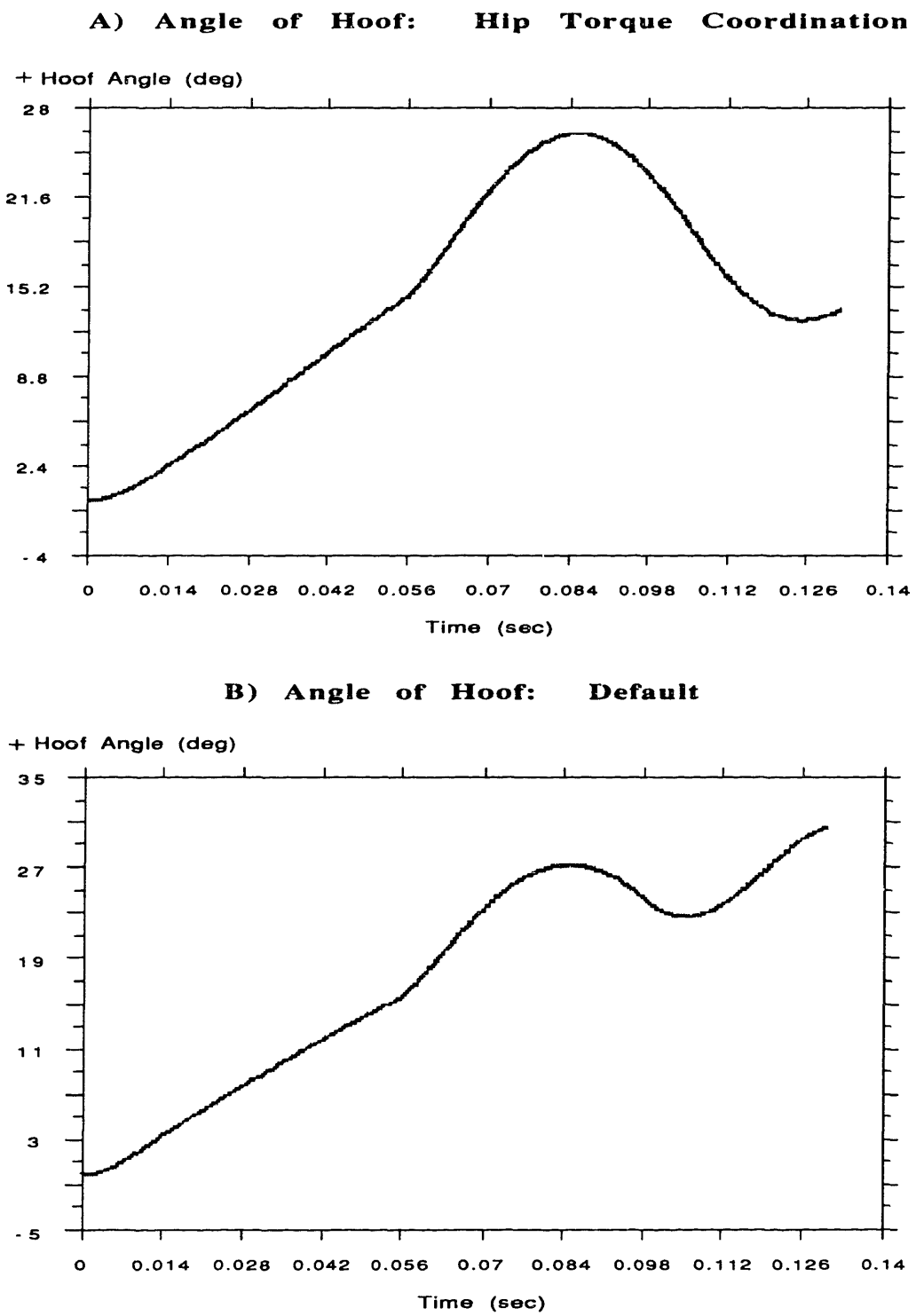


Figure 5-3: Graphs of the angle of the hoof during the stance phase from the simulation of the monopod . The hip torque coordination scheme successfully reduced the angle of the hoof, thereby preventing the rolling of the hoof, whereas the original default mode resulted in the rolling of the hoof. However, the hip torque coordination scheme severely affected the body attitude control.

least with the presentl hardware used on the monopod, because sensing the hip torque was extremely problematic. The sweeping force induced from the hip torque is obtained from the pressure transducers on the hip acutator. The pressure transducers read the pressures on both sides of the hip actuator piston, and the differential pressure is then used to derive the sweeping force. However, as shown in figure 5-5, the pressures read from the hip actuator are oscillatory, and it is difficult to obtain the accurate value of the sweeping force. Idealy we could obtain the horizontal and vertical forces at the hoof with a force plate, but then again a force plate is not accesible in normal running situation. Therefore, we decided to search for alternative approaches.

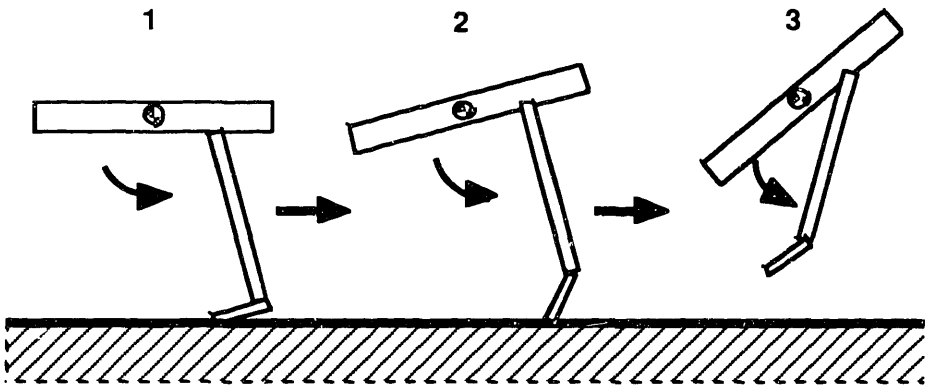


Figure 5-4: The fall of the monopod induced by the pitch disturbance. Because of the natural tendency of the monopod to nose-down when running, the hip torque applied for body attitude correction results mostly in pushing the leg backward during the stance phase. This action increases the horizontal force about the hoof. Then the hip torque coordination control scheme tries to reduce the horizontal force by disturbing the body attitude (state 1). The body is further nosed down at liftoff (states 2). The body attitude error grows quickly to failure (state 3).

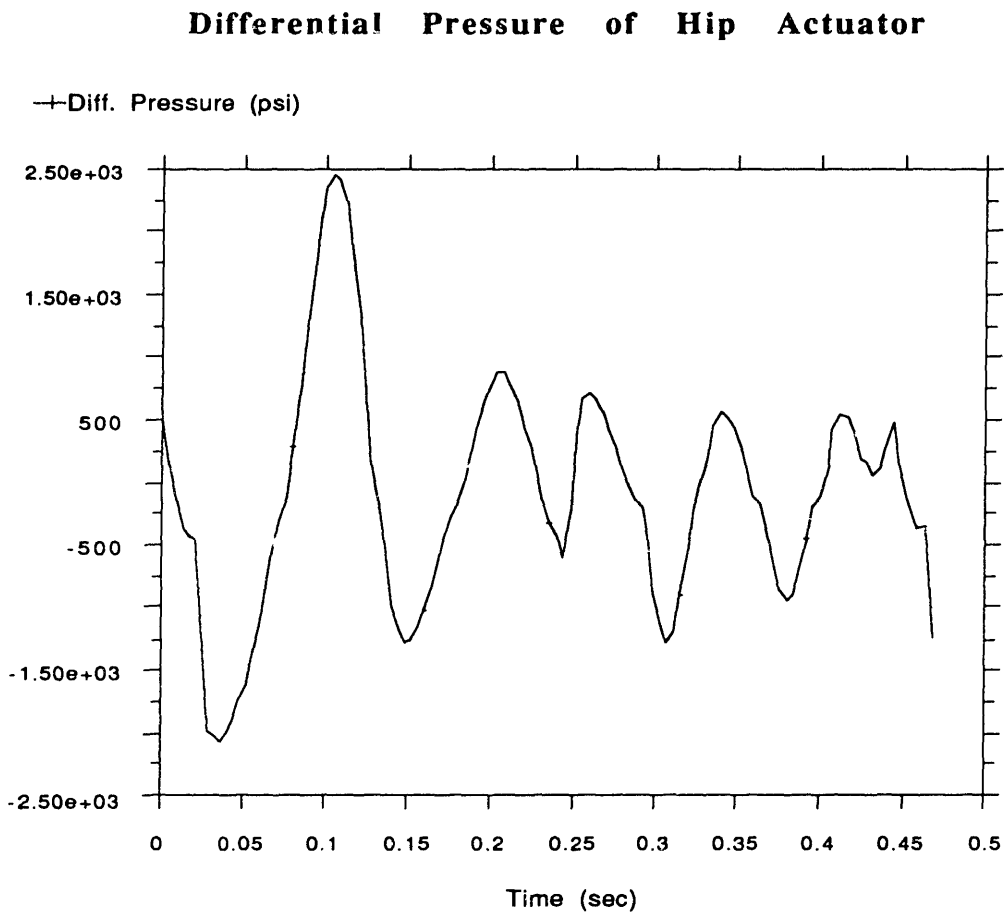


Figure 5-5: Reading of the differential pressure in the piston chambers of the hip actuator duing a single stance phase. The pressure readings are oscillatory.

Chapter 6
Limiting Inertial Loading on Hoof

The addition of a hoof to the monopod improved the forward velocity control of the monopod, but it introduced a new problem; rolling of the hoof. From an analysis, the rolling of the hoof was found to be strongly influenced the hip torque applied for body attitude control. We studied the method of coordinating the hip torque with downwards force at the hoof, but the unsatisfactory result from the hip torque coordination method motivated us to investigate the dynamical nature of the force applied at the hoof. Intuitively it was clear that strong relationships existed among the direction of the resultant force applied at the hinge of the hoof, the forward velocity at touchdown, leg angle at touchdown and leg angle during the stance phase. However, it was not obvious what the exact relationships between these variables were. How much of an influence does the leg angle at touchdown have on the direction of the force induced at the hoof? What is the precise relationship between the angle of the leg and the direction of force at the hoof? We attempted to answer these questions in the hope that the answers would lead us to better control of the monopod.

6.1 Simple Model

The first step was to study the dynamical nature of the monopod using an extremely simple model, shown in figure 6-1. The simplified model consists of a large mass, M, representing the body, and a small mass, m, representing the hoof. The two masses are connected by a massless leg spring with stiffness K, and the small mass m is connected to the ground by horizontal and vertical springs of stiffness k_g. These springs represent the compliance in the ground and the rubber pad on the bottom of the hoof. The masses of the body and the hoof are modelled as point masses with no inertia, and the leg springs and the hoof pad springs are modeled as linear springs. This model represents the stance phase of an one legged running machine with no control action; the hip torque and the thrust do not enter the picture. Parameters from the physical machine are used for the parameters of the

model. The Lagrangian method was used to derive the equations of motion for the simplified model (Appendix E for detailed derivations):

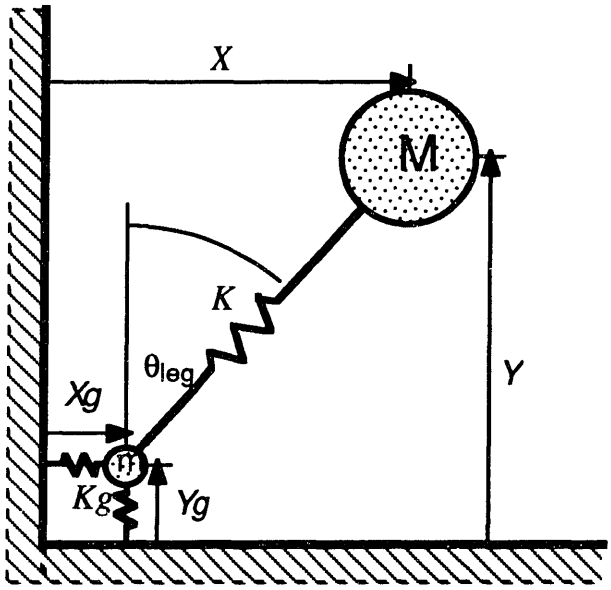


Figure 6-1: The diagram of the simplified model of the monopod. The simple model consists of a large point mass, M, for the body and a small point mass, m, for the hoof. The two masses are connected by a massless spring with stiffness K. The ground and the rubber padding on the bottom of the hoof are modelled as horizontal and vertical linear springs of stiffness k_g.

$$
\begin{aligned}
\frac{d^2x_g}{dt^2} &= -\frac{k_g}{m}(x_g - x_{go}) + \frac{K}{m}(x - x_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2+(y-y_g)^2)}}\right) \\
\frac{d^2y_g}{dt^2} &= -\frac{k_g}{m}(y_g - y_{go}) + \frac{K}{m}(y - y_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2+(y-y_g)^2)}}\right) - G \\
\frac{d^2x}{dt^2} &= -\frac{K}{M}(x - x_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2+(y-y_g)^2)}}\right) \\
\frac{d^2y}{dt^2} &= -\frac{K}{M}(y - y_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2+(y-y_g)^2)}}\right) - G
\end{aligned}
\qquad (6.1)
$$

where

x_g is the horizontal position of the hoof with respect to the touchdown point on the ground,

y_g is the vertical position of the hoof with respect to the ground,

x is the horizontal position of the body with respect to the touchdown point on the ground,

y is the vertical position of the body with respect to the ground,

M is the mass of the body,

m is the mass of the hoof,

K is the stiffness of the leg spring,

k_g is the stiffness of the hoof pad spring and the ground,

r_o is the length of the leg when the leg spring is at rests, and

G is the gravitational acceleration.

The horizontal and vertical components of the resultant force applied at the hinge of the hoof are equal to the forces induced at the horizontal and vertical hoof pad springs. The horizontal and vertical forces at the hoof are

$$F_x = K_g (x_g - x_{go}) = -m \frac{d^2 x_g}{dt^2} - M \frac{d^2 x}{dt^2}$$
$$F_y = K_g (y_g - y_{go}) = -m \frac{d^2 y_g}{dt^2} - M \frac{d^2 y}{dt^2} - (m + M) G \qquad (6.2)$$

where

F_x is the horizontal component of the force applied at the hoof, and

F_y is the vertical component of the force applied at the hoof.

Since the mass of the hoof (m) is much smaller than the mass of the body (M) (less than 1%) and since the simulation results show that the peak accelerations of the foot are in a similar range to the peak accelerations of the body, the mass of the hoof can be assumed to be negligible. After setting m equal to zero and also after some more algebraic manipulations (refer to Appendix E for detail), the ratio of F_x and F_y becomes

$$\frac{F_x}{F_y} \approx \tan \theta_{leg} \qquad (6.3)$$

where

θ_{leg} is the angle of the leg with respect to the vertical.

This equation implies that the angle of the leg is approximately the same as the angle of the resultant force vector applied at the hoof with respect to the vertical.

$$\phi_{force} \approx \theta_{leg} \tag{6.4}$$

where

ϕ_{force} is the angle of the resultant force applied at the hoof with respect to vertical.

Therefore the rolling of the hoof can be prevented by limiting the maximum leg angle during the stance phase.

6.1.1 Simulation of the Simplified Model

This hypothesis was tested by computer simulations of the simplified model. Two initial conditions are used as the control variables: the angle of the body's velocity vector with respect to vertical at touchdown, and the angle of the leg with respect to vertical at touchdown. The angle of the leg at touchdown varied by 10 degree increments from -20 degrees to 20 degrees. For each leg angle at touchdown, the angle of the body's velocity vector was varied from -30 degrees to 30 degrees with increments of 10 degrees. In all cases the speed of the body at touchdown is 1 m/sec. The outputs from the simulation are the angle of the leg, the angle of the force vector at the hoof, and the horizontal force and the vertical force at the hoof. For the complete graphical results of the simulations refer to Appendix G. Some of the results are shown in figure 6-2. Several points are selected from the simulation results to show how closely the angle of the resultant force vector applied at the hoof, ϕ_f follows the angle of the leg, θ_{leg}. The average error is approximately 2 %, which is very good, except during the first 0.01 second after landing where the angle of the force at the hoof, ϕ_f changes from zero to the angle of the leg. This phenomenon is due to the high frequency oscillations of the hoof's pad springs. The oscillations also influence the horizontal force F_x and the vertical force F_z at the hoof, as shown in figure 6-2. Apparently, the oscillation of the pad springs are dominant until the leg spring "settles in." This phenomenon was predicted by the earlier discussion concerning chattering. and it does not seem to have much effect on the result since the angle of the force at the hoof, ϕ_f during the initial period is smaller than the angle of the leg. Therefore, this result from the simulation verifies the result of equation (6.4), which states that the angle of the force at the hoof is approximately equal to the angle of the leg during the stance phase.

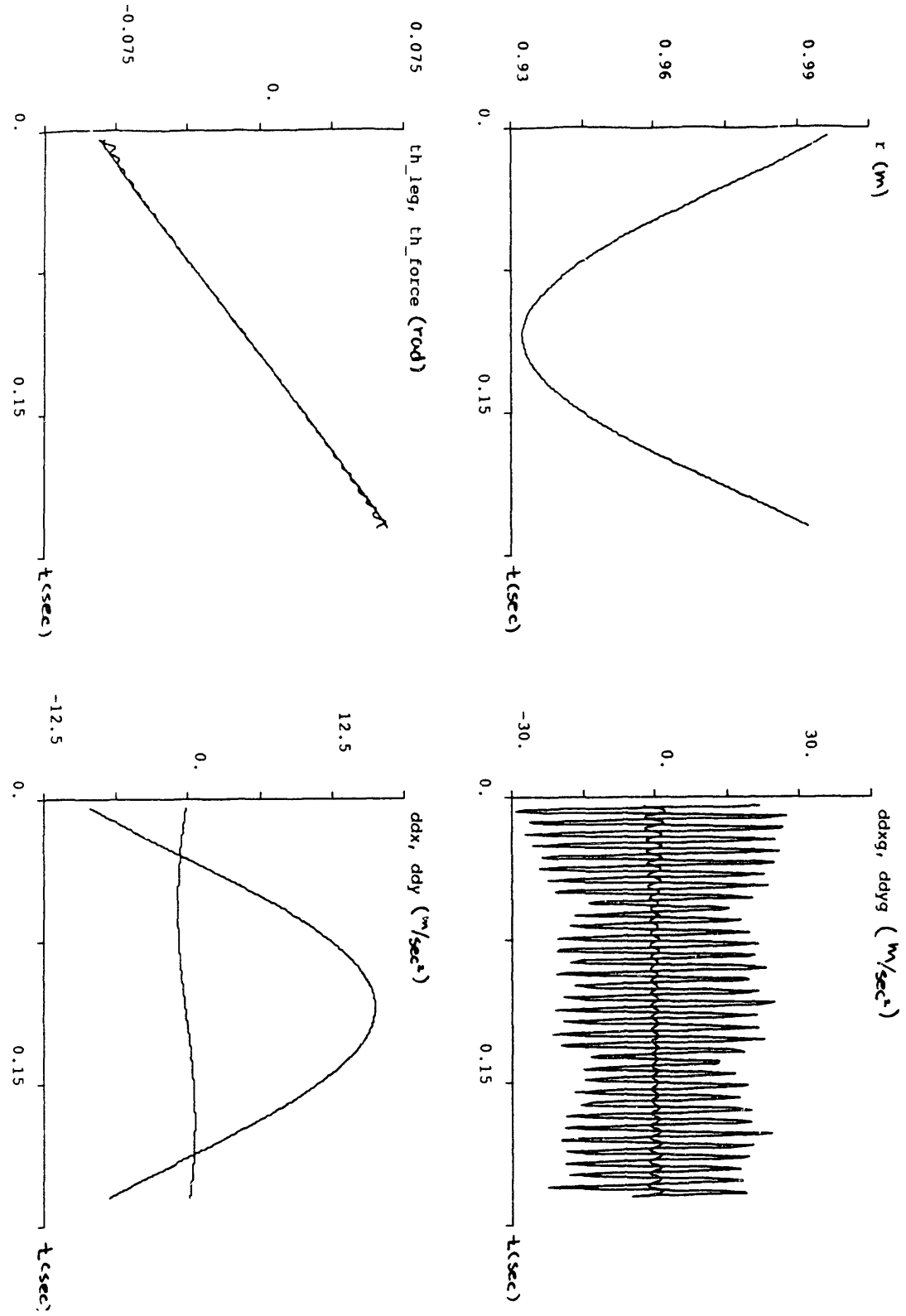


Figure 6-2: Outputs from the simulation of the simple model. The results show that the angle of the force at the hoof is approximately equal to the angle of the leg during the stance phase.

6.2 Rolling of the Hoof Revisited

The analysis of the simplified model of the one-legged running machine shows that, in the absence of the hip torque, the angle of the force applied at the hinge of the hoof, ϕ_f, is a function of the angle of the leg. To limit the maximum angle of the force at the hoof, the maximum angle of the leg during the stance phase must be limited. We propose three possible approaches: the model-based approach, the tabulation approach and the maximum acceleration limitation approach.

Model Based Approach

Linearize the simplified model and obtain an analytical solution for the leg angle during the stance phase. The maximum leg angle during the stance phase can be calculated while the monopod is still in flight phase, and the angle of the leg can be adjusted accordingly. This approach can be easily implemented, however it has a serious short coming; the model is too simple to produce an accurate prediction of the leg angle of the monopod. The monopod is more complex than the simple model: the hip joint is not located at the center of mass, and the leg is articulated and has rotary rather than telescoping joints. For this reason, this approach is not pursued any further.

Tabulation Approach

Another way of predicting the maximum angle of the leg during the stance phase is to tabulate the maximum leg angle for each case of initial conditions; the velocity of the center of mass and the angle of the leg at the touchdown. Once the table is complete, the control system can look up the range of leg angles at touchdown which would result in a leg angle which remained within the desired range during the stance phase. The range of feasible leg angles at touchdown can then be used as the maximum lower bound or the least upper bound of the leg angle at touchdown. In this scheme the foot would be placed at the position specified by the forward velocity control if that position was within the bounds shown by the table. If not, the boundary itself would be used as the desired leg angle at touchdown.

However, the tabulation approach is also diffucult to implement for a couple of reasons. The first is the number of state variables we would want to use to index the table:

the leg angle and the direction and magnitude of the velocity vector of the center of the mass at touchdown. With three dimensions a table which was fine grained enough to use would be huge and would take a long time to fill. The second reason is that our control over setting up the inital conditions for a table entry which we would like to fill is limited. The distribution of data points in the table would be uneven, and so would take even longer to fill.

Maximum Acceleration Limitation Approach

For this approach, we took a slightly different view from the other two approaches for directly limiting the maximum leg angle. Recalling the equations (6-2), the approximate ratio of the horizontal force F_x and the vertical force F_y can be written as

$$\frac{F_x}{F_y} \approx \frac{\frac{d^2x}{dt^2}}{\frac{d^2y}{dt^2} + G} \tag{6.5}$$

where

- x is the horizontal position of the body,
- y is the vertical position of the body, and
- G is gravitational acceleration.

The gravitational acceleration is a constant and thus is not a controllable variable. Therefore, the ratio of the horizontal force F_x and the vertical force F_y is a function of the horizontal and the vertical accelerations of the center of the mass of the monopod. Experiments showed that the vertical accelerations are approximately constant during stance regardless of the forward speed (figure 6-3); the average peak vertical acceleration when the monopod is running in place is 30 m/sec^2 whereas the average peak vertical acceleration when the monopod is running with a forward velocity of 0.9m/sec is 35 m/sec^2. This shows that the ratio of the horizontal force F_x and the vertical force F_y depends mostly on the horizontal acceleration of the center of mass of the system. Therefore the angle of the resultant force vector at the hoof ϕ_{force} due to the inertial loading of the monopod during the stance phase can be reduced by reducing the horizontal acceleration during the stance phase. We implemented this approach and the following sections discuss the results of this implementation.

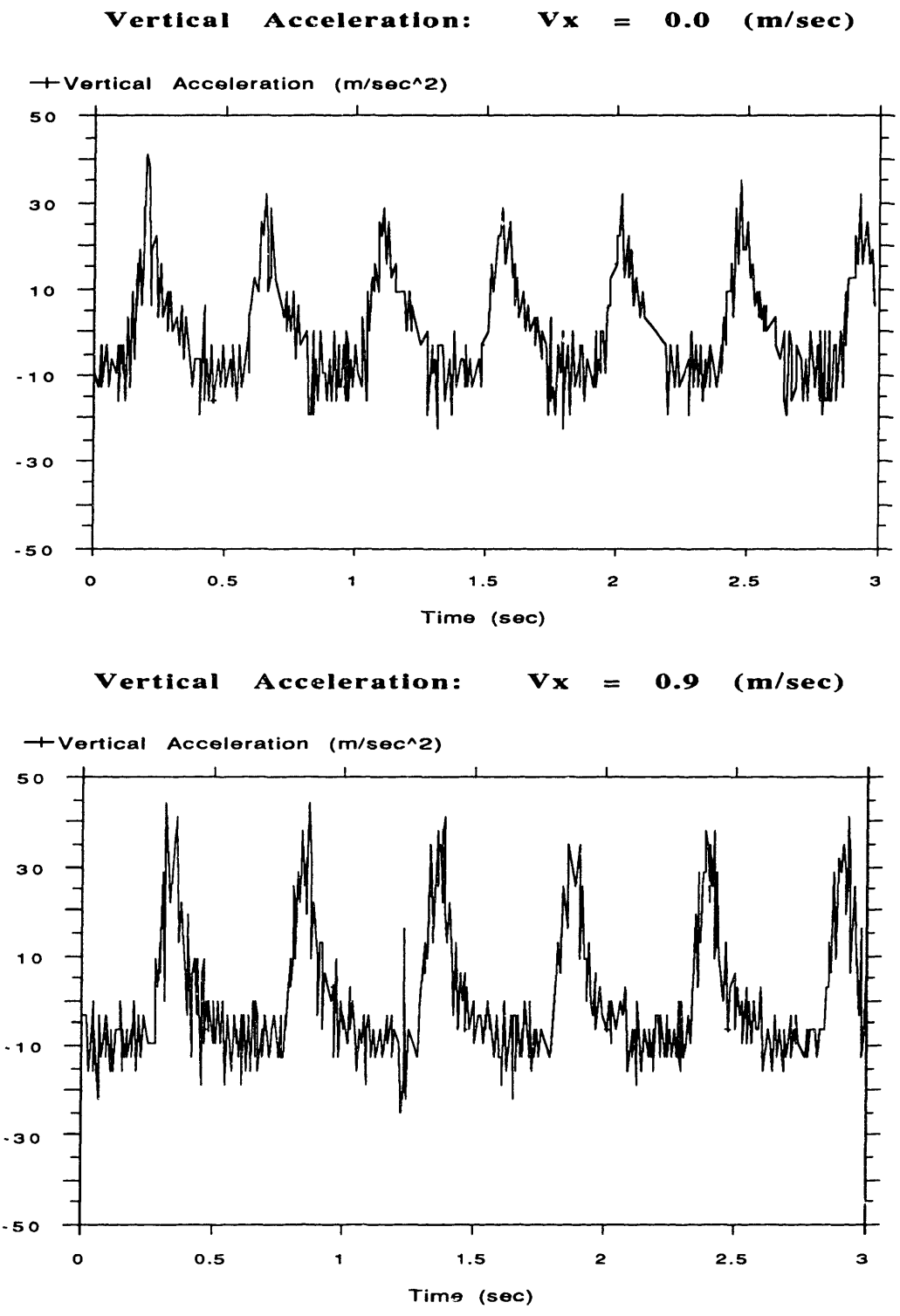


Figure 6-3: The vertical accelerations of the monopod from two runs with different forward velocities; 0.0 m/sec and 0.9 m/sec. The average peak vertical accelerations are 30 m/sec^2 and 35 m/sec^2, implying that the vertical acceleration is relatively insensitive to the changes in the forward velocity.

6.3 Maximum Acceleration Limitation

The control system for the monopod controls the forward running speed of the machine by placing the foot with respect to the neutral point. If the foot is placed at the neutral point, the net acceleration over the stance phase is zero. If the foot is displaced from the neutral point, the machine experiences a net acceleration over the stance phase that is proportional to the magnitude of the displacement of the foot placement from the neutral point. In this section, we discuss adjusting the angle of the heel at the touchdown in order to enhance the forward velocity control. Then we investigate the relationship between the net acceleration and the deviation of the foot placement from the neutral point. This relationship is then used to control the forward speed of the monopod with a hoof.

6.3.1 Heel Angle Adjustment

Before delving into further discussion on controlling the acceleration of the monopod, we note that the forward speed control strategy was developed initially for running machines with a telescoping leg. On these machines the thrust force passes through the center of the mass of the system along the axis of the leg. Such thrust force does not cause a torque about center of mass of the system. However, this is not always true for the monopod. Originally the monopod was designed so that the thrust force would go through the center of the mass when the leg was near vertical. This design criteria was achieved by having the length of the foot equal to the distance from the center of the mass to the hip joint. The toe is directly beneath the center of mass when the leg is vertical and the angle of the heel is near horizontal. This configuration makes the system quasi-statically stable -- the monopod will remain standing when it is placed on the ground carefully with its toe directly beneath the center of mass. The control system for the monopod was designed to take advantage of this fact; the thrust force would pass approximately through the center of the mass when the monopod was running in place.

In the original control system of the monopod, the setpoint of the foot spring was set to one of two fixed points: θ_{heel_max} during thrust and θ_{heel_min} during flight and compression. θ_{heel_max} was adjusted depending on the desired hopping height during the experiment, but θ_{heel_min} was constant (approximately 10 degrees.) θ_{heel_min} was set as low as possible to minimize the coupling while still preventing heel contact. The problem of heel contact was eliminated by the addition of the hoof. This method of controlling the thrust worked adequately in the past. However, note that such method ensures the

alignment of the thrust force with the center of the mass only when the machine is running in place, and the possible inducement of the disturbance in the forward speed control otherwise raises the need for more systematic way of evaluating the misaligned thrust force.

If the leaf spring foot of the monopod is modelled as the cantilever beam, then only the force exerted perpendicular to the cantilever beam is stored as the elastic energy. Similarly, the force induced by the recoiling of the cantilever is roughly perpendicular to the beam itself. Therefore the thrust force exerted by the monopod's leaf spring foot is roughly perpendicular to the angle of the foot. If the body attitude control keeps the body angle at the desired angle (usually horizontal), then whether the thrust force at the toe or the hoof will be aligned with the center of the mass will then depend primarily on the angle of the foot spring. Since the length of the foot is equal to the distance between the center of the mass and the hip joint, the alignment of the thrust force with the center of the mass can be achieved by having the foot spring perpendicular to the leg as shown in figure 6-4. Obviously, achieving the continuous alignment of the thrust force with the center of the mass is difficult since the angle of the foot and the angle of the leg are continuously changing independently as the leaf spring foot deflects under load during the stance phase. But, it is possible to choose the setpoint of the angle of the heel such that the average thrust force during the stance phase is aligned with the center of the mass (figure 7-5). Such treatment is expected to minimize the torque caused by the thrust force about the center of the mass, thus minimizing the disturbance in the forward speed control.

When the monopod is running at a steady forward velocity, the toe or the hoof is placed at the neutral point at the touchdown, and the leg travels symmetrically about the neutral point. In addition, if the bouncing motion of the monopod is governed by the relatively stiff passive foot spring only (no thrust), then setting the setpoint of the foot spring to horizontal will imply that the average foot spring force will be aligned with the center of the mass. This setting also implies that the original method of controlling the thrust will result in approximately the zero average torque inducement about the center of the mass while the monopod is running forward at a constant velocity. However, this method fails when the monopod is accelerating, which is our biggest concern at this point.

In order to achieve zero average torque about the center of mass when the machine is accelerating as well as when it is running at a constant forward velocity, we adjust the setpoint of the foot spring by an amount that is proportional to the shift of the average leg angle θ_{leg_diff} during the stance phase. The thrust vector lies roughly in the middle of the leg travel during the stance phase. An example of this approach is shown in figure 6-6.

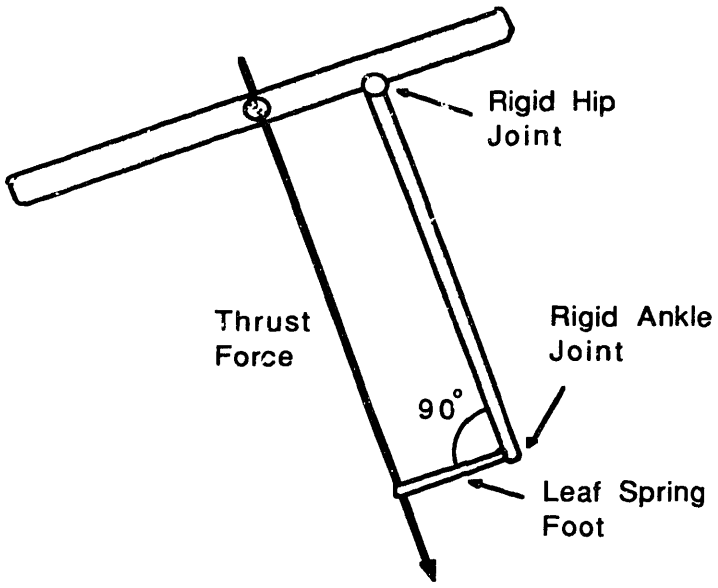


Figure 6-4: Treating the leaf spring foot of the monopod as a cantilever beam and the ankle and hip joints as rigid, the thrust force of the leaf spring foot is perpendicular to the foot angle. Because the length of the foot is approximately equal to the distance between the center of the mass and the hip joint, the thrust force can be roughly aligned with the center of the mass by having the foot perpendicular to the leg.

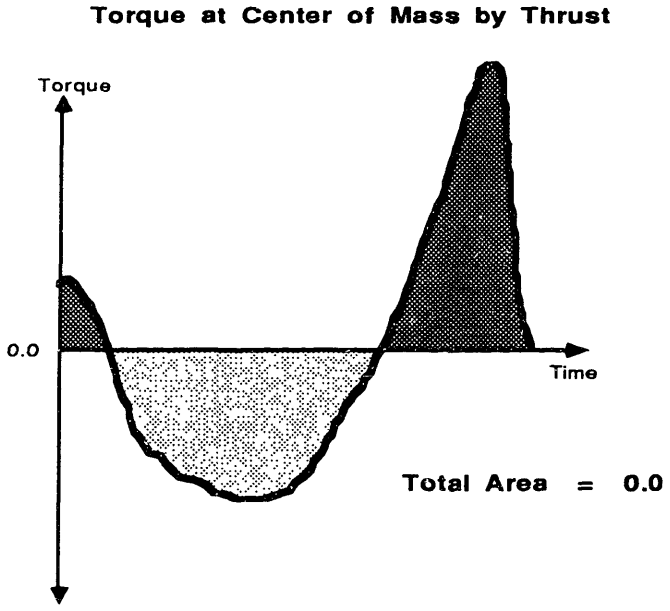


Figure 6-5: The thrust force cannot be aligned with the center of the mass continuously throughout the stance phase because the angle of the foot and the angle of the leg are continuously changing independently as the leaf spring foot deflects under load. The misaligned thrust force induces torque about the center of the mass. However, the setpoint of the angle of the foot spring can be chosen such that the average torque induced by the thrust about the center of the mass will be roughly zero. Such treatment is expected to minimize the disturbance in the forward speed control.

However, in order to employ this method of foot spring set point adjustment, we need a rough idea of the range of the leg's angular travel during the stance phase. To be able to predict the leg angle during the stance phase, the simplified model of the one-legged running machine from the section 6-1 was used. The computer simulation of the simplified model showed that the total leg angle travel θ_{leg_travel} during the stance phase does not vary substantially as the foot placement is displaced from the neutral point (figures 6-7); the figure 7-7-b shows the plot of θ_{leg_travel} versus $\theta_{leg\_touchdown}$ from the simulations of the simple model in which θ_{leg_travel} varies approximately 6 deg for whole range of $\theta_{leg\_touchdown}$ from -13 deg to 18 deg. (The neutral point foot placement is achieved when the $\theta_{leg\_touchdown}$ is roughly -4.3 deg. We call the leg angle which results in the foot placement at the neutral point, the *neutral angle* as shown in figure 6-7. The above result implies that the average leg angle during the stance phase shifted as the monopod accelerated by an approximately equal amount to the deviation of the leg angle from the neutral angle, θ_{accel}. This algorithm was added to the control system and the setpoint of the foot spring at the touchdown is adjusted according to

$$\theta_{foot\_o} = K_{heel}\,\theta_{accel} + \theta_{foot\_offset} \tag{6.6}$$

where

θ_{foot_o}	is the setpoint of the foot spring at touchdown,
θ_{accel}	is the deviation of the leg angle from the neutral angle,
θ_{foot_offset}	is the offset in setpoint to compensate for the effect of thrust (roughly 5 deg, determined empirically), and
K_{heel}	is the proportional factor (roughly 1 to 1.5).

The effect of the foot spring setpoint adjustment on the forward speed control was tested in simulation, and the plots of the forward velocity tracking performance and the thrust force component perpendicular to the alignment with the center of the mass are shown in figure 6-8. The forward velocity from the original control system resulted in overshoot whenever the monopod accelerated. The overshoot is due to the torque induced about the center of the mass by the misaligned thrust force. Such disturbances are expected to be more serious for larger accelerations. On the other hand, the modified control system resulted in relatively smoother transitions in the forward speed. The plots of the component of the thrust force perpendicular to the effective leg show that the average thrust force is roughly aligned with the center of the mass for the modified control system whereas it is not for the original control system. Now that we are equipped with better thrust control, we are ready to proceed to the next step: acceleration control.

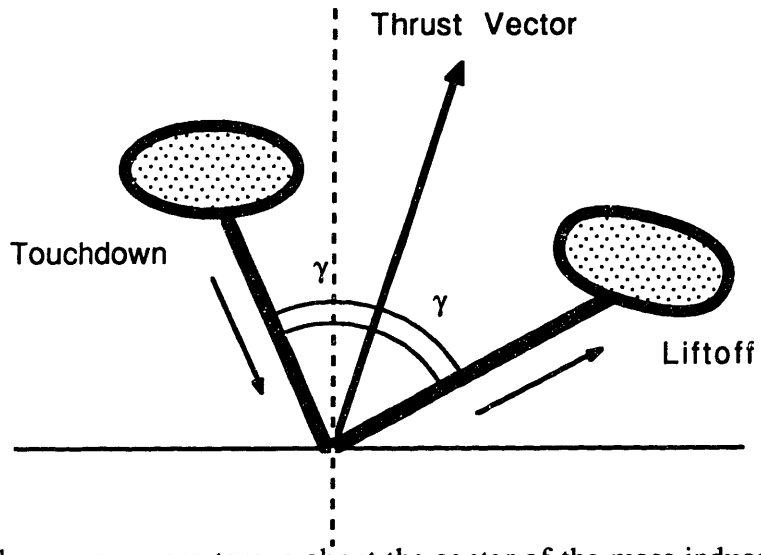
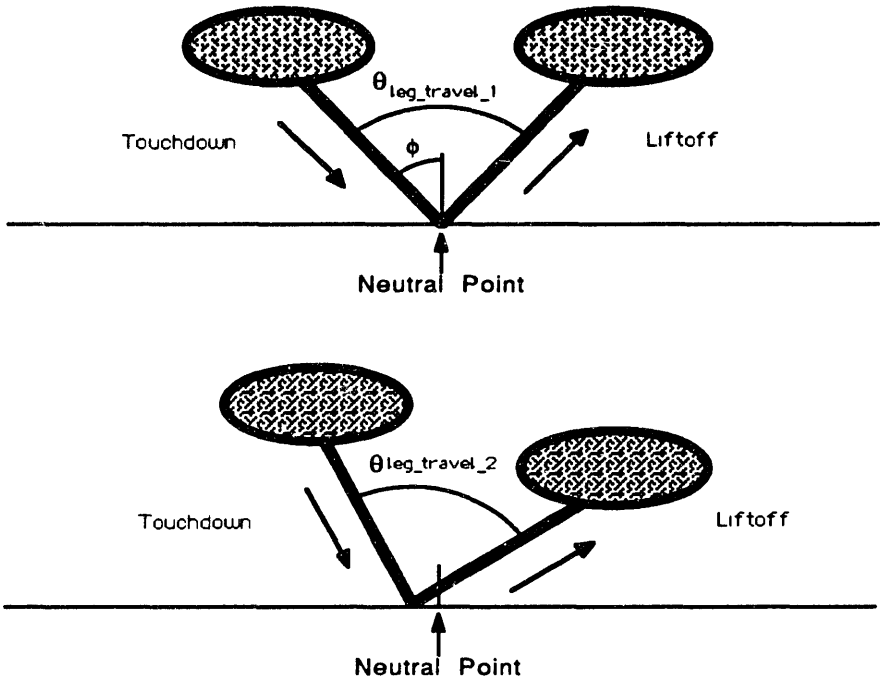


Figure 6-6: The average zero torque about the center of the mass induced by the thrust force can be achieved when the monopod is accelerating as well as when it is running at a constant forward speed if the average thrust force vector lies roughly in the middle of the leg travel during the stance phase.

6.3.2 Foot Placement and Net Acceleration

The rolling of the hoof during the stance phase is caused by two driving forces; the torque at the hip joint for correcting the body attitude and the inertial loading at the hoof due to the change in momentum of the system. Coordinating the hip torque of the body attitude control to the vertical loading at the hoof was unsuccessful because of the disturbance to body attitude. As a result, the hip torque coordination is not pursued further, and maximum acceleration limitation is adopted as the technique for preventing the rolling of the hoof. However, the body attitude control system still exerts a torque at the hip during the stance phase, and we propose to experimentally find the range of the forward accelerations in which the hip torque would not cause the hoof to roll.

In the forward speed control system of the monopod, the deviation of the foot placement from the neutral point is used to induce forward acceleration, and the deviation of the angle of the leg from the neutral angle is used here as the control variable for the net acceleration. The relationship between the neutral angle and the net forward acceleration of the monopod is first investigated through computer simulation and then verified with experiments on the physical machine.



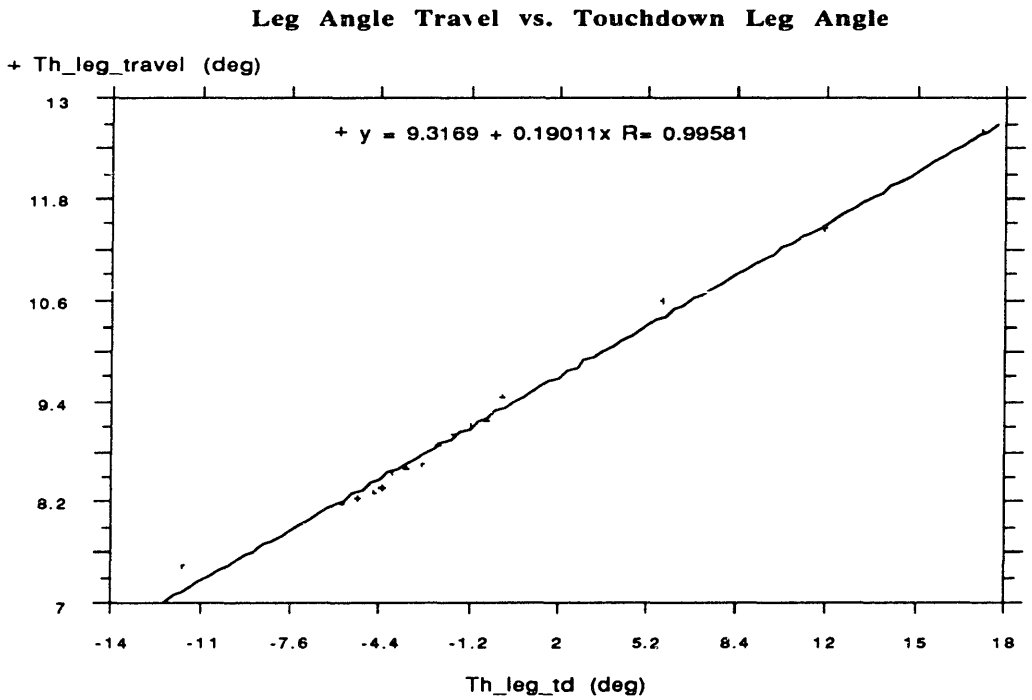
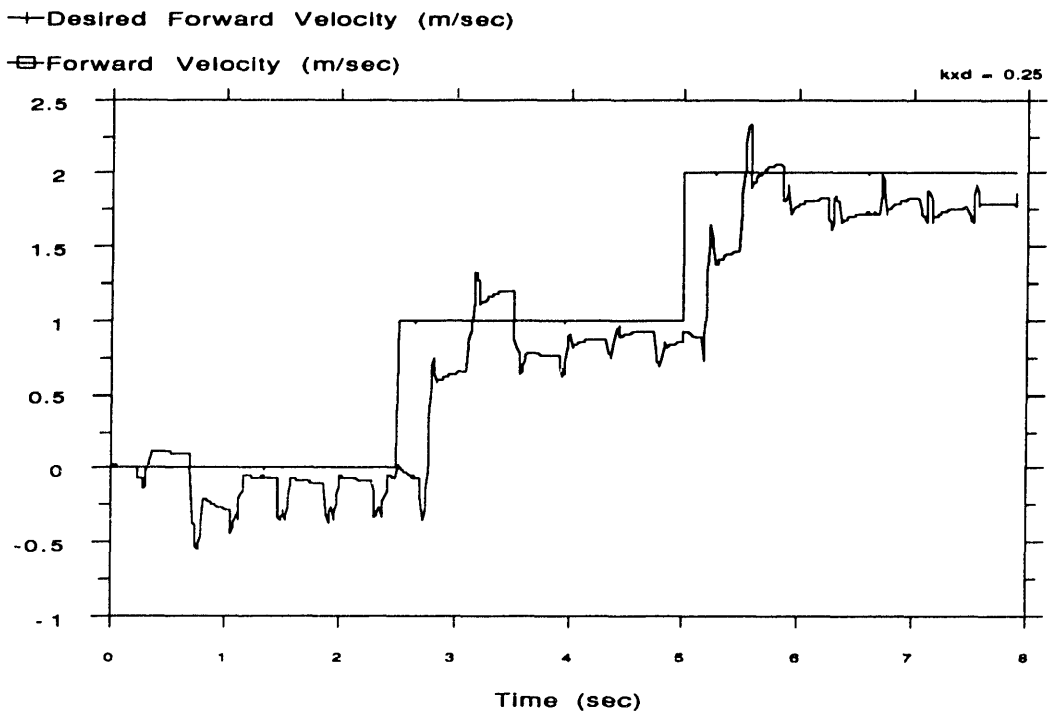
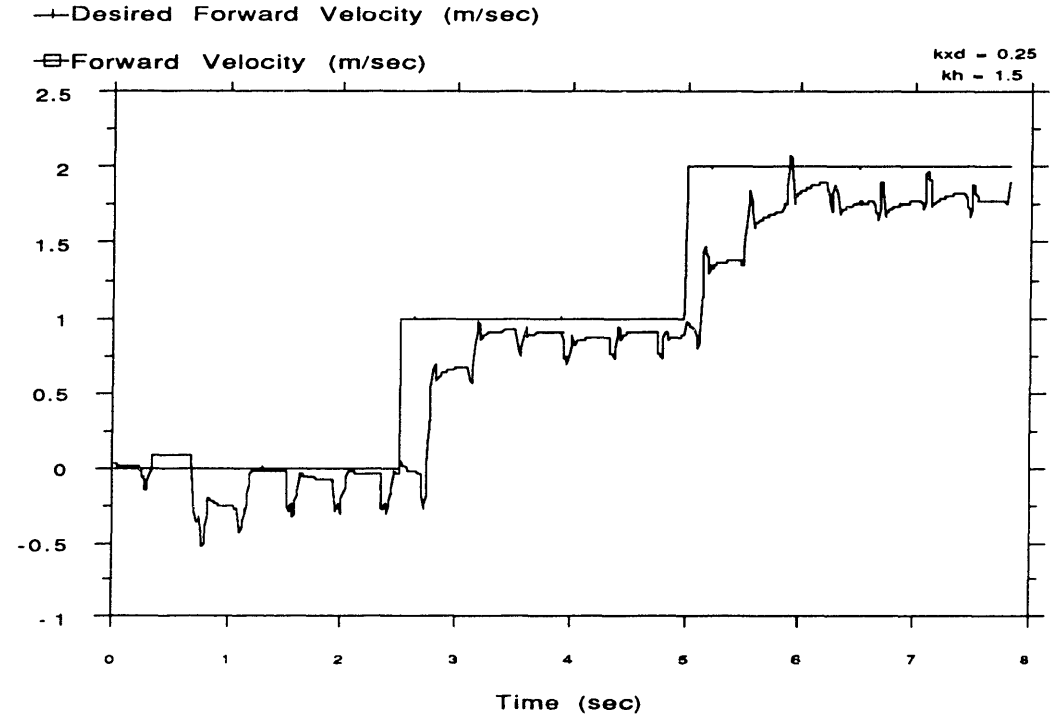


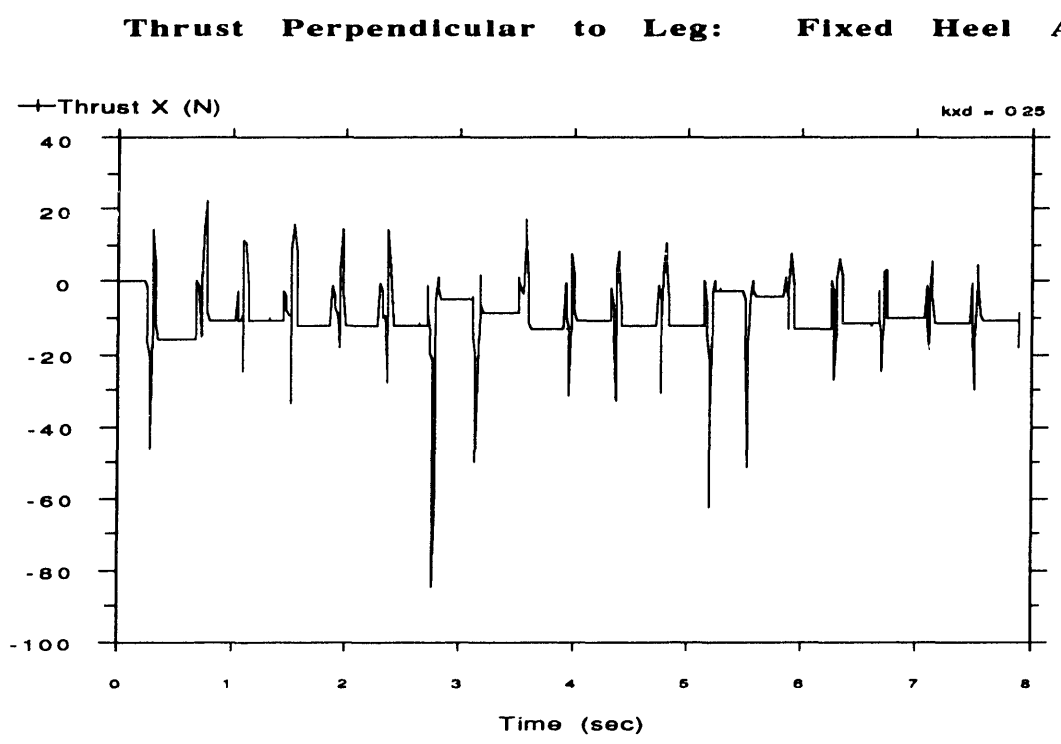
Figure 6-7: The computer simulation of the simplified model showed that the total leg angle travel θ_{leg_trave} during the stance phase varies little as the foot placement at the touchdown is varied from the neutral point as shown in figure 7-7-a. This observation implies that the average leg angle during the stance phase shifted as the monopod accelerated by roughly equal amount to the deviation of the leg angle from the neutral angle. Figure 7-7-b shows the plot of θ_{leg_travel} versus $\theta_{leg\_touchdown}$ where the neutral angle is about -4.3 deg. It is evident that the θ_{leg_travel} is insensitive to changes in $\theta_{leg\_touchdown}$.

Forward Velocity Tracking: Fixed Heel Angle



Forward Velocity Tracking: Varied Heel Angle





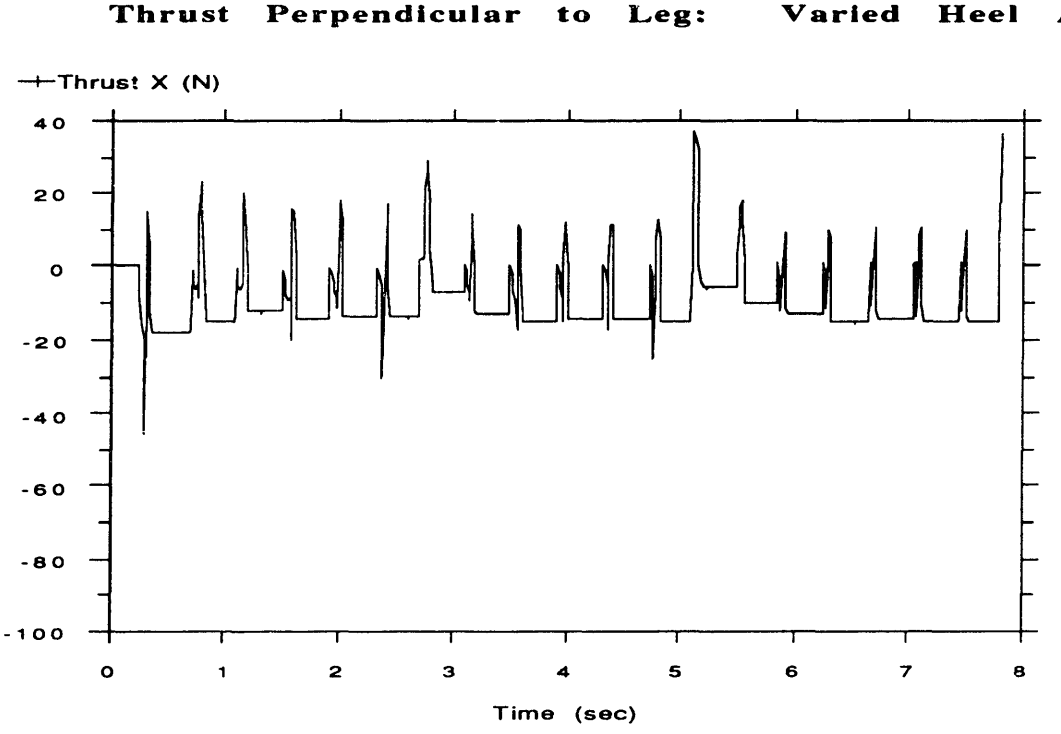


Figure 6-8: Plots of the forward velocity tracking performances and the corresponding thrust force component perpendicular to the alignment with the center of the mass. The forward velocity control from the original control system resulted in overshoot whenever the monopod was commanded to accelerate. The control system with heel angle adjusting scheme shows much smoother transitions in forward speed.

Simulation

In the simulation using the model from section 4.3.1, the model of the monopod ran at several forward velocities. The net forward acceleration of the center of the mass is plotted against the deviation from the neutral angle in figure 6-9. The forward velocity at touchdown was varied from 0.0 m/sec to 2.0 m/sec, and three least square lines were fitted for steps where the touchdown forward speed varied from 0.0 m/sec to 0.3 m/sec, from 1.3 m/sec and 2.0 m/sec, and from 0.0 m/sec to 2.0 m/sec (combined). The equations of the fitted lines are

$$V_{x\_diff} = 6.1887\ \theta_{accel} + 0.52409 \qquad (6.7)$$
$$R = 0.97245$$

$$V_{x\_diff} = 9.0594\ \theta_{accel} + 0.65264 \qquad (6.8)$$
$$R = 0.94717$$

and

$$V_{x\_diff} = 8.0782\ \theta_{accel} + 0.66216 \qquad (6.9)$$
$$R = 0.97211$$

where

V_{x_diff} is the net change in forward velocity over the stance phase (m/sec),

θ_{accel} is the deviation of the leg angle from the neutral angle (degree),

R is the correlation of the data with respect to the fitted line.

respectively. As shown from the plots, the relationship between the V_{x_diff} and the θ_{accel} is linear in general, regardless of the forward velocities. For the low forward speed the data follow the linear fit very closely, but for high forward speed the data points do not follow the linear fit as well. This result is expected since the forward velocity of the center is measured and the forward velocity is expected to be quite influenced by the angular velocity of the body with respect to the hip joint. The leg angle travel increases as the forward velocity of the monopod increases, and thus the pitching of the body increases as well; this result is a consequence of the law of conservation of angular momentum. Therefore, the measure of the forward speed is much more susceptible to noises caused by the rocking of the body. The relationship between the slope of the plots, and the forward touchdown velocity is also an important factor in designing the strategy for controlling the monopod. The simulation results show that the slope increases as the forward touchdown velocity increases. This result was verified with experiments on the physical machine.

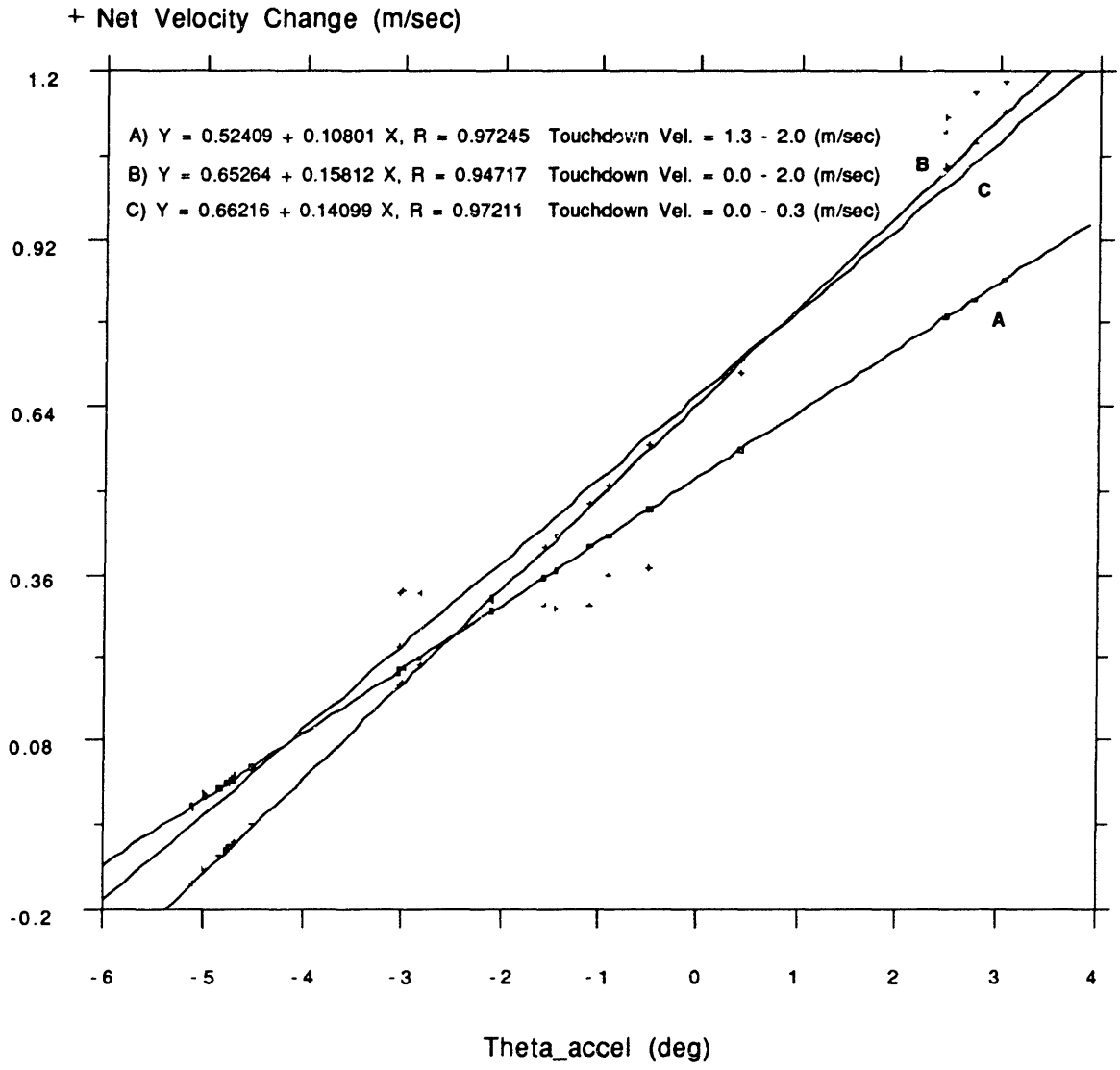


Figure 6-9: The plot of the simulation results of the net velocity change over the stride versus the deviation from the neutral angle. Straight lines a), b) and c) are fitted for the data with forward speed at touchdown between 0.0 m/sec and 0.3 m/sec, between 1.3 m/sec and 2.0 m/sec, and of all steps combined. Straight lines are fit to the data with relatively good correlations.

Experimental Results

The plot of the net velocity change during stance phase (V_{x_diff}) versus the deviation of the leg angle from the neutral angle (θ_{accel}) from experiments with the physical machine are shown in figure 6-10. As with the simulations described above, the monopod ran at different forward velocities, and the simple line fits are performed to the data collected and the expressions of the fitted lines are

$$V_{x\_diff} = 0.049358\ \theta_{accel} + 0.031579 \qquad (6.10)$$
$$R = 0.91237$$

for the forward velocities of 0.0 to 0.3 m/sec,

$$V_{x\_diff} = 0.054505\ \theta_{accel} + 0.0065538 \qquad (6.11)$$
$$R = 0.89278$$

for the forward velocities of 0.5 to 1.0 m/sec, and

$$V_{x\_diff} = 0.051745\ \theta_{accel} + 0.016216 \qquad (6.12)$$
$$R = 0.89457$$

for the combined forward velocities of 0.0 to 1.0 m/sec. The good correlations of the data to the straight line confirm that the net velocity change over the stance phase is related linearly to the deviation of the leg angle from the neutral angle. The actual experiments also confirm that the rate of V_{x_diff} change with respect to θ_{accel} increases as the forward touchdown velocity increases. Although the rate, $\Delta V_{x\_diff} / \Delta\theta_{accel}$, varies with the change in forward velocity, the magnitude of the change in rates does not seem significant. For this reason, the change in rate, $\Delta V_{x\_diff} / \Delta\theta_{accel}$, is not included as a factor in the strategy design.

6.3.3 Rolling or Falling?

Analysis presented in previous sections showed that the horizontal acceleration of the running machine during the stance phase induces inertial loading (horizontal force F_x) at the hinge of the hoof. Furthermore, we also understand the relationship between the horizontal acceleration of the machine during the stance phase, which is represented as the net change in forward velocity during the stance phase V_{x_diff} in this case, and the control variable, the deviation from the neutral angle, θ_{accel}. Now it is a matter of finding the minimum acceleration of the monopod which causes the rolling of

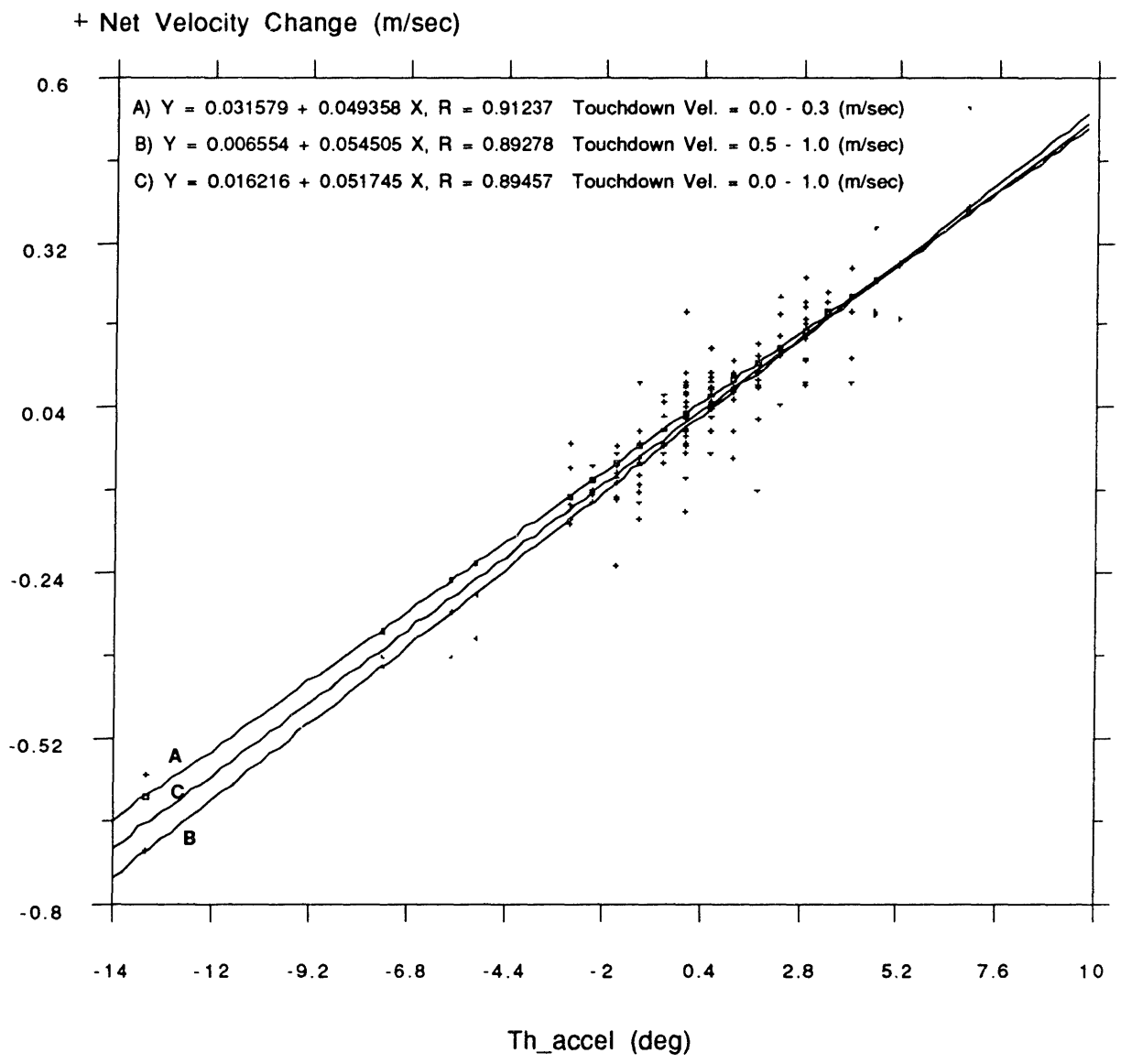


Figure 6-10: The plots of the experimental results of the net velocity change over the stride versus the deviation from the neutral angle. Straight lines a), b) and c) are fitted for the data with forward speed at touchdown between 0.0 m/sec and 0.3 m/sec, between 0.5 m/sec and 1.0 m/sec, and of all steps combined. Straight lines are fit to the data with relatively good correlations. Although the slope of the fitted lines vary with the change in the touchdown forward speed, the variance in the slope is not significant enough to be included as a factor in the control strategy design.

the hoof. This maximum acceleration can be used as the upper bound of the acceleration allowed in the control system.

At this point, I mentioned to a friend the plan of limiting the maximum horizontal acceleration to prevent the rolling of the hoof and he commented that excessive forward acceleration of a running system also seems to cause another problem---falling over. When I thought about what he said, it certainly made sense. Intuitively, the horizontal acceleration should effect the stability of the running cycle. I remember from a personal experience that when I tried to accelerate too quickly, my body would tip over and finally I would fall forward on my nose. This phenomenon is also present in the running of the robots since they must obey the same laws of physics. This intuition raises a question; would the hoof roll before the machine falls over, or vice versa? We could take whichever happens first and use it to establish the maximum acceptable forward acceleration. The nature of "falling over" is further investigated. Better understanding of the mechanism of "falling over" is expected to help us in finding the forward acceleration limits of the monopod.

Falling Over: When and Why? - Hypothesis

Why does a person or a machine such as the monopod fall over when it accelerates excessively? Is excessive acceleration the real cause of the machine's falling over? After some investigating, I soon concluded that the answer is no. The excessive acceleration of the running system is not the direct cause of the system's falling over. The rest of this section describes why I came to this conclusion .

The real cause of the machine's falling over is intimately coupled to the profile of the leg angle and thrusting action during the stance phase. To be more precise, the actual deciding factor for the machine's falling over is the trajectory of the center of the mass of the system. The direction and the magnitude of the velocity vector of the system's center of the mass determines this trajectory.

Assuming that the thrust at liftoff stays approximately constant for simplicity, the maximum height and the flight duration of the machine during the flight phase will depend on the direction of the center of the mass's velocity at lift-off. This relationship is shown in figure 6-11---the assumption of constant thrust at liftoff needs to be verified, though. In other words, with a given amount of kinetic energy at lift-off, the trajectory of the center of

mass is governed by the law of conservation of momentum. If we call the angle of the velocity vector of the center of mass at liftoff with respect to vertical $\theta_{liftoff}$, the profile of the trajectory of the center of mass is governed by

$$X_{cg} = (V_o \sin \theta_{lift-off})\, t$$
$$Y_{cg} = (V_o \cos \theta_{lift-off})\, t - \frac{1}{2} G t^2 \qquad (6.13)$$

where

X_{cg} is the horizontal position of the center of mass of the system with respect to the location of the center of mass at liftoff,

Y_{cg} is the vertical position of the center of the mass of the system with respect to the location of the center of mass at liftoff,

V_o is the magnitude of the velocity vector of the center of mass at liftoff,

$\theta_{lift-off}$ is the angle of the velocity vector of the center of mass with respect to vertical at liftoff,

t is time since liftoff

G is gravitational acceleration.

Now, understanding the dynamics of the trajectory of the flight phase using basic physics, the first intuitive requirement to avoid falling over is that the maximum height of the body must be greater than the length of the leg. Otherwise the foot will not clear the ground. Another intuitive requirement is that the monopod must stay in the air long enough to allow the hip actuator to swing the leg forward to its desired angle before touchdown. Combining these two factors, the monopod must stay above the ground clearing height longer than the maximum time required by the hip actuator to swing the leg forward. Otherwise, the monopod is likely to land prematurely, and fall over. I propose this situation is exactly what causes the running system to fall over---the monopod lifts off with such a steep angle $\theta_{lift-off}$ and not enough speed that the flight time is insufficient for the hip actuator to swing the leg forward to the angle required for stability of the running cycle.

Experiments showed that the liftoff speed increased from 0.9 m/sec to 1.35 m/sec linearly as the forward velocity varied from zero to 1.4 m/sec (figure 6-12) with the slope of approximately 0.38. However, this relative low slope indicates that $\theta_{lift-off}$ would be

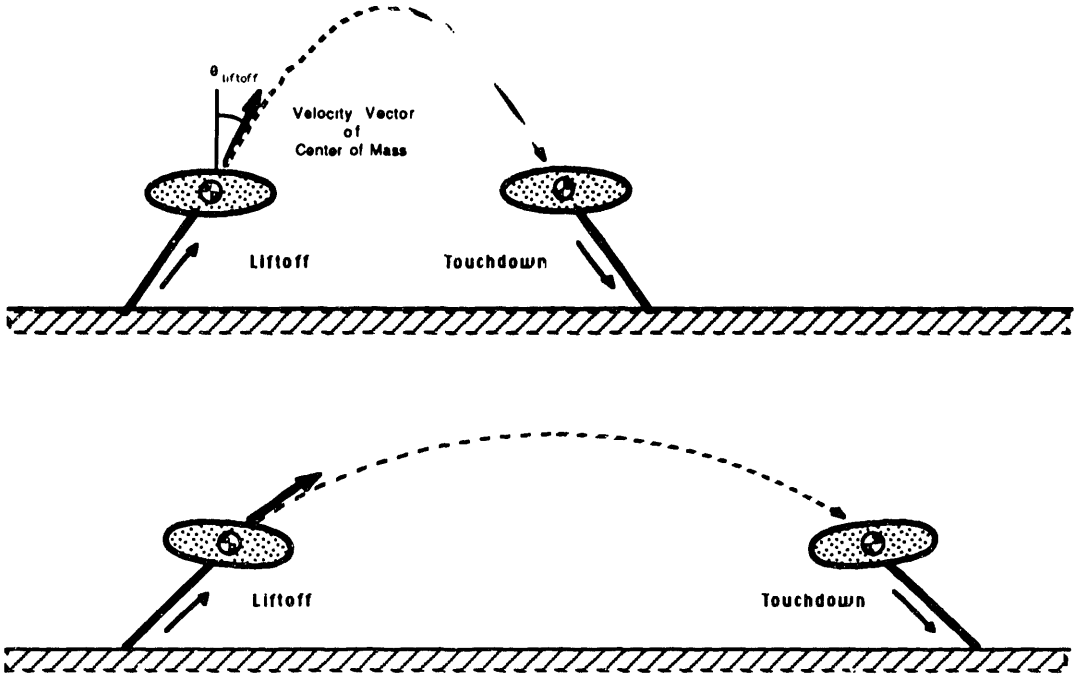


Figure 6-11:For a given amount of thrusting energy at the lift-off, the maximum height and flight duration of the machine during flight depend on θ_{lift_off}. It is expected that the phenomenon of "falling over" is a direct consequence of the trajectory of the machine in the flight phase rather than the forward acceleration.

the dominant factor in determining the trajectory of the monopod during the flight phase, which in turn determine whether the monopod is likely to fall or not. I would then expect there to exist a maximum forward velocity for the monopod. The maximum forward velocity allowed at lift-off is

$$V_{x\_max} = V_o \sin \theta_{lift\text{-}off\_max} \tag{6.14}$$

where

$\theta_{lift\text{-}off\_max}$ is the maximum $\theta_{lift\text{-}off}$ allowed without causing the falling over.

This reasoning implies that the acceleration of the machine does not have a direct impact on falling over, but rather that the acceleration would lead to the machine's falling over only if the forward acceleration during the stance phase would cause the forward velocity at lift-off to be greater than the maximum forward velocity allowed, V_{x_max}. This suggests that the maximum allowable acceleration for the monopod will be limited by two factors: the forward acceleration which causes the rolling of the hoof and the maximum forward velocity at lift-off. This analysis is expected to be helpful in designing a control strategy. However, the explanation of the mechanism leading to falling over is still only a hypothesis, and it must be experimentally verified.

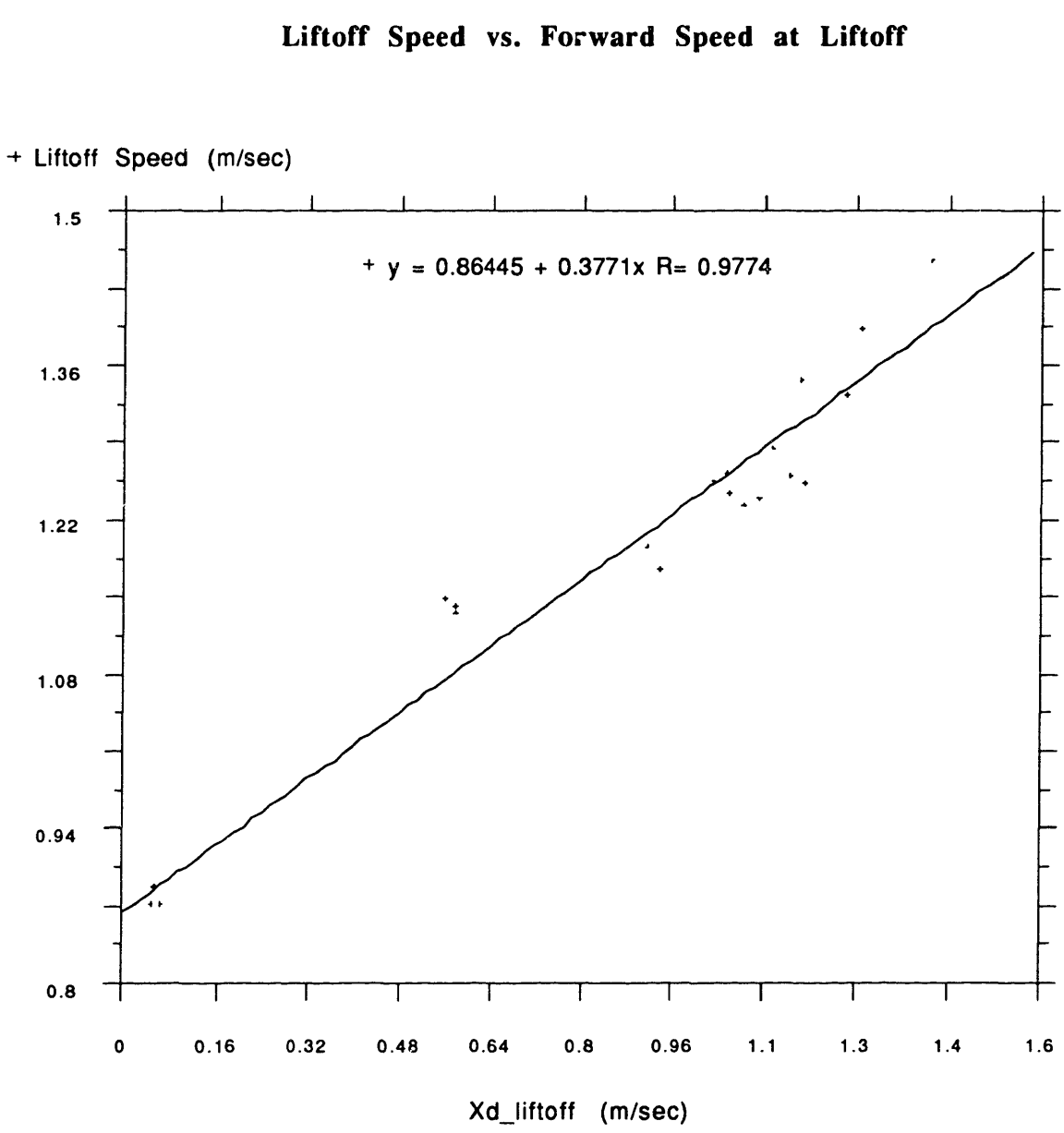


Figure 6-12: Graph of liftoff speed versus the forward velocity at liftoff. The results show that the liftoff speed increases linearly as the forward velocity at liftoff varied from zero to 1.4 m/sec with the slope of approximately 0.38.

Falling Over: Experimental Results

To verify the mechanism of "falling over" proposed above, experiments are performed in which the monopod is pushed to the limit -- to the verge of instability -- causing either falling forward on its nose or rolling of the hoof (which also eventually causes falling). Table 6-1 shows the states of the monopod during the stance phase immediately prior to the failure, and the results show excellent agreement with the proposed mechanism of "falling over".

The forward velocity of the monopod at the touchdown (V_{x_td}) and the net forward velocity change during the stance phase (V_{x_diff}) are used as the control variables for the experiment, and the observed relevant variables are the forward velocity at lift-off (V_{x_lo}), the deviation of the leg angle from the neutral angle (θ_{accel}), the angle of the leg with respect to the vertical at lift-off (θ_{leg_lo}) and the angle of the velocity vector of the center of mass with respect to the vertical ($\theta_{lift-off}$). The forward velocity at the touchdown (V_{x_td}) is varied from 0.3 m/sec to 1.24 m/sec, and the net velocity change during the stance phase (V_{x_diff}) is independently varied from 0.13 m/sec per stance to 0.65 m/sec per stance.

Data from five runs, each ending in failure, are listed in table 6-1. Only run *A* failed due to the rolling of the hoof, the other four runs ended because the machine fell forward. The rolling of the hoof is detected visually and also by the deflection of the foot --- the rolling of the hoof causes a sudden momentary relaxation of the foot spring. Figure 6-13 shows the foot deflection from two bounces, one from a normal bounce and one from a bounce where the hoof rolled. Table 6-1 shows that the runs where the monopod failed by falling forward are characterized by the high forward velocities at lift-off (between 1.2 m/sec and 1.5 m/sec with an average of 1.33 m/sec), and also by a relatively large angle for the velocity vector at lift-off (between73 degree and 80 degree with an average of 76.9 degree). However, the value for V_{x_diff} vary from 0.13 m/sec to 0.65 m/sec without much correlation. These observations imply that the monopod falls forward whenever it reaches a forward velocity of approximately 1.3 m/sec at lift-off independent of the forward acceleration during the stance phase. On the other hand, run A is characterized by a high net change in forward velocity during the stance phase (approximately 0.64 m/sec per stance). The forward velocity at lift-off is relatively small compared to the other runs, and thus the angle of the velocity vector is also small compared to the others. However, note that V_{x_lo} and the $\theta_{lift-off}$ is not the deciding factor in whether the hoof rolls or not. Run D failed by falling forward, although in this case the failure could have been caused by either

falling forward or rolling of the hoof, or even both because both V_{x_lo} and V_{x_diff} were high.

	V_{x_td} (m/sec)	V_{x_lo} (m/sec)	V_{x_diff} (m/sec)	θ_{leg_lo} (deg)	θ_{accel} (deg)	$\theta_{lift-off}$ (deg)
A	0.30	0.94	0.64	21.2	12.6	59.0
B	1.24	1.37	0.13	24.5	2.46	79.6
C	0.95	1.40	0.45	25.4	8.15	76.8
D	0.58	1.23	0.65	25.2	10.4	73.3
F	0.83	1.32	0.49	26.2	5.36	77.9

Table 6-1: The results from five runs in which the monopod fell, either by falling forward or by rolling of the hoof. Only run A failed due to the rolling of the hoof. The other four runs failed due to falling forward. The run in which the monopod failed by falling forward is characterized by the high forward velocity at the lift-off of roughly 1.3 m/sec, whereas the run in which the monopod failed because of the rolling of the hoof is characterized by the high net change in the forward velocity during the stance phase (0.64 m/sec).

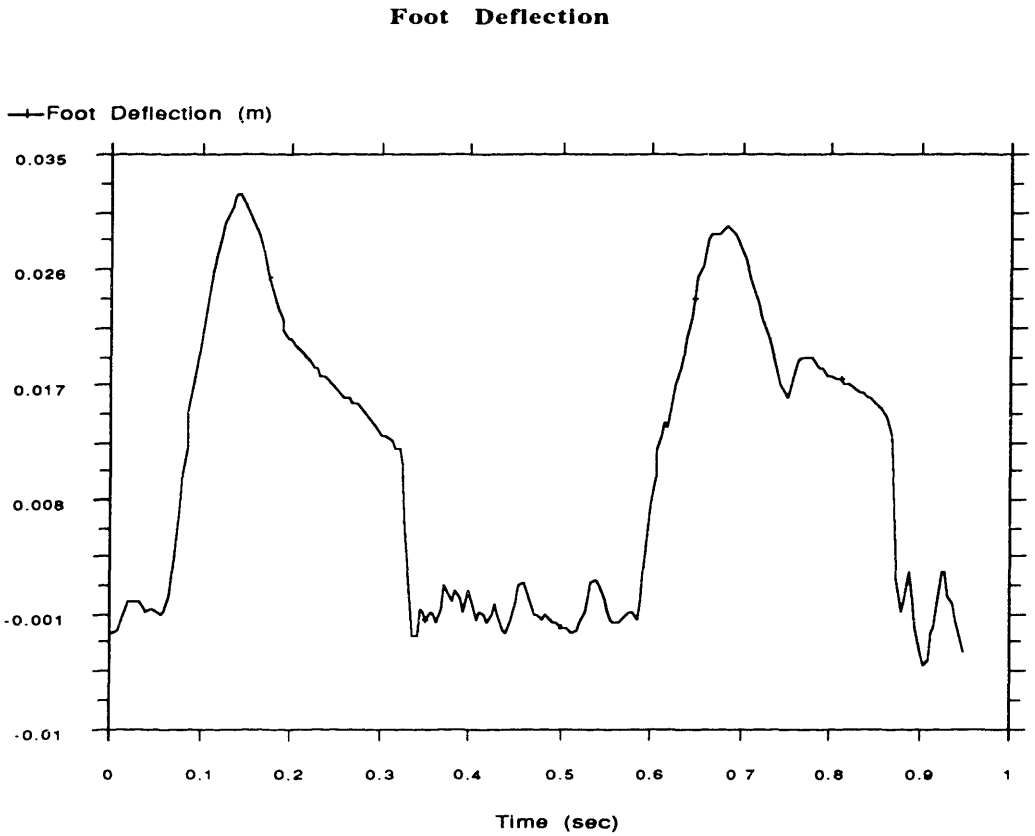


Figure 6-13: The plot of the foot deflection of two bounces; one from a normal bounce and one from a bounce where the hoof rolled. The rolling of the hoof causes sudden momentary relaxation of the foot spring.

Overall, the experiment supported the proposed hypothesis of the conditions which cause "falling over". With this experiment, we also empirically determinedthe maximum forward acceleration (0.64 m/sec per stance) and the maximum forward velocity (1.3 m/sec) that define the boundaries of the region in which the monopod can perform safely. And, best of all, this result can be effectively implemented in the control system. The maximum forward velocity can probably be increased by having stronger thrust and hip actuator and stiffer foot spring which would yield less leg sweep and faster liftoff.

6.4 Control Strategy: Design, Implementation, and Test

The goal of the new control system is to prevent the monopod from failures due to either rolling of the hoof or "falling over". These new goals are in addition to the primary functions performed by the original control system: stabilizing the running cycle by regulating forward speed, hopping height and body attitude. From analysis and experiments, we learned that the rolling of the hoof is induced by two factors: the hip torque from the body attitude correction procedure, and the inertial loading at the hoof due to the horizontal acceleration of the monopod. In addition, we also learned that the monopod would lose stability and "fall over" when it reached a certain forward velocity.

6.4.1 Design and Implementation

The core of the control strategy is the direct implementation of the the information obtained from the previous section; regulations of the monopod's forward velocity at lift-off and forward acceleration per stance within the allowed region shown in figure 6-14. The break point in the forward velocity, V_x^* is

$$V_x^* = V_{x\_max} - V_{x\_diff\_max} \tag{6.15}$$

where

V_x^*	is the forward velocity of the break point,
V_{x_max}	is the maximum forward velocity allowed, and
$V_{x\_diff\_max}$	is the maximum forward acceleration per stance allowed.

While the monopod is in the flight phase, the forward speed control of the original control system will calculate the desired position of the foot at touchdown and the corresponding

deviation of the leg angle from the neutral angle, θ_{accel}. Then, using the previously discussed linear fit from the plot ofV_{x_diff} versus θ_{accel} in the form

$$V_{x\_diff} = m\,\theta_{accel} + b \tag{6.16}$$

where

m is the slope of the linear fit, $\Delta V_{x\_diff} / \Delta\theta_{accel}$, and

b is the vertical axis intersection point,

the control system will predict the value of the forward acceleration during the stance phase. At this point, the control system compares the present forward velocity to the break point forward velocity. If V_x is smaller than$V_x{}^*$, then the maximum forward acceleration per stance becomes the effective maximum forward acceleration per stance allowed, $maxV_{x\_diff}$. If not, then the effective maximum forward acceleration per stance allowed $maxV_{x\_diff}$ becomes

$$maxV_{x\_diff} = V_{x\_max} - V_x. \tag{6.17}$$

Having done this, the control system then compares the predicted forward acceleration per stance to the effective maximum forward acceleration per stance allowed. If V_{x_diff} is smaller than $maxV_{x\_diff}$, then the desired position of the foot placement and the corresponding deviation of the leg angle from the neutral angle are unchanged. However, if V_{x_diff} is greater than $maxV_{x\_diff}$, then the value of the V_{x_diff} is replaced by the $maxV_{x\_diff}$, and the new V_{x_diff} is transformed to the corresponding θ_{accel} by using the linear fit expression

$$\theta_{accel} = \frac{1}{m}\left(V_{x\_diff} - b \right) \tag{6.18}$$

In this way, the control system will ensure that the monopod will always operate in a region where the hoof will not roll and it will not fall over.

Hip Torque Revisited

So far we have discussed the strategy of controlling the maximum forward acceleration and the maximum forward velocity to prevent the hoof from rolling and the machine

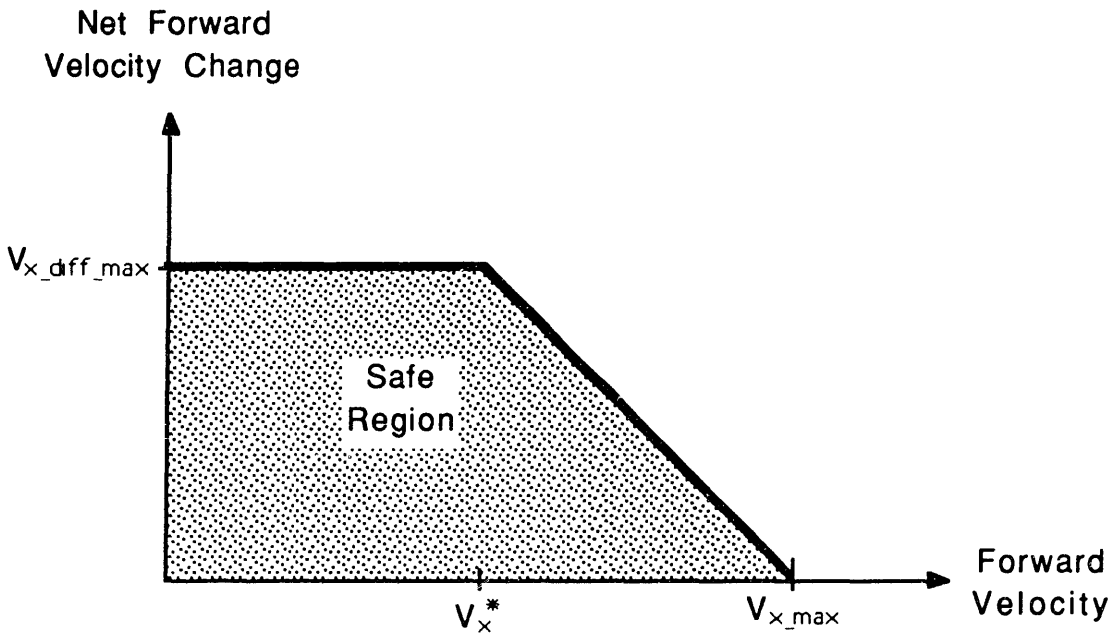


Figure6-14: Diagram of the boundaries of the allowed region in which the monopod can perform safely. The boundaries are determined by the maximum forward speed and the maximum acceleration which were found experimentally.

from falling over. Because of the reasons discussed in the previous section, we have ignored the possibility of controlling the hip torque from the body attitude correction to prevent the hoof from rolling. As we concluded previously, it is still true that setting a maximum torque introduces a disturbance in pitch which worsens the performance of the monopod. However, the rolling of the hoof is influenced by both the hip torque applied during the stance phase and the forward acceleration, and therefore, it is necessary to discuss how large a hip torque should be applied.

The hip torque that can be applied without causing the hoof to roll seems to depend on forward speed. For instance, when the monopod is running in place, the forward acceleration of the monopod is zero, and the hip torque is the only source of the horizontal component of the force induced at the hinge of the hoof. However, as the forward velocity of the monopod increases, regardless of whether the monopod is running at the steady speed or is accelerating, the magnitude of deceleration and acceleration the monopod experiences at touchdown and lift-off increase as well. In other words, the monopod decelerates suddenly upon touchdown, storing kinetic energy in the foot spring. The

monopod regains its velocity during the stance phase, and reaches the lift-off velocity (equal to the touchdown velocity in the case of steady-state running) by the end of the stance phase. An example of this cycle is shown in figure 6-15 where the monopod is running in forward speed of 1.05 m/sec; the states 100, 200, and 300 correspond to the compression phase, thrust phase, and flight phase respectively.

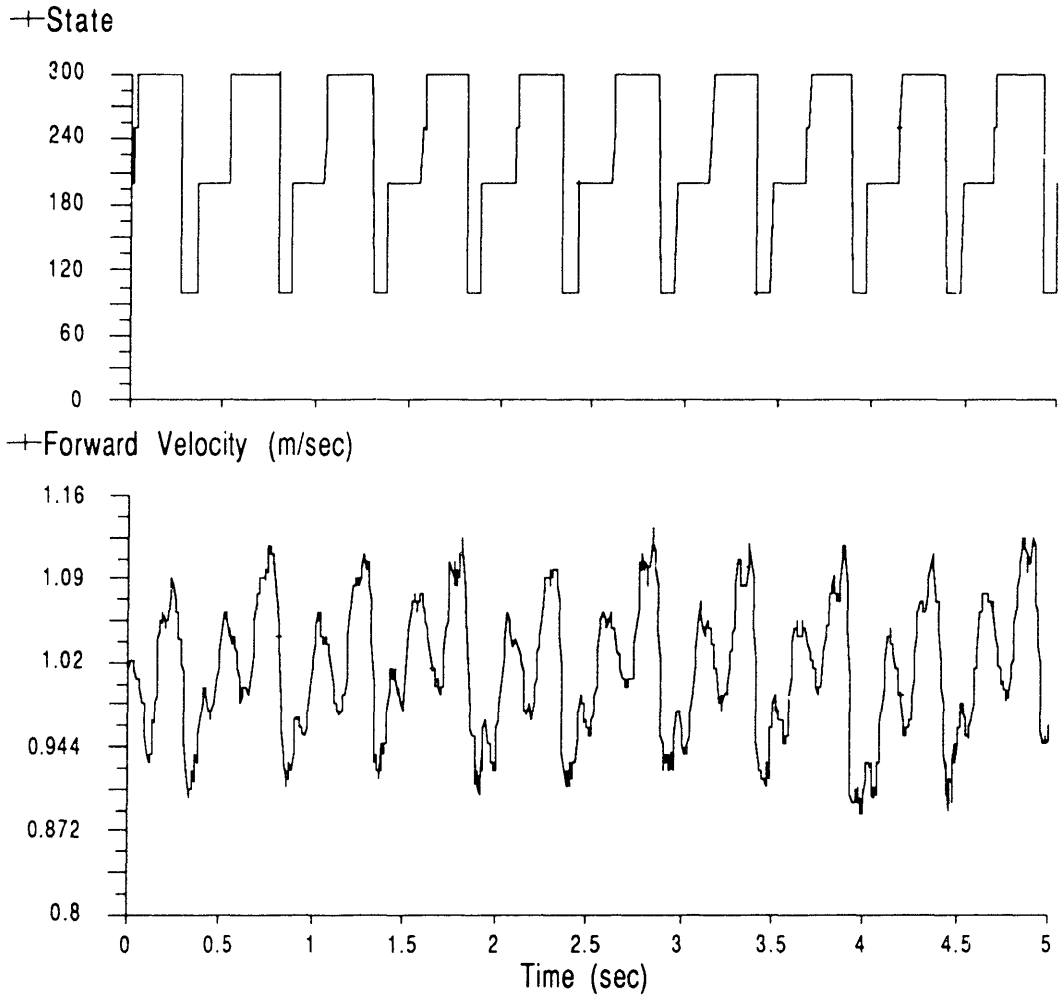


Figure 6-14: Plots of the state and forward velocity of the monopod. The monopod is running at an average forward speed of 1.05 m/sec. The states 100, 200 and 300 correspond to the compression, the thrust and the flight phases respectively. The forward velocity decreases upon landing and increases back to the desired velocity.

The monopod goes through a cycle of deceleration and acceleration regardless of the average forward velocity.

The monopod tends to nose down when running, and as a consequence, the hip torque of the body attitude control results in pushing the leg backward. This effect is in the opposite direction from the inertial loading induced at the toe or the hoof as the monopod decelerates upon touchdown. In addition, also note that the major portion of the hip torque for body correction is applied during the first half of the stance phase. This point is illustrated by the profile of the signal to the hip actuator valve during the stance phase shown in figure 6-16. A negative value of $T2$ corresponds to a hip torque which will push the body upward and the leg backward. This implies that part of the horizontal force at the hoof induced by hip torque will be canceled by the inertial loading at the beginning half of the stance phase. As a result, when forward speed increases, a larger hip torque can be applied safely during the first part of stance without causing the hoof to roll.

The hip torque applied during the stance phase is roughly proportional to the proportional and derivative gains of the linear servo for the body attitude control, K_p and K_v. Therefore, the above result can be directly implemented in the control system by adjusting the gains of the linear servo for the hip torque so that they are proportional to the forward velocity of the monopod.

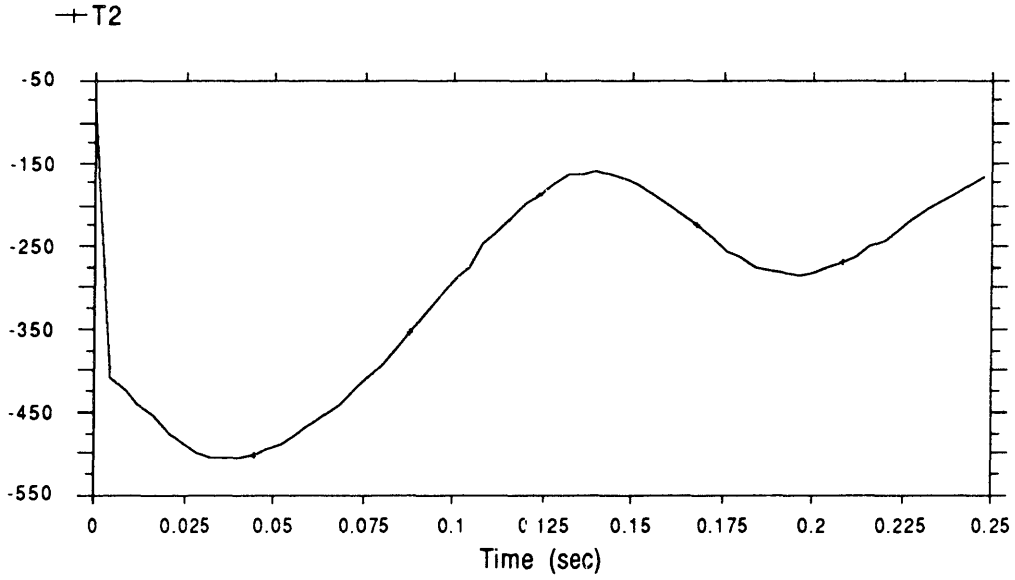


Figure 6-16: Plot of $T2$, the signal sent to the hip actuator servo valve, during the stance phase. A negative value of $T2$ implies that the hip torque is pushing the body upward and the leg backward. The major portion of the hip torque for body attitude correction is applied in the first half of the stance phase.

6.4.2 Test of the Proposed Control Strategy

The control strategy discussed so far was successfully added to the control system of the monopod. The experments showed that the maximum forward velocity and the maximum forward acceleration per stance are approximately 1 m/sec and 0.5 m/sec per stance respectively with about 10 % safety factor. The break point forward velocity V_x^* is roughly 0.5 m/sec. The line fit of

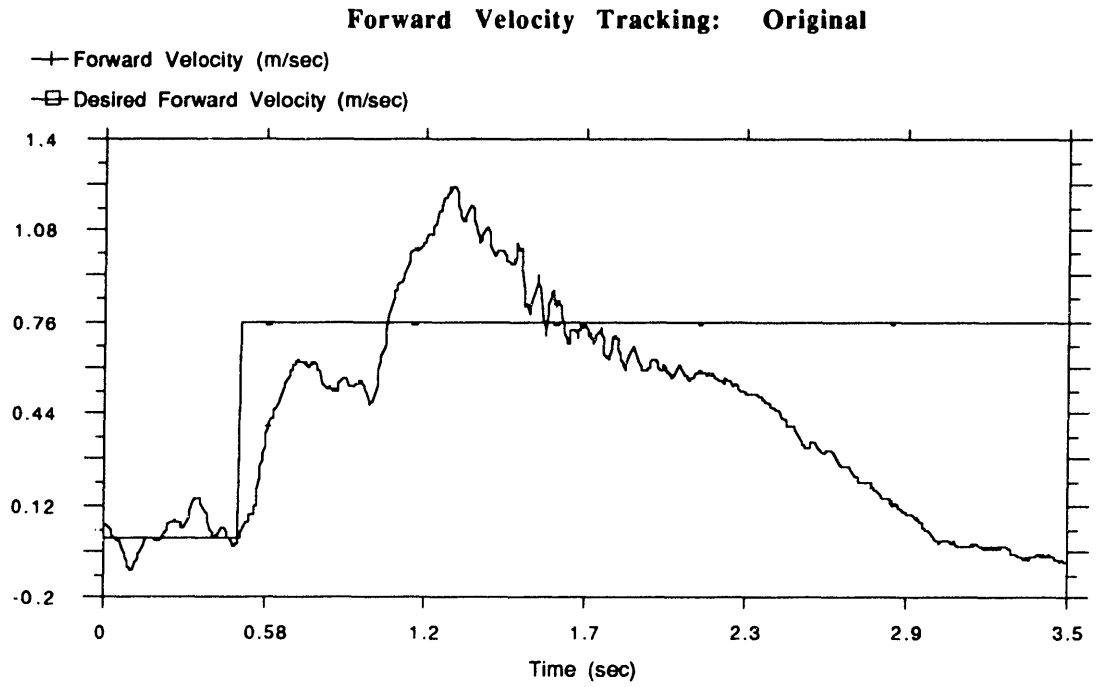
$$V_{x\_diff} = 0.051745\ \theta_{accel} + 0.016216$$

is used to predict the forward acceleration per stance prior to the touchdown. The gains of the linear servo for body attitude control at the maximum forward speed could be increased to about twice the magnitude of the gains when the monopod is running in place. With the improved control system, the monopod could easily run at an average forward speed of 1 m/sec without falling forward or rolling the hoof. Figure 6-17 shows data from runs with the original and the improved control systems. The command for the forward speed and the forward speed correction factor $k_{\dot{x}}$ are such that without these enhancements to the control the monopod would fall or hoof would roll. However, with the improved control system, the monopod did not fall over and the hoof did not roll. Furthermore, the monopod successfully tracked the desired forward velocity and also maintained the body attitude to the desired angle (nominally horizontal) quite successfully.

6.5 Further Improvement: Mechanical Design

Obviously the quantitative values of the control system such as maximum forward velocity, maximum forward acceleration and maximum hip torque are strongly dependent on the mechanical design of the monopod. The issue of the shape of the hoof was briefly discussed previously concerning the rolling of the hoof. And now, I would like to point out two areas for future improvement: choosing an appropriate stiffness for the foot spring and system adjusting the body inertia.

Is a stiff foot spring better or worse? It depends what the goal is. Modelling the monopod as a mass-spring oscillating system, the frequency of the cyclic motion is



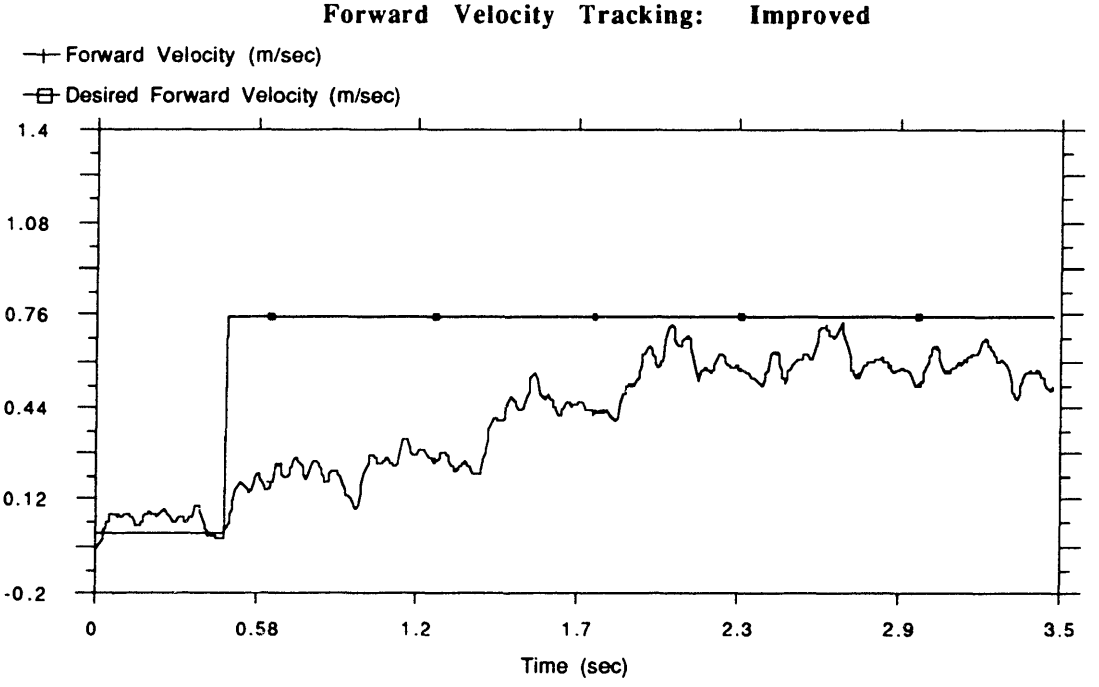


Figure 6-17: Data from runs with the original and the improved control systems. The command for the forward speed and the forward speed correction factor are such that without the enhancements to the control the monopod would fall or hoor would roll. In this case, the monopod with the original control system tried to accelerate too much resulting in failure whereas the monopd with the improved control system successfully tracked the desired forward velocity.

$$\omega = \sqrt{\frac{k}{m}} \tag{6.19}$$

where

ω is the angular frequency of the motion,

m is the mass of the system, and

k is the stiffness of the spring.

A stiffer foot spring increases the speed at liftoff thereby increasing the maximum forward speed of the monopod. In addition, the stiffer foot spring decreases the duration of the stance phase by the square root of the increase in the stiffness of the foot spring. But, what does shorter duration of the stance phase mean? Shorter stance duration means two things; the leg angle sweeps a shorter distance while the foot is on the ground for shorter period allowing less time available for body attitude correction. The smaller leg angles are beneficial for two reasons: first, a smaller horizontal force is induced at the hoof due to the inertial loading, and second, when the leg is swept forward, the disturbance to the pitch angle during the flight phase is smaller. On the other hand, the shorter time for the body attitude correction is not good. Developing some sort of cost function of the foot spring's stiffness with respect to the overall performance would be useful.

Another possible mechanical improvement is a body inertia adjustment system, or simply put, a second leg or a tail. One major problem in controlling the monopod is the disturbances to the body attitude caused by the swing of the leg during flight and the asymmetry in design. Because of this asymmetry, the monopod runs nose down. This nose down body attitude increases the chance that the hoof will roll. There are two possible solutions: the addition of another leg or the addition of a tail, both for the purpose of counter-oscillation. In fact, the kangaroo counter-oscillates the tail to compensate for the swinging of the leg during the flight. Whichever method is selected, such a counter-oscillation system is an absolute necessity for a high performance running system.

Chapter 7
Conclusion and Future Work

The leg is a crucial component in a legged system, especially for running machines, and there are numerous properties of a leg design which would be desirable. A leg should be mechanically simple, light weight, sturdy, have a low unsprung mass and good resilience, have low friction in the mechanism, and should include simple sensors. We have built numerous running machines at the Leg Lab, ranging from a one legged planar hopper to a three dimensional biped. Most of these machines used a telescoping leg with an air spring. However, the telescoping leg has several limitations. It is heavy and bulky, mechanically complex, the seals leak and create high friction, the long linear leg length potentiometers are non standard equipment, and the leg has an large moment of inertia. In addition to these limitations, the fact that animals have articulated legs and run much better than our machnes motivated us to explore the articulated leg design and test it on the monopod.

Although the articulated leg eliminated the limitations present in the telescoping leg, it introduced another problem; the impact of the heel against the ground. This impact disturbed the control of forward speed, and as the result, the monopod could not run in place well. Increasing the angle of the foot avoided the heel impacts, but it increased coupling between the vertical and horizontal motion. An alternative foot design with a hoof was proposed and tested. It successfully eliminated the heel impact against the ground and allowed the foot to be nearly levered, but it brought forth another problem, the rolling of the hoof. The rolling of the hoof was caused largely by the hip torque from the body attitude control and the inertial loading from the acceleration of the monopod during the stance phase. Initially, we tried the method of coordinating the hip torque to the downwards force induced at the hoof. Simulation results showed marginal success in the method of the hip torque coordination, but even in simulation, the excessive pitch disturbance induced by this method caused problems. The pitch disturbance caused this method to fail when it was implemented on the actual machine.

After concluding that the hip torque coordination method was not promising, we moved on to the method of limiting the forward acceleration during the stance phase. We obtained a linear relationship between the deviation of the leg angle from the neutral angle and the net change in the forward velocity over the stance phase. This relationship enabled us to predict the approximate forward acceleration of the monopod. We then established values for the maximum forward speed and maximum net velocity change during the stance phase within which the monopod can perform without falling over or causing the hoof to roll. By operating only within this region, rolling of the hoof and falling over were successfully avoided.

Future Work: Knee

We hope to replace telescoping legs with articulated legs in future running machines. However, because of its inability to retract substantially, the current articulated leg design is not adequate for running machines with more than one leg. With a single leg, the swing phase occurs during flight when the foot is clear off the ground. But for a machine with more than one leg the idle leg must remain clear of the ground while the other leg is in stance---the idle leg must be able to shorten to less than the shortest length seen by the support leg during stance.

One way to obtain a large retraction is to add a knee-like joint. The ankle pivot and tendon could then be eliminated because vertical thrust could be obtained by a combination of knee and hip motions. However, this motion is complicated because of the kinematic coupling between the hip and knee joints---both purely vertical movement and purely horizontal movement require use of the same joints. Therefore, use of the ankle as the primary actuator while using the knee joint mainly for gross changes in leg length during flight seems to be the best choice. The final mechanical design of such a ankle-knee articulated leg has not been considered yet. It will be the next step in the development of the project.

Bibliography

Alexander, R. McN. 1974. The mechanics of jumping by a dog. *J. Zoology (London)* 173:549-573.

Alexander, R. McN., Vernon, A. 1975. The mechanics of hopping by kangaroos (Macropodiadas). *J. Zoology (London)* 177:265-303.

Cavagna, G. A. 1970. Elastic bounce of the body. *J. Applied Physiology* 29:279-282.

Cavagna, G. A., Heglund, N. C., Taylor, C. R. 1977. Mechanical work in terrestrial locomotion: Two basic mechanisms for minimizing energy expenditure. *American J. Physiology* 233:R243-R261.

Dawson, T. J., Taylor, C. R. 1973. Energetic cost of locomotion in kangaroos. *Nature* 246:313-314.

Hildebrand, M. 1960. How animals run. *Scientific American* 148-157.

Hirose, S., Umetani, Y. 1980. The basic motion regulation system for a quadruped walking vehicle. *ASME Conference on Mechanisms.*

Hodgins, J., Koechling, J., Raibert, M. H. 1986. Running experiments with a planar biped. *Third International Symposium on Robotics Research*, G, Giralt, M. Ghallab (eds.). Cambridge: MIT Press.

McMahon, T. A. 1984. *Muscle, Reflexes, and Locomotion.* Princeton: Princeton University Press.

Raibert, M. H., Brown, H. B. Jr. 1984. Experiments in balance with a 2D one-legged hopping machine. *ASME J. Dynamic Systems, Measurement, and Control* 106:75-81.

Raibert, M. H., Chepponis, M., Brown, H. B. Jr. 1986. Running on four legs as though they were one. *IEEE J. Robotics and Automation*, 2.

Song, S. M., Vohnout, V. J., Waldron, K. H., Kinzel, G. L. 1981. Computer-aided design of a leg for an energy efficient walking machine. *Proceedings of 7th Applied Mechanisms Conference*, Kansas City, pp. VII-1-VII-7.

Vohnout, V. J., Alexander, K. S., Kinzel, G. L. 1983. The structural design of the legs for a walking vehicle. *Proceedings of 8th Applied Mechanisms Conference*, St. Louis, pp. 50-1-50-8.

Waldron, K. J., Kinzel, G. L. 1983. The relationship between actuator geometry and mechanical efficiency in robots. In *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy'81*, A. Morecki, G. Bianchi, K. Kedzior (eds.). Warsaw: Polish Scientific Publishers, 305-316.

Waldron, K. J., Vohnout, V. J., Pery, A., McGhee, R. B. 1984. Configuration design of the adaptive suspension vehicle. *International J. Robotics Research* 3:37-48.

Appendix A:
Physical Parameters of the monopod

Parameter	Metric Units	English Units
Lengths/angles		
Overall height	0.84 m	33 in
Overall length	1.02 m	40 in
Overall width	0.23 m	9 in
Hip height (leg fully extended)	0.74 m	29.2 in
Leg vertical travel	0.13 m	5.2 in
Leg sweep angle	±0.73 rad	±42 deg
Masses		
Total mass (body, leg, and coupling)	11.3 kg	25 lbm
Body mass	5.4 kg	11.8 lbm
Leg mass	0.73 kg	1.61 lbm
Leg mass, unsprung	0.063 kg	0.14 lbm
Ratio of total mass to unsprung leg mass	125:1	125:1
Moments of Inertia		
Body moment of inertia (about CG)	0.22 kg-m^2	750 lgm-in^2
Leg moment of inertia (about hip)	0.097 kg-m^2	330 lbm-in^2
Ratio of body to leg moments of inertia	2.3:1	2.3:1
Performance		
Ideal no load stroke time❖	0.023 s	0.023 s
Ideal no load sweep time❖	0.037 s	0.037 s
Static thrust❖	17.3 N	77 lb
Static hip torque❖	67 N-m	590 in-lb
Ratio of static thrust to weight❖	4.5:1	4.5:1
Work per Thrust Stroke❖	49.9 N-m	442 in-lb
Work per Sweep Stroke❖	82 N-m	724 in-lb
Leg spring stiffness	4060 N/m	23 lb/in

❖ Differential hydraulic pressure of 14mPa or 2000 psi.

Appendix B: Kinematics of the monopod

Symbol	Variable Description
ϕ	Angle of the body wrt horizontal (pitch)
θ_1	Angle of foot actuator lever arm wrt leg normal
θ_2	Angle of leg wrt body normal
θ_{leg}	Angle of leg wrt vertical
θ_3	Angle of foot lever arm wrt leg normal
θ_{heel}	Angle of foot lever arm wrt horizontal
θ_{toe}	Angle of line joining ankle and toe wrt horizontal
α_1	Angle of foot actuator wrt leg
α_2	Angle of hip actuator wrt body normal
ω_1	Foot actuator length, pivot to pivot
ω_2	Hip actuator length, pivot to pivot
$d\omega_1$	Foot actuator displacement ($\omega_1 - \omega_{2\_0}$)
$d\omega_2$	Hip actuator displacement (ω_2 - ω_{2_0})
r_1	Effective moment arm of foot actuator about hip
r_2	Effective moment arm of hip actuator about hip
δ_f	Foot deflection perpendicular to foot length
x	Horizontal position of the body CG
z	Height of the body CG about ground
$legx$	Horizontal distance from toe to body CG
$legz$	Vertical distance from toe to body CG

Table B-1: Kinematic variables for monopod

The kinematic configuration fo the monopod is completely determined by four measured variables: ϕ, θ_1, θ_2, and δ_f. Given these four position variables, we want to find θ_{leg}, θ_{heel}, θ_{toe}, $legx$, and $legz$. By inspection of figure B-1, we see at once that

$$\theta_{leg} = \theta_2 + \phi \tag{B.1}$$

From the four-bar linkage defined by a,c,b, and d, we see that

Symbol	Parameter	Metric Units	English Units
a	Foot-actuator lever arm length	0.0381 m	1.500 in
b	Foot lever arm length	0.0439 m	1.73 in
$c + h$	Leg length, hip to ankle	0.610 m	24.0 in
d	Tendon length	0.48 m	19.0 in
e	Hip-to-body-CG offset	0.1651 m	6.500 in
f	Foot length, ankle to toe	0.188 m	7.4 in
g	Body half length	0.51 m	20.0 in
h	Hip to foot-actuator-lever-pivot distance	0.1016 m	4.00 in
i	Hip actuator lever length	0.0381 m	1.500 in
k	Hip to hip-actuator-pivot distance	0.1091 m	4.295 in
ω_{1_o}	Foot actuator length at center of travel	0.1016 m	4.000 in
ω_{2_o}	Hip actuator length at center of travel	0.1022 m	4.025 in
θ_{3_o}	Value of θ_3 when $\theta_1 = 0$	0.19 rad	11 deg
θ_{toe_o}	Value of θ_{toe} when $\theta_{heel} = 0$ and $\delta_f = 0$	0.33 rad	19 deg
α_{2_o}	Value of α_2 when $\theta_2 = 0$	1.215 rad	69.6 deg

Table B-2: Kinematic constants of the monopod

$$c + a \sin \theta_1 = d + b \sin \theta_3, \tag{B.2}$$

$$\sin \theta_3 = \frac{c - d}{b} + \frac{a}{b} \sin \theta_1 \tag{B.3}$$

$$= \sin \theta_{3\_0} + \frac{a}{b} \sin \theta_1, \tag{B.4}$$

where $\sin \theta_{3\_0} = (c - d)/b$ is the value of $\sin \theta_3$ when $\theta_1 = 0$. Therefore,

$$\theta_3 = \arcsin(\sin \theta_{3\_0} + \frac{a}{b} \sin \theta_1). \tag{B.5}$$

Also by inspection

$$\theta_{heel} = \theta_{leg} + \theta_3. \tag{B.6}$$

If we assume that foot length f is constant, and foot deflection δ_f is measured along an arc about the ankle, then we can say

$$\theta_{toe} + \frac{\delta_f}{f} \approx \theta_{heel} + \theta_{toe\_0} \tag{B.7}$$

$$= \theta_{leg} + \theta_3 + \theta_{toe\_o}. \tag{B.8}$$

Therefore

$$\theta_{toe} \approx \theta_{leg} + \theta_3 + \theta_{toe\_o} - \frac{\delta_f}{f}. \tag{B.9}$$

We find the horizontal and vertical distances from toe to hip by adding components of each of the three link lengths:

$$legx = -f \cos \theta_{toe} + (c + h) \sin \theta_{leg} + e \cos \phi, \tag{B.10}$$

$$legz = f \sin \theta_{toe} + (c + h) \cos \theta_{leg} - e \sin \phi. \tag{B.11}$$

Actuator Kinematics

Given angles q1 and q2, we want to find the actuator lengths and displacements w1, w2 and dw1, dw2, and moment arms r1 and r2. With reference to figure B-2, we add length components parallel to the leg for actuator 1 to get

$$\omega_1 \cos \alpha_1 + a \sin \theta_2 + a \sin \theta_1 = h. \tag{B.12}$$

Therefore

$$\omega_1 = \frac{h - a (\sin \theta_1 + \sin \theta_2)}{\cos \alpha_1}. \tag{B.13}$$

By considering length components perpendicular to the leg we get

$$a \cos \theta_1 = \omega_1 \sin \alpha_1 + a \cos \theta_2 \tag{B.14}$$

or

$$\omega_1 = \frac{a (\cos \theta_1 - \cos \theta_2)}{\sin \alpha_1}. \tag{B.15}$$

We combine this with (B.13) to get

$$\frac{h - a (\sin \theta_1 + \sin \theta_2)}{\cos \alpha_1} = \frac{a (\cos \theta_1 - \cos \theta_2)}{\sin \alpha_1}. \tag{B.16}$$

or

$$\tan \alpha_1 = \frac{\cos \theta_1 - \cos \theta_2}{\frac{h}{a} - \sin \theta_1 - \sin \theta_2} \approx \alpha_1 \tag{B.17}$$

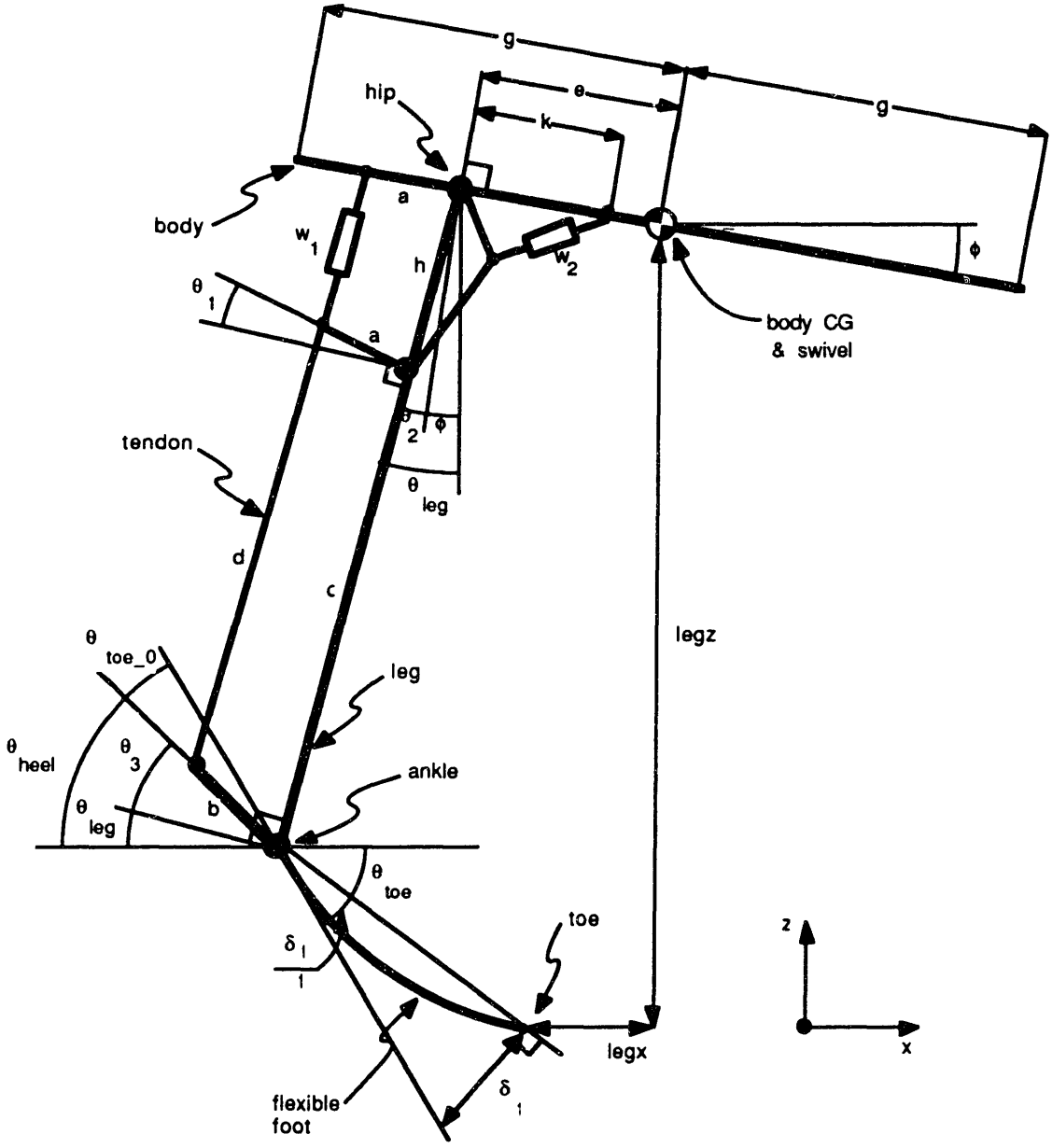


Figure B-1: monopod kinematics. The figure shows the overall configuration of the monopod with labels giving some of the nomenclature. Symbols are shown for the kinematic variables and parameters, which are further defined in tables B-1 and B-2.

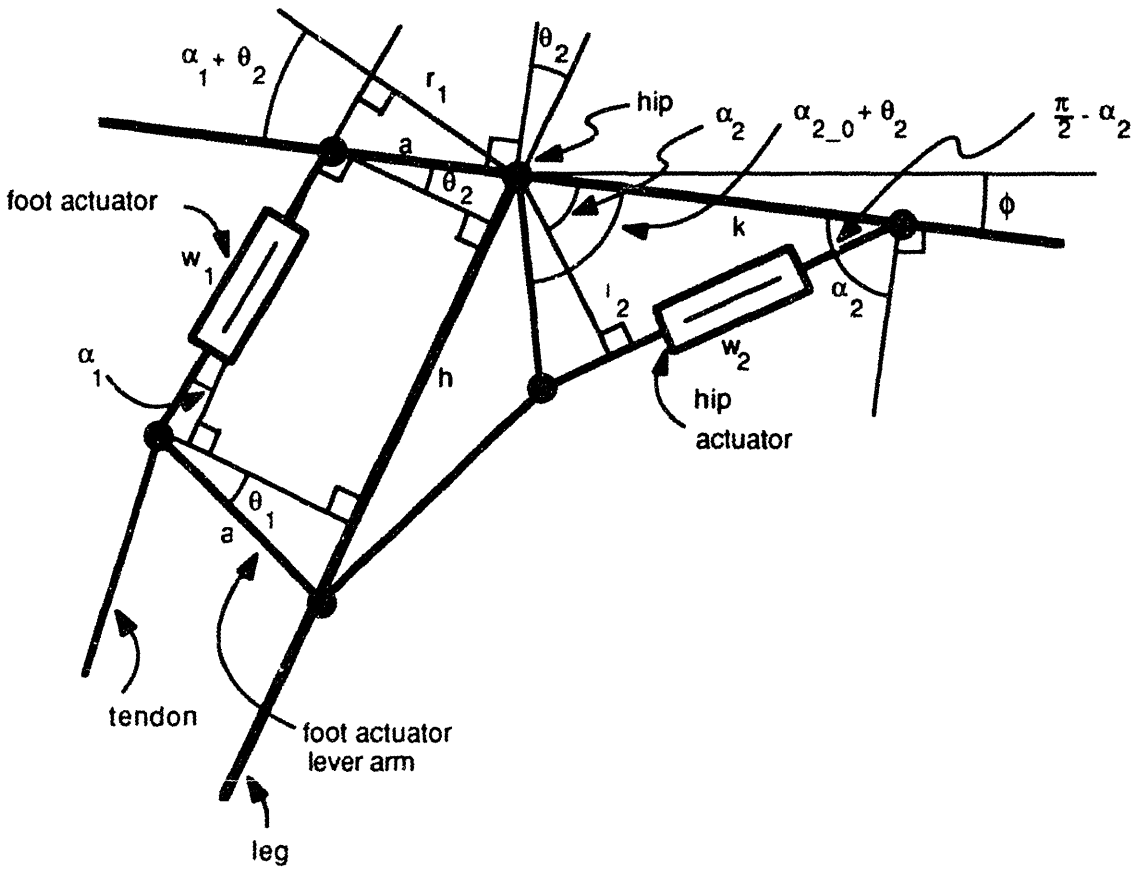


Figure B-2: Actuator kinematics. The figure shows the geometry of the hip and actuator linkages.

The maximum possible value of a is 0.12 rad (6.9 deg), so $\tan \alpha_1 \approx \alpha_1$ with less than 0.5% error. Also $\cos \alpha_1 \approx 1$ within 1%, so (B.13) becomes

$$\omega_1 \approx h - a(\sin \theta_1 + \sin \theta_2). \tag{B.18}$$

By design, $\omega_{1\_o} = h$. Therefore

$$d\omega_1 = \omega_1 - \omega_{1\_o} = \omega_1 - h \approx -a(\sin \theta_1 + \sin \theta_2). \tag{B.19}$$

From figure B-2 we see that

$$r_1 = a \cos(\alpha_1 + \theta_2). \tag{B.20}$$

For actuator 2, we apply the law of cosines to the triangle formed by i, k and ω_2:

$$\omega_2^2 = i^2 + k^2 - 2\, i\, k \cos(\alpha_{2\_o} + \theta_2). \tag{B.21}$$

We see also that

$$r_2 = i \cos(\alpha_{2\_o} + \theta_2 - \alpha_2) \approx i \cos\theta_2 \qquad \text{(B.22)}$$

for $\alpha_2 \approx \alpha_{2\_o}$. The worst error in r_2 due to this assumption is 14% which occurs only at the limit of hip travel. If necessary, we can find α_2 and use the exact form of (B.22). Law of cosines gives

$$i^2 = k^2 + \omega_2^2 - k\,\omega_2 \cos(\frac{\pi}{2} - \alpha_2) \qquad \text{(B.23)}$$

from which

$$\alpha_2 = \frac{\pi}{2} - \arccos(\frac{k^2 + \omega_2^2 - i^2}{k\,\omega_2}). \qquad \text{(B.24)}$$

Appendix C: Planarizer

We have designed a mechanism that will permit the monopod to run on a treadmill, while constraining its motion to the plane. The main function of the planarizer is to restrict motion of the monopod in all degrees of freedom, except those in the sagittal plane---fore and aft, up and down, and pitch rotation. In fact, the function of the planarizer is same as of the tether boom used to constrain the planar biped described in earlier chapters. However, the tether boom turns the machine as the machines moves fore and aft. If the monopod were to travel forward on the treadmill while tethered by a boom, the legs would no longer sweep in the direction of the moving belt.

The planarizer should be rigid enough to eliminate non-planar motions, and should have minimum influence on the dynamics of the monopod. Low friction is thus very desirable. It is also important that the size of the planarizer permits it to be mounted on the treadmill. The major criteria for design of the planarizer are that it

- Allows motions within the sagittal plane, while preventing motions out of the plane.
- Does not disturb dynamics.
- Is of adequate size.

We have considered several designs, some of which are shown in figure C-1---the arm structure, the X-Y table, and the linear-sliding boom. However, none of these designs satisfied all the criteria mentioned above. For example, the arm structure is vulnerable to the side thrust applied by the robot due to its long arm, and requires a bulky and heavy arm in order to be rigid enough. Furthermore, the distribution of inertia is even only over a limited range of orientations, and otherwise it disturbs the dynamics of the robot. The X-Y table satisfies the first and third criteria, but it does not satisfy the second criterion because of its uneven distribution of mass. The linear-sliding boom meets all the criteria. However, the arm has to be relatively long, and the space available was the limiting factor.

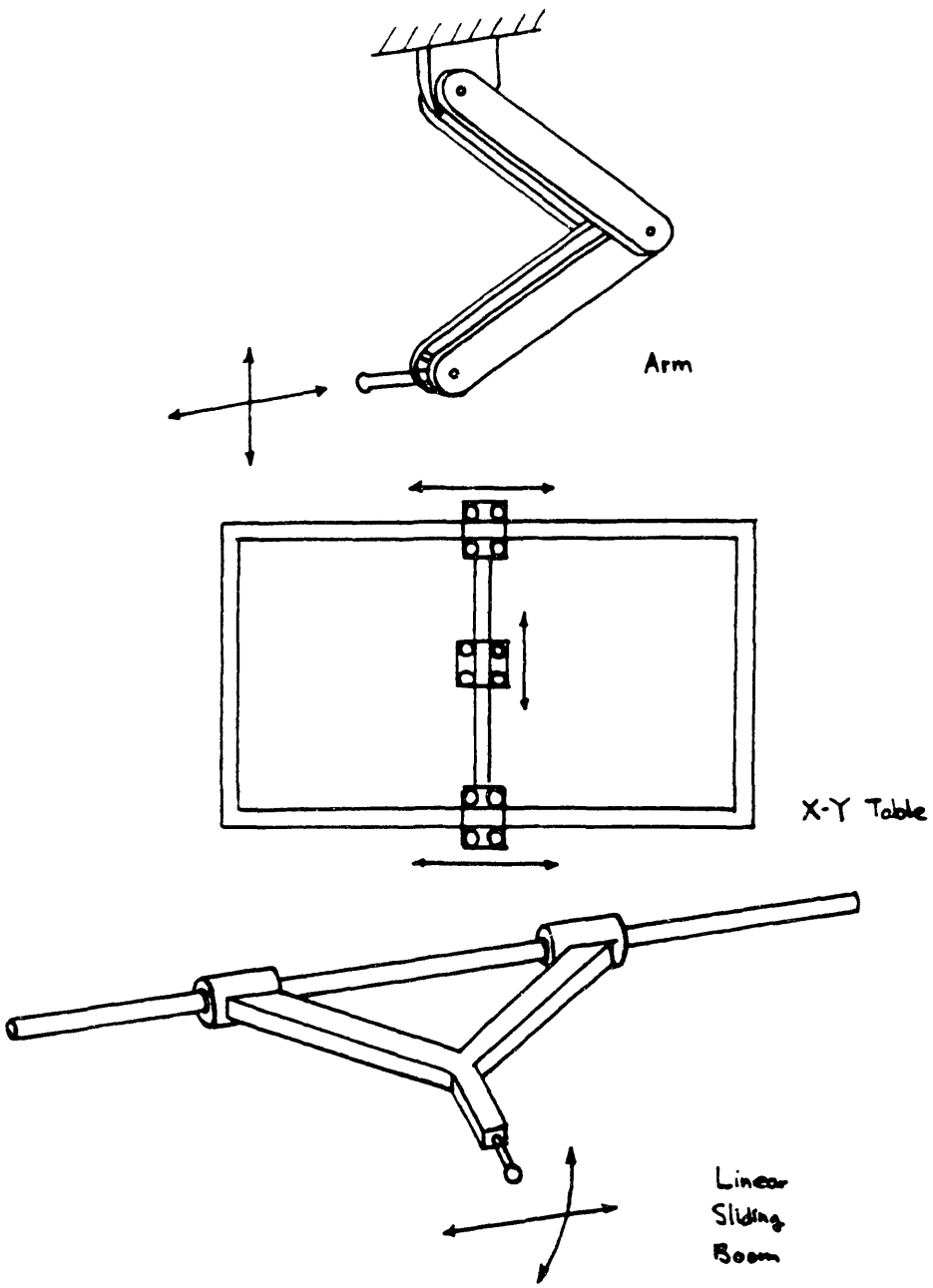


Figure C-1: Three designs of planarizer: A) Arm structure: It is vulnerable to the side thrust applied by the robot due to its long arm, and requires a bulky and heavy arm in order to be rigid enough. The distribution of inertia is even only over a limited range of orientations, and otherwise it disturbs the dynamics of the robot. B) X-Y table: The distribution of inertia is uneven. C) Linear-sliding boom: The arm has to be relatively long, and the space available is thus the limiting factor.

The schematic of the design we have selected is shown in figure C-2. This mechanism operates in Cartesian coordinates. It consists of two vertical linear slides and rails, two horizontal linear slides and rails, pulleys, and cables. The running machine is mounted on the vertical linear slide #1, which rides on the vertical rail #1. The vertical rail #1, in turn, is mounted on the two horizontal linear slides as shown, whereas the vertical rail #2 is stationary. The force transmitting elements are the pulleys and cables. They are mounted in such a way that the vertical movement of vertical slide #1 will cause exactly the same movement in vertical slide #2, but the horizontal movement will not affect the vertical slide #2. Therefore the inertia of the machine's vertical movement can be adjusted to be the same as the inertia of horizontal movement by adding weights on the vertical slide #2. The major drawback is friction in the cables and pulleys.

The design permits measurement of location of the machine by measuring the rotation of the wheels riding on the horizontal and vertical slides using rotary digital encoders. The motion controller chip HCTL-1000 of Hewlett-Packard is used as the decoder and counter for the quadrature signal from the digital encoder. The interface to the UNIBUS of the VAX computer is designed and implemented. The schematics of the interface and the bus timing circuits are shown in figure C-3. In addition, a special coupling mechanism was designed for mounting the monopod and also for supplying the hydraulic power to the monopod. Figure C-4 shows the engineering drawing of the coupling. The coupling is consisted of three parts; the housing, the outer tube and the inner tube. The outer tube and the inner tube are a pressfit, forming a single piece which connects to the planarizer whereas the housing connects to the monopod. The high pressure supply hydraulic oil flows through the inner tube to the monopod, and the low pressure return hydraulic oil flows through the gap between the inner tube and the outer tube, back to the pump. The housing mounts on the outer tube with rotary bearing inbetween so that the monopod is free to rotate with respect to the tubes which are fixed to the vertical slide of the planarizer. The rotary digital encoder is mounted on the outer tube, measuring the rotational position of the housing, which in same as the position of the monopod, with respect to the outer tube.

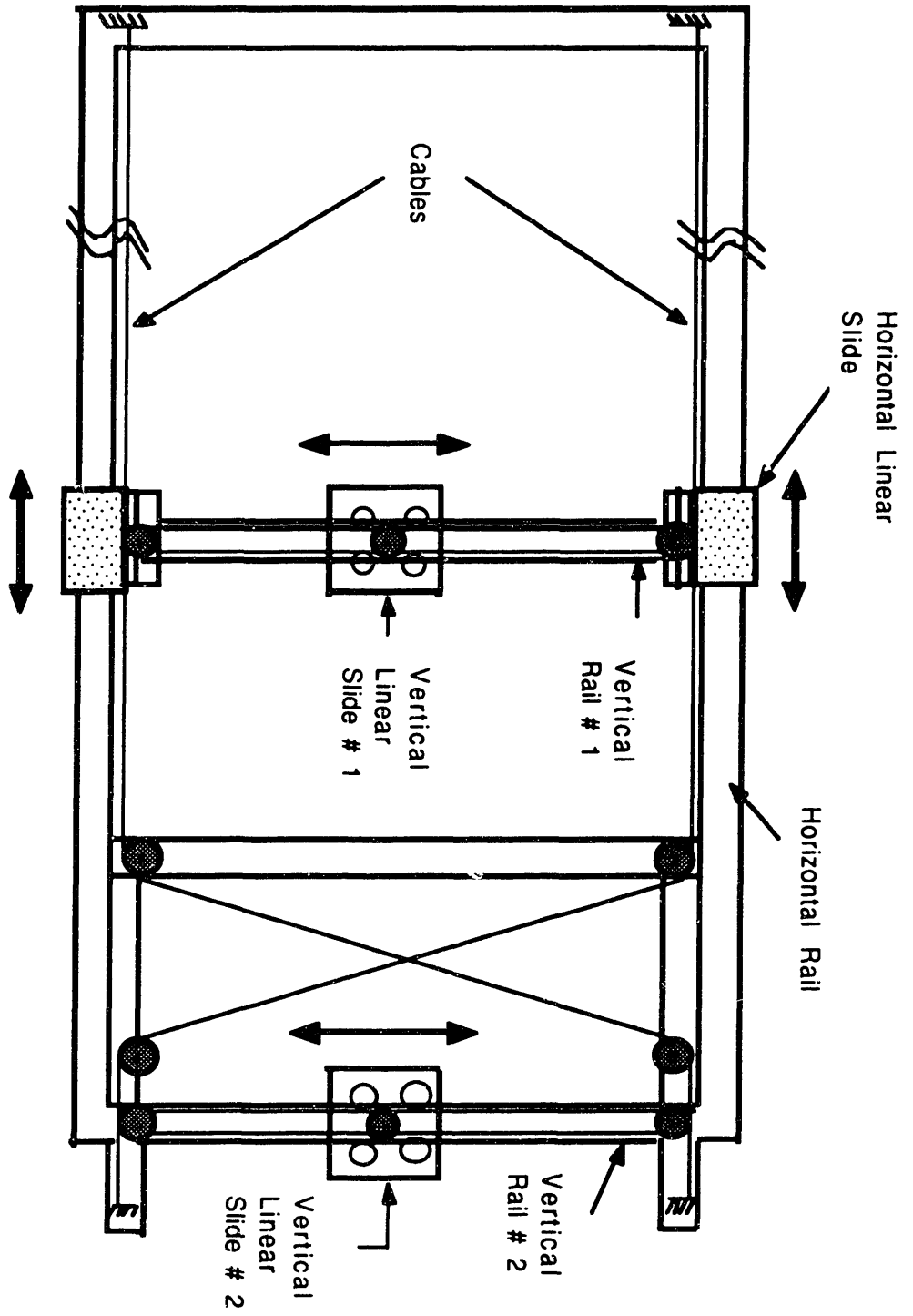
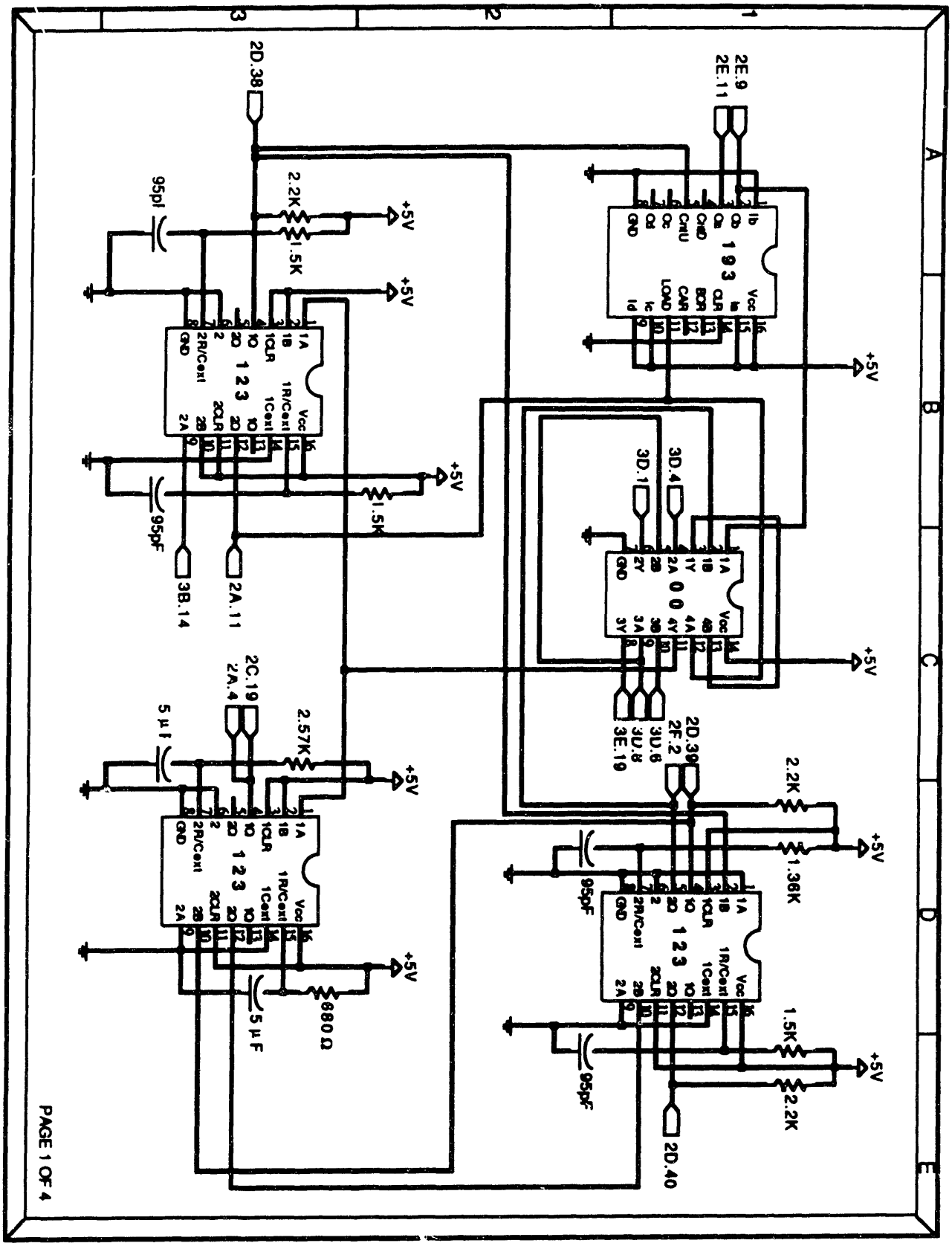
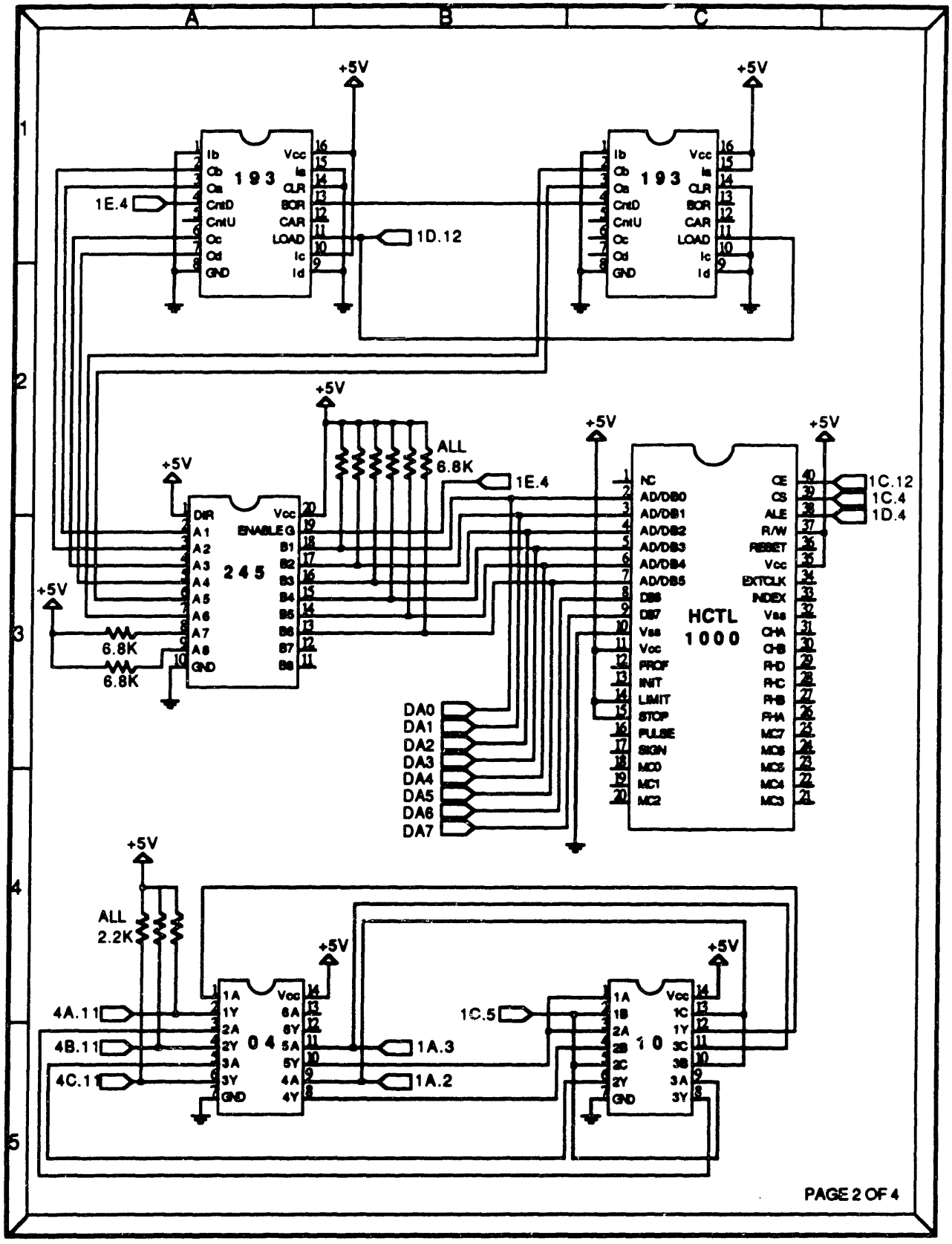
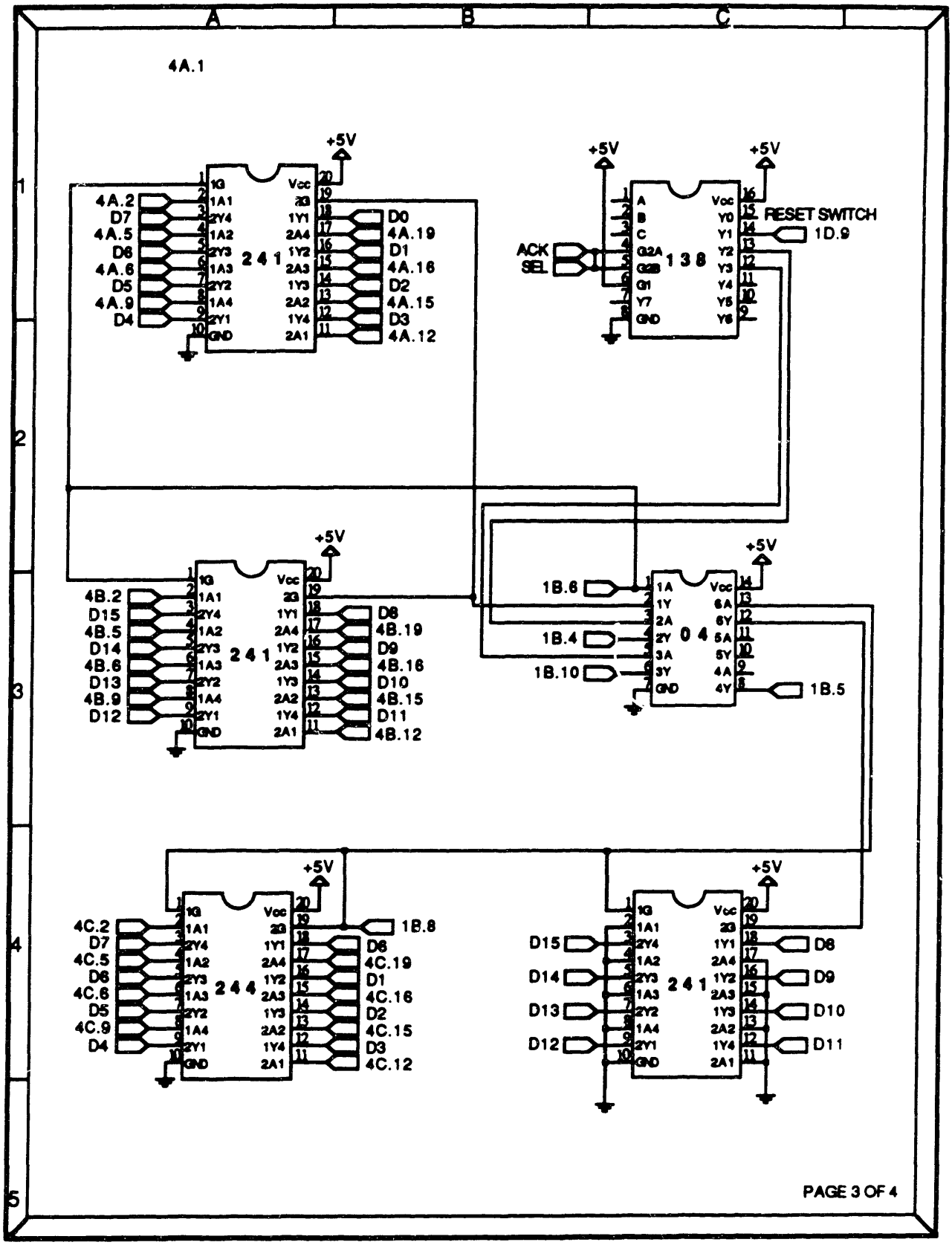


Figure C-2: Schematic of planarizer. The inertia of the machine's vertical movement can be adjusted to be the same as the inertia of horizontal movement by adding weights to the vertical slide #2. The major drawback is friction in the cables and pulleys.


2E.9
2E.11
2D.38
95pf
2.2K
1.5K
+5V
3D.1
3D.4
3B.14
2A.11
2C.19
2A.4
5 µ f
2.57K
880 Ω
5 µ F
2D.39
2F.2
3U.6
3U.8
3E.19
1.36K
1.5K
2D.40
95pF
PAGE 1 OF 4


A
B
C
1
2
3
4
5
+5V
193
1E.4
1D.12
245
ALL
6.8K
HCTL
1000
1C.12
1C.4
1D.4
DA0
DA1
DA2
DA3
DA4
DA5
DA6
DA7
ALL
2.2K
04
10
4A.11
4B.11
4C.11
1A.3
1A.2
1C.5
PAGE 2 OF 4


A
B
C
1
2
3
4
5
4A.1
+5V
241
4A.2
D7
4A.5
D6
4A.6
D5
4A.9
D4
D0
4A.19
D1
4A.16
D2
4A.15
D3
4A.12
138
ACK
SEL
RESET SWITCH
1D.9
241
4B.2
D15
4B.5
D14
4B.6
D13
4B.9
D12
D8
4B.19
D9
4B.16
D10
4B.15
D11
4B.12
04
1B.6
1B.4
1B.10
1B.5
244
4C.2
D7
4C.5
D6
4C.6
D5
4C.9
D4
1B.8
D8
4C.19
D1
4C.16
D2
4C.15
D3
4C.12
241
D15
D14
D13
D12
D8
D9
D10
D11
PAGE 3 OF 4

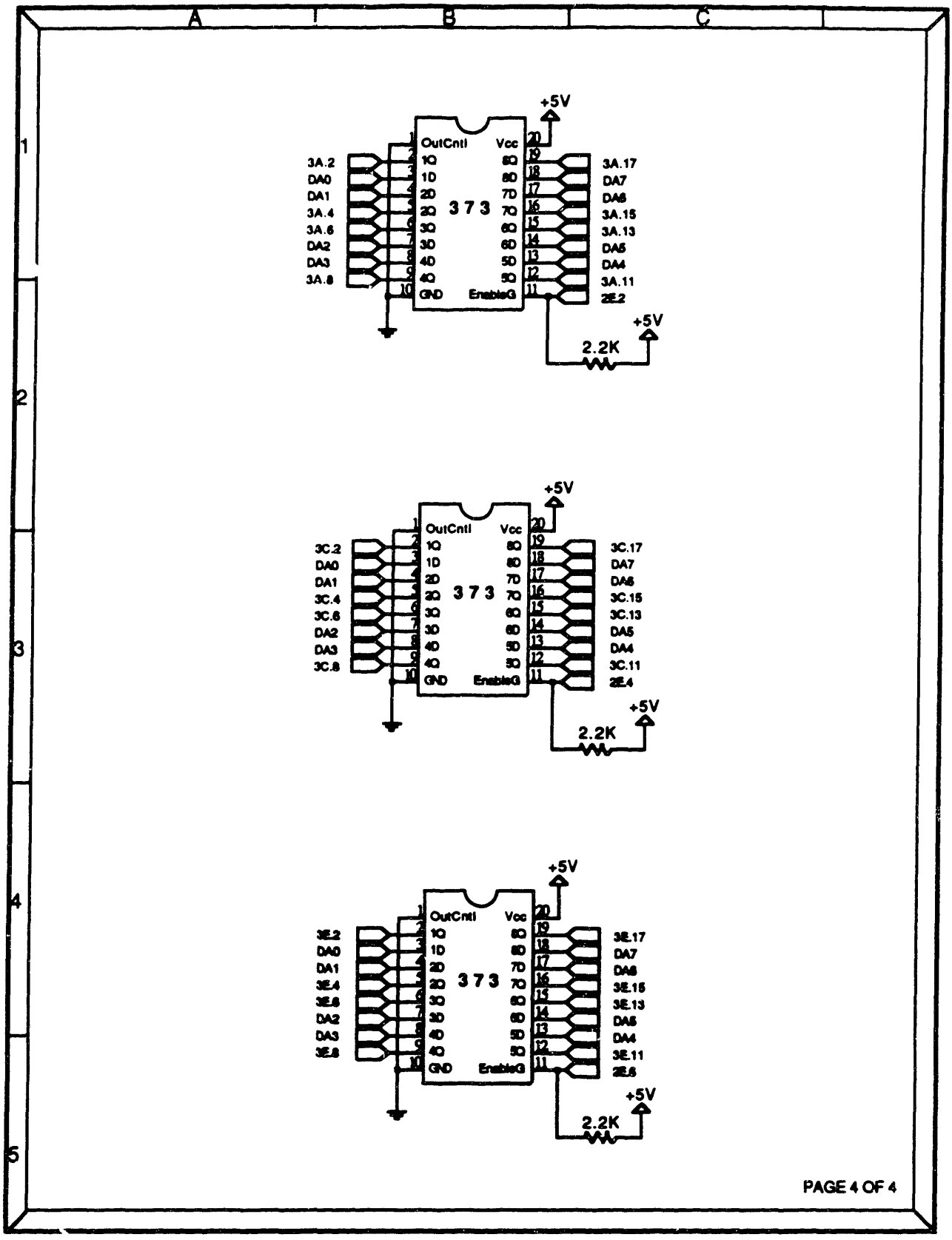
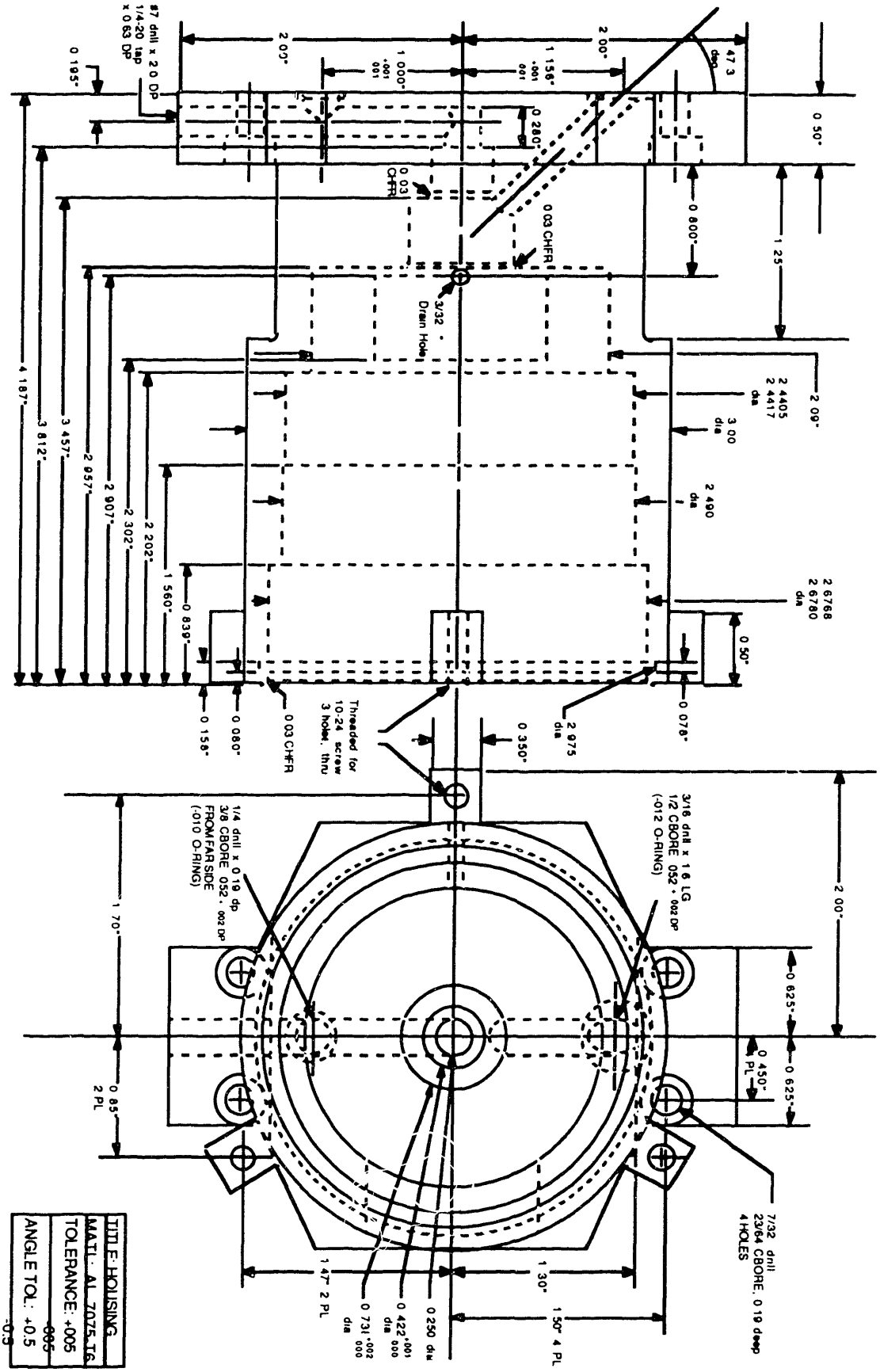
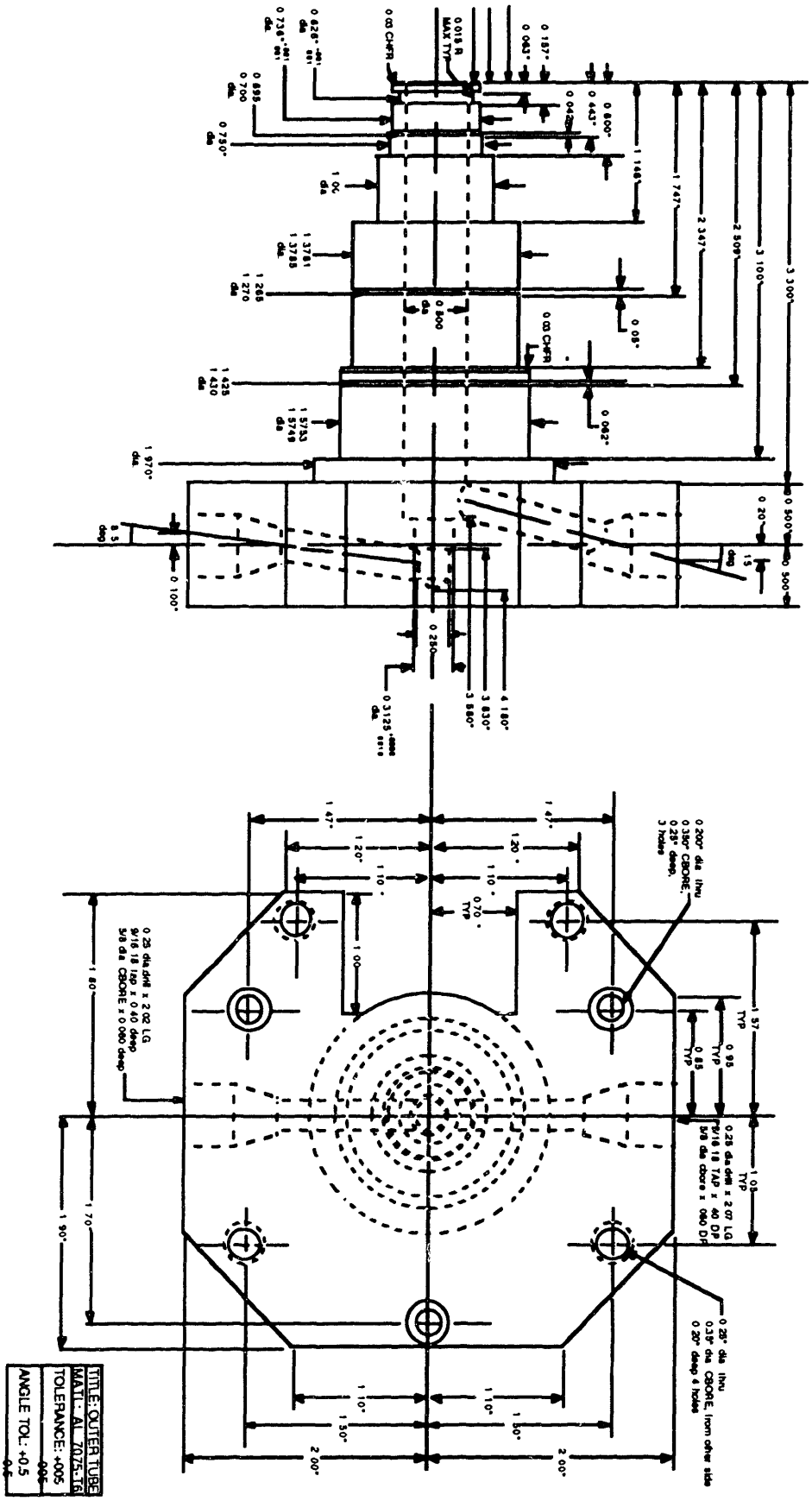


Figure C-3: Schematic of the bus timing and interface circuits for supporting the HCTL-1000 encoder's quadrature signal decoding and counting chip to the UNIBUS of the VAX computer.


#7 drill x 2 0 DP
1/4-20 tap
x 0 63 DP
0 195"
2 00"
1 158"
1 000"
47 3 deg
0 50"
0 280"
0 03 CHFR
0 600"
1 25
3/32" Drain Hole
4 187"
3 812"
3 457"
2 957"
2 907"
2 302"
2 202"
1 560"
0 839"
0 158"
0 080"
0 03 CHFR
Threaded for 10-24 screw 3 holes, thru
2 975 dia
0 350"
0 078"
0 50"
2 6768
2 6780 dia
2 490 dia
3 00 dia
2 4405
2 4417 dia
2 09"
3/16 drill x 1 6 LG 1/2 CBORE 052" +002 DP (-012 O-RING)
1/4 drill x 0 19 dp 3/8 CBORE 052" +002 DP FROM FAR SIDE (-010 O-RING)
1 70"
2 00"
0 625"
0 450" PL
0 625"
0 85" 2 PL
7/32 drill 23/64 CBORE, 0 19 deep 4 HOLES
1 47" 2 PL
1 30"
1 50" 4 PL
0 250 dia
0 422" +001 -000 dia
0 731" +002 -000 dia
TITLE: HOUSING
MATL: AL 7075-T6
TOLERANCE: +005 -005
ANGLE TOL: +0.5 -0.5


TITLE: OUTER TUBE
MATL: AL 7075-T6
TOLERANCE: ±005
ANGLE TOL: ±0.5

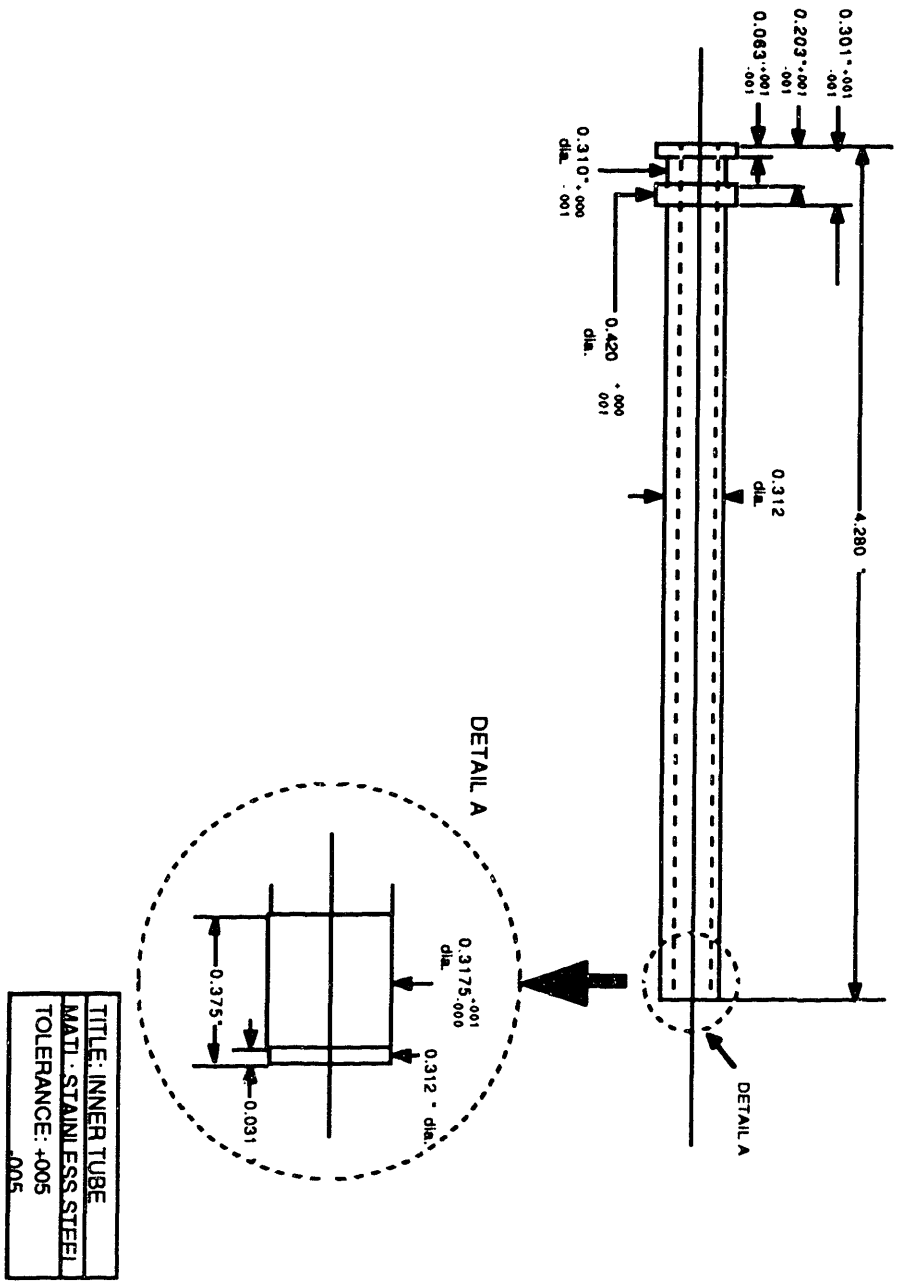


Figure C-4: Engineering drawings of the parts for the coupling between the monopod and the planarizer The coupling consists of three parts: the housing, the outer tube and the inner tube. The outer tube and the inner tube are a pressfit, forming a single piece which connects to the planarizer wheras the housing connects to the monopod. The high pressure supply hydraulic oil flows through the inner tube to the monopod, and the low pressure return hydraulic oil flows through the gap between the inner tube and the outer tube, back to the pump. The housing mounts on the outer tube with rotary bearing inbetween so that the monopod is free to rotate with respect to the tubes which are fixed to the vertical slide of the planarizer. The rotary optical encoder disk is mounted on the outer tube, and the rotational position of the housing with respect to the outer tube is read as the rotary position of the monopod.

Appendix D:
Derivation of Equations of Motion of the monopod

With the model of the monopod discussed in chapter 5, the equations of motion of the monopod are derived using the Lagrangian dynamics. The generalized coordinates chosen are horizontal and vertical positions of the center of the body X_{body}, Y_{body}, the angle of the body Θ_{body}, the angle of the leg Θ_{leg}, the angle of the foot Θ_{foot} and the angle of the hoof Θ_{hoof}. The kinetic coenergy T^*, the potential energy L, and the virtual work δW are

$$\begin{aligned} T^* = & \frac{1}{2} M_{body} (\dot{X}_{body}{}^2 + \dot{Y}_{body}{}^2) + \frac{1}{2} I_{body} \dot{\theta}_{body}{}^2 + \frac{1}{2} M_{leg} (\dot{X}_{leg}{}^2 + \dot{Y}_{leg}{}^2) \\ & + \frac{1}{2} I_{leg} \dot{\theta}_{leg}{}^2 + \frac{1}{2} M_{foot} (\dot{X}_{foot}{}^2 + \dot{Y}_{foot}{}^2) + \frac{1}{2} I_{foot} \dot{\theta}^2 \\ & + \frac{1}{2} M_{hoof} (\dot{X}_{hoof}{}^2 + \dot{Y}_{hoof}{}^2) + \frac{1}{2} I_{hoof} \dot{\theta}_{hoof}{}^2 \end{aligned} \tag{D.1}$$

$$\begin{aligned} V = & (M_{body} Y_{body} + M_{leg} Y_{leg} + M_{foot} Y_{foot} + M_{hoof} Y_{hoof}) G \\ & + \frac{1}{2} K_{ankle} (\theta_{foot} + \theta_{leg} - \theta_{ankleo})^2 + \frac{1}{2} K_{toe} [\theta_{hoof} + \theta_{foot} - \theta_{toeo}]^2 \\ & + \frac{1}{2} K_{g1} [(X_{g1} - Xg_{1\_td})^2 + Y_{g1}{}^2] + \frac{1}{2} K_{g2} [(X_{g2} - X_{g2\_td})^2 + Y_{g2}{}^2] \end{aligned} \tag{D.2}$$

$$\begin{aligned} \delta W = & -B_{g1} \dot{X}_{g1} \delta X_{g1} - B_{g2} \dot{X}_{g2} \delta X_{g2} - B_{g1} \dot{Y}_{g1} \delta Y_{g1} - B_{g2} \dot{Y}_{g2} \delta Y_{g2} \\ & - B_{ankle} (\dot{\theta}_{foot} + \dot{\theta}_{leg}) (\delta\theta_{foot} + \delta\theta_{leg}) - B_{toe} (\dot{\theta}_{hoof} + \dot{\theta}_{foot}) (\delta\theta_{hoof} + \delta\theta_{foot}) \\ & + \tau_{hip} \delta\theta_{leg} - \tau_{hip} \delta\theta_{body} \end{aligned} \tag{D.3}$$

where

M_{body} is the mass of the body,
M_{leg} is the mass of the leg,
M_{foot} is the mass of the foot,
M_{hoof} is the mass of the hoof,
I_{body} is the moment of inertia of the body,
I_{leg} is the moment of inertia of the leg,

I_{foot} is the moment of inertia of the foot,
I_{hoof} is the moment of inertia of the hoof,
X_{leg} is the horizontal position of the center of mass of the leg,
X_{foot} is the horizontal position of the center of mass of the foot,
X_{hoof} is the horizontal position of the center of mass of the hoof,
X_{g1} is the horizontal position of the hoof's contact point 1,
X_{g1_td} is the horizontal position of the hoof's contact point 1 at touchdown,
X_{g2} is the horizontal position of the hoof's contact point 2,
X_{g2_td} is the horizontal position of the hoof's contact point 2 at touchdown,
Y_{leg} is the vertical position of the center of mass of the leg,
Y_{foot} is the vertical position of the center of mass of the foot,
Y_{hoof} is the vertical position of the center of mass of the foot,
δ_{Xg1} is the horizontal virtual displacement of hoof's contact point 1,
δ_{Xg2} is the horizontal virtual displacement of hoof's contact point 2,
δ_{Yg1} is the vertical virtual displacement of the hoof's contact point 1,
δ_{Yg2} is the vertical virtual displacement of the hoof's contact point 2,
$\delta\theta_{body}$ is the angular virtual displacement of the body,
$\delta\theta_{leg}$ is the angular virtual displacement of the leg,
$\delta\theta_{foot}$ is the angular virtual displacement of the foot,
$\delta\theta_{hoof}$ is the angular virtual displacement of the hoof,
θ_{ankleo} is the setpoint angle for the ankle spring, and
θ_{toeo} is the setpoint angle for the toe spring.

The variables that are not the chosen generalized coordinates are substituted by the kinematic relations

$$\begin{aligned} X_{hip} &= X_{body} - E\cos\theta_{body} \\ Y_{hip} &= Y_{body} - E\sin\theta_{body} \end{aligned} \tag{D.4}$$

$$\begin{aligned} X_{leg} &= X_{hip} + \frac{L_{leg}}{2}\sin\theta_{leg} \\ Y_{leg} &= Y_{hip} - \frac{L_{leg}}{2}\cos\theta_{leg} \end{aligned} \tag{D.5}$$

$$\begin{aligned} X_{ankle} &= X_{hip} + L_{leg}\sin\theta_{leg} \\ Y_{ankle} &= Y_{hip} - L_{leg}\cos\theta_{leg} \end{aligned} \tag{D.6}$$

$$X_{foot} = X_{ankle} + \frac{L_{foot}}{2} \cos \theta_{foot}$$
$$Y_{foot} = Y_{ankle} - \frac{L_{foot}}{2} \sin \theta_{foot} \quad \text{(D.7)}$$

$$X_{toe} = X_{ankle} + L_{foot} \cos \theta_{foot}$$
$$Y_{toe} = Y_{ankle} - L_{foot} \sin \theta_{foot} \quad \text{(D.8)}$$

$$X_{hoof} = X_{toe} + \frac{L_{hoof}}{2} \sin \theta_{hoof}$$
$$Y_{hoof} = Y_{toe} - \frac{L_{hoof}}{2} \cos \theta_{hoof} \quad \text{(D.9)}$$

$$X_{g1} = X_{toe} + L_{hoof} \sin \theta_{hoof} - \frac{W_{hoof}}{2} \cos \theta_{hoof}$$
$$Y_{g1} = Y_{toe} - L_{hoof} \cos \theta_{hoof} - \frac{W_{hoof}}{2} \sin \theta_{hoof} \quad \text{(D.10)}$$

$$X_{g2} = X_{toe} + L_{hoof} \sin \theta_{hoof} + \frac{W_{hoof}}{2} \cos \theta_{hoof}$$
$$Y_{g2} = Y_{toe} - L_{hoof} \cos \theta_{hoof} + \frac{W_{hoof}}{2} \sin \theta_{hoof} \quad \text{(D.11)}$$

where

X_{hip} is the horizontal position of the hip joint,
Y_{hip} is the vertical position of the hip joint,
X_{ankle} is the horizontal position of the ankle joint,
Y_{ankle} is the vertical position of the ankle joint,
X_{toe} is the horizontal position of the toe joint (hoof's hinge),
Y_{toe} is the vertical position of the toe joint (hoof's hinge),
E is the distance between the center of the mass and the hip joint,
L_{leg} is the length of the leg,
L_{foot} is the length of the foot,
L_{hoof} is the length of the hoof, and
W_{hoof} is the width of the hoof.

The equations of the motion of the monopod are then derived from

$$\frac{d}{dt}\left(\frac{\partial \alpha}{\partial \dot{\xi}}\right) - \frac{\partial \alpha}{\partial \xi} = F_{\xi} \quad \text{(D.12)}$$

where

α is the Lagrangian term, $T^* - V$,

ζ is the generalized coordinate, and

F_ξ is the generalized force of the generalized coordinate ζ.

The generalized forces F_ξ's are related to the generalized work by

$$\delta W = F_{\xi_1}\,\delta\xi_1 + F_{\xi_2}\,\delta\xi_2 + ... + F_{\xi_n}\,\delta\xi_n \tag{D.13}$$

Most of the algebraic manipulations were performed using MATHEMATICA✣, and the resulting equations of the motion are

$$\frac{d^2}{dt^2}\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & A_{56} \\ A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} \end{bmatrix} \begin{Bmatrix} X_{body} \\ Y_{body} \\ \theta_{body} \\ \theta_{leg} \\ \theta_{foot} \\ \theta_{hoof} \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{Bmatrix} \tag{D.14}$$

and the coefficients are

$A_{11} = M_{body} + M_{foot} + M_{hoof} + M_{leg}$
$A_{12} = 0.0$
$A_{13} = E\,sin(\theta_{body})\,(M_{foot} + M_{hoof} + M_{leg})$
$A_{14} = L_{leg}\,cos(\theta_{leg})\,(M_{foot} + M_{hoof} + M_{leg}/2)$
$A_{15} = -L_{foot}\,sin(\theta_{foot})\,(M_{foot}/2 + M_{hoof})$
$A_{16} = L_{hoof}\,M_{hoof}\,cos(\theta_{hoof})/2$

$A_{21} = 0.0$
$A_{22} = M_{body} + M_{foot} + M_{hoof} + M_{leg}$
$A_{23} = -E\,cos(\theta_{body})\,(M_{foot} + M_{hoof} + M_{leg})$
$A_{24} = L_{leg}\,sin(\theta_{leg})\,(M_{foot} + M_{hoof} + M_{leg}/2)$
$A_{25} = -L_{foot}\,cos(\theta_{foot})\,(M_{foot}/2 + M_{hoof})$
$A_{26} = L_{hoof}\,M_{hoof}\,sin(\theta_{hoof})/2$

$A_{31} = E\,sin(\theta_{body})\,(M_{foot} + M_{hoof} + M_{leg})$

✣ MATHEMATICA is a trademark of Wolfram Research Inc.

$$A_{32} = -E\cos(\theta_{body})\,(M_{foot} + M_{hoof} + M_{leg})$$
$$A_{33} = I_{body} + E^2\,(M_{foot} + M_{hoof} + M_{leg})$$
$$A_{34} = E\,L_{leg}\,(M_{foot} + M_{hoof} + M_{leg}/2)\sin(\theta_{body} - \theta_{leg})$$
$$A_{35} = E\,L_{foot}\,(M_{foot}/2 + M_{hoof})\cos(\theta_{body} + \theta_{foot})$$
$$A_{36} = E\,L_{hoof}\,(M_{hoof}/2)\sin(\theta_{body} - \theta_{hoof})$$

$$A_{41} = L_{leg}\cos(\theta_{leg})\,(M_{foot} + M_{hoof} + M_{leg}/2)$$
$$A_{42} = L_{leg}\sin(\theta_{leg})\,(M_{foot} + M_{hoof} + M_{leg}/2)$$
$$A_{43} = E\,L_{leg}\,(M_{foot} + M_{hoof} + M_{leg}/2)\sin(\theta_{body} - \theta_{leg})$$
$$A_{44} = I_{leg} + L_{leg}^2\,(M_{foot} + M_{hoof} + M_{leg}/4)$$
$$A_{45} = -L_{foot}\,L_{leg}\,(M_{foot}/2 + M_{hoof})\sin(\theta_{leg} + \theta_{foot})$$
$$A_{46} = L_{hoof}\,L_{leg}\,(M_{hoof}/2)\cos(\theta_{leg} - \theta_{hoof})$$

$$A_{51} = -L_{foot}\sin(\theta_{foot})\,(M_{foot}/2 + M_{hoof})$$
$$A_{52} = -L_{foot}\cos(\theta_{foot})\,(M_{foot}/2 + M_{hoof})$$
$$A_{53} = E\,L_{foot}\,(M_{foot}/2 + M_{hoof})\cos(\theta_{body} + \theta_{foot})$$
$$A_{54} = -L_{foot}\,L_{leg}\,(M_{foot}/2 + M_{hoof})\sin(\theta_{leg} + \theta_{foot})$$
$$A_{55} = I_{foot} + L_{foot}^2\,(M_{foot}/4 + M_{hoof})$$
$$A_{56} = -L_{foot}\,L_{hoof}\,(M_{hoof}/2)\sin(\theta_{foot} + \theta_{hoof})$$

$$A_{61} = L_{hoof}\,M_{hoof}\cos(\theta_{hoof})/2$$
$$A_{62} = L_{hoof}\,M_{hoof}\sin(\theta_{hoof})/2$$
$$A_{63} = E\,L_{hoof}\,(M_{hoof}/2)\sin(\theta_{body} - \theta_{hoof})$$
$$A_{64} = L_{hoof}\,L_{leg}\,(M_{hoof}/2)\cos(\theta_{leg} - \theta_{hoof})$$
$$A_{65} = -L_{foot}\,L_{hoof}\,(M_{hoof}/2)\sin(\theta_{foot} + \theta_{hoof})$$
$$A_{66} = I_{hoof} + L_{hoof}^2\,M_{hoof}/4$$

$$\begin{aligned} B_1 = {} & K_{g1}\,X_{g1\_td} + K_{g2}\,X_{g2\_td} - (K_{g1} + K_{g2})\,X_{body} - (B_{g1} + B_{g2})\,\dot{X}_{body} + \\ & E\,(K_{g1} + K_{g2})\cos(\theta_{body}) - E\,(M_{foot} + M_{hoof} + M_{leg})\,\dot{\theta}_{body}^2\cos(\theta_{body}) - \\ & (K_{g1} + K_{g2})\,L_{foot}\cos(\theta_{foot}) + L_{foot}\,(M_{foot}/2 + M_{hoof})\,\dot{\theta}_{foot}^2\cos(\theta_{foot}) + \\ & ((K_{g1} - K_{g2})\,W_{hoof}\cos(\theta_{hoof}))/2 - (B_{g1} + B_{g2})\,L_{leg}\,\dot{\theta}_{leg}\cos(\theta_{leg}) - \\ & (B_{g1} + B_{g2})\,E\,\dot{\theta}_{body}\sin(\theta_{body}) + (B_{g1} + B_{g2})\,L_{foot}\,\dot{\theta}_{foot}\sin(\theta_{foot}) - \\ & (K_{g1} + K_{g2})\,L_{hoof}\sin(\theta_{hoof}) + (L_{hoof}\,M_{hoof}\,\dot{\theta}_{hoof}^2\sin(\theta_{hoof}))/2 + \\ & \dot{\theta}_{hoof}\,(-((B_{g1} + B_{g2})\,L_{hoof}\cos(\theta_{hoof})) + ((-B_{g1} + B_{g2})\,W_{hoof}\sin(\theta_{hoof}))/2) - \\ & (K_{g1} + K_{g2})\,L_{leg}\sin(\theta_{leg}) + L_{leg}\,(M_{foot} + M_{hoof} + M_{leg}/2)\,\dot{\theta}_{leg}^2\sin(\theta_{leg}) \end{aligned}$$

$$\begin{aligned} B_2 = {} & -(G\,(M_{body} + M_{foot} + M_{hoof} + M_{leg})) - (K_{g1} + K_{g2})\,Y_{body} - (B_{g1} + B_{g2})\,\dot{Y}_{body} + \\ & (B_{g1} + B_{g2})\,E\,\dot{\theta}_{body}\cos(\theta_{body}) + (B_{g1} + B_{g2})\,L_{foot}\,\dot{\theta}_{foot}\cos(\theta_{foot}) + \\ & (K_{g1} + K_{g2})\,L_{hoof}\cos(\theta_{hoof}) - (L_{hoof}\,M_{hoof}\,\dot{\theta}_{hoof}^2\cos(\theta_{hoof}))/2 + \end{aligned}$$

$$(K_{g1} + K_{g2})\, L_{leg}\, \cos(\theta_{leg}) - (M_{foot} + M_{hoof} + M_{leg}/2)\, \dot{\theta}_{leg}^{2}\, \cos(\theta_{leg}) +$$
$$E\ (K_{g1} + K_{g2})\, \sin(\theta_{body}) - E\ (M_{foot} + M_{hoof} + M_{leg})\, \dot{\theta}_{body}^{2}\, \sin(\theta_{body}) +$$
$$(K_{g1} + K_{g2})\, L_{foot}\, \sin(\theta_{foot}) - L_{foot}\, (M_{foot}/2 + M_{hoof})\, \dot{\theta}_{foot}^{2}\, \sin(\theta_{foot}) +$$
$$((K_{g1} - K_{g2})\, W_{hoof}\, \sin(\theta_{hoof}))/2 + \dot{\theta}_{hoof}\, (((B_{g1} - B_{g2})\, W_{hoof}\, \cos(\theta_{hoof}))/2 +$$
$$(B_{g1} + B_{g2})\, L_{hoof}\, \sin(\theta_{hoof})) - (B_{g1} + B_{g2})\, L_{leg}\, \dot{\theta}_{leg}\, \sin(\theta_{leg})$$

$$B_3 = -\tau_{hip} - (B_{g1} + B_{g2})\, E\ E\ \dot{\theta}_{body} + E\ G\ (M_{foot} + M_{hoof} + M_{leg})\, \cos(\theta_{body}) +$$
$$E\ (K_{g1} + K_{g2})\, Y_{body}\, \cos(\theta_{body}) + (B_{g1} + B_{g2})\, E\ \dot{Y}_{body}\, \cos(\theta_{body}) -$$
$$E\ (K_{g1} + K_{g2})\, L_{hoof}\, \cos(\theta_{body} - \theta_{hoof}) +$$
$$(E\ L_{hoof}\, M_{hoof}\, \dot{\theta}_{hoof}^{2}\, \cos(\theta_{body} - \theta_{hoof}))/2 -$$
$$E\ (K_{g1} + K_{g2})\, L_{leg}\, \cos(\theta_{body} - \theta_{leg}) +$$
$$E\ L_{leg}\, (M_{foot} + M_{hoof} + M_{leg}/2)\, \dot{\theta}_{leg}^{2}\, \cos(\theta_{body} - \theta_{leg}) +$$
$$E\ (K_{g1}\, X_{g1\_td} + K_{g2}\, X_{g2\_td})\, \sin(\theta_{body}) - E\ (K_{g1} + K_{g2})\, X_{body}\, \sin(\theta_{body}) -$$
$$(B_{g1} + B_{g2})\, E\ \dot{X}_{body}\, \sin(\theta_{body}) - (B_{g1} + B_{g2})\, E\ L_{foot}\, \dot{\theta}_{foot}\, \cos(\theta_{body} + \theta_{foot}) -$$
$$E\ (K_{g1} + K_{g2})\, L_{foot}\, \sin(\theta_{body} + \theta_{foot}) +$$
$$E\ L_{foot}\, (M_{foot}/2 + M_{hoof})\, \dot{\theta}_{foot}^{2}\, \sin(\theta_{body} + \theta_{foot}) -$$
$$(E\ (K_{g1} - K_{g2})\, W_{hoof}\, \sin(-\theta_{body} + \theta_{hoof}))/2 +$$
$$\dot{\theta}_{hoof}\, (((-B_{g1} + B_{g2})\, E\ W_{hoof}\, \cos(\theta_{body} - \theta_{hoof}))/2 +$$
$$(B_{g1} + B_{g2})\, E\ L_{hoof}\, \sin(-\theta_{body} + \theta_{hoof})) +$$
$$(B_{g1} + B_{g2})\, E\ L_{leg}\, \dot{\theta}_{leg}\, \sin(-\theta_{body} + \theta_{leg})$$

$$B_4 = K_{ankle}\, (\theta_{ankleo} - \theta_{foot} - \theta_{leg}) + \tau_{hip} - (B_{ankle} + (B_{g1} + B_{g2})\, L_{leg}^{2})\, \dot{\theta}_{leg} +$$
$$E\ (K_{g1} + K_{g2})\, L_{leg}\, \cos(\theta_{body} - \theta_{leg}) -$$
$$(E\ L_{leg}\, (M_{foot} + M_{hoof} + M_{leg}/2)\, \dot{\theta}_{body}^{2}\, \cos(\theta_{body} - \theta_{leg}))/2 +$$
$$((K_{g1} - K_{g2})\, L_{leg}\, W_{hoof}\, \cos(\theta_{hoof} - \theta_{leg}))/2 +$$
$$L_{leg}\, (K_{g1}\, X_{g1\_td} + K_{g2}\, X_{g2\_td})\, \cos(\theta_{leg}) -$$
$$(K_{g1} + K_{g2})\, L_{leg}\, X_{body}\, \cos(\theta_{leg}) - (B_{g1} + B_{g2})\, L_{leg}\, \dot{X}_{body}\, \cos(\theta_{leg}) +$$
$$L_{foot}\, L_{leg}\, (M_{foot}/2 + M_{hoof})\, \dot{\theta}_{foot}^{2}\, \cos(\theta_{foot} + \theta_{leg}) +$$

$$(L_{hoof}\, L_{leg}\, M_{hoof}\, \dot{\theta}_{hoof}^{2}\, \sin(\theta_{hoof} - \theta_{leg}))/2 -$$

$$\dot{\theta}_{hoof}\, ((B_{g1} + B_{g2})\, L_{hoof}\, L_{leg}\, \cos(\theta_{hoof} - \theta_{leg}) +$$

$$((B_{g1} - B_{g2})\, L_{leg}\, W_{hoof}\, \sin(\theta_{hoof} - \theta_{leg}))/2) -$$

$$(G\ L_{leg}\, (M_{foot} + M_{hoof} + M_{leg}/2)\, \sin(\theta_{leg}))/2 -$$

$$(K_{g1} + K_{g2})\, L_{leg}\, Y_{body}\, \sin(\theta_{leg}) - (B_{g1} + B_{g2})\, L_{leg}\, \dot{Y}_{body}\, \sin(\theta_{leg}) -$$

$$(K_{g1} + K_{g2})\, Lfoot\, L_{leg}\, \cos(\theta_{foot} + \theta_{leg}) +$$

$$(B_{g1} + B_{g2})\, E\ L_{leg}\, \dot{\theta}_{body}\, \sin(-\theta_{body} + \theta_{leg}) +$$

$$\dot{\theta}_{foot}\, (-B_{ankle} + (B_{g1} + B_{g2})\, L_{foot}\, L_{leg}\, \sin(\theta_{foot} + \theta_{leg})) +$$

$$(K_{g1} + K_{g2})\, Lhoof\, L_{leg}\, \sin(-\theta_{hoof} + \theta_{leg})$$

$$B_5 = K_{ankle}\, (\theta_{ankleo} - \theta_{foot} - \theta_{leg}) - K_{toe}\, (\theta_{foot} + \theta_{hoof} - \theta_{toeo}) -$$

$$(B_{ankle} + B_{toe} + (B_{g1} + B_{g2})\, L_{foot}^{2})\, \dot{\theta}_{foot} +$$

$$G\, L_{foot}\, (M_{foot}/2 + M_{hoof})\, \cos(\theta_{foot}) + (K_{g1} + K_{g2})\, L_{foot}\, Y_{body}\, \cos(\theta_{foot}) +$$

$$(B_{g1} + B_{g2})\, L_{foot}\, \dot{Y}_{body}\, \cos(\theta_{foot}) +$$

$$(L_{foot}\, L_{hoof}\, M_{hoof}\, \dot{\theta}_{hoof}^{2}\, \cos(\theta_{foot} + \theta_{hoof}))/2 +$$

$$L_{foot}\, L_{leg}\, (M_{foot}/2 + M_{hoof})\, \dot{\theta}_{leg}^{2}\, \cos(\theta_{foot} + \theta_{leg}) -$$

$$L_{foot}\, (K_{g1}\, X_{g1\_td} + K_{g2}\, X_{g2\_td})\, \sin(\theta_{foot}) + (K_{g1} + K_{g2})\, L_{foot}\, X_{body}\, \sin(\theta_{foot}) +$$

$$(B_{g1} + B_{g2})\, L_{foot}\, \dot{X}_{body}\, \sin(\theta_{foot}) - (B_{g1} + B_{g2})\, E\ L_{foot}\, \dot{\theta}_{body}\, \cos(\theta_{body} + \theta_{foot}) -$$

$$E\ (K_{g1} + K_{g2})\, L_{foot}\, \sin(\theta_{body} + \theta_{foot}) +$$

$$E\ L_{foot}\, (M_{foot}/2 + M_{hoof})\, \dot{\theta}_{body}^{2}\, \sin(\theta_{body} + \theta_{foot}) -$$

$$(K_{g1} + K_{g2})\, L_{foot}\, L_{hoof}\, \cos(\theta_{foot} + \theta_{hoof}) -$$

$$((K_{g1} - K_{g2})\, L_{foot}\, W_{hoof}\, \sin(\theta_{foot} + \theta_{hoof}))/2 +$$

$$\dot{\theta}_{hoof}\, (-B_{toe} + ((B_{g1} - B_{g2})\, L_{foot}\, W_{hoof}\, (-\cos(\theta_{foot} + \theta_{hoof})))/2 +$$

$$(B_{g1} + B_{g2})\, L_{foot}\, L_{hoof}\, \sin(\theta_{foot} + \theta_{hoof})) -$$

$$(K_{g1} + K_{g2})\, L_{foot}\, L_{leg}\, \cos(\theta_{foot}) + \theta_{leg}) +$$

$$\dot{\theta}_{leg}\, (-B_{ankle} + (B_{g1} + B_{g2})\, L_{foot}\, L_{leg}\, \sin(\theta_{foot} + \theta_{leg}))$$

$$B_6 = -(K_{toe}\, (\theta_{foot} + \theta_{hoof} - \theta_{toeo})) -$$

$$(B_{toe} + (B_{g1} + B_{g2})\, L_{hoof}^{2} + ((B_{g1} + B_{g2})\, W_{hoof}^{2})/4)\, \dot{\theta}_{hoof} +$$

$$E\ (K_{g1} + K_{g2})\ L_{hoof}\ \cos(\theta_{body} - \theta_{hoof}) -$$

$$(E\ L_{hoof}\ M_{hoof}\ \dot{\theta}_{body}^{2}\ \cos(\theta_{body} - \theta_{hoof}))/2\ +$$

$$L_{hoof}\ (K_{g1}\ X_{g1\_td}\ + K_{g2}\ X_{g2\_td}\ )\ \cos(\theta_{hoof}) - (K_{g1} + K_{g2})\ L_{hoof}\ X_{body}\ \cos(\theta_{hoof}) +$$

$$((K_{g1} - K_{g2})\ W_{hoof}\ Y_{body}\ \cos(\theta_{hoof}))/2\ +$$

$$(L_{foot}\ L_{hoof}\ M_{hoof}\ \dot{\theta}_{foot}^{2}\ \cos(\theta_{foot} + \theta_{hoof}))/2\ -$$

$$((K_{g1} - K_{g2})\ L_{leg}\ W_{hoof}\ \cos(\theta_{hoof} - \theta_{leg}))/2\ -$$

$$\dot{\theta}_{body}\ (((B_{g1} - B_{g2})\ E\ W_{hoof}\ \cos(\theta_{body} - \theta_{hoof}))/2\ +$$

$$(B_{g1} + B_{g2})\ E\ L_{hoof}\ \sin(\theta_{body} - \theta_{hoof})) - (G\ L_{hoof}\ M_{hoof}\ \sin(\theta_{hoof}))/2\ +$$

$$(W_{hoof}\ (K_{g1}\ X_{g1\_td}\ - K_{g2}\ X_{g2\_td}\ )\ \sin(\theta_{hoof}))/2\ -$$

$$((K_{g1} - K_{g2})\ W_{hoof}\ X_{body}\ \sin(\theta_{hoof}))/2 - (K_{g1} + K_{g2})\ L_{hoof}\ Y_{body}\ \sin(\theta_{hoof}) +$$

$$\dot{Y}_{body}\ (((B_{g1} - B_{g2})\ W_{hoof}\ \cos(\theta_{hoof}))/2 - (B_{g1} + B_{g2})\ L_{hoof}\ \sin(\theta_{hoof})) -$$

$$\dot{X}_{body}\ ((B_{g1} + B_{g2})\ L_{hoof}\ \cos(\theta_{hoof}) + ((B_{g1} - B_{g2})\ W_{hoof}\ \sin(\theta_{hoof}))/2) -$$

$$(K_{g1} + K_{g2})\ L_{foot}\ L_{hoof}\ \cos(\theta_{foot} + \theta_{hoof}) +$$

$$(E\ (K_{g1} - K_{g2})\ W_{hoof}\ \sin(-\theta_{body} + \theta_{hoof}))/2\ -$$

$$((K_{g1} - K_{g2})\ L_{foot}\ W_{hoof}\ \sin(\theta_{foot} + \theta_{hoof}))/2\ +$$

$$\dot{\theta}_{foot}\ (-B_{toe} + ((B_{g1} - B_{g2})\ L_{foot}\ W_{hoof}\ (-\cos(\theta_{foot} + \theta_{hoof})))/2\ +$$

$$(B_{g1} + B_{g2})\ L_{foot}\ L_{hoof}\ \sin(\theta_{foot} + \theta_{hoof})) -$$

$$(L_{hoof}\ L_{leg}\ M_{hoof}\ \dot{\theta}_{leg}^{2}\ \sin(\theta_{hoof} - \theta_{leg}))/2\ -$$

$$\dot{\theta}_{leg}\ ((B_{g1} + B_{g2})\ L_{hoof}\ L_{leg}\ \cos(\theta_{hoof} - \theta_{leg}) +$$

$$((B_{g1} - B_{g2})\ L_{leg}\ W_{hoof}\ \sin(\theta_{hoof} - \theta_{leg}))/2) -$$

$$(K_{g1} + K_{g2})\ L_{hoof}\ L_{leg}\ \sin(-\theta_{hoof} + \theta_{leg})$$

Appendix E:
Derivation of Equations of Motion of the Simple Model and Direction of the Force at the Hoof

Given the model shown in figure 7-1, the equations of the motion for the model are derived using the Lagrangian dynamic method. The generalized coordinates, ζ, chosen are the horizontal position of the body mass M with respect to a reference point, x, the vertical position of the body mass M with respect to the ground, y, the horizontal position of the foot mass m with respect to a reference point, x_g, and the vertical position of the foot mass m with respect to the ground, y_g. The horizontal reference point is the touchdown position of the foot. The kinetic coenergy T^*, the potential energy V, and the virtual work δW are

$$T^* = \frac{1}{2} m \dot{x}_g^2 + \frac{1}{2} m \dot{y}_g^2 + \frac{1}{2} M \dot{x}^2 + \frac{1}{2} M \dot{y}^2 \tag{E.1}$$

$$\begin{aligned} V = {} & \frac{1}{2} k_g (x_g - x_{g\_o})^2 + \frac{1}{2} k_g (y_g - y_{g\_o})^2 \\ & + \frac{1}{2} K \left(\sqrt{(x - x_g)^2 + (y - y_g)^2} - r_o \right)^2 \\ & + (m y_g + M y) G \end{aligned} \tag{E.2}$$

$$\delta W = 0.0 \tag{E.3}$$

where

K	is the stiffness of the leg spring,
k_g	is the stiffness of the rubber foot pad spring,
x_{g_o}	is the horizontal reference position (the touchdown position),
y_{g_o}	is the vertical reference position (= 0.0), and
G	is the gravity.

Note that there is no external force applied and no dissipative element in the model. Thus the virtual work is zero. The equations of the motion are then derived by

$$\frac{d}{dt}\left(\frac{\partial \alpha}{\partial \dot{\xi}}\right) - \frac{\partial \alpha}{\partial \xi} = 0.0 \tag{E.4}$$

where

α is the Lagrangian term, $T^* - V$, and

ζ is the generalized coordinate.

The resulting equations of the motion are

$$\begin{aligned}
\frac{d^2x_g}{dt^2} &= -\frac{k_g}{m}(x_g - x_{go}) + \frac{K}{m}(x - x_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2 + (y-y_g)^2)}}\right) \\
\frac{d^2y_g}{dt^2} &= -\frac{k_g}{m}(y_g - y_{go}) + \frac{K}{m}(y - y_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2 + (y-y_g)^2)}}\right) - G \\
\frac{d^2x}{dt^2} &= -\frac{K}{M}(x - x_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2 + (y-y_g)^2)}}\right) \\
\frac{d^2y}{dt^2} &= -\frac{K}{M}(y - y_g)\left(1 - \frac{r_o}{\sqrt{((x-x_g)^2 + (y-y_g)^2)}}\right) - G
\end{aligned} \tag{E.5}$$

Next, let us derive the relationship of the angle of the force vector induced at the foot (or the hinge of the hoof in the case for the monopod) with respect to the state of the system. Since the horizontal and vertical components of the resultant force induced at the foot (hinge of the hoof) are equal to the forces of the horizontal and vertical rubber foot (hoof) pad springs,

$$\begin{aligned}
F_x &= k_g\,(x_g - x_{g_o}) \\
&= -m\,\ddot{x}_g + K\,(x - x_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right)
\end{aligned} \tag{E.6}$$

$$\begin{aligned}
F_y &= k_g\,(y_g - y_{g_o}) \\
&= -m\,\ddot{y}_g + K\,(y - y_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right) - m\,G
\end{aligned} \tag{E.7}$$

where

F_x is the horizontal component of the force applied at the foot (hoof), and

F_y is the vertical component of the force applied at the foot (hoof).

By substituting

$$K\,(x - x_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right) = -M\,\ddot{x} \tag{E.8}$$

$$K\,(y - y_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right) = -M\,\ddot{y} - M\,G \tag{E.9}$$

into the equations (E.6) and (E.7), then we get

$$F_x = k_g\,(x_g - x_{g\_o}) = -\,m\,\ddot{x}_g - M\,\ddot{x} \tag{E.10}$$

$$F_y = k_g\,(y_g - y_{g\_o}) = -\,m\,\ddot{y}_g - M\,\ddot{y} - (m + M)\,G \tag{E.11}$$

Since the mass of the foot (or the hoof) m is much smaller than the mass of the body M (less than 1 %) and since the peak accelerations of the foot are in a similar range to that of the body, the mass of the hoof can be safely assumed to have negligible effect. Then

$$F_x = k_g\,(x_g - x_{g\_o}) \approx -\,M\,\ddot{x} \tag{E.12}$$

$$F_y = k_g\,(y_g - y_{g\_o}) \approx -\,M\,\ddot{y} - M\,G \tag{E.13}$$

which implies that the ratio of F_x and F_y is

$$\frac{F_x}{F_Y} \approx \frac{\ddot{x}}{\ddot{y} + G} \tag{E.14}$$

Substituting

$$\ddot{x} = -\frac{K}{M}(x - x_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right) \tag{E.15}$$

$$\ddot{y} = -\frac{K}{M}(y - y_g)\left(1 - \frac{r_o}{\sqrt{(x - x_g)^2 + (y - y_g)^2}}\right) - G \tag{E.16}$$

into the equation (E,14) yields

$$\frac{F_x}{F_Y} \approx \frac{(x - x_g)}{(y - y_g)}. \tag{E.17}$$

Since

$$(x - x_g) = r\,\sin\,\theta_{leg} \tag{E.18}$$

$$(y - y_g) = r\,\cos\,\theta_{leg} \tag{E.19}$$

where

r is the length of the leg, and

θ_{leg} is the angle of the leg with respect to the vertical.

Therefore, the ratio of the Fx and Fy is approximately equal to the tangent of the angle of the leg;

$$\frac{F_x}{F_y} \approx tan\ \theta_{leg} \tag{E.20}$$

which implies that

$$\phi_{force} \approx \theta_{leg} \tag{E.21}$$

where

ϕ_{force} is the angle of the resultant force applied at the foot (hoof) with respect to vertical.

Appendix F:
Code for Computer Simulation of the Monopod

```
#include <math.h>
#include <malloc.h>
#include <stdio.h>
#include "pixels.h"
#include "views.h"

#define TINY 1.0e-30;

typedef struct monopod_structure
{
 /* Simulation parameters. */
 float gravity;
 float body_mass;
 float leg_mass;
 float foot_mass;
 float hoof_mass;
 float body_I;
 float leg_I;
 float foot_I;
 float hoof_I;

 /* State variables. */

 /* Positions */
 float foot_angle;
 float leg_angle;
 float body_angle;
 float hoof_angle;
 float body_x;
 float body_y;
 float deltaf;


 /* Velocities */
 float foot_angled;
 float leg_angled;
 float body_angled;
 float hoof_angled;
 float body_xd;
 float body_yd;
 float deltafd;

 /* Accelerations */
 float foot_angledd;
 float body_angledd;
 float leg_angledd;
 float hoof_angledd;
 float body_xdd;
 float body_ydd;
```

```
float deltafdd;

float hip_torque;
float hip_torque_cal;
float fx;
float fy;
float ankle_stiffness;
float ankle_viscosity;
float toe_stiffness;
float toe_viscosity;
float ground_stiffness;
float ground1_stiffness;
float ground_viscosity;
float ground1_viscosity;
float ground2_stiffness;
float ground2_viscosity;

/* Controller variables. */

float hoof1_x;
float hoof1_y;
float hoof2_x;
float hoof2_y;
float hoof1_xd;
float hoof1_yd;
float hoof2_xd;
float hoof2_yd;
float hoof1_xtd;
float hoof2_xtd;
float hoof1_y_previous;
float hoof2_y_previous;
float body_x_touchdown;
float time;
float time_touchdown;
float time_takeoff;
float time_touchdown_1;
float time_touchdown_2;
float time_takeoff_1;
float time_takeoff_2;
float time_stance_1;
float time_stance_2;
float time_stance;
float foot_angle_des;
float foot_angle_desired;
float leg_angle_desired;
float hoof_angle_desired;
float body_xd_desired;
float kxd;
float kp;
float kv;
float kp_flight;
float kv_flight;
float flight;
float stance;
float cg_print;
float accel;
float xd_td;
float xd_lo;
```

```
float xd_diff;
float theta_accel;
float foot_f;
float zforce;
float toef;
float downf;
float sidef;
float sidef_th1;
float sidef_abs;
float sidef_sign;
float hip_torque_sign;
float kt2;

/* Other useful variables. */

float display_interval;
float time_step;
float foot_length;
float leg_length;
float body_length;
float hoof_length;
float hoof_width;
float e;
float ankle_x;
float ankle_y;
float ankle_yd;
float body_front_x;
float body_front_y;
float body_back_x;
float body_back_y;
float hip_x;
float hip_y;
float toe_x;
float toe_y;
float theta2;
float theta2d;
float thetaheel;
float thetaheeld;
float foot_angled_square;
float cg_offset;
float foot_angle_offset;
float roll_counter;
float th_leg_heel;

float hoof1_x_display;
float hoof1_y_display;
float hoof2_x_display;
float hoof2_y_display;
float toe_x_display;
float toe_y_display;
float ankle_x_display;
float ankle_y_display;
float hip_x_display;
float hip_y_display;
float body_front_x_display;
float body_front_y_display;
float body_back_x_display;
float body_back_y_display;
```

```
  float new_time;
  float eps;
  float h1;
  float hmin;


} Hopper, *pHopper;



pView pv;
pHopper ph;


extern float last_x1,last_y1,last_x2,last_y2;

main()
{
  View v;
  Hopper h;


  printf("Monopod Simulation\n");

  init_window();
  clear_window();

  pv = &v;
  pv->user_left_x = -1.0;
  pv->user_right_x = 1.0;
  pv->user_bottom_y = -1.0;
  pv->user_top_y = 1.0;
  pv->display_left_x = 0.0;
  pv->display_right_x = 496.0;
  pv->display_bottom_y = 844.0;
  pv->display_top_y = 0.0;
  pv->display = 0.0;
  pv->x_conversion_factor = 248.0 /2.5;
  pv->y_conversion_factor = -211.0/2.5;

  describe_view( pv );

  vdraw_line(-0.9, -0.9, 0.9, 0.9, SET, pv);
  vdraw_line(0.9, -0.9, -0.9, 0.9, SET, pv);
  vdraw_line(-0.9, -0.9, 0.9, -0.9, SET, pv);
  vdraw_line(0.9, -0.9, 0.9, 0.9, SET, pv);
  vdraw_line(0.9, 0.9, -0.9, 0.9, SET, pv);
  vdraw_line(-0.9, 0.9, -0.9, -0.9, SET, pv);

  printf("Press RETURN to continue...\n");
  getchar();


  ph = &h;
  ph->h1 = 0.001;  /* s */
  ph->time_step = 0.0001;
  ph->hmin = 0.00000;
```

```
  ph->time = 0.0;
  ph->new_time = 0.0;
  ph->eps = 0.01;
  ph->display_interval = 1;
  ph->flight = 1.0;
  ph->stance = 0.0;
  ph->gravity = 9.81000;  /* m/s^2 */
  ph->hoof_length = 0.05;
  ph->hoof_width = 0.07;
  ph->foot_length = 0.188;
  ph->leg_length = 0.610;
  ph->body_length = 0.7;
  ph->e = 0.16;
  ph->body_mass = 6.4;              /* kg */
  ph->leg_mass = 0.73;
  ph->foot_mass = 0.1;
  ph->hoof_mass = 0.05;
  ph->body_I = 0.4;                         /* kg-m^2 */
  ph->leg_I =  0.097;
  ph->foot_I = 0.0097;
  ph->hoof_I = 0.003;
  ph->ankle_stiffness =  300.0 ;
  ph->ankle_viscosity = 1.0;
  ph->toe_stiffness = 10.;
  ph->toe_viscosity = 0.1;
  ph->ground_stiffness =  20000.0;
  ph->ground_viscosity =  500.0;
  ph->ground1_stiffness =  0.0;
  ph->ground2_stiffness =  0.0;
  ph->ground1_viscosity =  0.0;
  ph->ground2_viscosity =  0.0;
  ph->kxd = 0.05;
  ph->kp = 500.0;
  ph->kv = 50.0;
  ph->kp_flight = 5000.0;
  ph->kv_flight = 500.0;
  ph->kt2 = 0.0 /* 0.54 */;
  ph->sidef_th1 = 0;
  ph->roll_counter = 0;

/* state variables */

  ph->foot_angle_des = 0.2;
  ph->foot_angle_desired = 0.2;
  ph->hoof_angle_desired = 0.0;
  ph->body_x = 0.1;
  ph->body_y = 0.9;                         /* m */
  ph->body_angle = 0.14;
  ph->leg_angle =  0.14;
  ph->foot_angle = 0.2;
  ph->hoof_angle = 0.0;
  ph->body_xd = 0.0;
  ph->body_xd_desired = 2.0;
  ph->body_yd  = 0.6;
  ph->foot_angled = 0.0;
  ph->body_angled = 0.0;
  ph->leg_angled = 0.0;
  ph->hoof_angled = 0.0;
```

```
    ph->hoof1_xtd = 0.0;
    ph->hoof2_xtd = 0.0;
    ph->hoof1_yd = 0.0;
    ph->hoof2_yd = 0.0;
    ph->hoof1_y_previous = 0.0;
    ph->hoof2_y_previous = 0.0;

    ph->body_xdd = 0.0;
    ph->body_ydd  = 0.0;
    ph->foot_angledd = 0.0;
    ph->body_angledd = 0.0;
    ph->leg_angledd = 0.0;
    ph->hoof_angledd = 0.0;

    ph->leg_angle_desired = ph->leg_angle;
    ph->foot_angled_square = 0.0;

    ph->cg_offset = -0.22;
    ph->foot_angle_offset = 0.1;


    ph->hip_torque = 0.0;
    ph->time_stance_1 = 0.0;
    ph->time_stance_2 = 0.0;
    ph->time_stance = 0.0;
    ph->time_takeoff_1 = 0.0;
    ph->time_takeoff_2 = 0.0;
    ph->time = 0.0;
    ph->time_touchdown_1 = 0.0;
    ph->time_touchdown_2 = 0.0;

/*****************************************************************/

    init_hopper (ph, pv);

    printf("Press RETURN to continue...\n");
    getchar();
    run_hopper (ph, pv, 1000000);

    printf("Press RETURN to continue...\n");
    getchar();

    done_with_window();
}

/*****************************************************************/
/*        Initial Conditions for Monopod                         */
/*****************************************************************/

init_hopper(ph, pv)
    pHopper ph;
    pView pv;
{
    float hoof1_x, hoof1_y, hoof2_x, hoof2_y, toe_x , toe_y ,
          ankle_x , ankle_y , hip_x , hip_y , body_front_x ,
          body_front_y, body_back_x , body_back_y;

    clear_window();
```

```
vdraw_line(-1.0, 0.0, 11.0, 0.0, XOR, pv);

ph->body_front_x = ph->body_x + (ph->body_length /2 *
                                  cos(ph->body_angle));
ph->body_front_y = ph->body_y + (ph->body_length /2 *
                                  sin(ph->body_angle));
ph->body_back_x = ph->body_x - (ph->body_length /2 *
                                  cos(ph->body_angle));
ph->body_back_y = ph->body_y - (ph->body_length /2 *
                                  sin(ph->body_angle));

ph->hip_x = ph->body_x - (ph->e * cos(ph->body_angle));
ph->hip_y = ph->body_y - (ph->e * sin(ph->body_angle));

ph->ankle_x = ph->hip_x + (ph->leg_length * sin(ph->leg_angle));
ph->ankle_y = ph->hip_y - (ph->leg_length * cos(ph->leg_angle));

ph->toe_x = ph->ankle_x + (ph->foot_length * cos(ph->foot_angle));
ph->toe_y = ph->ankle_y - (ph->foot_length * sin(ph->foot_angle));

ph->hoof1_x = ph->toe_x + (ph->hoof_length * sin(ph->hoof_angle))
        - (ph->hoof_width / 2 * cos(ph->hoof_angle));
ph->hoof1_y = ph->toe_y - (ph->hoof_length * cos(ph->hoof_angle))
        - (ph->hoof_width / 2 * sin(ph->hoof_angle));
ph->hoof2_x = ph->toe_x + (ph->hoof_length * sin(ph->hoof_angle))
        + (ph->hoof_width / 2 * cos(ph->hoof_angle));
ph->hoof2_y = ph->toe_y - (ph->hoof_length * cos(ph->hoof_angle))
        + (ph->hoof_width / 2 * sin(ph->hoof_angle));



hoof1_x = ph->hoof1_x;
hoof1_y = ph->hoof1_y;
hoof2_x = ph->hoof2_x;
hoof2_y = ph->hoof2_y;
toe_x = ph->toe_x;
toe_y = ph->toe_y;
ankle_x = ph->ankle_x;
ankle_y = ph->ankle_y;
hip_x = ph->hip_x;
hip_y = ph->hip_y;
body_front_x = ph->body_front_x;
body_front_y = ph->body_front_y;
body_back_x = ph->body_back_x;
body_back_y = ph->body_back_y;

vdraw_line(toe_x , toe_y , hoof1_x, hoof1_y, XOR, pv);
ph->toe_x_display = last_x1;
ph->toe_y_display = last_y1;
ph->hoof1_x_display = last_x2;
ph->hoof1_y_display = last_y2;

vdraw_line(toe_x , toe_y , hoof2_x, hoof2_y, XOR, pv);
ph->hoof2_x_display = last_x2;
ph->hoof2_y_display = last_y2;
```

```
  vdraw_line(hoof1_x,hoof1_y,hoof2_x, hoof2_y, XOR, pv);


  vdraw_line(toe_x , toe_y , ankle_x , ankle_y , XOR , pv);


  ph->ankle_x_display = last_x2;
  ph->ankle_y_display = last_y2;

  vdraw_line(ankle_x , ankle_y , hip_x , hip_y , XOR , pv);

  ph->hip_x_display = last_x2;
  ph->hip_y_display = last_y2;

  vdraw_line(body_front_x , body_front_y , body_back_x ,
             body_back_y , XOR , pv);

  ph->body_front_x_display = last_x1;
  ph->body_front_y_display = last_y1;
  ph->body_back_x_display = last_x2;
  ph->body_back_y_display = last_y2;
}

/*****************************************************************/
/*          Numerical Integration of Monopod          */
/*****************************************************************/

run_hopper (ph, pv, n_iterations)
    pHopper ph;
    pView pv;
    int n_iterations;


{
 float  hoof1_near_zero, hoof2_near_zero, small;

 float  hoof1_x, hoof1_y, hoof2_x, hoof2_y,
      toe_x , toe_y , ankle_x , ankle_y ,
      hip_x , hip_y , body_front_x , body_front_y,
      body_back_x   , body_back_y, ka, kb, kc, kd, ke;
 int a,b,c,d,e, counter,window_counter;


 small = 0.0001;
 counter = 0;
 window_counter = 0;

 for (c=0; c<n_iterations; ++c)
 {

    ph->new_time = ph->h1 + ph->time;

    integrate (ph);


    ph->ankle_yd = ph->body_yd - (ph->e * cos(ph->body_angle) *
                                  ph->body_angled) +
                                   (ph->leg_length *
```

```
                                    sin(ph->leg_angle) *
                                    ph->leg_angled);


ph->hoof1_x = ph->body_x
            - ph->e * cos(ph->body_angle)
                + ph->foot_length * cos(ph->foot_angle)
                - ph->hoof_width /2 * cos(ph->hoof_angle)
                + ph->hoof_length * sin(ph->hoof_angle)
                + ph->leg_length * sin(ph->leg_angle);

ph->hoof1_y = ph->body_y
            - ph->hoof_length * cos(ph->hoof_angle)
                - ph->leg_length * cos(ph->leg_angle)
                - ph->e * sin(ph->body_angle)
                - ph->foot_length * sin(ph->foot_angle)
                - ph->hoof_width / 2 * sin(ph->hoof_angle);

ph->hoof2_x = ph->body_x
            - ph->e * cos(ph->body_angle)
                + ph->foot_length * cos(ph->foot_angle)
                + ph->hoof_width /2 * cos(ph->hoof_angle)
                + ph->hoof_length * sin(ph->hoof_angle)
                + ph->leg_length * sin(ph->leg_angle);;

ph->hoof2_y = ph->body_y
            - ph->hoof_length * cos(ph->hoof_angle)
                - ph->leg_length * cos(ph->leg_angle)
                - ph->e * sin(ph->body_angle)
                - ph->foot_length * sin(ph->foot_angle)
                + ph->hoof_width / 2 * sin(ph->hoof_angle);


ph->hoof1_xd = ph->hoof_length * cos(ph->hoof_angle) * ph->hoof_angled
             + ph->leg_length * cos(ph->leg_angle) * ph->leg_angled
                + ph->body_xd
                + ph->e * ph->body_angled * sin(ph->body_angle)
                - ph->foot_length * ph->foot_angled * sin(ph->foot_angle)
                + ph->hoof_width /2 * ph->hoof_angled * sin(ph->hoof_angle);

ph->hoof2_xd = ph->hoof_length * cos(ph->hoof_angle) * ph->hoof_angled
             + ph->leg_length * cos(ph->leg_angle) * ph->leg_angled
                + ph->body_xd
                + ph->e * ph->body_angled * sin(ph->body_angle)
                - ph->foot_length * ph->foot_angled * sin(ph->foot_angle)
                - ph->hoof_width /2 * ph->hoof_angled * sin(ph->hoof_angle);


ph->hoof1_yd = - (ph->e * cos(ph->body_angle) * ph->body_angled)
             - (ph->foot_length * cos(ph->foot_angle) *
                    ph->foot_angled)
                - (ph->hoof_width /2 * cos(ph->hoof_angle) *
                    ph->hoof_angled)
                + ph->body_yd
                + (ph->hoof_length * ph->hoof_angled *
```

```
                    sin(ph->hoof_angle))
                + (ph->leg_length * ph->leg_angled *
                    sin(ph->leg_angle));

  ph->hoof2_yd = - (ph->e * cos(ph->body_angle) * ph->body_angled)'
              - (ph->foot_length * cos(ph->foot_angle) *
                    ph->foot_angled)
                + (ph->hoof_width /2 * cos(ph->hoof_angle) *
                    ph->hoof_angled)
                + ph->body_yd
                + (ph->hoof_length * ph->hoof_angled *
                    sin(ph->hoof_angle))
                + (ph->leg_length * ph->leg_angled *
                    sin(ph->leg_angle));




  if (ph->hoof1_y <= 0.0 && ph->hoof1_y_previous > 0.0) {
       ph->hoof1_xtd = ph->hoof1_x;
       ph->time_touchdown_1 = ph->time;
       ph->flight = 0.0;
       ph->stance = 1.0;
       ph->xd_td = ph->body_xd;

}

  if (ph->hoof2_y <= 0.0 && ph->hoof2_y_previous > 0.0) {
       ph->hoof2_xtd = ph->hoof2_x;
       ph->time_touchdown_2 = ph->time;
       ph->flight = 0.0;
       ph->stance = 1.0;
       ph->xd_td = ph->body_xd;
}

  if (ph->hoof1_y_previous > 0.0 && ph->hoof2_y_previous > 0.0 &&
        ph->stance == 1.0)
       ph->theta_accel = ph->cg_print - ph->cg_offset - ph->leg_angle;

  if (ph->hoof1_y >= 0.0 && ph->hoof1_y_previous < 0.0) {
       ph->time_takeoff_1 = ph->time;
       ph->flight = 1.0;
       ph->stance = 0.0;
       ph->xd_lo = ph->body_xd;
}
  if (ph->hoof2_y >= 0.0 && ph->hoof2_y_previous < 0.0) {
       ph->time_takeoff_2 = ph->time;
       ph->flight = 1.0;
       ph->stance = 0.0;
       ph->xd_lo = ph->body_xd;
}

 if ((ph->hoof1_y_previous < 0.0 || ph->hoof2_y_previous < 0.0) &&
        (ph->hoof1_y >= 0.0 && ph->hoof2_y >= 0.0)) {
       ph->xd_diff = ph->xd_lo - ph->xd_td;
       printf("%f,%f,%f\n",ph->xd_td,ph->xd_diff,ph->theta_accel);
```

```
}

 if (ph->time_touchdown_1 < ph->time_touchdown_2) {
      ph->time_touchdown = ph->time_touchdown_1;
}
 else ph->time_touchdown = ph->time_touchdown_2;

 if (ph->time_takeoff_1 > ph->time_takeoff_2) {
      ph->time_takeoff = ph->time_takeoff_1;
}
 else ph->time_takeoff = ph->time_takeoff_2;



 if (ph->hoof1_y <= 0.0){
      ph->ground1_stiffness = ph->ground_stiffness;
      ph->ground1_viscosity = ph->ground_viscosity;
}
 else {
       ph->ground1_stiffness = 0.0;
      ph->ground1_viscosity = 0.0;
}
 if (ph->hoof2_y <= 0.0){
      ph->ground2_stiffness = ph->ground_stiffness;
      ph->ground2_viscosity = ph->ground_viscosity;
}
 else {
       ph->ground2_stiffness = 0.0;
      ph->ground2_viscosity = 0.0;
}



 if (ph->hoof1_y > 0.0 && ph->hoof2_y > 0.0) {
      ph->roll_counter = 0;
      ph->time_stance = ph->time_takeoff - ph->time_touchdown;

      ph->flight = 1.0;
      ph->stance = 0.0;
      ph->cg_print = asin(ph->body_xd * ph->time_stance /
                          (ph->leg_length * 2));
      ph->accel = ph->kxd * (ph->body_xd - ph->body_xd_desired);

      ph->leg_angle_desired =  ph->cg_print + ph->accel - ph->cg_offset;


      ph->foot_angle_des = ph->cg_print - ph->accel +
        ph->foot_angle_offset;


      if ((ph->cg_print - ph->accel) < 0.0)
        ph->foot_angle_des = ph->foot_angle_offset;
      ph->foot_angle_desired = ph->foot_angle_des;

      if ((ph->hoof1_y > 0.05 && ph->ankle_yd >= 0.0) ||
         ph->ankle_yd < 0.0) {
       ph->hip_torque = - (ph->kp_flight * (ph->leg_angle -
                                    ph->leg_angle_desired)) -
```

```
                                    (ph->kv_flight * ph->leg_angled);

        }
        else  ph->hip_torque = 0.0;

}
  else {

        ph->foot_angled_square = ph->foot_angled * ph->foot_angled;

        if (ph->foot_angled >= 1.0) {
          ph->foot_angle_desired = ph->foot_angle_des + 0.25;

        }

        ph->hip_torque_cal =  (ph->kp * (ph->body_angle - 0.0)) +
                    (ph->kv * ph->body_angled);
        ph->foot_f = ph->ankle_stiffness * (ph->foot_angle_desired -
                                            ph->foot_angle);
        ph->zforce = ph->foot_f * cos(ph->foot_angle);
        ph->toef = - ph->hip_torque / ph->leg_length;
        ph->downf = ph->zforce + (ph->toef * sin(ph->leg_angle));
        ph->sidef = -(ph->toef*cos(ph->leg_angle))-
                          (ph->foot_f*sin(ph->foot_angle));

        ph->sidef_abs = sqrt(ph->sidef * ph->sidef);
        ph->sidef_sign = ph->sidef / ph->sidef_abs;


        if (ph->roll_counter == 0 || ph->roll_counter >= 3) {
          ph->roll_counter = 0;
          if (ph->downf > (ph->sidef_abs + ph->sidef_th1)) {
            ph->hip_torque = ph->hip_torque_cal;
          }
          else {
            ph->hip_torque = ph->hip_torque_cal +
             (ph->kt2 * ph->sidef_sign *
              (ph->downf - ph->sidef_abs - ph->sidef_th1));

            ph->roll_counter = ph->roll_counter + 1;

          }


        }
        if (ph->roll_counter > 0)
          ph->roll_counter = ph->roll_counter + 1;
}



 ph->deltaf = (ph->foot_angle - ph->foot_angle_desired)
              * ph->foot_length;
 ph->deltafd = ph->foot_angled * ph->foot_length;
 ph->thetaheel = ph->foot_angle;
 ph->thetaheeld = ph->foot_angled;
 ph->theta2 = - ph->leg_angle;
```

```
ph->theta2d = - ph->leg_angled;


ph->hoof1_y_previous = ph->hoof1_y;
ph->hoof2_y_previous = ph->hoof2_y;



if (++counter > ph->display_interval)
    {

   ph->body_front_x = ph->body_x +(ph->body_length /2 *
                                    cos(ph->body_angle));
   ph->body_front_y = ph->body_y +(ph->body_length /2 *
                                    sin(ph->body_angle));
   ph->body_back_x = ph->body_x -(ph->body_length /2 *
                                    cos(ph->body_angle));
   ph->body_back_y = ph->body_y -(ph->body_length /2 *
                                    sin(ph->body_angle));

      ph->hip_x = ph->body_x - (ph->e * cos(ph->body_angle));
      ph->hip_y = ph->body_y - (ph->e * sin(ph->body_angle));
      ph->ankle_x = ph->hip_x + (ph->leg_length * sin(ph->leg_angle));
      ph->ankle_y = ph->hip_y - (ph->leg_length * cos(ph->leg_angle));
      ph->toe_x = ph->ankle_x + (ph->foot_length *
                                 cos(ph->foot_angle));
      ph->toe_y = ph->ankle_y - (ph->foot_length *
                                 sin(ph->foot_angle));


      hoof1_x = ph->hoof1_x;
      hoof1_y = ph->hoof1_y;
      hoof2_x = ph->hoof2_x;
      hoof2_y = ph->hoof2_y;
      toe_x = ph->toe_x;
      toe_y = ph->toe_y;
      ankle_x = ph->ankle_x;
      ankle_y = ph->ankle_y;
      hip_x = ph->hip_x;
      hip_y = ph->hip_y;
      body_front_x = ph->body_front_x;
      body_front_y = ph->body_front_y;
      body_back_x = ph->body_back_x;
      body_back_y = ph->body_back_y;

      (int) window_counter = body_back_x / 3.5;
      hoof1_x = hoof1_x - 3.5 * (int) window_counter;
      hoof2_x = hoof2_x - 3.5 * (int) window_counter;
      hoof1_y = hoof1_y - 3.5 * (int) window_counter;
      hoof2_y = hoof2_y - 3.5 * (int) window_counter;
      toe_x = toe_x - 3.5 * (int) window_counter;
      ankle_x = ankle_x - 3.5 * (int) window_counter;
      hip_x = hip_x - 3.5 * (int) window_counter;
      body_front_x = body_front_x - 3.5 * (int) window_counter;
      body_back_x = body_back_x - 3.5 * (int) window_counter;
```

```
_vdraw_line((float)ph->toe_x_display,
            (float)ph->toe_y_display,
            (float)ph->hoof1_x_display,
            (float)ph->hoof1_y_display, XOR);

_vdraw_line((float)ph->toe_x_display,
            (float)ph->toe_y_display,
            (float)ph->hoof2_x_display,
            (float)ph->hoof2_y_display, XOR);

_vdraw_line((float)ph->hoof1_x_display,
            (float)ph->hoof1_y_display,
            (float)ph->hoof2_x_display,
            (float)ph->hoof2_y_display, XOR);

_vdraw_line((float)ph->toe_x_display,
            (float)ph->toe_y_display,
            (float)ph->ankle_x_display,
            (float)ph->ankle_y_display, XOR);

_vdraw_line((float)ph->ankle_x_display,
            (float)ph->ankle_y_display,
            (float)ph->hip_x_display,
            (float)ph->hip_y_display, XOR);

_vdraw_line((float)ph->body_front_x_display,
            (float)ph->body_front_y_display,
            (float)ph->body_back_x_display,
            (float)ph->body_back_y_display, XOR);

vdraw_line(toe_x , toe_y , hoof1_x, hoof1_y, XOR, pv);
ph->toe_x_display = last_x1;
ph->toe_y_display = last_y1;
ph->hoof1_x_display = last_x2;
ph->hoof1_y_display = last_y2;

vdraw_line(toe_x , toe_y , hoof2_x, hoof2_y, XOR, pv);
ph->hoof2_x_display = last_x2;
ph->hoof2_y_display = last_y2;


vdraw_line(hoof1_x,hoof1_y,hoof2_x, hoof2_y, XOR, pv);

vdraw_line(toe_x , toe_y , ankle_x , ankle_y , XOR , pv);

ph->ankle_x_display = last_x2;
ph->ankle_y_display = last_y2;

vdraw_line(ankle_x , ankle_y , hip_x , hip_y , XOR , pv);

ph->hip_x_display = last_x2;
ph->hip_y_display = last_y2;


vdraw_line(body_front_x , body_front_y , body_back_x ,
           body_back_y , XOR , pv);

ph->body_front_x_display = last_x1;
```

```
            ph->body_front_y_display = last_y1;
            ph->body_back_x_display = last_x2;
            ph->body_back_y_display = last_y2;

            counter = 0;

          }
    }

}

/*****************************************************************/

integrate (ph)
    pHopper ph;

{

  int *nok, *nbad, nok_value, nbad_value;
  void derivs();
  void odeint();
  void rkqc();
  void free_vector(), free_ivector();

  float y[12], dydt[12], *vector();
  int *ivector();

/*  y = vector(1,12); */
/*  dydt = vector(1,12); */

  y[1] = ph->body_x;
  y[2] = ph->body_y;
  y[3] = ph->body_angle;
  y[4] = ph->leg_angle;
  y[5] = ph->foot_angle;
  y[6] = ph->hoof_angle;
  y[7] = ph->body_xd;
  y[8] = ph->body_yd;
  y[9] = ph->body_angled;
  y[10] = ph->leg_angled;
  y[11] = ph->foot_angled;
  y[12] = ph->hoof_angled;



  nok_value = 0;
  nbad_value = 0;
  nok = &nok_value;
  nbad = &nbad_value;

  odeint(ph,y,12,ph->time,ph->new_time,ph->eps,ph->h1,ph->hmin,nok,nbad,derivs,rkqc);


  ph->body_x = y[1];
  ph->body_y = y[2];
  ph->body_angle = y[3];
  ph->leg_angle = y[4];
  ph->foot_angle = y[5];
```

```
  ph->hoof_angle = y[6];
  ph->body_xd = y[7];
  ph->body_yd = y[8];
  ph->body_angled = y[9];
  ph->leg_angled = y[10];
  ph->foot_angled = y[11];
  ph->hoof_angled = y[12];

  ph->time = ph->new_time;

  derivs(ph,ph->time,y,dydt);

  ph->body_xdd = dydt[7];
  ph->body_ydd = dydt[8];
  ph->body_angledd = dydt[9];
  ph->leg_angledd = dydt[10];
  ph->foot_angledd = dydt[11];
  ph->hoof_angledd = dydt[12];



}


void derivs (ph,x,y,dydt)
   float x;
    float y[12], dydt[12];
    pHopper ph;


{
 float **A, *B, **matrix(), d, *vector(), fabs();
 int n, *indx, *ivector();
 void ludcmp(), lubksb(), free_vector(), free_ivector(),
    free_matrix();
 float tbody_x, tbody_xd, tbody_y, tbody_yd,tbody_angle,tbody_angled,
     tleg_angle,tleg_angled,tfoot_angle,tfoot_angled,
     thoof_angle,thoof_angled;

 A = matrix(1,12,1,12);
 B = vector(1,12);
 indx = ivector(1,12);


 tbody_x = y[1];
 tbody_y = y[2];
 tbody_angle = y[3];
 tleg_angle = y[4];
 tfoot_angle = y[5];
 thoof_angle = y[6];
 tbody_xd = y[7];
 tbody_yd = y[8];
 tbody_angled = y[9];
 tleg_angled = y[10];
 tfoot_angled = y[11];
 thoof_angled = y[12];

A[1][1] = 1.0;
```

```
A[1][2] = 0.0;
A[1][3] = 0.0;
A[1][4] = 0.0;
A[1][5] = 0.0;
A[1][6] = 0.0;
A[1][7] = 0.0;
A[1][8] = 0.0;
A[1][9] = 0.0;
A[1][10] = 0.0;
A[1][11] = 0.0;
A[1][12] = 0.0;

A[2][1] = 0.0;
A[2][2] = 1.0;
A[2][3] = 0.0;
A[2][4] = 0.0;
A[2][5] = 0.0;
A[2][6] = 0.0;
A[2][7] = 0.0;
A[2][8] = 0.0;
A[2][9] = 0.0;
A[2][10] = 0.0;
A[2][11] = 0.0;
A[2][12] = 0.0;

A[3][1] = 0.0;
A[3][2] = 0.0;
A[3][3] = 1.0;
A[3][4] = 0.0;
A[3][5] = 0.0;
A[3][6] = 0.0;
A[3][7] = 0.0;
A[3][8] = 0.0;
A[3][9] = 0.0;
A[3][10] = 0.0;
A[3][11] = 0.0;
A[3][12] = 0.0;

A[4][1] = 0.0;
A[4][2] = 0.0;
A[4][3] = 0.0;
A[4][4] = 1.0;
A[4][5] = 0.0;
A[4][6] = 0.0;
A[4][7] = 0.0;
A[4][8] = 0.0;
A[4][9] = 0.0;
A[4][10] = 0.0;
A[4][11] = 0.0;
A[4][12] = 0.0;

A[5][1] = 0.0;
A[5][2] = 0.0;
A[5][3] = 0.0;
A[5][4] = 0.0;
A[5][5] = 1.0;
A[5][6] = 0.0;
A[5][7] = 0.0;
```

```
A[5][8] = 0.0;
A[5][9] = 0.0;
A[5][10] = 0.0;
A[5][11] = 0.0;
A[5][12] = 0.0;

A[6][1] = 0.0;
A[6][2] = 0.0;
A[6][3] = 0.0;
A[6][4] = 0.0;
A[6][5] = 0.0;
A[6][6] = 1.0;
A[6][7] = 0.0;
A[6][8] = 0.0;
A[6][9] = 0.0;
A[6][10] = 0.0;
A[6][11] = 0.0;
A[6][12] = 0.0;

A[7][1] = 0.0;
A[7][2] = 0.0;
A[7][3] = 0.0;
A[7][4] = 0.0;
A[7][5] = 0.0;
A[7][6] = 0.0;
A[7][7] = ph->body_mass + ph->foot_mass +
          ph->hoof_mass + ph->leg_mass;
A[7][8] = 0.;
A[7][9] = ph->e*(ph->foot_mass + ph->hoof_mass + ph->leg_mass)*
          sin(tbody_angle);
A[7][10] = (ph->leg_length*cos(tleg_angle)*
          (2*ph->foot_mass + 2*ph->hoof_mass + ph->leg_mass))/2;
A[7][11] = -(ph->foot_length*(ph->foot_mass + 2*ph->hoof_mass)*
          sin(tfoot_angle))/2;
A[7][12] = (ph->hoof_length*ph->hoof_mass*cos(thoof_angle))/2;

A[8][1] = 0.0;
A[8][2] = 0.0;
A[8][3] = 0.0;
A[8][4] = 0.0;
A[8][5] = 0.0;
A[8][6] = 0.0;
A[8][7] = A[7][8];
A[8][8] = ph->body_mass + ph->foot_mass + ph->hoof_mass +
          ph->leg_mass;
A[8][9] = -(ph->e*cos(tbody_angle)*(ph->foot_mass +
          ph->hoof_mass + ph->leg_mass));
A[8][10] = (ph->leg_length*(2*ph->foot_mass + 2*ph->hoof_mass +
          ph->leg_mass)*sin(tleg_angle))/2;
A[8][11] = -(ph->foot_length*cos(tfoot_angle)*
          (ph->foot_mass + 2*ph->hoof_mass))/2;
A[8][12] = (ph->hoof_length*ph->hoof_mass*sin(thoof_angle))/2;

A[9][1] = 0.0;
A[9][2] = 0.0;
A[9][3] = 0.0;
A[9][4] = 0.0;
A[9][5] = 0.0;
```

```
A[9][6] = 0.0;
A[9][7] = A[7][9];
A[9][8] = A[8][9];
A[9][9] = ph->body_I + ph->e*ph->e*ph->foot_mass + ph->e * ph->e *
          ph->hoof_mass + ph->e*ph->e*ph->leg_mass;
A[9][10] = (ph->e*ph->leg_length*(4*ph->foot_mass + ph->leg_mass)*
          sin(tbody_angle - tleg_angle))/2;
A[9][11] = (ph->e*ph->foot_length*cos(tbody_angle + tfoot_angle)*
          (ph->foot_mass + 2*ph->hoof_mass))/2;
A[9][12] = (ph->e*ph->hoof_length*ph->hoof_mass*
          sin(tbody_angle - thoof_angle))/2;

A[10][1] = 0.0;
A[10][2] = 0.0;
A[10][3] = 0.0;
A[10][4] = 0.0;
A[10][5] = 0.0;
A[10][6] = 0.0;
A[10][7] = A[7][10];
A[10][8] = A[8][10];
A[10][9] = A[9][10];
A[10][10] = ph->leg_I + ph->leg_length*ph->leg_length*ph->foot_mass +
          ph->leg_length*ph->leg_length*ph->hoof_mass +
          (ph->leg_length*ph->leg_length*ph->leg_mass)/4;
A[10][11] = -(ph->foot_length*ph->leg_length*(ph->foot_mass +
          2*ph->hoof_mass)*sin(tfoot_angle + tleg_angle))/2;
A[10][12] = (ph->hoof_length*ph->leg_length*ph->hoof_mass*
          cos(thoof_angle - tleg_angle))/2;

A[11][1] = 0.0;
A[11][2] = 0.0;
A[11][3] = 0.0;
A[11][4] = 0.0;
A[11][5] = 0.0;
A[11][6] = 0.0;
A[11][7] = A[7][11];
A[11][8] = A[8][11];
A[11][9] = A[9][11];
A[11][10] = A[10][11];
A[11][11] = ph->foot_I + (ph->foot_length*ph->foot_length*ph->foot_mass)/4 +
          ph->foot_length*ph->foot_length*ph->hoof_mass;
A[11][12] = -(ph->foot_length*ph->hoof_length*ph->hoof_mass*
          sin(tfoot_angle + thoof_angle))/2;

A[12][1] = 0.0;
A[12][2] = 0.0;
A[12][3] = 0.0;
A[12][4] = 0.0;
A[12][5] = 0.0;
A[12][6] = 0.0;
A[12][7] = A[7][12];
A[12][8] = A[8][12];
A[12][9] = A[9][12];
A[12][10] = A[10][12];
A[12][11] = A[11][12];
A[12][12] = ph->hoof_I + (ph->hoof_length*ph->hoof_length*ph->hoof_mass)/4:
```

```
B[1] = tbody_xd;
B[2] = tbody_yd;
B[3] = tbody_angled;
B[4] = tleg_angled;
B[5] = tfoot_angled;
B[6] = thoof_angled;

B[7] = (ph->hoof_width*cos(thoof_angle)*(ph->ground1_stiffness - ph-
>ground2_stiffness))/2 +
        ph->foot_length*cos(tfoot_angle)*tfoot_angled*tfoot_angled*(ph->foot_mass/2 +
ph->hoof_mass) +
        ph->ground1_stiffness*(ph->hoof1_xtd - tbody_x) +
        ph->ground2_stiffness*(ph->hoof2_xtd - tbody_x) -
        ph->e*cos(tbody_angle)*tbody_angled*tbody_angled*(ph->foot_mass + ph-
>hoof_mass + ph->leg_mass) +
     (ph->ground1_stiffness + ph->ground2_stiffness)*(ph->e*cos(tbody_angle) - ph-
>foot_length*cos(tfoot_angle) -
         ph->hoof_length*sin(thoof_angle) - ph->leg_length*sin(tleg_angle)) +
    (ph->ground1_viscosity + ph->ground2_viscosity)*(-(ph-
>hoof_length*cos(thoof_angle)*thoof_angled) -
         ph->leg_length*cos(tleg_angle)*tleg_angled - tbody_xd -
         ph->e*tbody_angled*sin(tbody_angle) +
         ph->foot_length*tfoot_angled*sin(tfoot_angle)) +
        (ph->hoof_length*ph->hoof_mass*thoof_angled*thoof_angled*sin(thoof_angle))/2
-
        (ph->hoof_width*thoof_angled*(ph->ground1_viscosity - ph-
>ground2_viscosity)*sin(thoof_angle))/2 +
        ph->leg_length*tleg_angled*tleg_angled*(ph->foot_mass + ph->hoof_mass + ph-
>leg_mass/2)*sin(tleg_angle);

B[8] = -(ph->hoof_length*ph->hoof_mass*cos(thoof_angle)*thoof_angled*thoof_angled)/2
+
        (ph->hoof_width*cos(thoof_angle)*thoof_angled*(ph->ground1_viscosity - ph-
>ground2_viscosity))/2 +
        ph->leg_length*cos(tleg_angle)*tleg_angled*tleg_angled*(-ph->foot_mass - ph-
>hoof_mass - ph->leg_mass/2) -
        ph->gravity*(ph->body_mass + ph->foot_mass + ph->hoof_mass + ph->leg_mass) +
     (ph->ground1_stiffness + ph->ground2_stiffness)*
                (-tbody_y + ph->hoof_length*cos(thoof_angle) + ph-
>leg_length*cos(tleg_angle) +
        ph->e*sin(tbody_angle) + ph->foot_length*sin(tfoot_angle)) +
     (ph->ground1_viscosity + ph->ground2_viscosity)*(ph-
>e*cos(tbody_angle)*tbody_angled +
         ph->foot_length*cos(tfoot_angle)*tfoot_angled - tbody_yd -
         ph->hoof_length*thoof_angled*sin(thoof_angle) -
         ph->leg_length*tleg_angled*sin(tleg_angle)) +
    ph->e*tbody_angled*tbody_angled*(-ph->foot_mass - ph->hoof_mass - ph-
>leg_mass)*sin(tbody_angle) +
    ph->foot_length*tfoot_angled*tfoot_angled*(-ph->foot_mass/2 - ph-
>hoof_mass)*sin(tfoot_angle) +
    (ph->hoof_width*(ph->ground1_stiffness - ph->ground2_stiffness)*sin(thoof_angle))/2;

B[9] = -ph->hip_torque + (ph->e*ph->hoof_length*ph->hoof_mass*cos(tbody_angle -
thoof_angle)*
        thoof_angled*thoof_angled)/2 -
        (ph->e*ph->hoof_width*cos(tbody_angle - thoof_angle)*thoof_angled*
        (ph->ground1_viscosity - ph->ground2_viscosity))/2 +
```

```
        ph->e*ph->leg_length*cos(tbody_angle - tleg_angle)*tleg_angled*tleg_angled*
        (ph->foot_mass + ph->hoof_mass + ph->leg_mass/2) +
        ph->e*ph->gravity*cos(tbody_angle)*(ph->foot_mass + ph->hoof_mass + ph-
>leg_mass) +
     (ph->ground1_stiffness + ph->ground2_stiffness)*
        (ph->e*tbody_y*cos(tbody_angle) -
        ph->e*ph->hoof_length*cos(tbody_angle - thoof_angle) -
      ph->e*ph->leg_length*cos(tbody_angle - tleg_angle) -
        ph->e*ph->foot_length*sin(tbody_angle + tfoot_angle)) +
      (ph->ground1_viscosity + ph->ground2_viscosity)*(-(ph->e*ph->e*tbody_angled) -
         ph->e*ph->foot_length*cos(tbody_angle + tfoot_angle)*tfoot_angled +
         ph->e*cos(tbody_angle)*tbody_yd - ph->e*tbody_xd*sin(tbody_angle) -
         ph->e*ph->hoof_length*thoof_angled*sin(tbody_angle - thoof_angle) -
         ph->e*ph->leg_length*tleg_angled*sin(tbody_angle - tleg_angle)) +
     ph->e*ph->ground1_stiffness*(ph->hoof1_xtd - tbody_x)*sin(tbody_angle) +
     ph->e*ph->ground2_stiffness*(ph->hoof2_xtd - tbody_x)*sin(tbody_angle) +
     ph->e*ph->foot_length*tfoot_angled*tfoot_angled*(ph->foot_mass/2 +
        ph->hoof_mass)*sin(tbody_angle + tfoot_angle) +
    (ph->e*ph->hoof_width*(ph->ground1_stiffness - ph-
>ground2_stiffness)*sin(tbody_angle - thoof_angle))/2;

B[10] = ph->hip_torque + (ph->leg_length*ph->hoof_width*cos(thoof_angle - tleg_angle)*
        (ph->ground1_stiffness - ph->ground2_stiffness))/2 +
     ph->foot_length*ph->leg_length*cos(tfoot_angle +
tleg_angle)*tfoot_angled*tfoot_angled*
        (ph->foot_mass/2 + ph->hoof_mass) +
     ph->ground1_stiffness*ph->leg_length*cos(tleg_angle)*(ph->hoof1_xtd - tbody_x) +
     ph->ground2_stiffness*ph->leg_length*cos(tleg_angle)*(ph->hoof2_xtd - tbody_x) -
     ph->e*ph->leg_length*cos(tbody_angle - tleg_angle)*tbody_angled*tbody_angled*
        (ph->foot_mass + ph->hoof_mass + ph->leg_mass/2) +
      (ph->ground1_stiffness + ph->ground2_stiffness)*(ph->e*ph-
>leg_length*cos(tbody_angle - tleg_angle) -
         ph->foot_length*ph->leg_length*cos(tfoot_angle + tleg_angle) -
        ph->leg_length*tbody_y*sin(tleg_angle) -
         ph->hoof_length*ph->leg_length*sin(thoof_angle - tleg_angle)) +
     (ph->ground1_viscosity + ph->ground2_viscosity)*(-(ph->hoof_length*ph->leg_length*
              cos(thoof_angle - tleg_angle)*thoof_angled) -
        ph->leg_length*ph->leg_length*tleg_angled - ph-
>leg_length*cos(tleg_angle)*tbody_xd -
        ph->leg_length*tbody_yd*sin(tleg_angle) -
        ph->e*ph->leg_length*tbody_angled*sin(tbody_angle - tleg_angle) +
        ph->foot_length*ph->leg_length*tfoot_angled*sin(tfoot_angle + tleg_angle)) -
    ph->gravity*ph->leg_length*(ph->foot_mass + ph->hoof_mass + ph-
>leg_mass/2)*sin(tleg_angle) +
    (ph->hoof_length*ph->leg_length*ph-
>hoof_mass*thoof_angled*thoof_angled*sin(thoof_angle -
       tleg_angle))/2 -
    (ph->leg_length*ph->hoof_width*thoof_angled*(ph->ground1_viscosity - ph-
>ground2_viscosity)*
       sin(thoof_angle - tleg_angle))/2;

B[11] = -(ph->toe_viscosity*thoof_angled) +
       (ph->foot_length*ph->hoof_length*ph->hoof_mass*cos(tfoot_angle +
thoof_angle)*
     thoof_angled*thoof_angled)/2 + tfoot_angled*(-ph->ankle_viscosity - ph->toe_viscosity)
-
     (ph->foot_length*ph->hoof_width*cos(tfoot_angle + thoof_angle)*thoof_angled*
       (ph->ground1_viscosity - ph->ground2_viscosity))/2 +
```

```
    ph->gravity*ph->foot_length*cos(tfoot_angle)*(ph->foot_mass/2 + ph->hoof_mass) +
     ph->foot_length*ph->leg_length*cos(tfoot_angle + tleg_angle)*tleg_angled*
        tleg_angled*(ph->foot_mass/2 + ph->hoof_mass) -
    ph->ankle_stiffness*(tfoot_angle - ph->foot_angle_desired) +
    ph->toe_stiffness*(-(tfoot_angle - ph->foot_angle_desired) - (thoof_angle - ph-
>hoof_angle_desired)) +
     (ph->ground1_stiffness + ph->ground2_stiffness)*(ph-
>foot_length*tbody_y*cos(tfoot_angle) -
         ph->foot_length*ph->hoof_length*cos(tfoot_angle + thoof_angle) -
         ph->foot_length*ph->leg_length*cos(tfoot_angle + tleg_angle) -
         ph->e*ph->foot_length*sin(tbody_angle + tfoot_angle)) +
    (ph->ground1_viscosity + ph->ground2_viscosity)*(-(ph->e*ph-
>foot_length*cos(tbody_angle +
        tfoot_angle)*tbody_angled) -
         ph->foot_length*ph->foot_length*tfoot_angled +
        ph->foot_length*cos(tfoot_angle)*tbody_yd +
         ph->foot_length*tbody_xd*sin(tfoot_angle) +
          ph->foot_length*ph->hoof_length*thoof_angled*sin(tfoot_angle + thoof_angle) +
          ph->foot_length*ph->leg_length*tleg_angled*sin(tfoot_angle + tleg_angle)) +
    ph->ground1_stiffness*ph->foot_length*(-ph->hoof1_xtd + tbody_x)*sin(tfoot_angle) +
    ph->ground2_stiffness*ph->foot_length*(-ph->hoof2_xtd + tbody_x)*sin(tfoot_angle) +
    ph->e*ph->foot_length*tbody_angled*tbody_angled*(ph->foot_mass/2 + ph-
>hoof_mass)*
        sin(tbody_angle + tfoot_angle) -
    (ph->foot_length*ph->hoof_width*(ph->ground1_stiffness - ph->ground2_stiffness)*
        sin(tfoot_angle + thoof_angle))/2;

B[12] = -(ph->e*ph->hoof_length*ph->hoof_mass*cos(tbody_angle - thoof_angle)*
        tbody_angled*tbody_angled)/2 -
    ph->toe_viscosity*tfoot_angled + (ph->foot_length*ph->hoof_length*
        ph->hoof_mass*cos(tfoot_angle + thoof_angle)*
         tfoot_angled*tfoot_angled)/2 - ph->toe_viscosity*thoof_angled +
    ph->ground1_stiffness*ph->hoof_length*cos(thoof_angle)*(ph->hoof1_xtd - tbody_x) +
    ph->ground2_stiffness*ph->hoof_length*cos(thoof_angle)*(ph->hoof2_xtd - tbody_x) +
    ph->toe_stiffness*(-(tfoot_angle - ph->foot_angle_desired) - (thoof_angle - ph-
>hoof_angle_desired)) +
     (ph->ground1_stiffness - ph->ground2_stiffness)*((ph-
>hoof_width*tbody_y*cos(thoof_angle))/2 -
         (ph->leg_length*ph->hoof_width*cos(thoof_angle - tleg_angle))/2 -
         (ph->e*ph->hoof_width*sin(tbody_angle - thoof_angle))/2 -
         (ph->foot_length*ph->hoof_width*sin(tfoot_angle + thoof_angle))/2) +
     (ph->ground1_stiffness + ph->ground2_stiffness)*(ph->e*ph->hoof_length*
        cos(tbody_angle - thoof_angle) -
         ph->foot_length*ph->hoof_length*cos(tfoot_angle + thoof_angle) -
         ph->hoof_length*tbody_y*sin(thoof_angle) + ph->hoof_length*
        ph->leg_length*sin(thoof_angle - tleg_angle)) +
    (ph->ground1_viscosity - ph->ground2_viscosity)*(-(ph->e*ph->hoof_width*
        cos(tbody_angle - thoof_angle)*tbody_angled)/2 -
         (ph->foot_length*ph->hoof_width*cos(tfoot_angle + thoof_angle)*tfoot_angled)/2 +
         (ph->hoof_width*cos(thoof_angle)*tbody_yd)/2 -
         (ph->hoof_width*tbody_xd*sin(thoof_angle))/2 -
         (ph->leg_length*ph->hoof_width*tleg_angled*sin(thoof_angle - tleg_angle))/2) +
    (ph->ground1_viscosity + ph->ground2_viscosity)*(-(ph->hoof_length*ph-
>hoof_length*thoof_angled) -
         (ph->hoof_width*ph->hoof_width*thoof_angled)/4 -
         ph->hoof_length*ph->leg_length*cos(thoof_angle - tleg_angle)*tleg_angled -
        ph->hoof_length*cos(thoof_angle)*tbody_xd -
        ph->hoof_length*tbody_yd*sin(thoof_angle) -
```

```
            ph->e*ph->hoof_length*tbody_angled*sin(tbody_angle - thoof_angle) +
            ph->foot_length*ph->hoof_length*tfoot_angled*sin(tfoot_angle + thoof_angle)) -
       (ph->gravity*ph->hoof_length*ph->hoof_mass*sin(thoof_angle))/2 +
       ph->ground1_stiffness*ph->hoof_width*(ph->hoof1_xtd/2 - tbody_x/2)*sin(thoof_angle)
-
       ph->ground2_stiffness*ph->hoof_width*(ph->hoof2_xtd/2 - tbody_x/2)*sin(thoof_angle)
-
       (ph->hoof_length*ph->leg_length*ph->hoof_mass*tleg_angled*tleg_angled*
           sin(thoof_angle - tleg_angle))/2;



  ludcmp (A, 12, indx, &d);
  lubksb (A, 12, indx, B);

  dydt[1] =  B[1];
  dydt[2] =  B[2];
  dydt[3] =  B[3];
  dydt[4] =  B[4];
  dydt[5] =  B[5];
  dydt[6] =  B[6];
  dydt[7] =  B[7];
  dydt[8] =  B[8];
  dydt[9] =  B[9];
  dydt[10] = B[10];
  dydt[11] = B[11];
  dydt[12] = B[12];

  free_matrix(A, 1,12,1,12);
  free_vector(B, 1,12);
  free_ivector (indx, 1,12);

}

void ludcmp(a,n,indx,d)
int n, *indx;
float **a, *d;

 {
   int i, imax, j, k;
   float big, dum, sum, temp;
   float *vv, *vector();
   void nrerror(), free_vector();

   vv=vector(1,n);
   *d=1.0;
   for (i=1; i<=n; i++) {
    big = 0.0;
    for (j=1; j<=n; j++)
          if ((temp=fabs(a[i][j])) > big) big=temp;
    if (big == 0.0) nrerror ("Singular matrix in routine LUDCMP");
    vv[i] = 1.0/big;
 }

  for (j=1; j<=n; j++) {
   for (i=1;i<j;i++) {
         sum = a[i][j];
         for (k=1;k<i;k++) sum -= a[i][k] * a[k][j];
```

```
            a[i][j] = sum;
        }
        big = 0.0;
        for (i=j; i<=n; i++) {
            sum = a[i][j];
            for (k=1;k<j;k++)
              sum -= a[i][k]*a[k][j];
            a[i][j] = sum;
            if ((dum=vv[i]*fabs(sum)) >= big) {
              big = dum;
              imax = i;
            }
        }

        if (j != imax) {
            for (k=1; k<=n; k++) {
              dum = a[imax][k];
              a[imax][k] = a[j][k];
              a[j][k] = dum;
            }
            *d = -(*d);
            vv[imax] = vv[j];
        }
        indx[j]=imax;
        if (a[j][j] == 0.0) a[j][j] = TINY;
        if (j != n) {
            dum = 1.0/(a[j][j]);
            for (i=j+1;i<=n;i++) a[i][j] *= dum;
        }
      }
      free_vector (vv, 1, n);
    }


    void lubksb (a,n, indx, b)
    float **a, b[];
    int n, *indx;

    {
      int i, ii=0, ip, j;
      float sum;

      for (i=1; i<=n; i++) {
        ip = indx[i];
        sum=b[ip];
        b[ip]=b[i];
        if (ii)
            for (j=ii; j<=i-1;j++) sum -= a[i][j]*b[j];
        else if (sum) ii=i;
        b[i] = sum;
      }

      for (i=n;i>=1;i--) {
        sum=b[i];
        for (j=i+1; j<=n; j++) sum -= a[i][j]*b[j];
        b[i]=sum/a[i][i];
      }
    }
```

```
void nrerror(error_text)
    char error_text;
{
  void exit();

  printf( "Numerical Recipes run-time error ... \n");
  printf( "%s\n", error_text);
  printf( "...now exiting to system ... \n");

  exit(1);
}

double *dvector(nl,nh)
    int nl,nh;

{
  double *v;

  v=(double *)malloc((unsigned)(nh-nl+1)*sizeof(double));
  if (!v) nrerror("allocation failure in dvector()");
  return v-nl;
}

float *vector(nl,nh)
    int nl,nh;

{
  float *v;

  v=(float *)malloc((unsigned)(nh-nl+1)*sizeof(float));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl;
}

int *ivector (nl,nh)
    int nl, nh;

{
  int *v;

  v=(int *)malloc((unsigned)(nh-nl+1)*sizeof(int));
  if (!v) nrerror("allocation failure in ivector()");
  return v-nl;
}

double **dmatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;

{
  int i;
  double **m;

  m=(double **) malloc((unsigned)(nch-nrl+1)*sizeof(double*));
  if (!m) nrerror("allocation failure 1 in matrix()");
  m -= nrl;

  for(i=nrl;i<=nrh;i++) {
```

```
    m[i]=(double *) malloc((unsigned)(nch-ncl+1)*sizecf(double));
    if (!m[i]) nrerror("allocation failure 2 in matrix()");
    m[i] -= ncl;
  }
  return m;
}

float **matrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;

{
  int i;
  float **m;

  m=(float **) malloc((unsigned)(nch-nrl+1)*sizeof(float*));
  if (!m) nrerror("allocation failure 1 in matrix()");
  m -= nrl;

  for(i=nrl;i<=nrh;i++) {
    m[i]=(float *) malloc((unsigned)(nch-ncl+1)*sizeof(float));
    if (!m[i]) nrerror("allocation failure 2 in matrix()");
    m[i] -= ncl;
  }
  return m;
}

void free_dvsctor (v, nl,nh)
double *v;
int nl, nh;
{
  free((char*) (v+nl));
}

void free_vector (v, nl,nh)
float *v;
int nl, nh;
{
  free((char*) (v+nl));
}

double dabs(x)
    double x;
{
  if (x < 0.00000000000000)
    x = -x;
  return (x);
}

float fabs(x)
    float x;
{
  if (x < 0.00000000000000)
    x = -x;
  return (x);
}


void free_ivector (v, nl, nh)
```

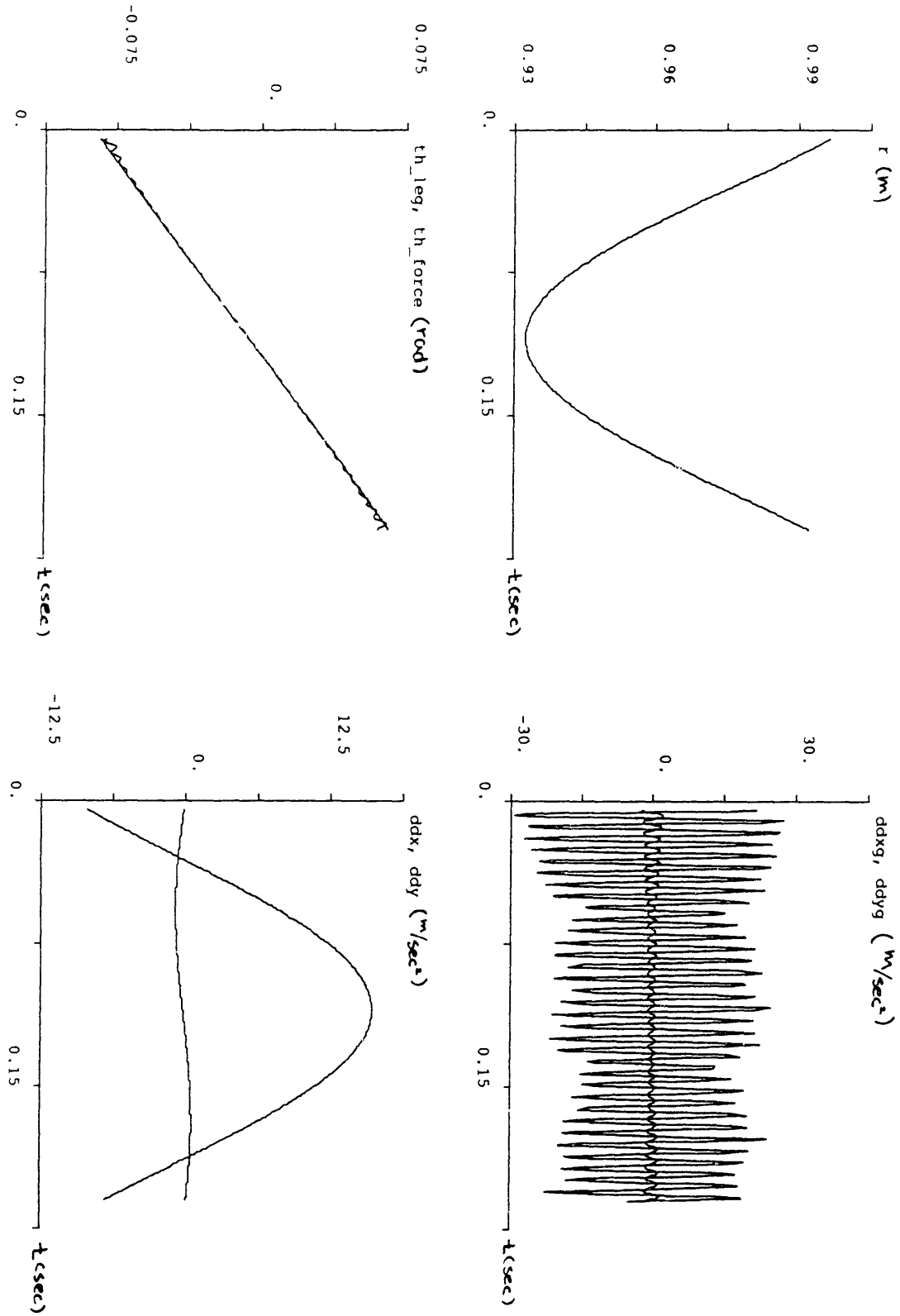
```
int *v, nl, nh;
{
 free((char*) (v+nl));
}

void free_dmatrix(m,nrl,nrh,ncl,nch)

double **m;
int nrl, nrh, ncl, nch;
{
 int i;

 for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));
 free((char*) (m+nrl));
}

void free_matrix(m,nrl,nrh,ncl,nch)

float **m;
int nrl, nrh, ncl, nch;
{
 int i;

 for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));
 free((char*) (m+nrl));
}


/*this file will use the Runge-Kutta method of approximating the integral
  of a function.  This file also makes use of variable step sizes.

          input variables must be passed to
          this function as follows :

          ystart   should be an array of initial
                conditions for each of the state
                variables.(xo,yo,theta0)

                   nvar     is the number of variables in the
                        ystart array

         x1     initial time

          x2      time after one time step
                        x2-x1=timestep

          eps     the accuracy specified for the
                        integrator

                   h1       is the first guess for the step size

          hmin    is the minimum stepsize allowed

          nok     is the # of good steps taken

                   nbad     is the number of bad steps
                        (they are retried)
```

```
                derivs   is a user supplied routine to calculate
              the derivatives for each state variable
                        at a certain time t

          rkqc    is the variable step function          */

#define MAXSTP 10000

int kmax=0,kount=0;
float *xp=0,**yp=0,dxsav=0;


void odeint(ph,ystart,nvar,x1,x2,eps,h1,hmin,nok,nbad,derivs,rkqc)
    pHopper ph;
    float ystart[],x1,x2,eps,h1,hmin;
    int nvar, *nok, *nbad;
    void (*derivs)();
    void (*rkqc)();

{
 int nstp,i;
 float xsav,x,hnext,hdid,h;
 float *yscal,*y,*dydx,*vector(),fabs();
 void nrerror(),free_vector();

 yscal=vector(1,nvar);
 y=vector(1,nvar);
 dydx=vector(1,nvar);
 x=x1;
 h=(x2>x1) ? fabs(h1) : -fabs(h1);

 kount = 0;

 for(i=1;i<=nvar;i++) y[i]=ystart[i];
 if (kmax>0) xsav=x-dxsav*2.0;
 for (nstp=1;nstp<=MAXSTP;nstp++) {
  (*derivs)(ph,x,y,dydx);
/*    printf("%f,%f,%f,%f,%f\n", y[2],dydx[2], dydx[8],h, ph->time); */
   for (i=1;i<=nvar;i++)
    yscal[i]=fabs(y[i])+fabs(dydx[i]*h)+TINY;
   if (kmax>0){
    if (fabs(x-xsav)>fabs(dxsav)) {
          if (kount<kmax-1) {
           xp[++kount]=x;
           for (i=1;i<=nvar;i++) yp[i][kount]=y[i];
           xsav=x;
          }
   }
  }
   if((x+h-x2)*(x+h-x1) > 0.0) h=x2-x;
   (*rkqc)(ph,y,dydx,nvar,&x,h,eps,yscal,&hdid,&hnext,derivs);
  if(hdid == h) ++(*nok); else ++(*nbad);
  if ((x-x2)*(x2-x1)>=0.0) {
   for (i=1;i<=nvar;i++) ystart[i]=y[i];
   if(kmax) {
          xp[++kount]=x;
          for(i=1;i<=nvar;i++)yp[i][kount]=y[i];
   }
```

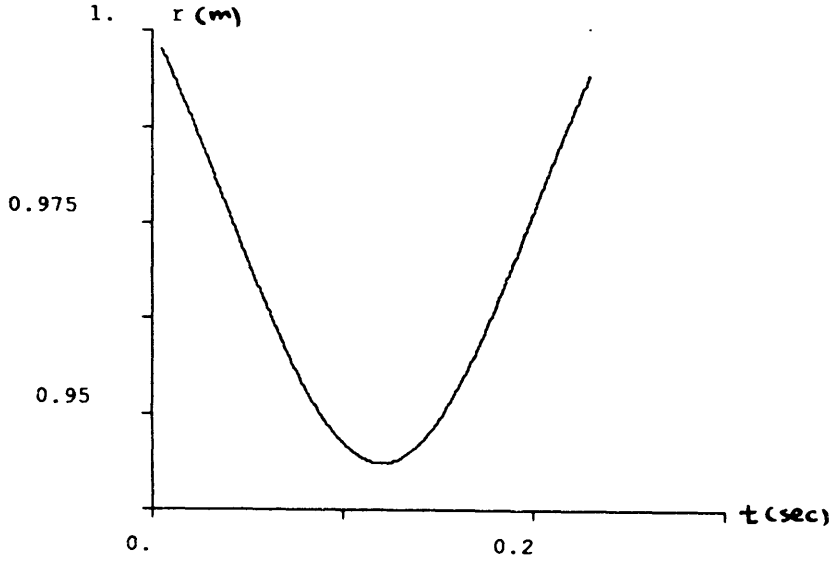
```
      free_vector(dydx,1,nvar);
      free_vector(y,1,nvar);
      free_vector(yscal,1,nvar);
      return;
    }
    if (fabs(hnext)<=hmin) nrerror("step size too small in ODEINT");
    h=hnext;
  }
  nrerror("Too many steps in routine ODEINT");
}


#define PGROW -0.20
#define PSHRINK -0.25
#define FCOR 0.06666666
#define SAFETY 0.9
#define ERRCON  6.0e-4

void rkqc(ph,y,dydx,n,x,htry,eps,yscal,hdid,hnext,derivs)
    pHopper ph;
     float y[],dydx[],*x,htry,eps,yscal[],*hdid,*hnext;
    void (*derivs)();
    int n;

{
  int i;
  float xsav,hh,h,temp,errmax;
  float *dysav,*ysav,*ytemp,*vector();
  void rk4(),nrerror(),free_vector();

  dysav=vector(1,n);
  ysav=vector(1,n);
  ytemp=vector(1,n);
  xsav=(*x);
  for(i=1;i<=n;i++) {
    ysav[i]=y[i];
    dysav[i]=dydx[i];
  }
  h=htry;
  for(;;) {
    hh=0.5*h;
    rk4(ph,ysav,dysav,n,xsav,hh,ytemp,derivs);
    *x=xsav+hh;
    (*derivs)(ph,*x,ytemp,dydx);
    rk4(ph,ytemp,dydx,n,*x,hh,y,derivs);
    *x=xsav+h;
    if(*x == xsav) nrerror("Stepsize too small in routine RKQC");
    rk4(ph,ysav,dysav,n,xsav,h,ytemp,derivs);
    errmax=0.0;
    for (i=1;i<=n;i++){
      ytemp[i]=y[i]-ytemp[i];
      temp=fabs(ytemp[i]/yscal[i]);
      if(errmax<temp)errmax=temp;
  }
    errmax /= eps;
    if (errmax <= 1.0){
      *hdid=h;
      *hnext=(errmax > ERRCON ?
```

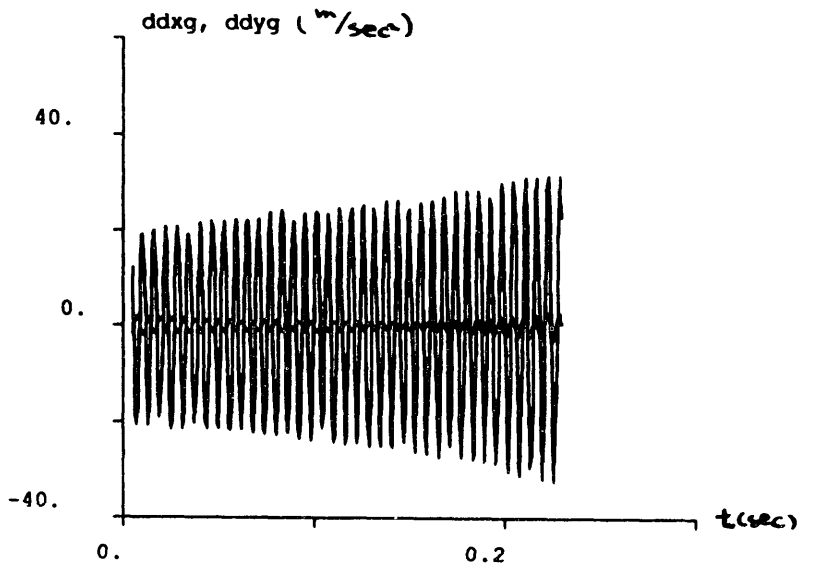
```
            SAFETY*h*exp(PGROW*log(errmax)) : 4.0*h);
        break;
      }
      h=SAFETY*h*exp(PSHRINK*log(errmax));
    }
    for (i=1;i<=n;i++) y[i] += ytemp[i]*FCOR;
    free_vector(ytemp,1,n);
    free_vector(dysav,1,n);
    free_vector(ysav,1,n);
}


void rk4(ph,y,dydx,n,x,h,yout,derivs)
    pHopper ph;
    float y[],dydx[],x,h,yout[];
    void (*derivs)();
    int n;

{
    int i;
    float xh,hh,h6,*dym,*dyt,*yt,*vector();
    void free_vector();

    dym=vector(1,n);
    dyt=vector(1,n);
    yt=vector(1,n);
    hh=h*0.5;
    h6=h/6.0;
    xh=x+hh;
    for (i=1;i<=n;i++) yt[i]=y[i]+hh*dydx[i];
    (*derivs)(ph,xh,yt,dyt);
    for (i=1;i<=n;i++) yt[i]=y[i]+hh*dyt[i];
    (*derivs)(ph,xh,yt,dym);
    for(i=1;i<=n;i++) {
      yt[i]=y[i]+h*dym[i];
      dym[i] += dyt[i];
    }
    (*derivs)(ph,x+h,yt,dyt);
    for(i=1;i<=n;i++)
      yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
    free_vector(yt,1,n);
    free_vector(dyt,1,n);
    free_vector(dym,1,n);
}
```

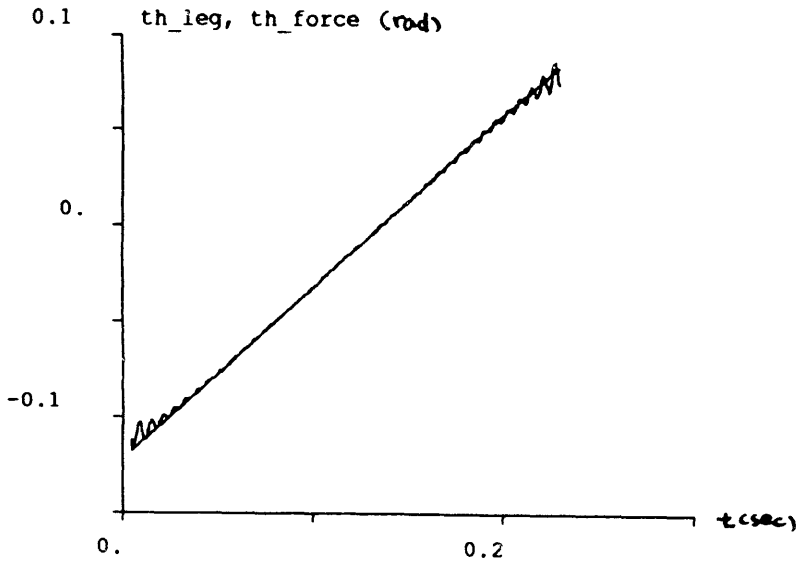
Appendix G:
Graphical Results of the Simple Model

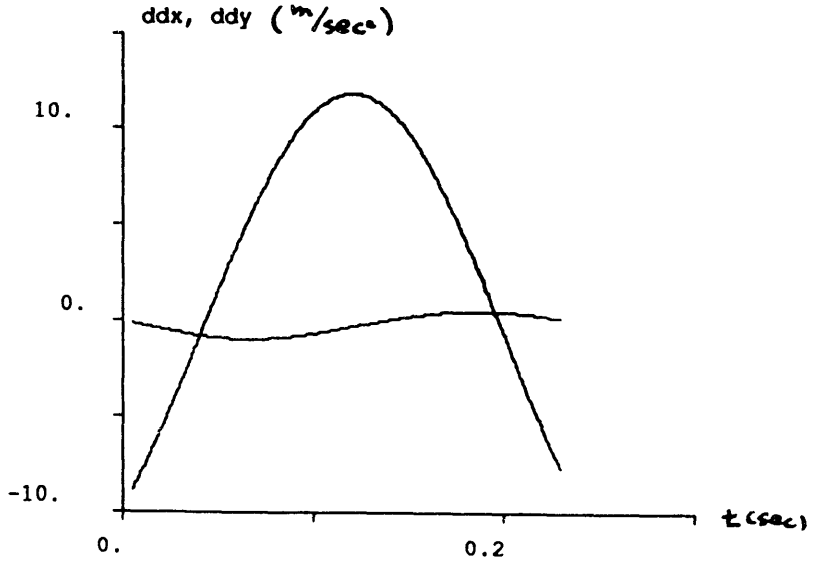


$\theta_{leg-o} = -7°$, $\theta_{ca} = -70°$

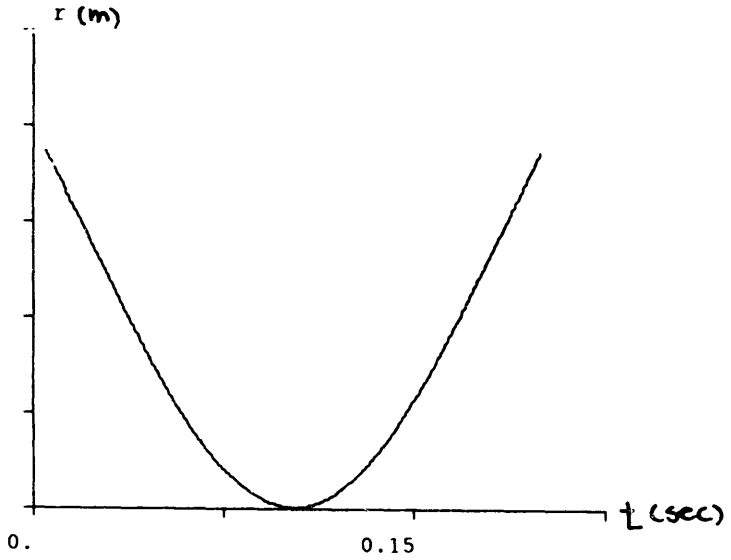


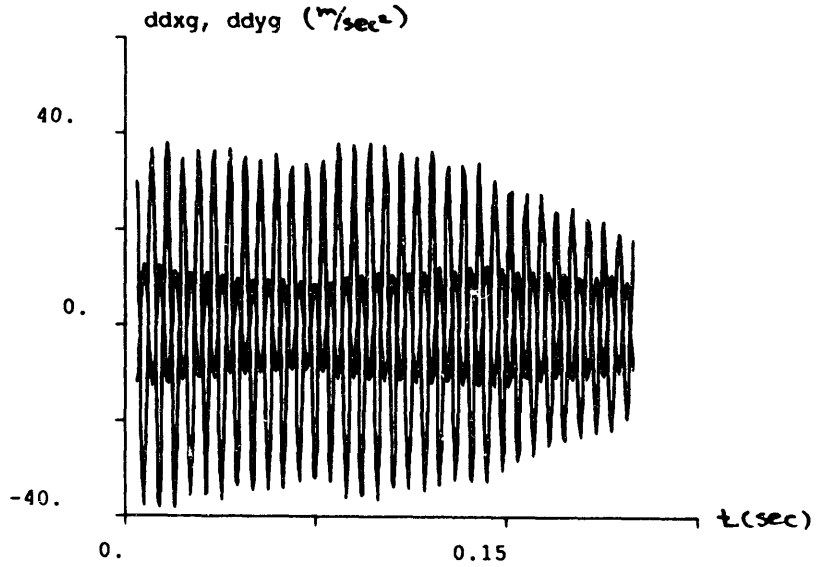


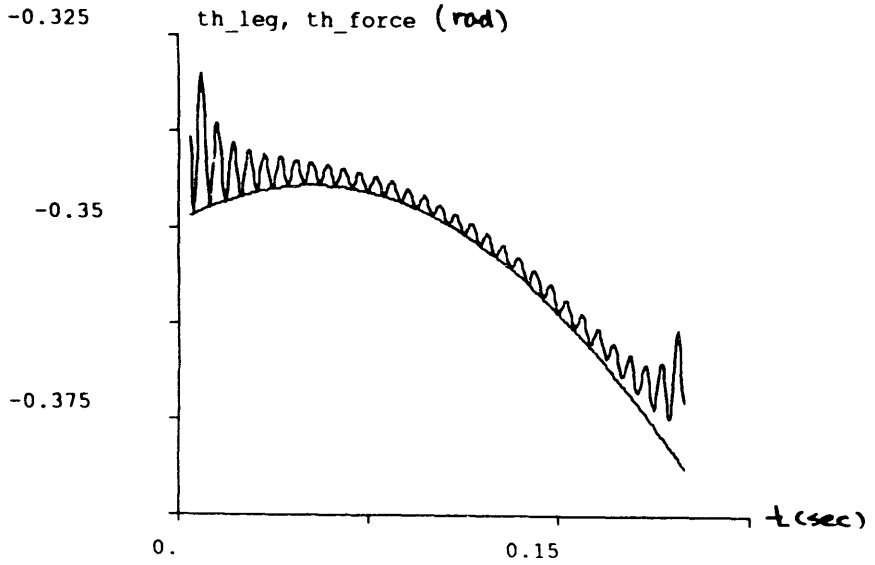


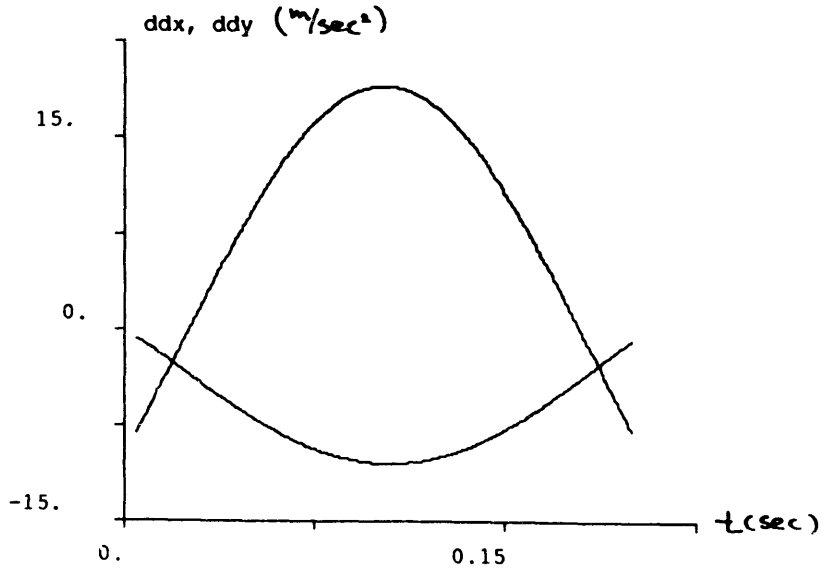


$\theta_{leg_0} = -20°$, $\theta_{fa} = -30°$

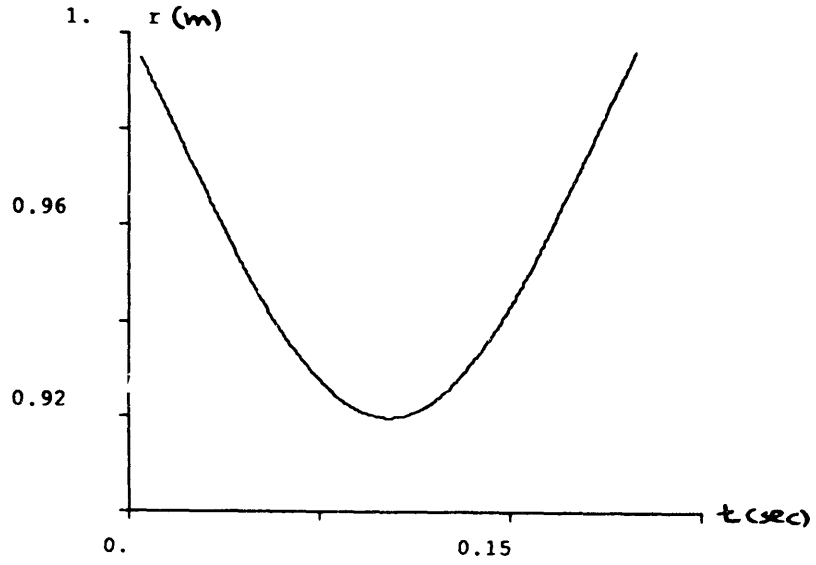


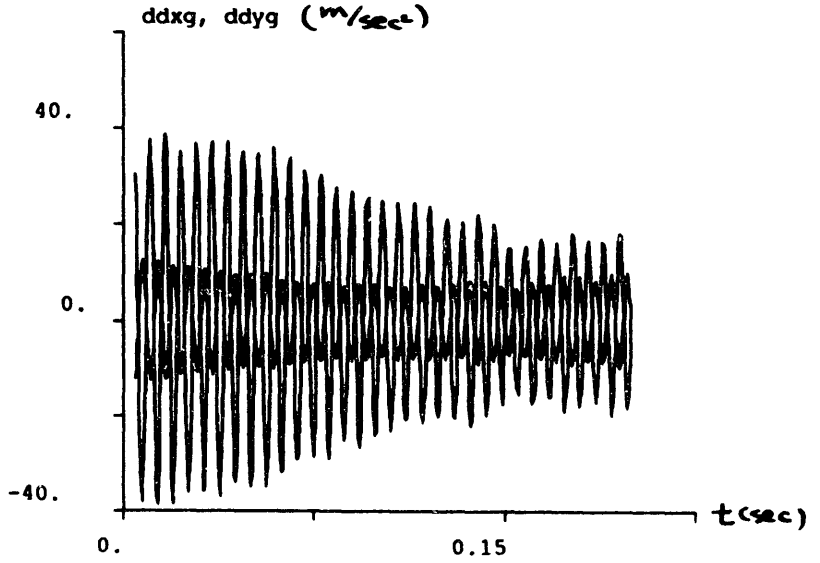


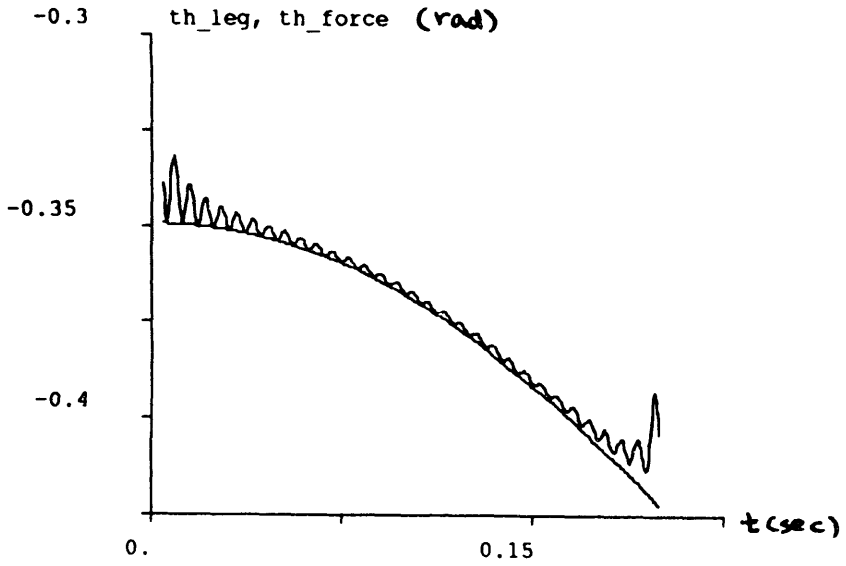


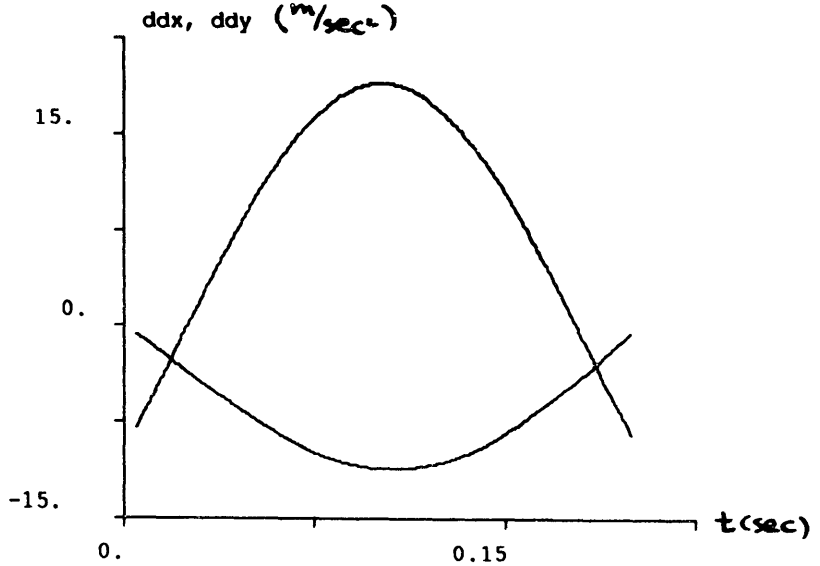


$\theta_{leg_0} = -26°$, $\theta_{ft} = -26°$

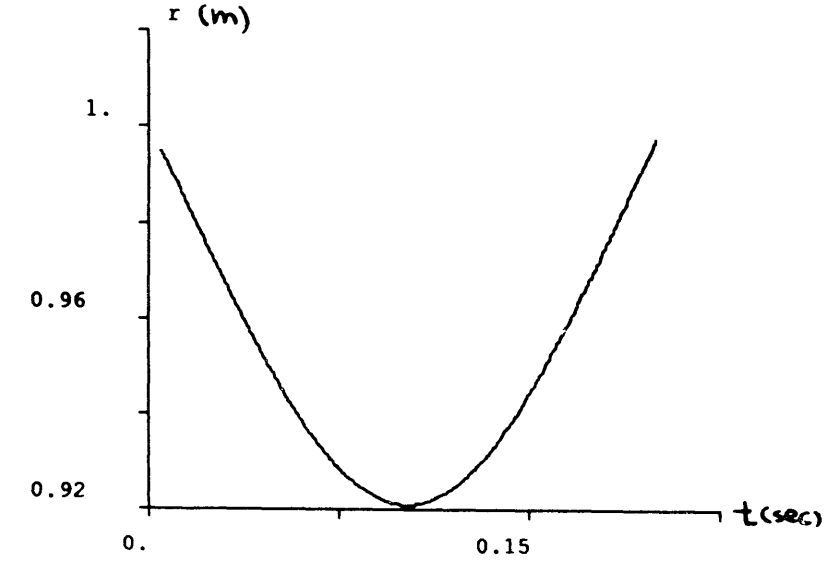


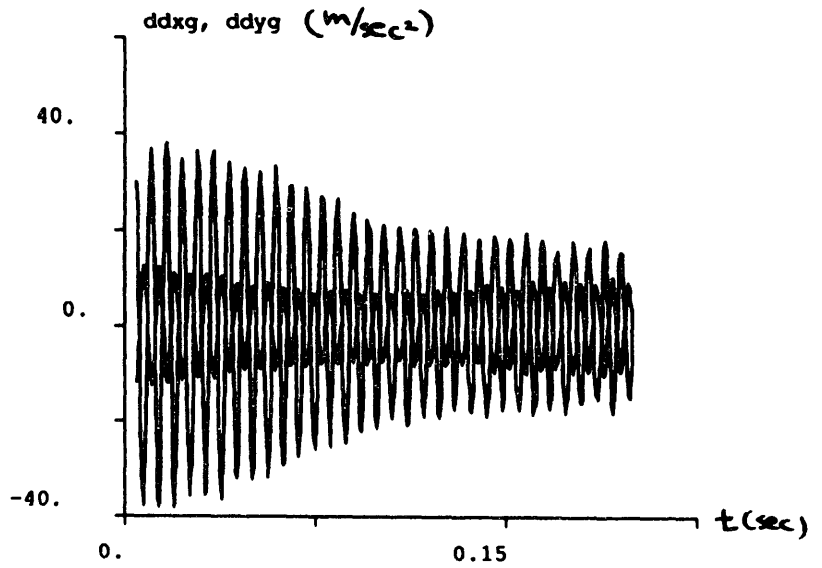


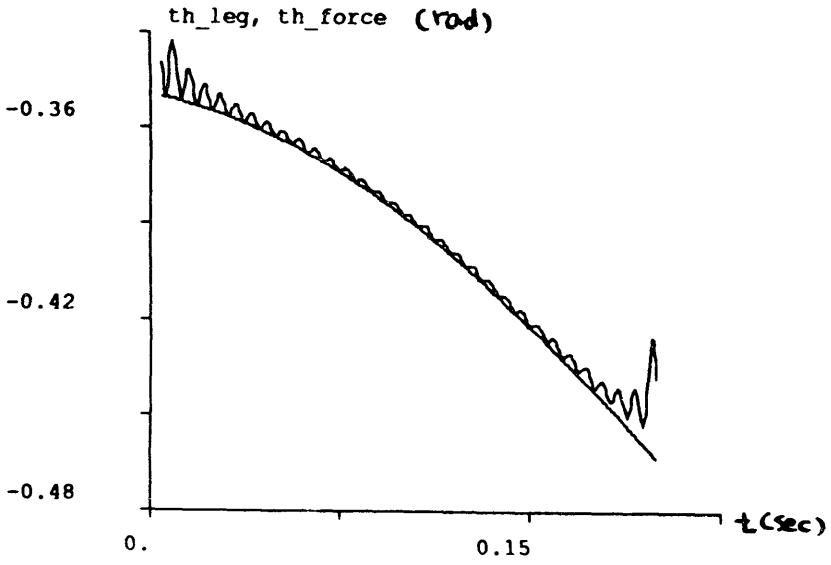


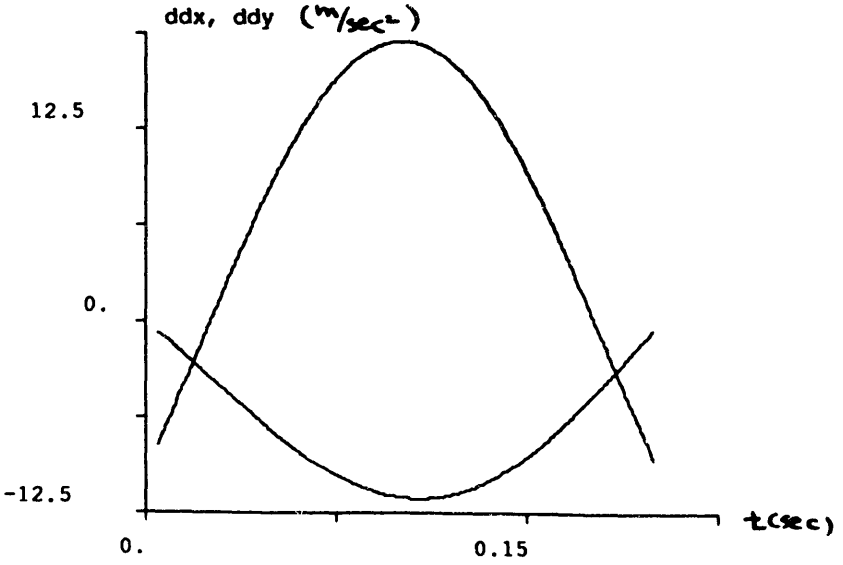


$\theta_{leg-0} = -20°$, $\theta_{ff} = 40°$

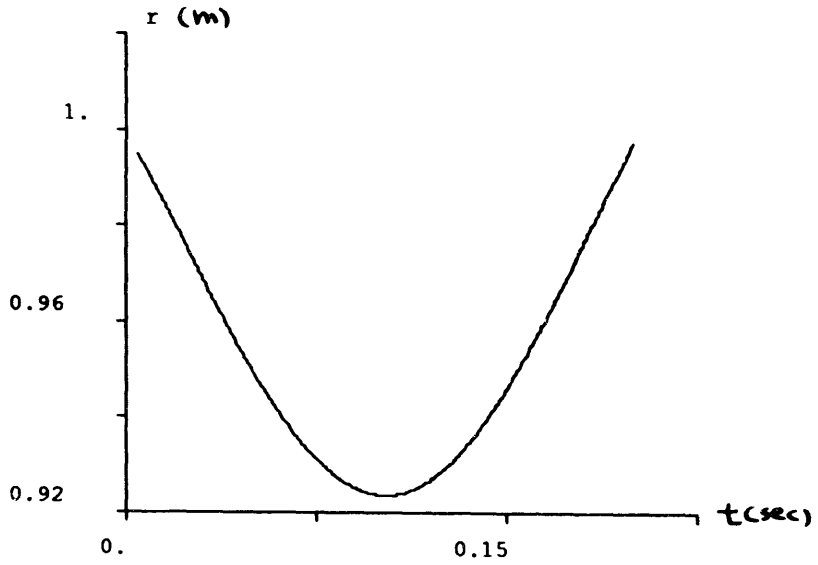


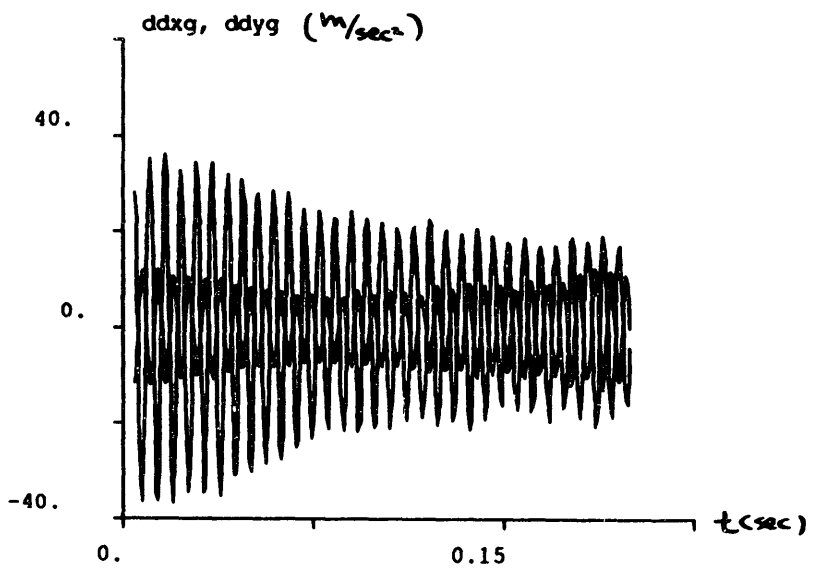


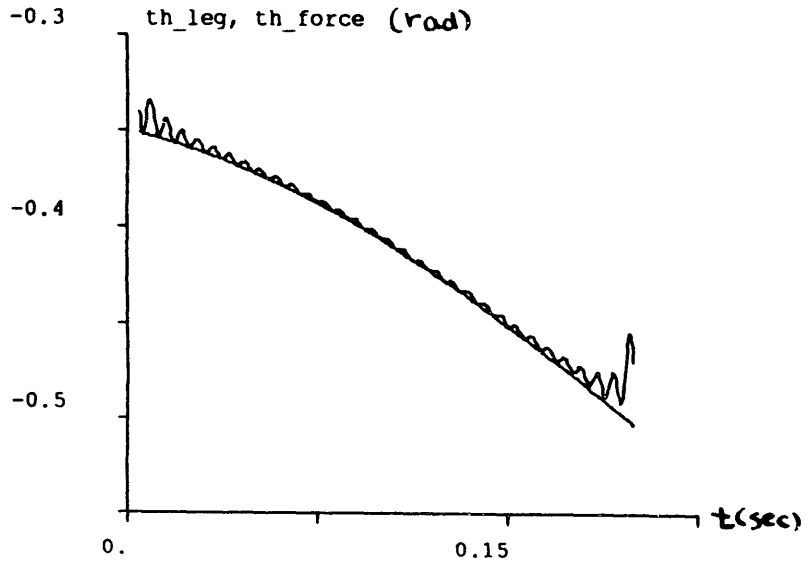


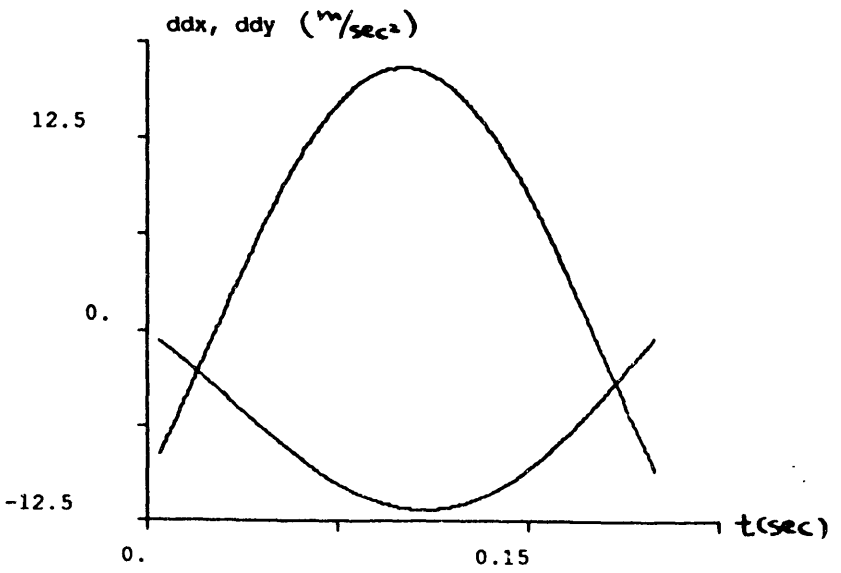


$\theta_{leg_0} = -20°$, $\theta_{td} = 0°$

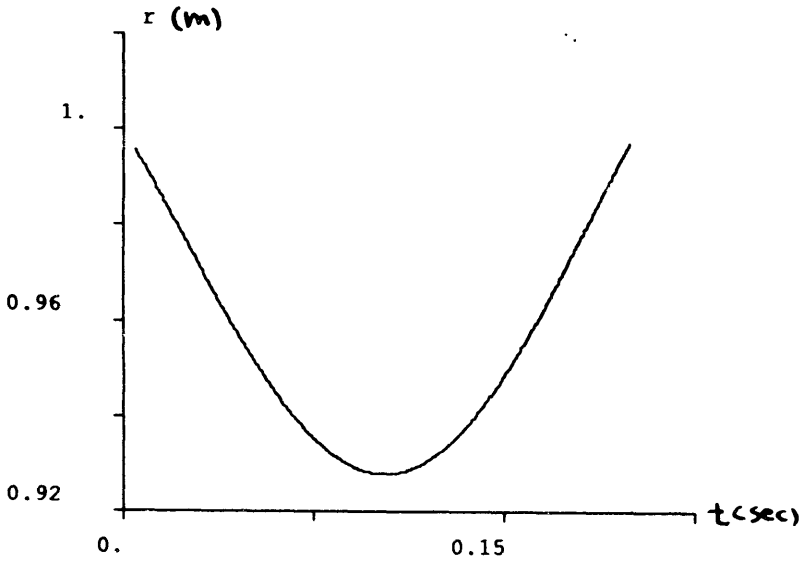


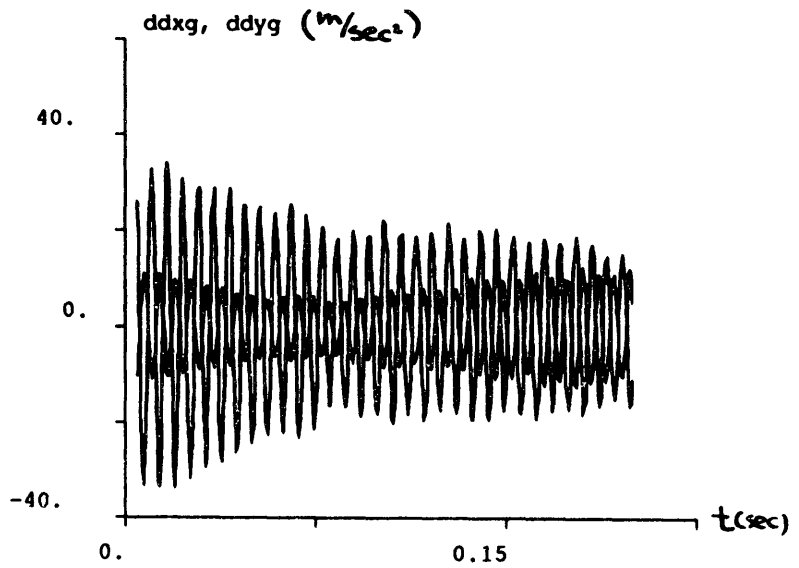


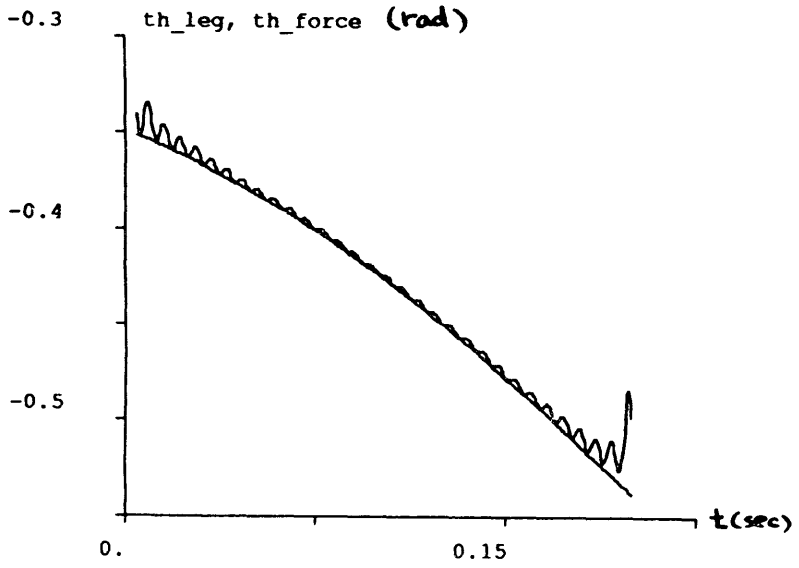


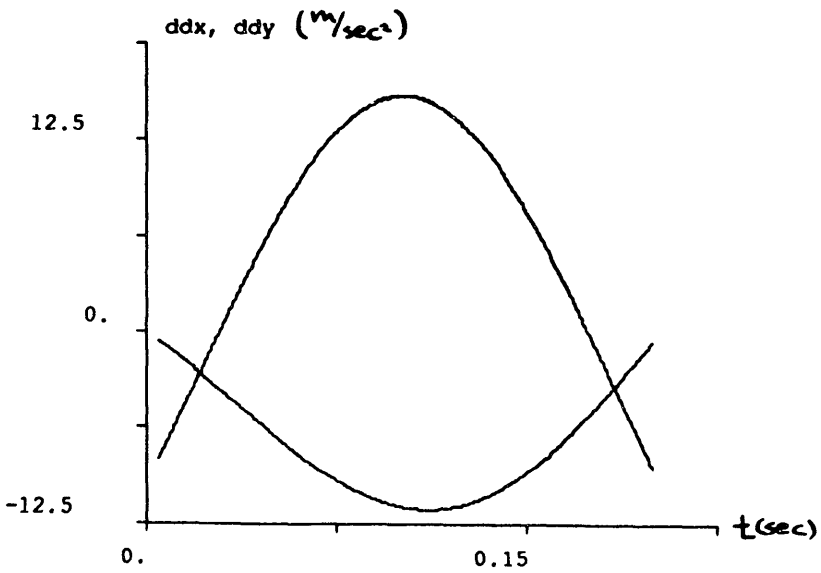


$\theta_{leg_0} = -20°$, $\theta_{ext} = 10°$

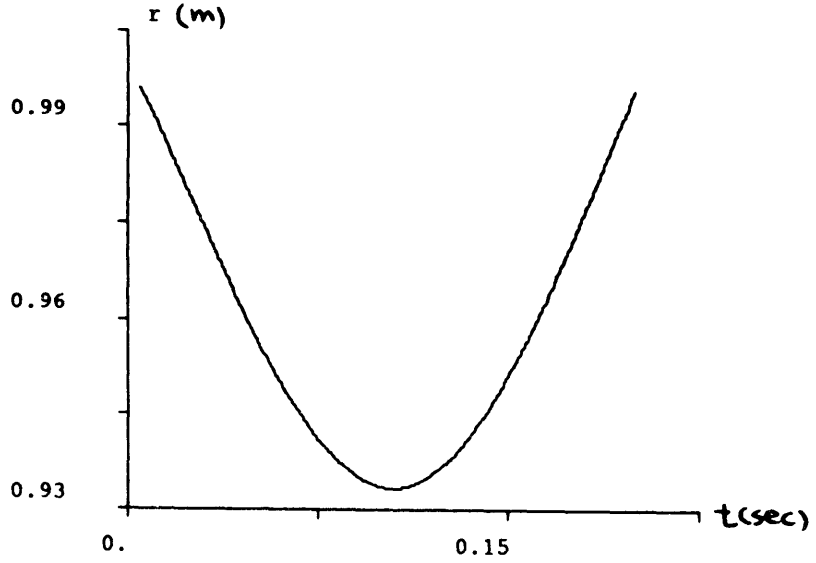


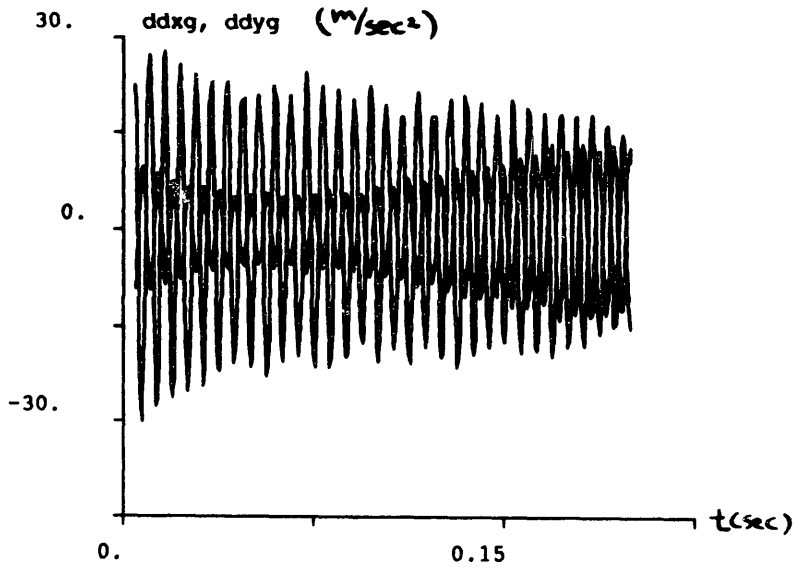


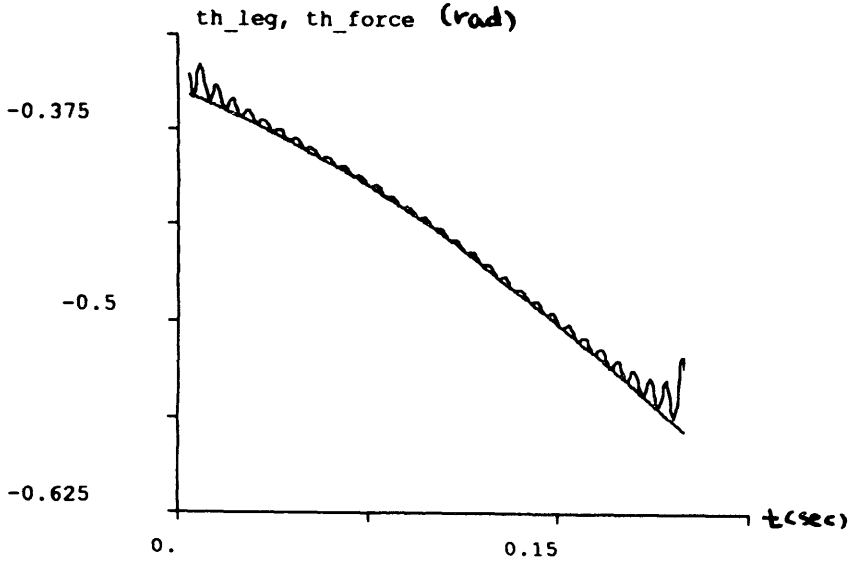


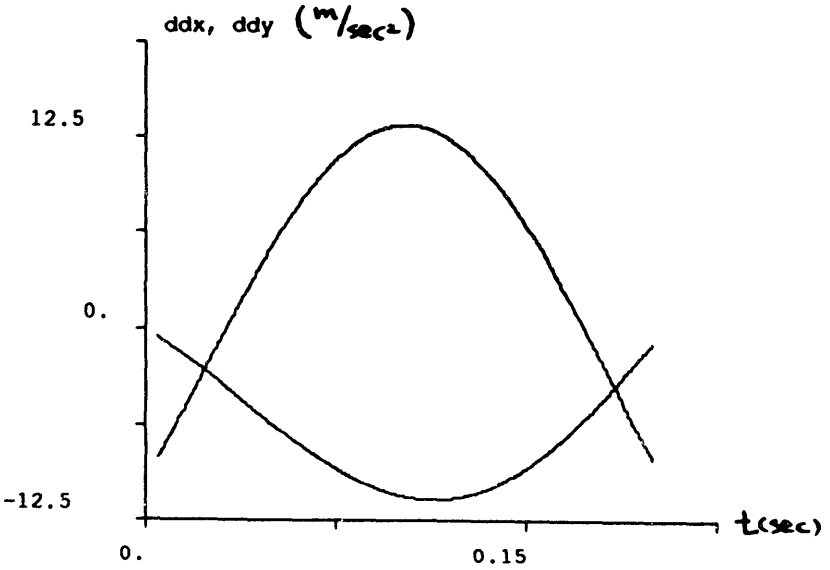


$\theta_{leg_0} = -20°$, $\theta_{td} = 20°$

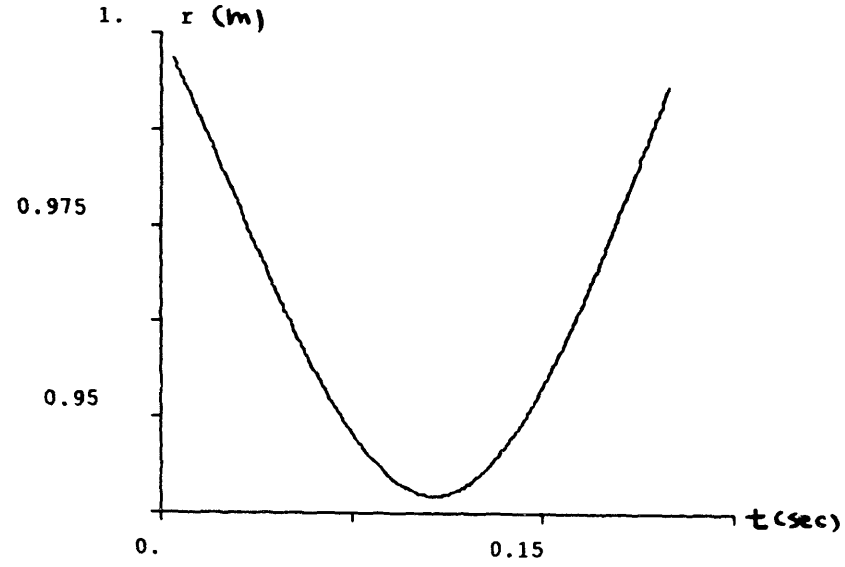


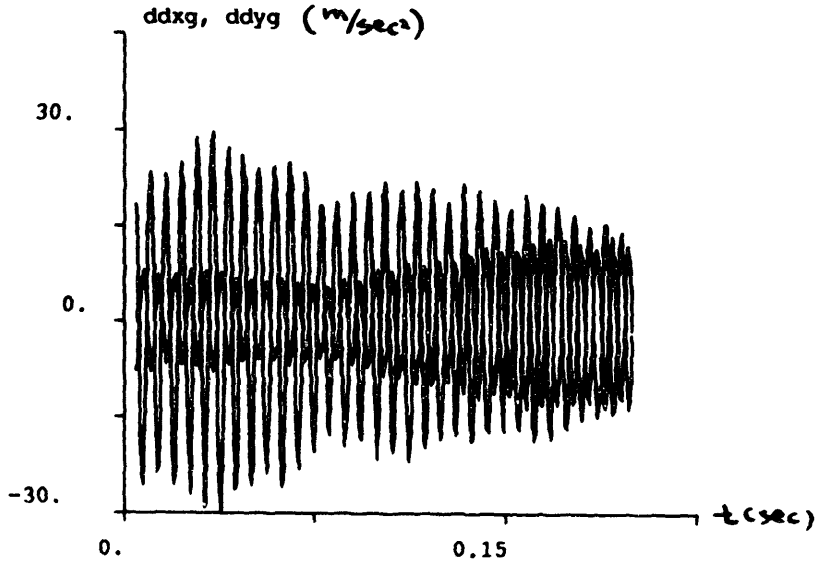


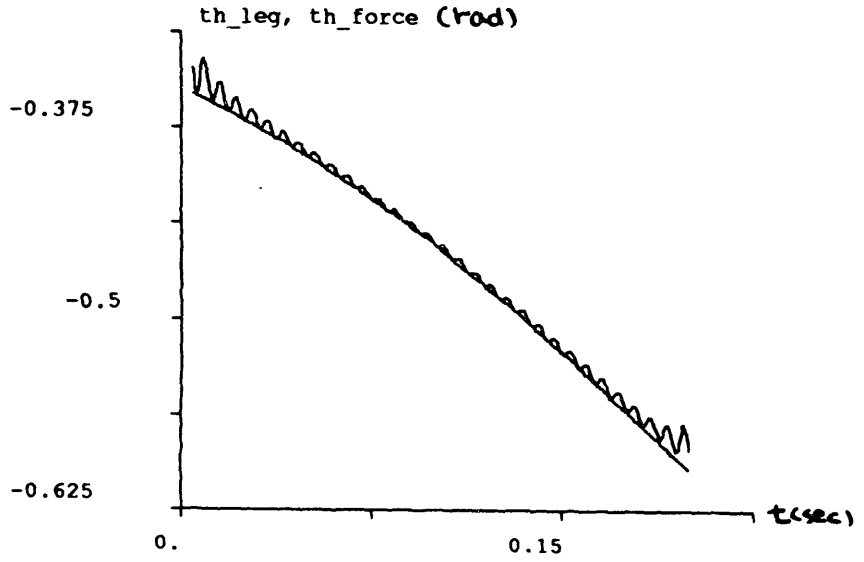


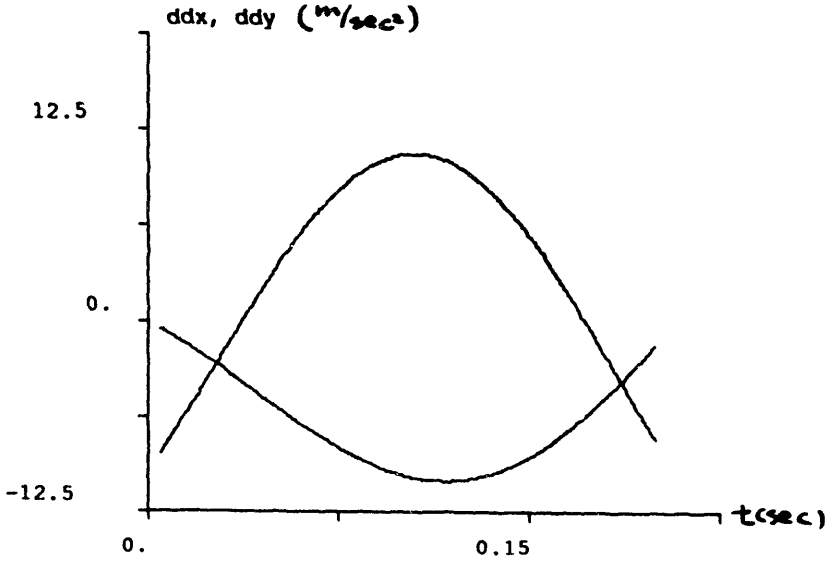


$\theta_{leg-0} = -20°$, $\theta_{td} = 30°$

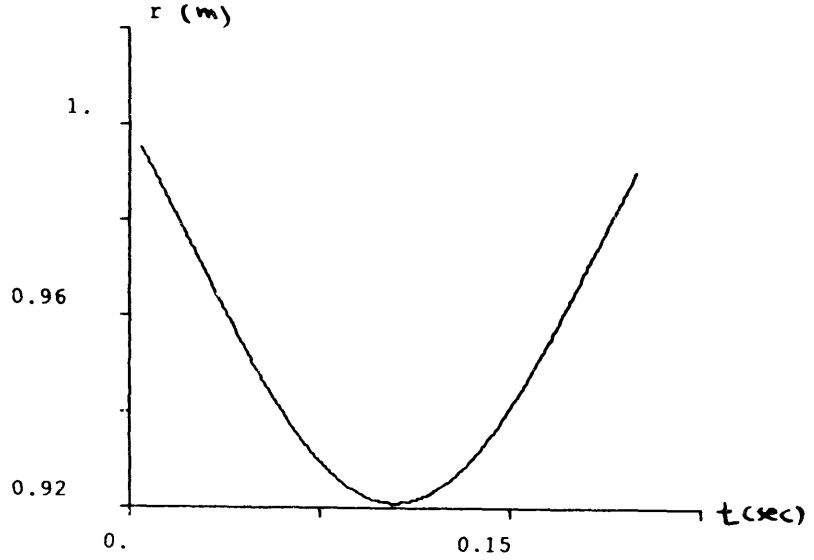


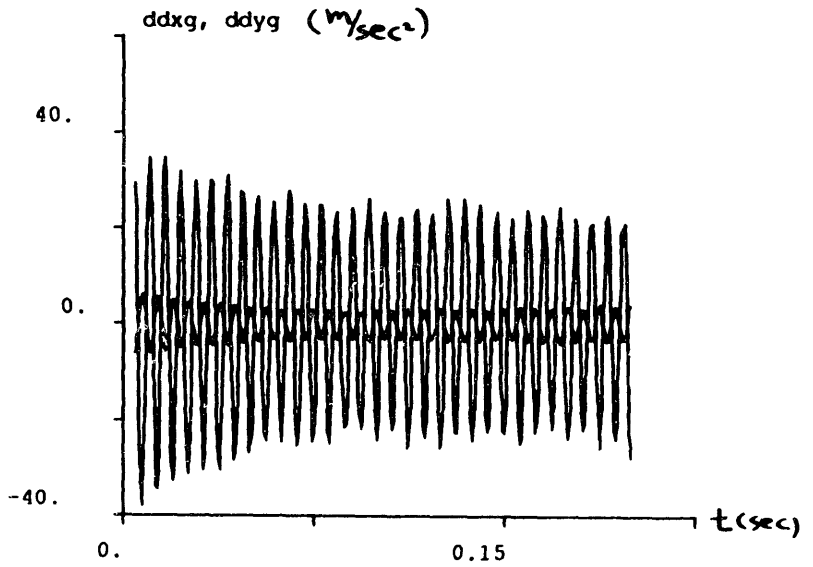


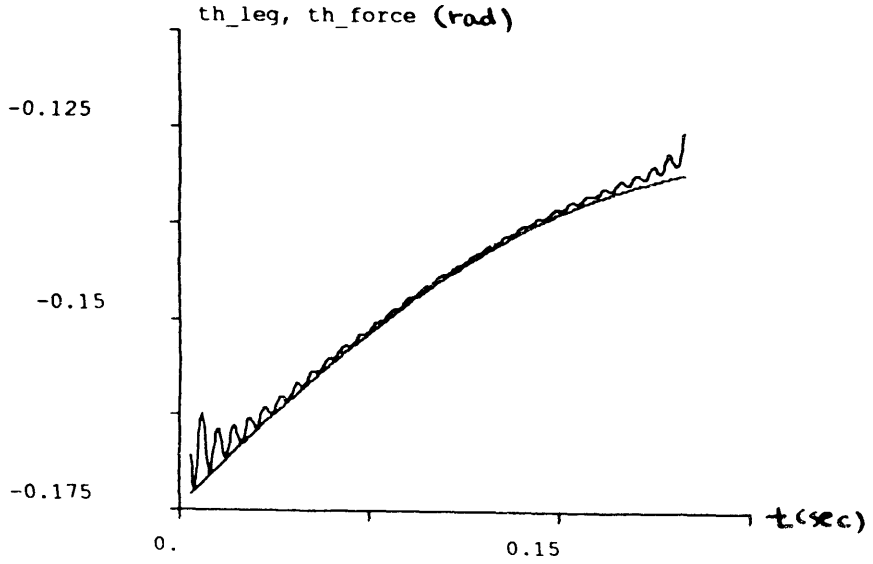


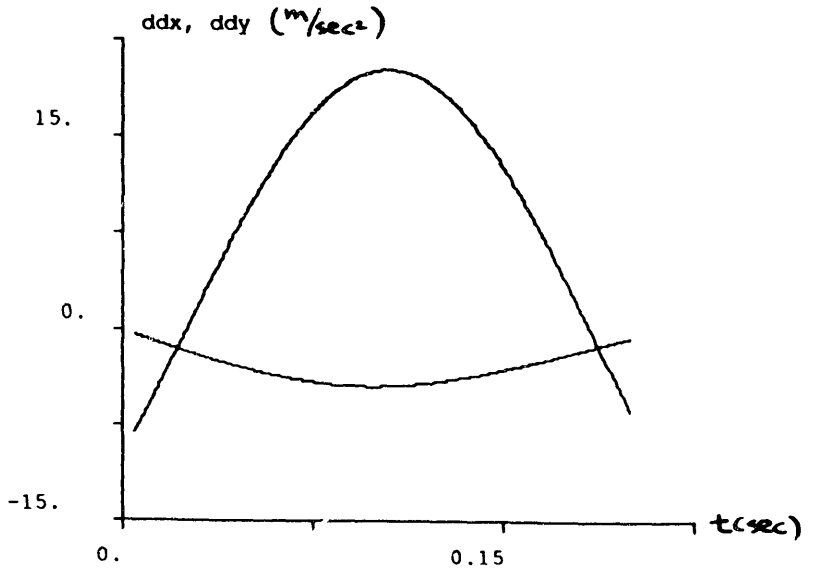


$\theta_{leg,0} = -10°$, $\theta_{td} = -30°$

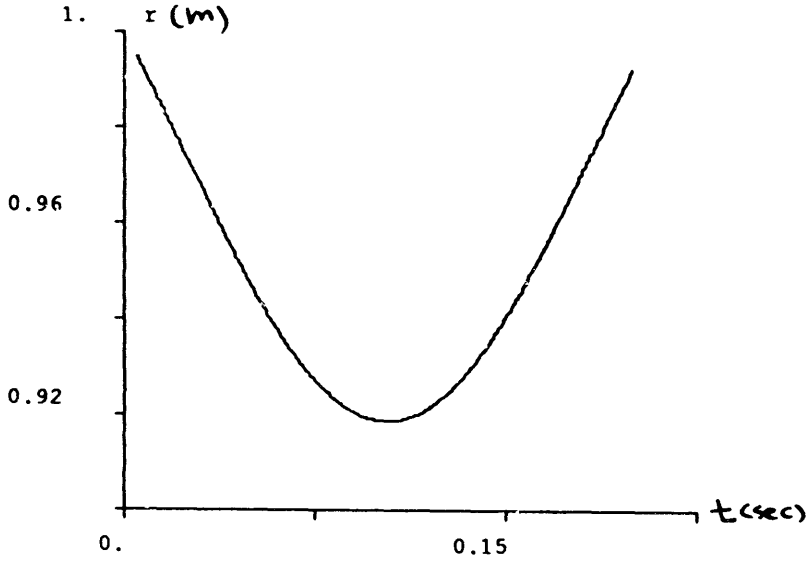


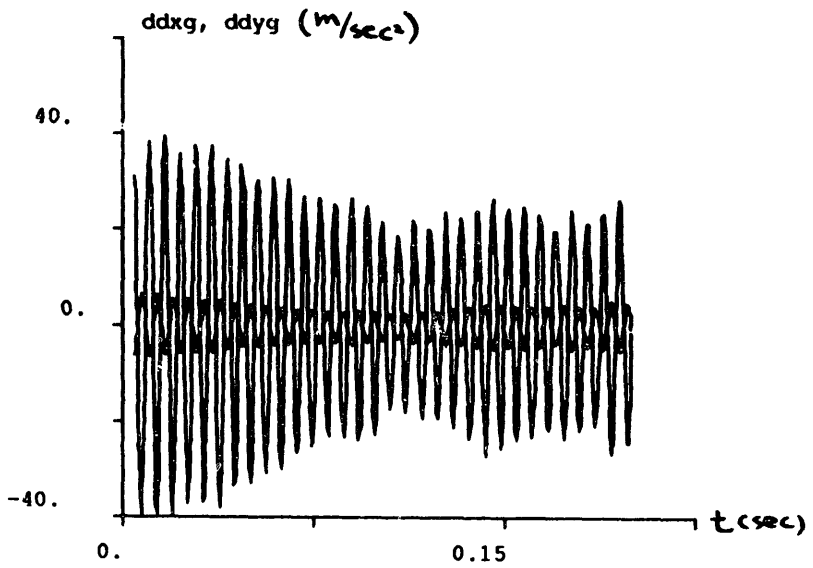


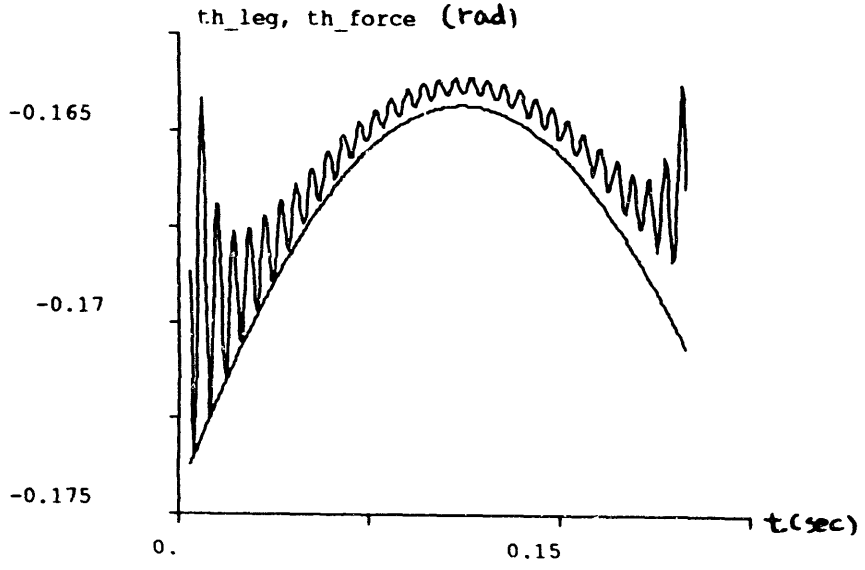


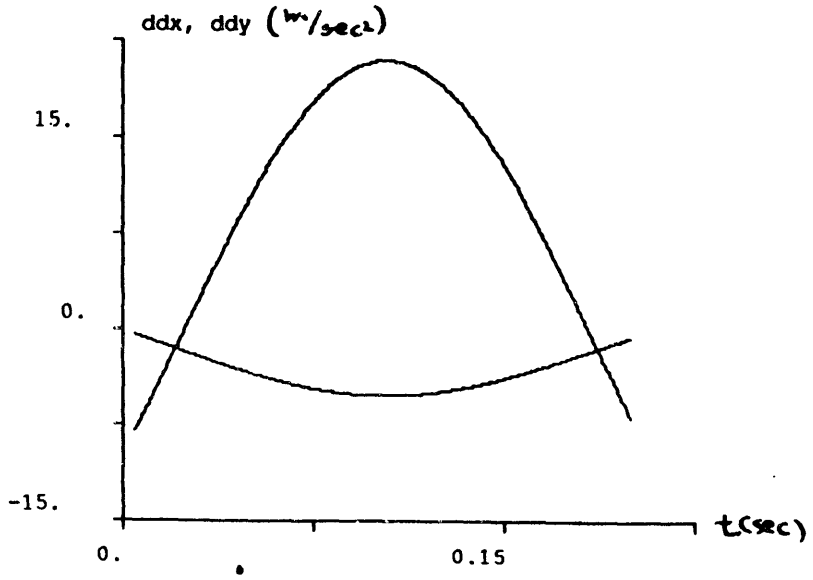


$\theta_{leg,0} = -10°$, $\theta_{rad} = -20°$

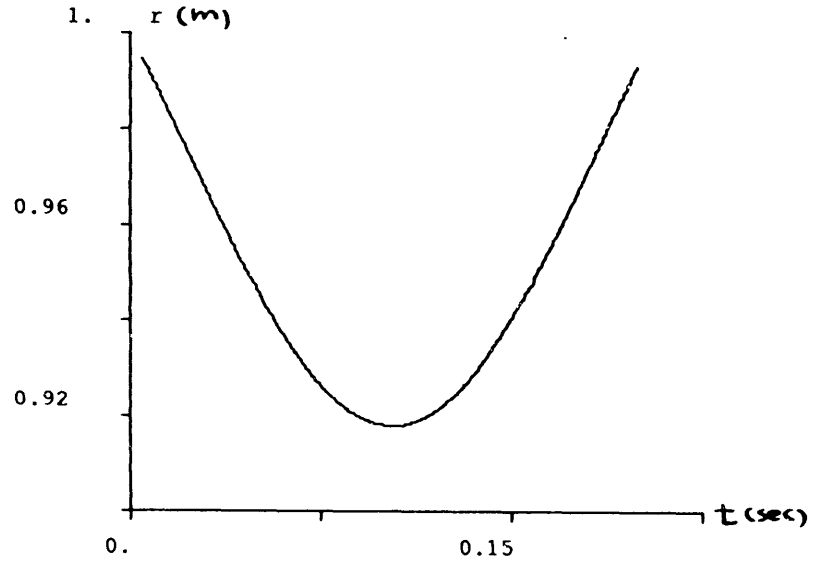


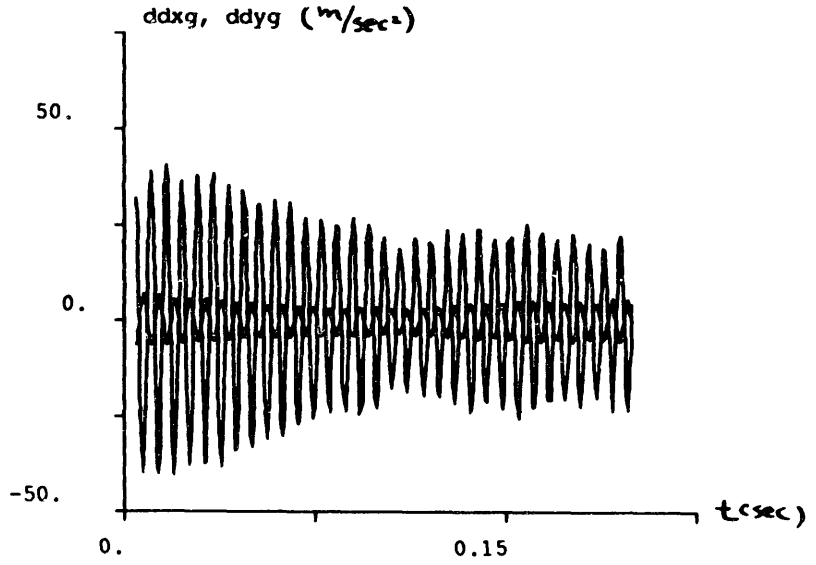


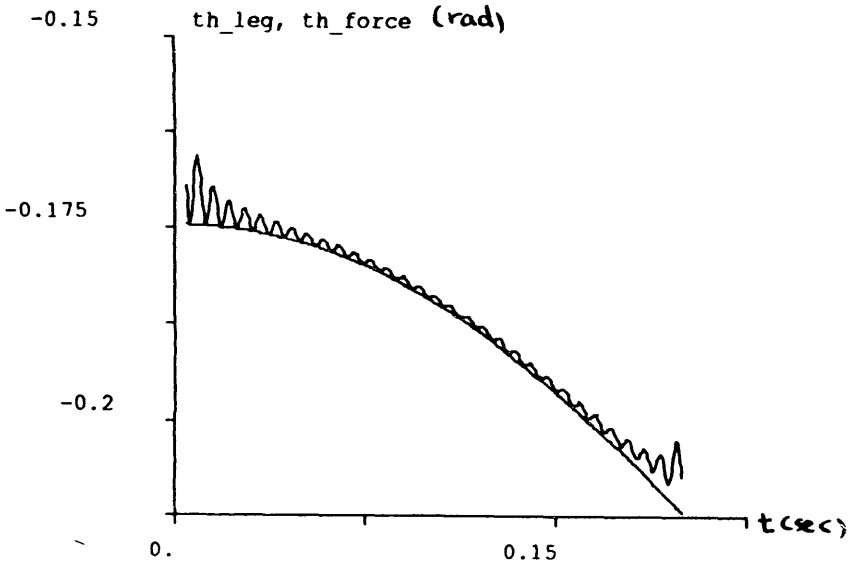


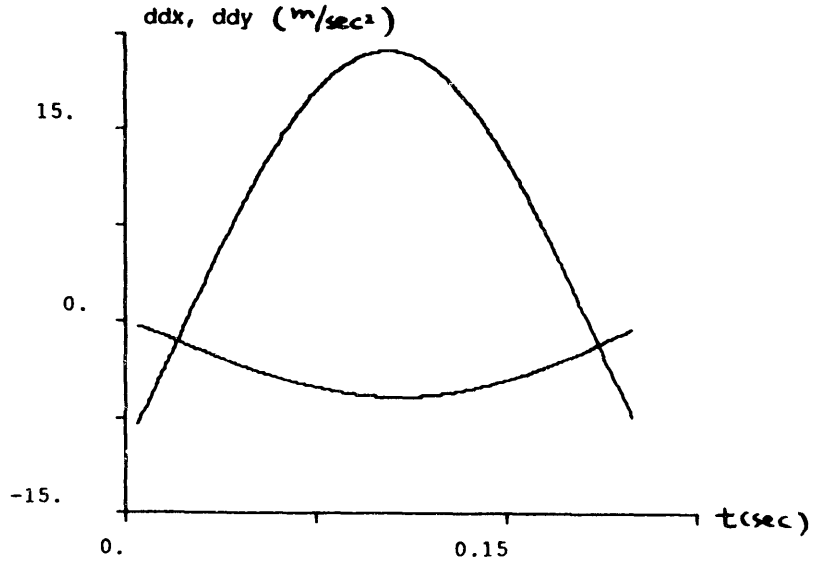


$\theta_{leg_0} = -10°$, $\theta_{td} = -10°$

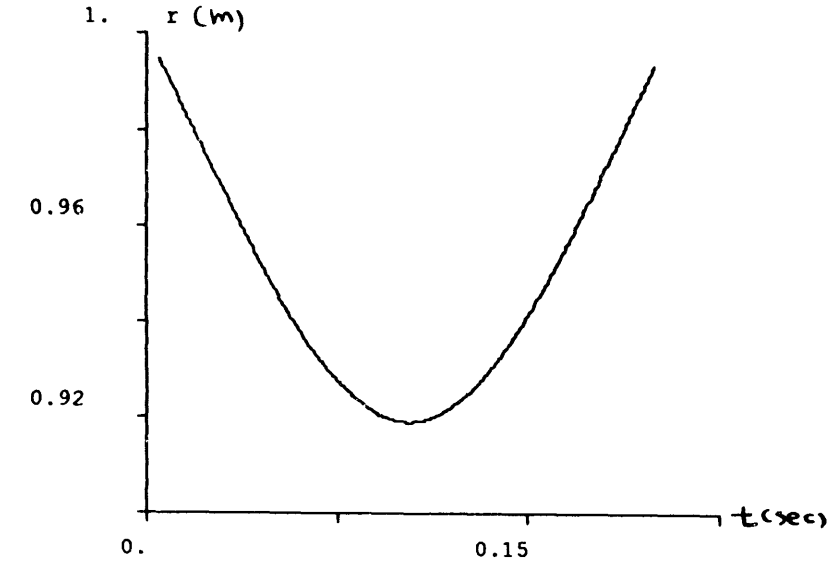


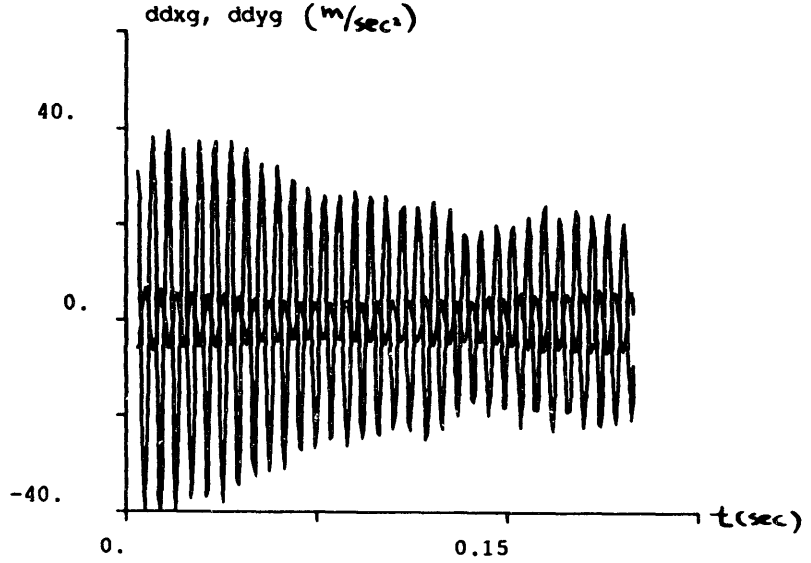


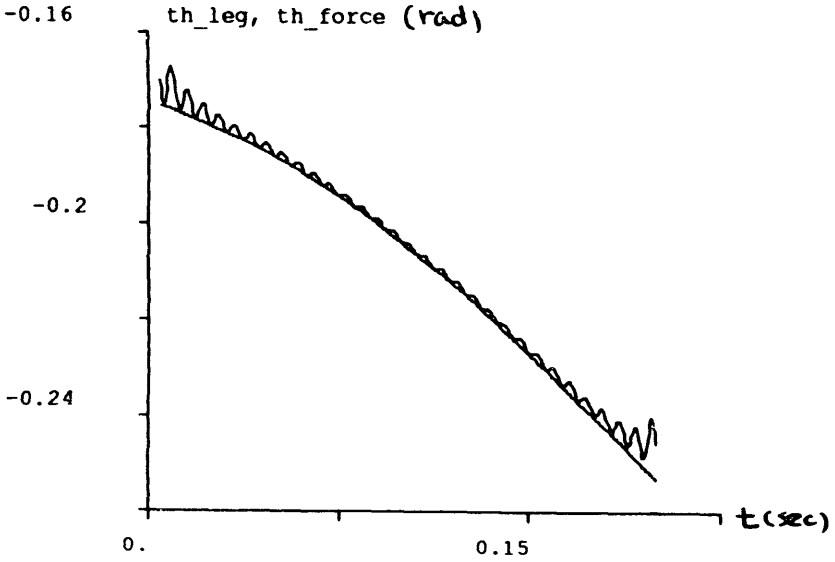


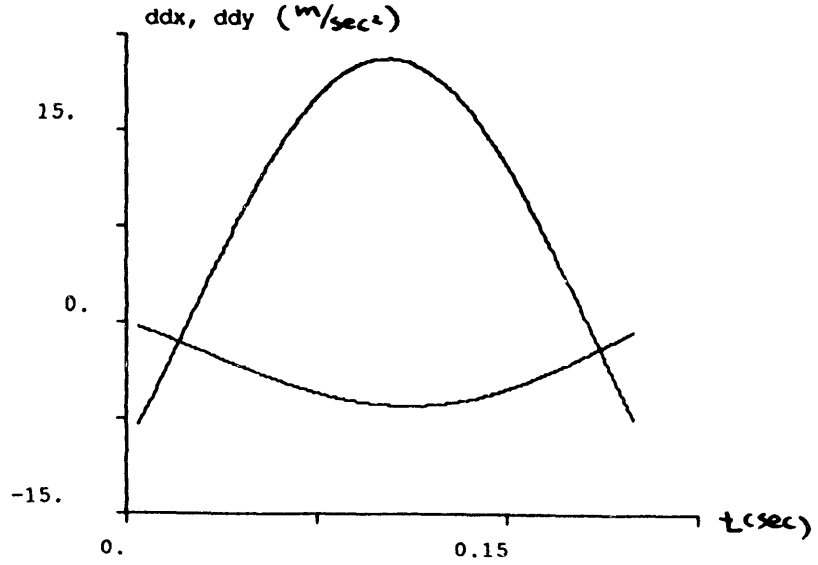


$\theta_{leg_{3-0}} = -10°$, $\theta_{rad} = 0°$

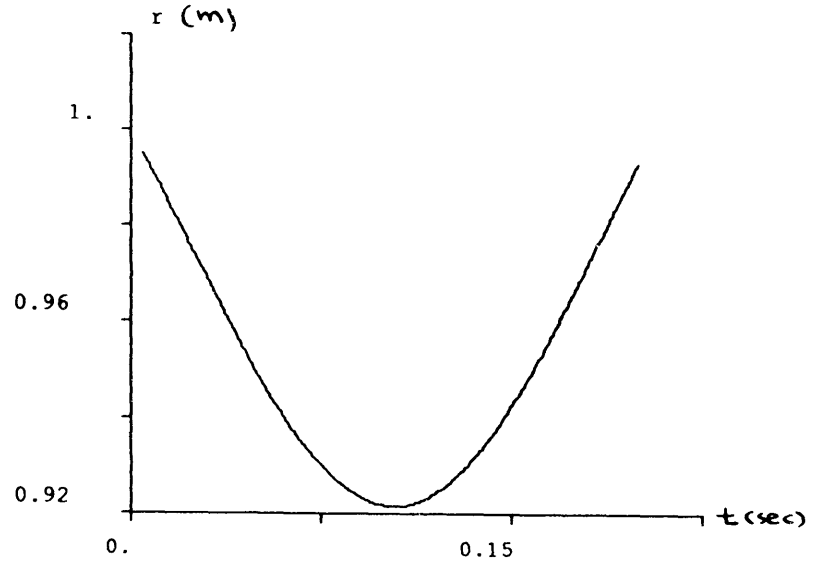


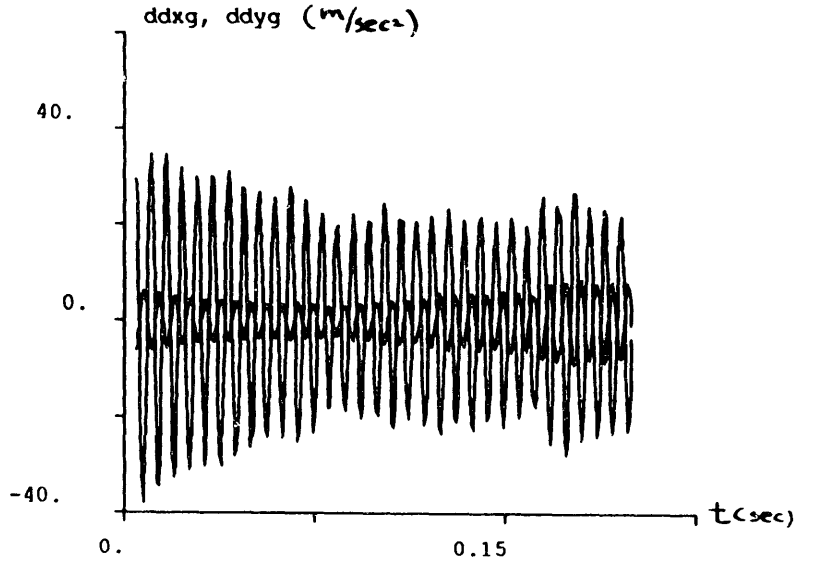


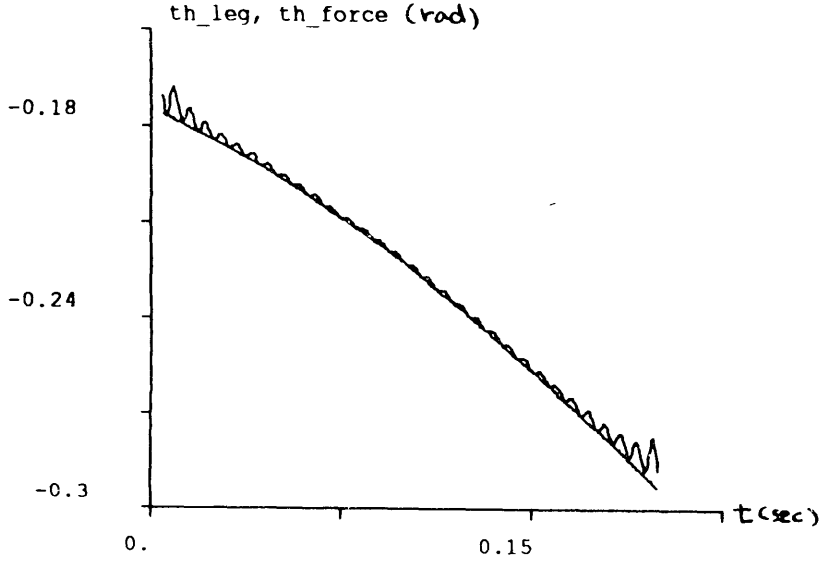


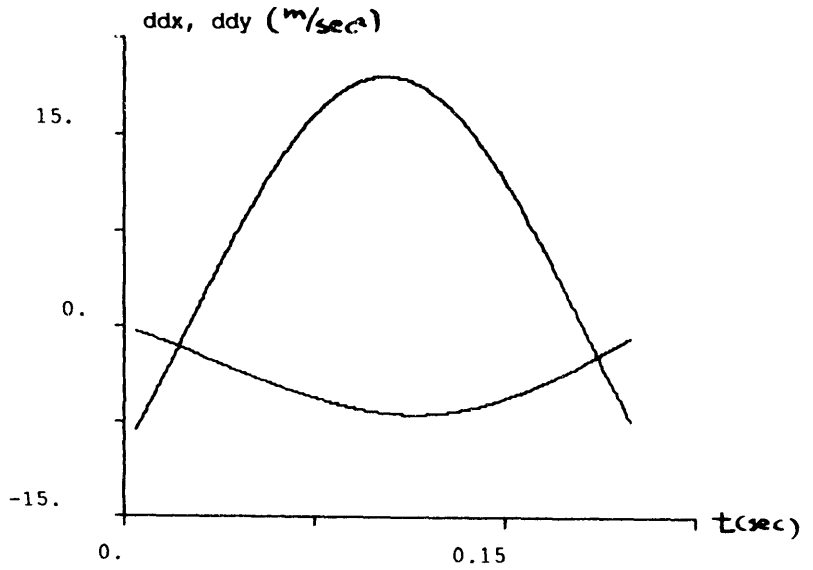


$\theta_{leg-o} = 40^\circ$, $\theta_{td} = 10^\circ$

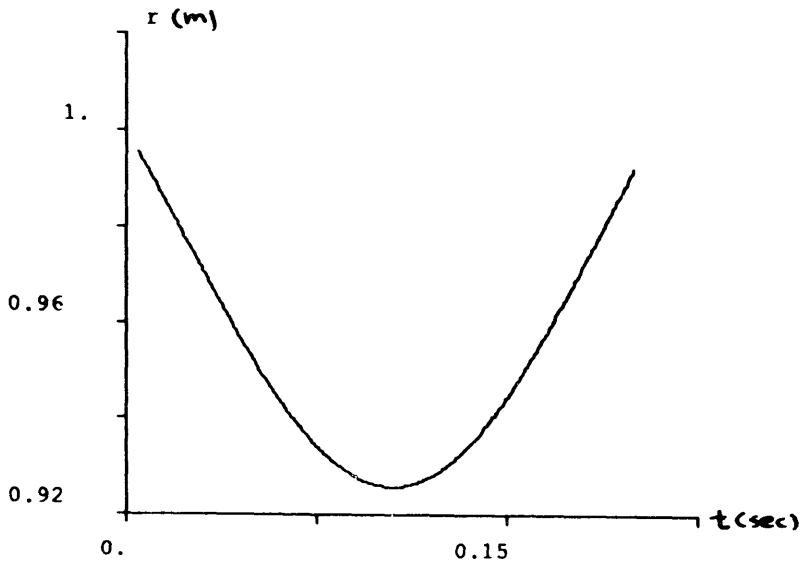


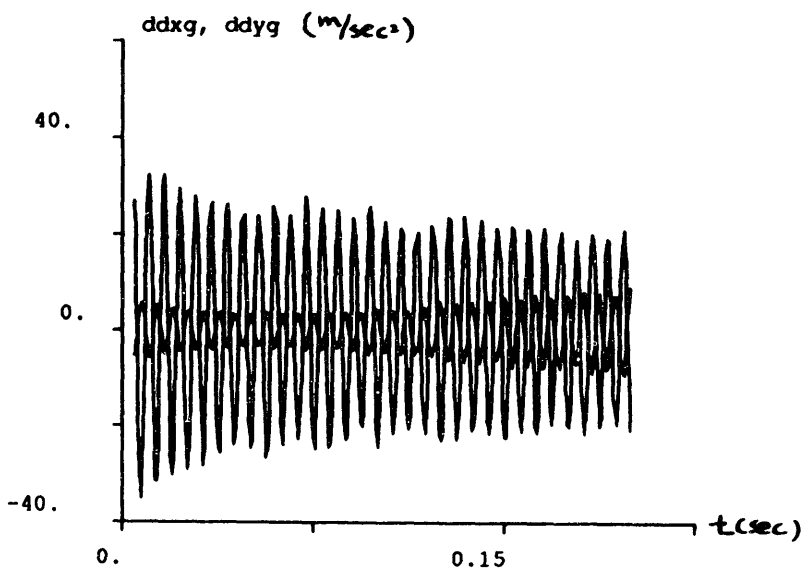


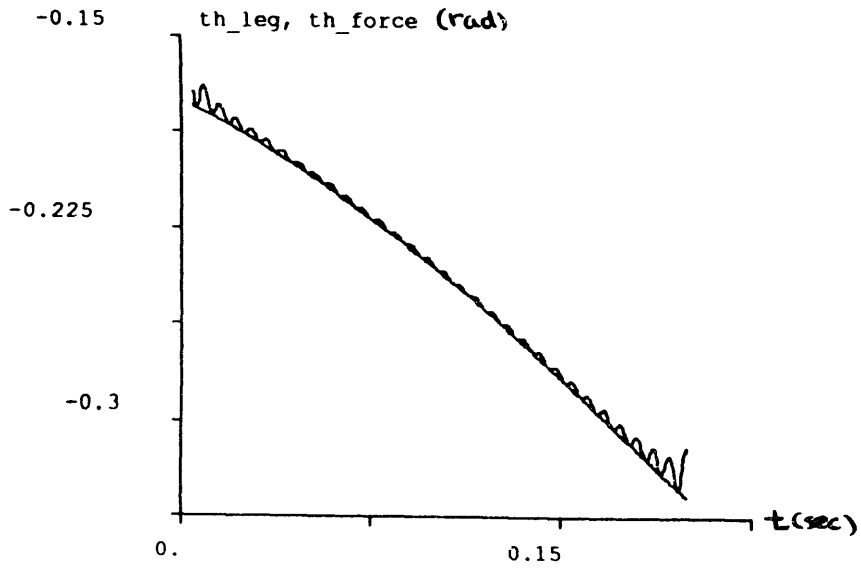


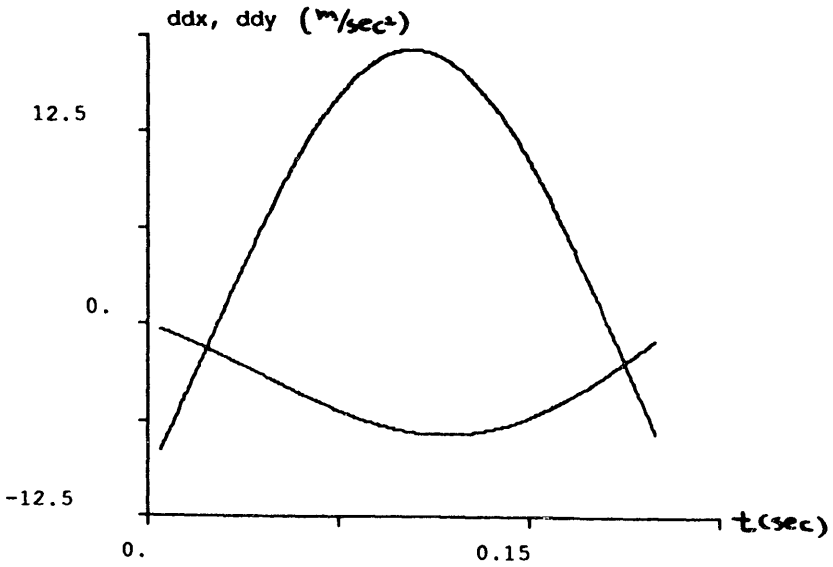


$\theta_{leg_0} = 10°$, $\theta_{th} = 20°$

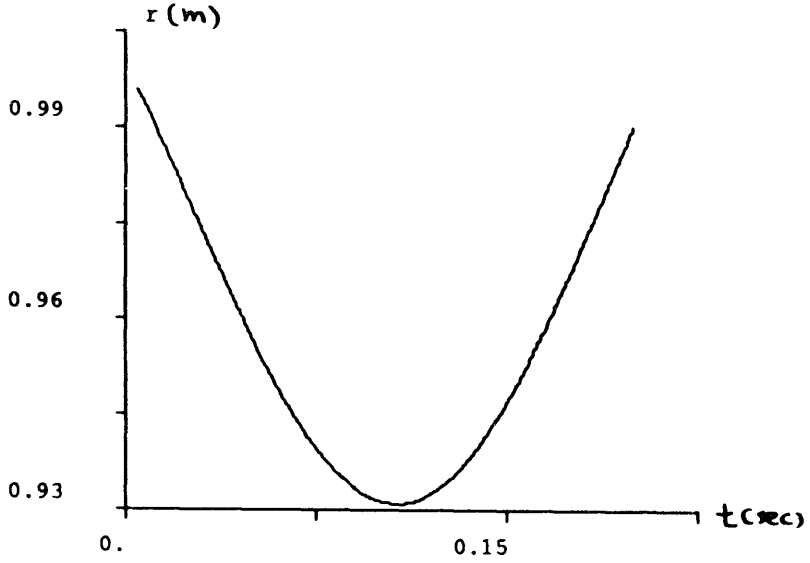


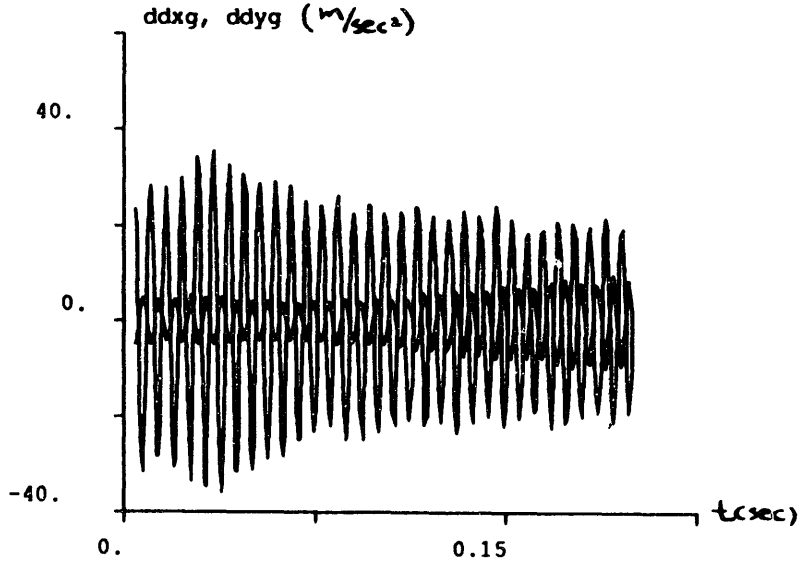


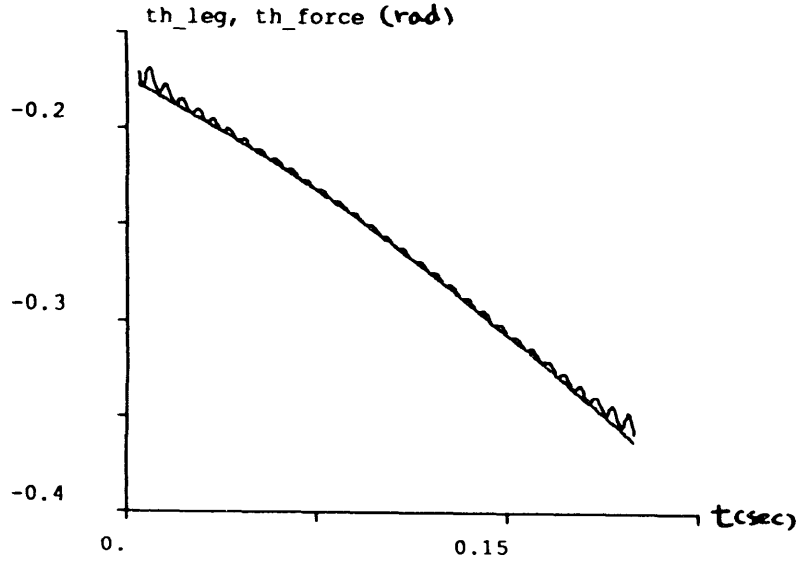


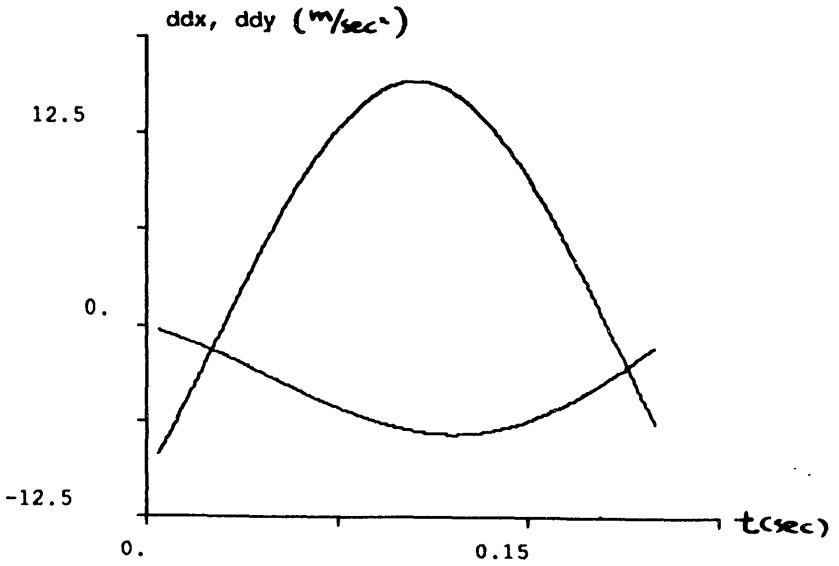


$\theta_{leg_0} = -10°$, $\theta_{ld} = 30°$

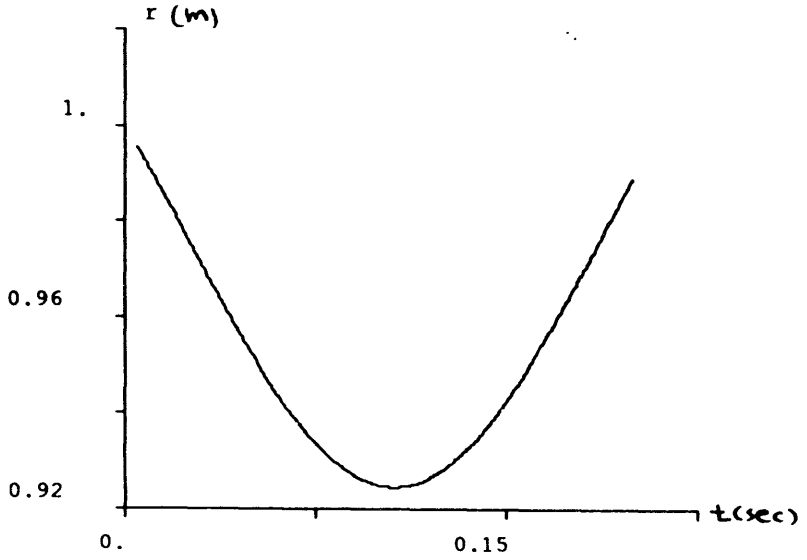


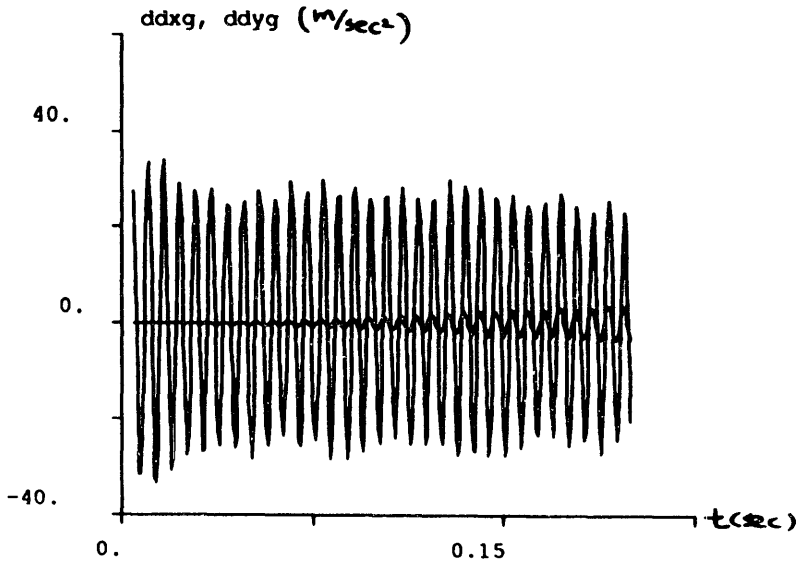


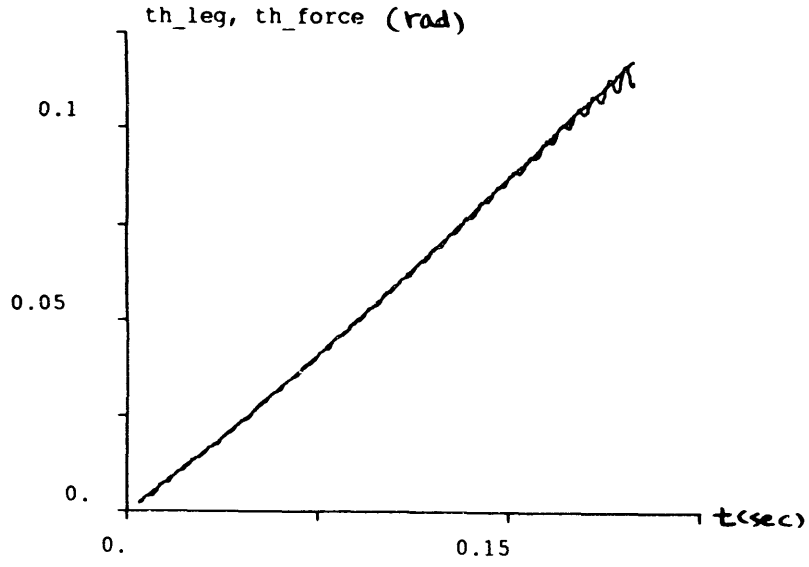


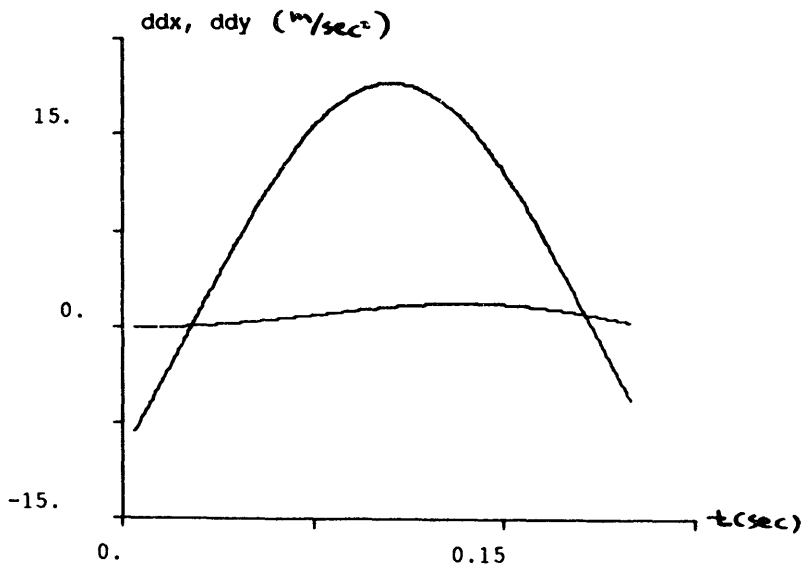


$\theta_{leg_0} = 0$, $\theta_{cd} = -30°$

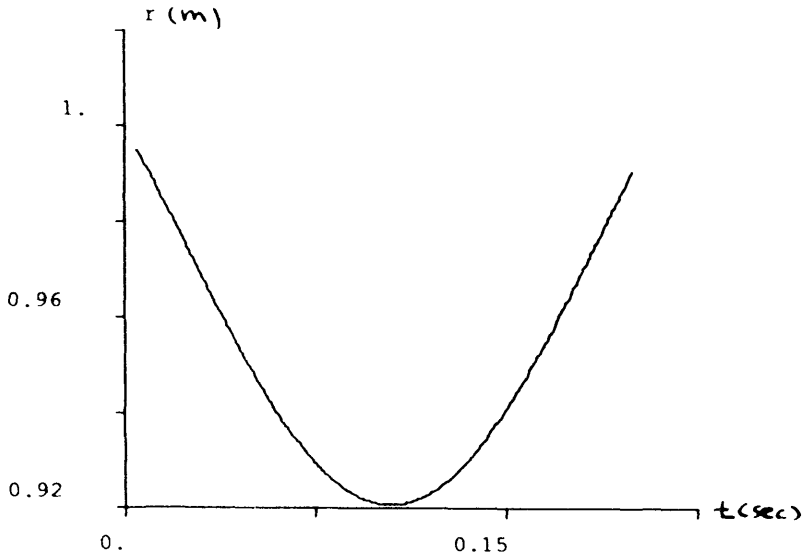


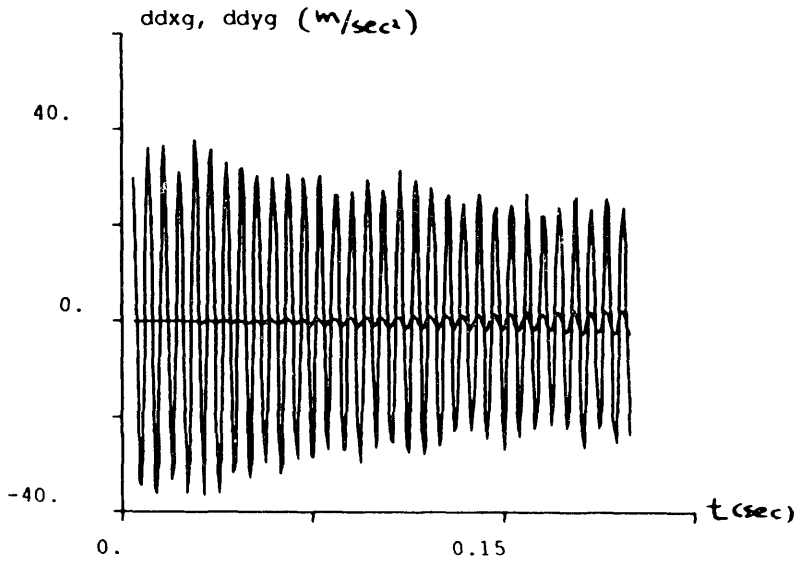


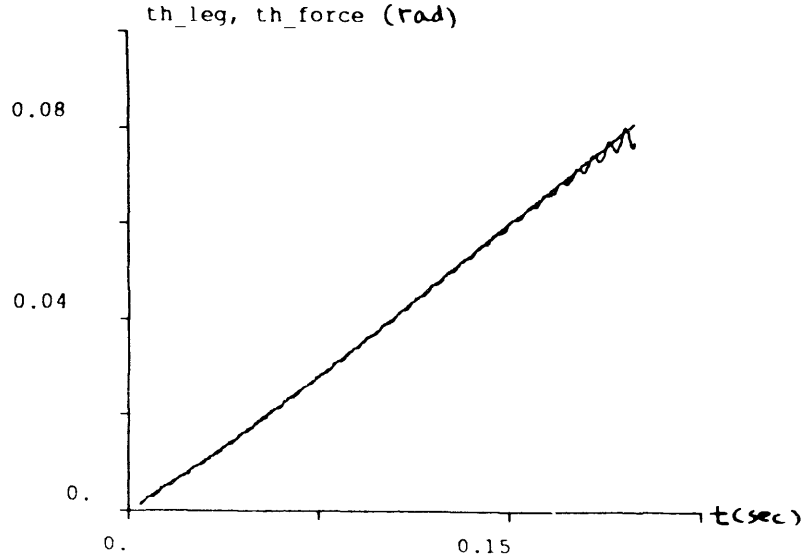


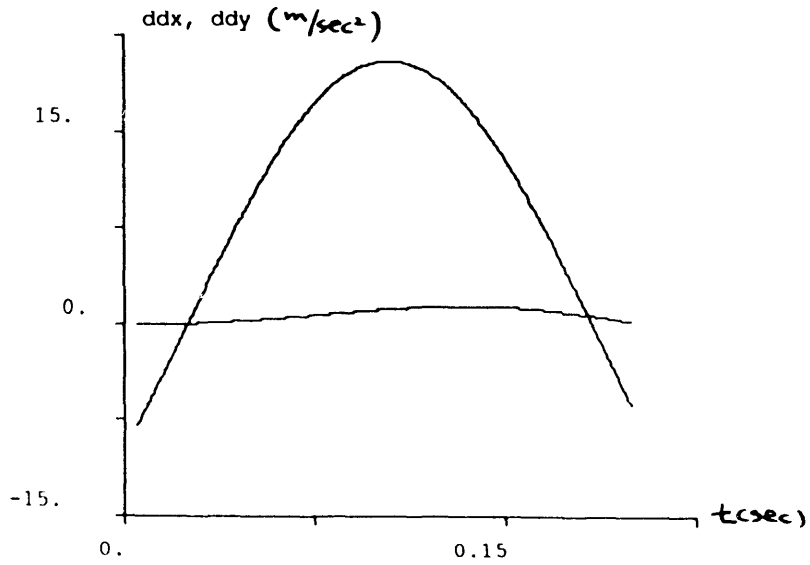


$\theta_{leg,0} = 0$, $\theta_{td} = -20°$

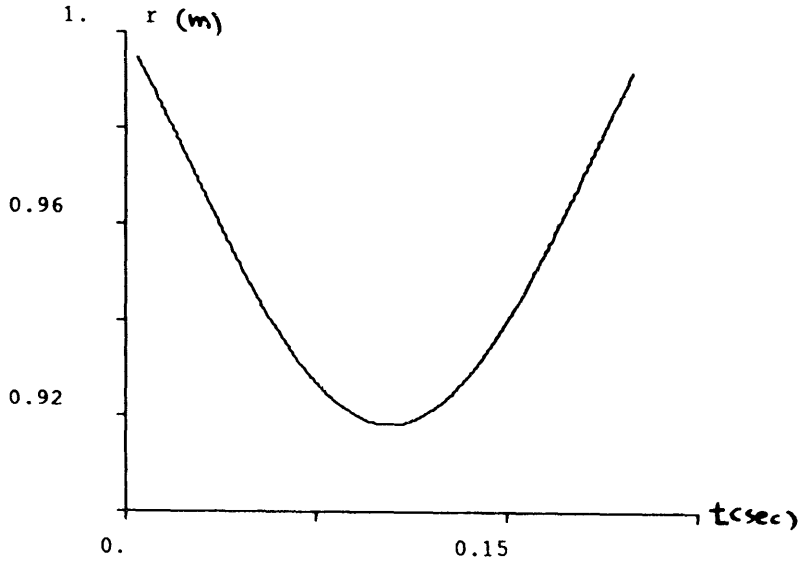


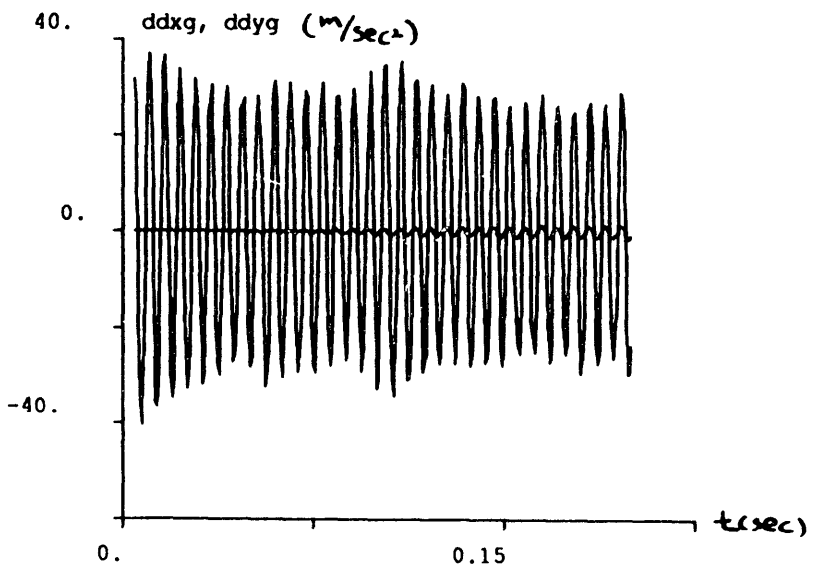


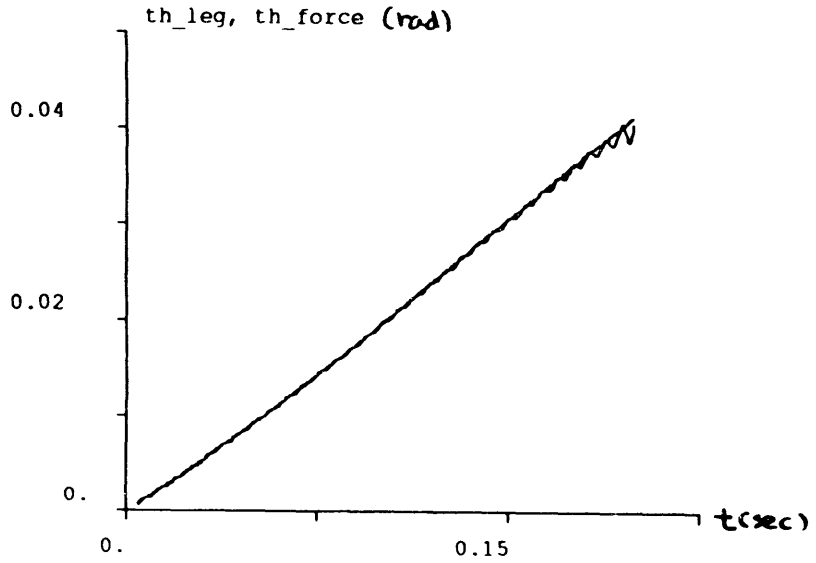


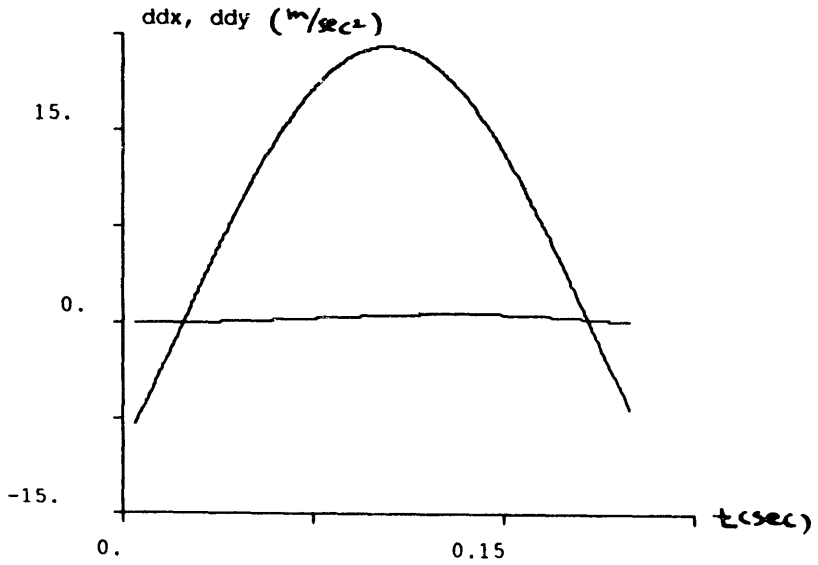


$\theta_{leg\_0} = 0$, $\theta_{td} = -10°$

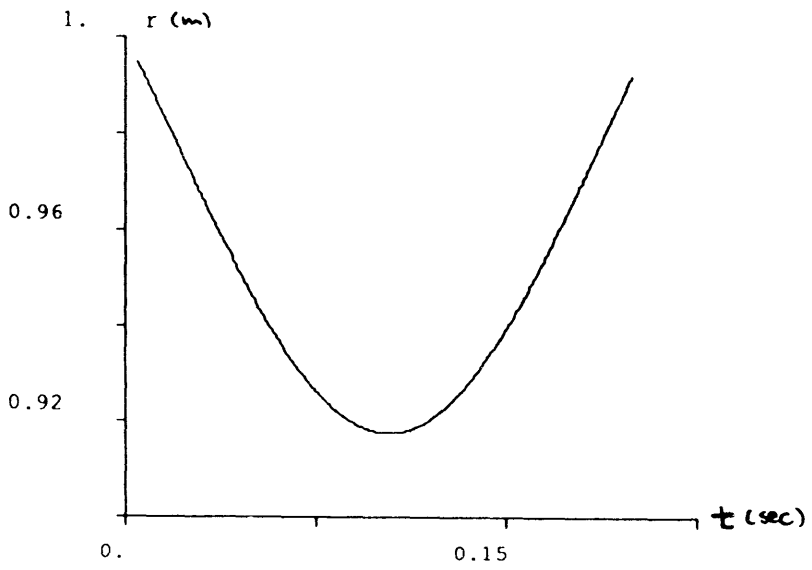


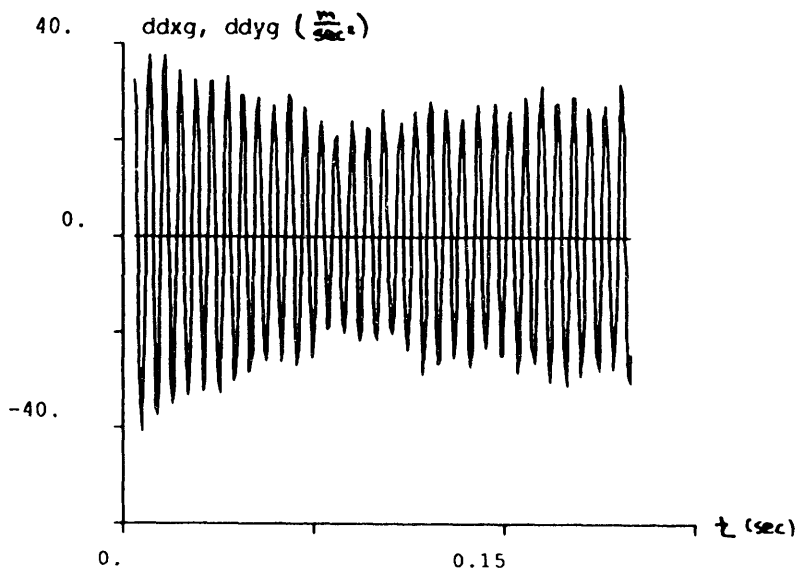


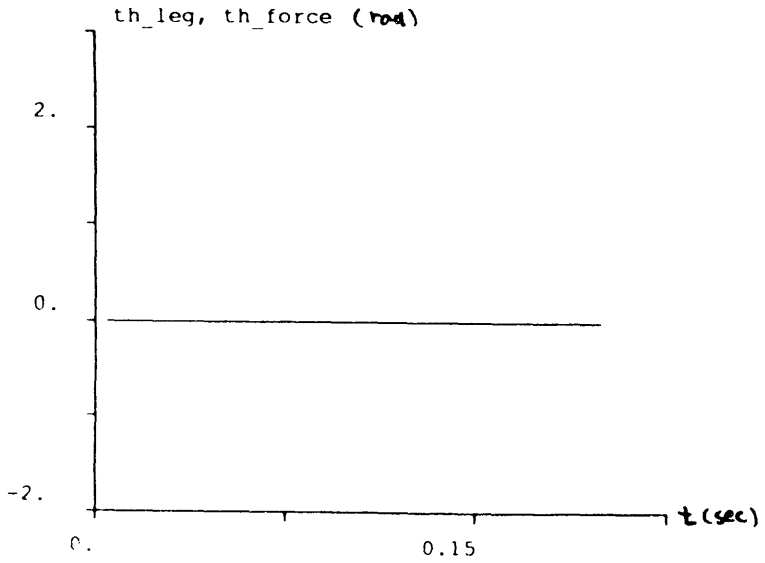


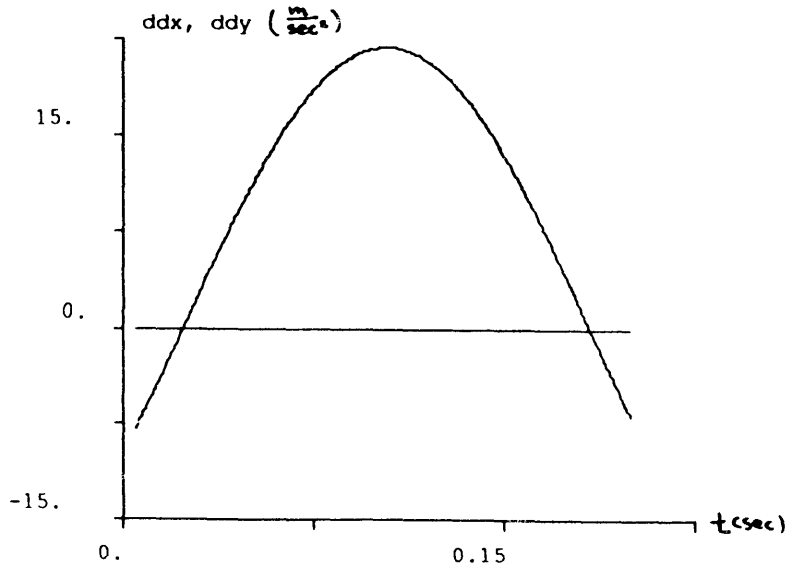


$\theta_{leg\_0} = 0$, $\theta_{td} = 0$

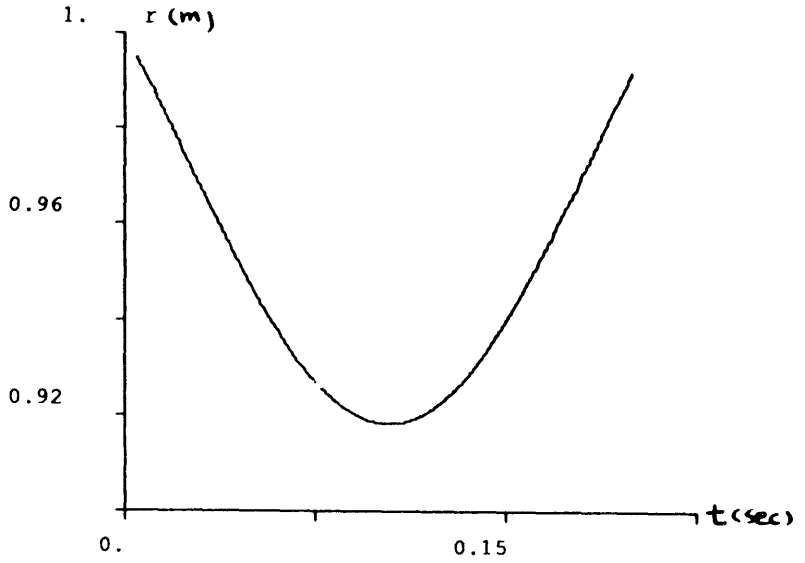


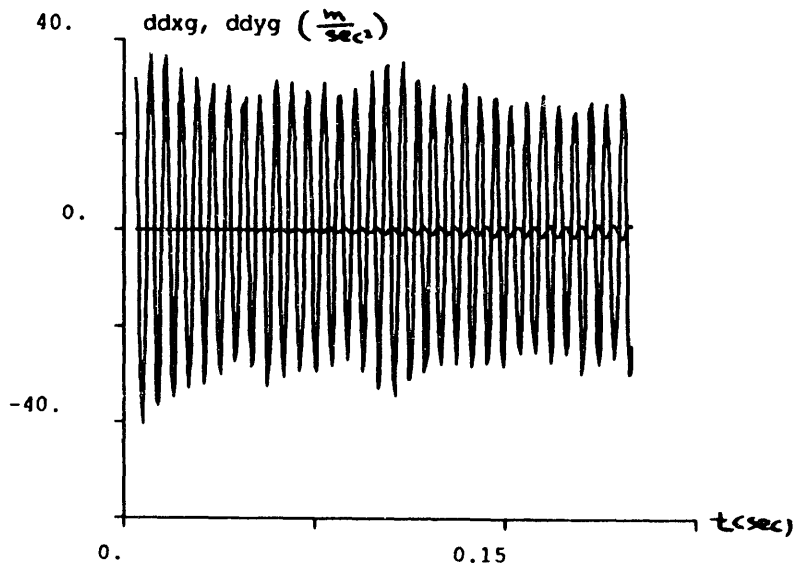


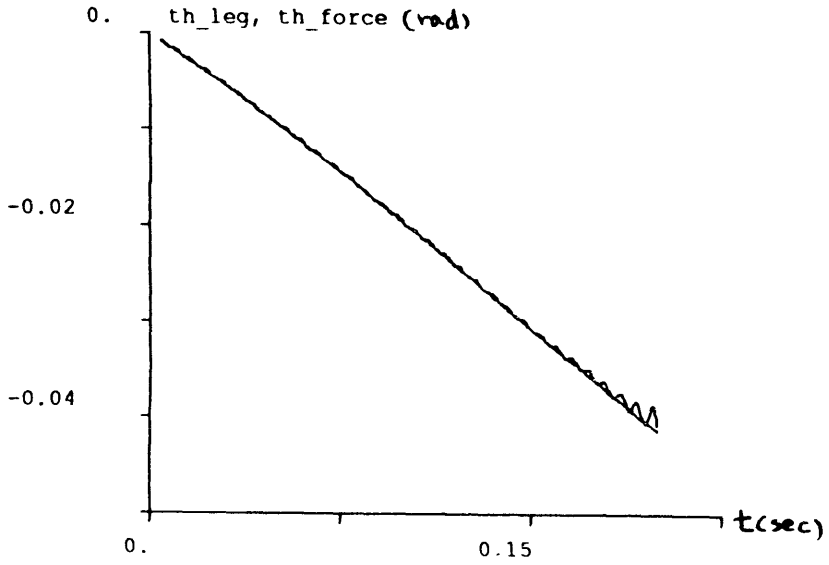


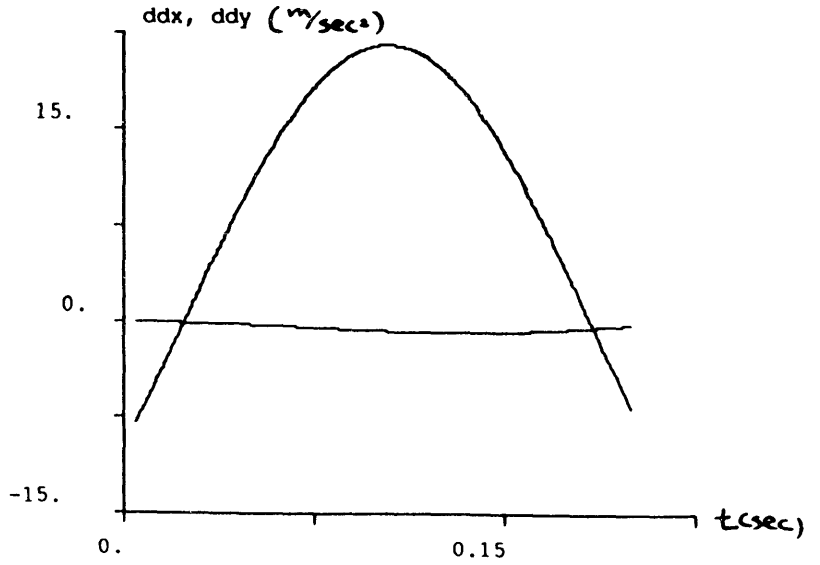


$\theta_{leg_0} = 0$, $\theta_{td} = 10°$

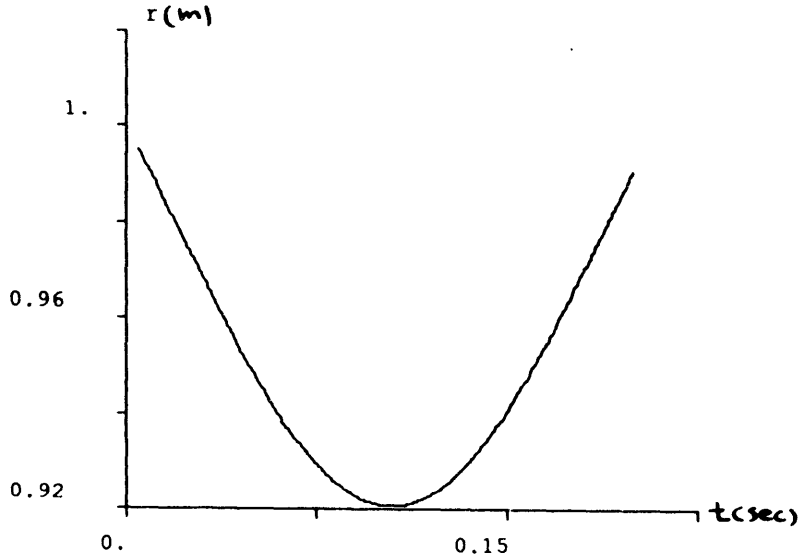


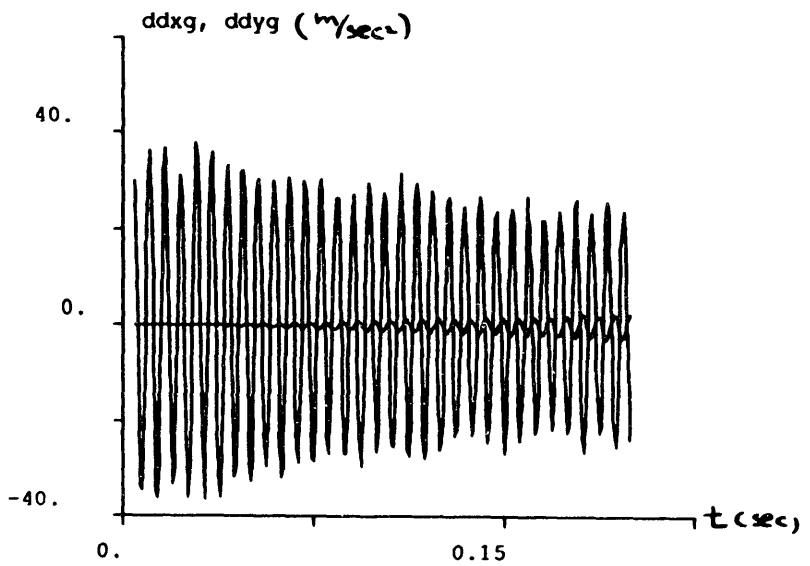


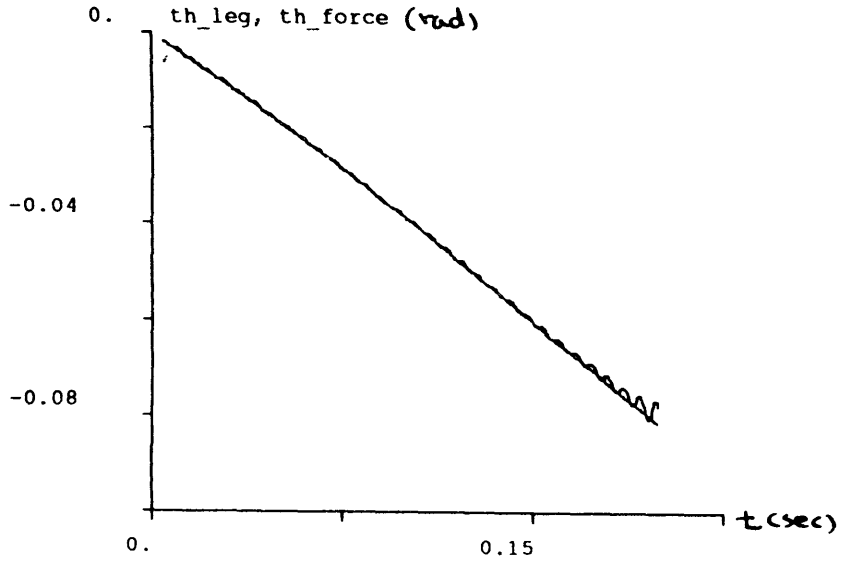


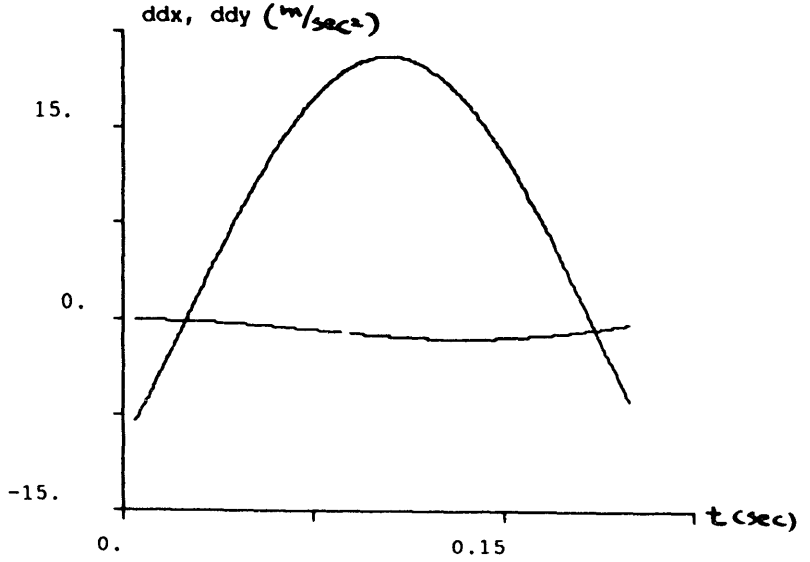


$\theta_{leg_0} = 0^\circ$, $\theta_{td} = 20^\circ$

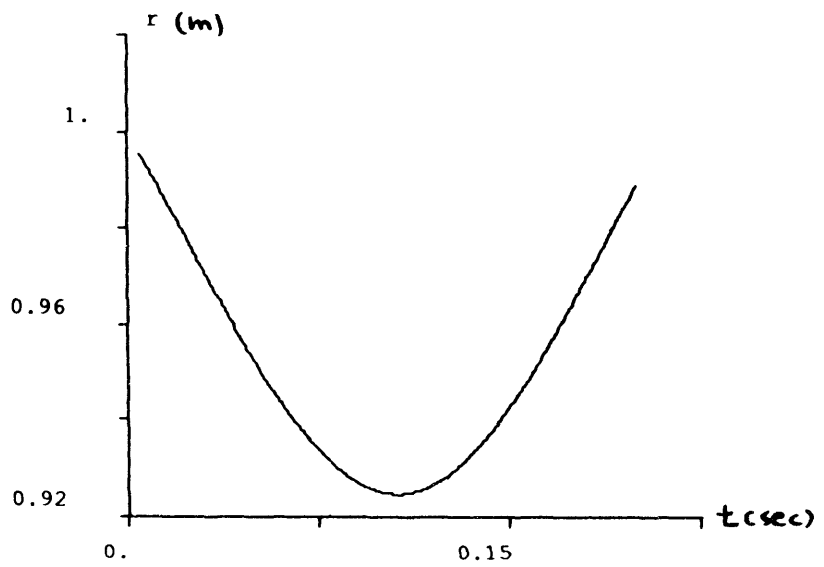


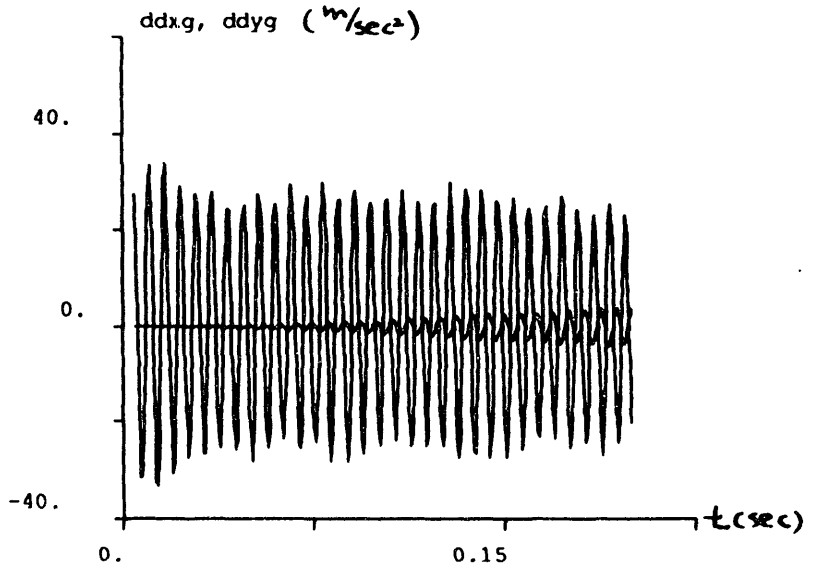


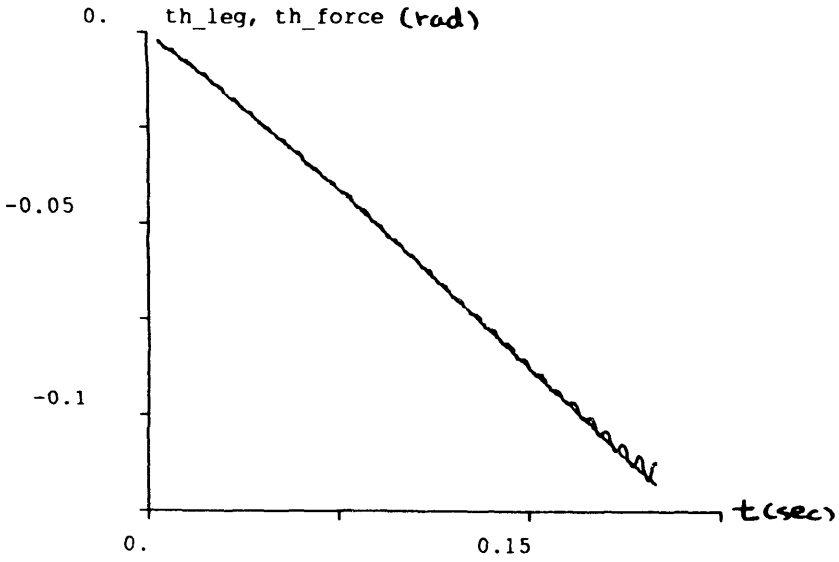


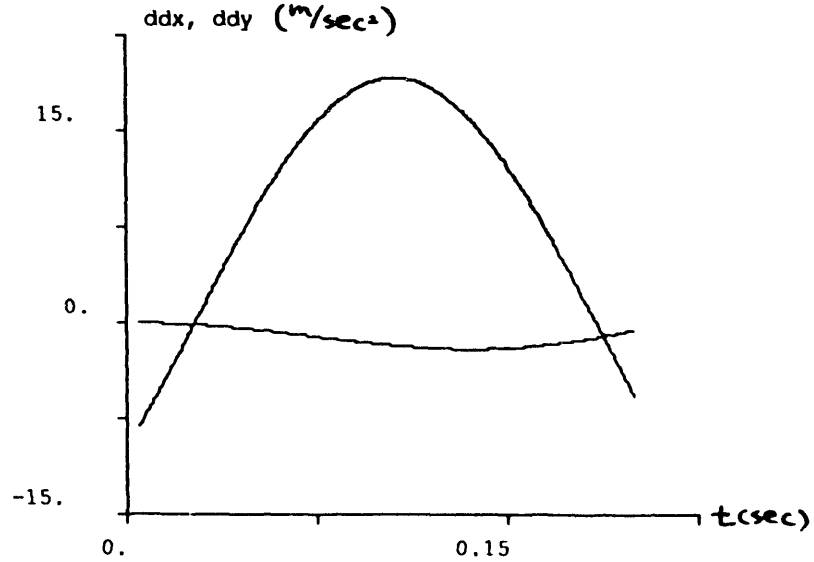


$\theta_{leg_0} = 0^\circ$, $\theta_{cd} = 30^\circ$

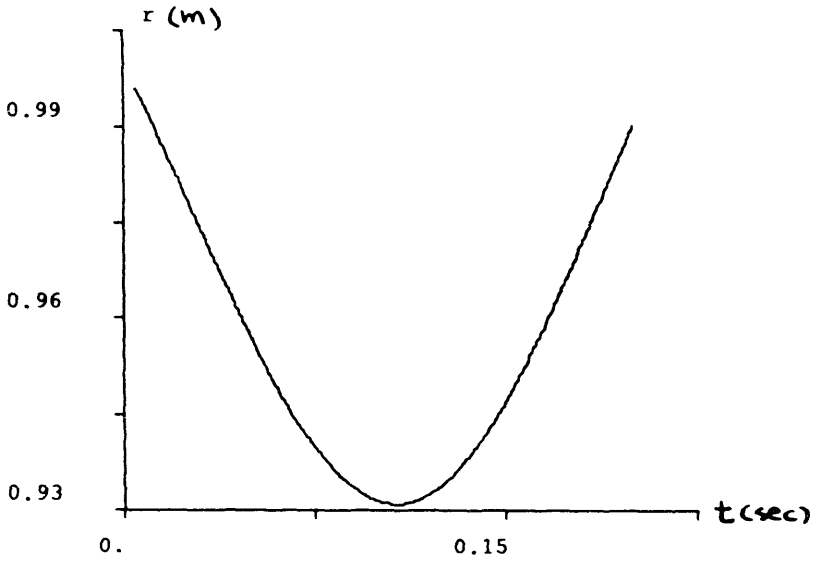


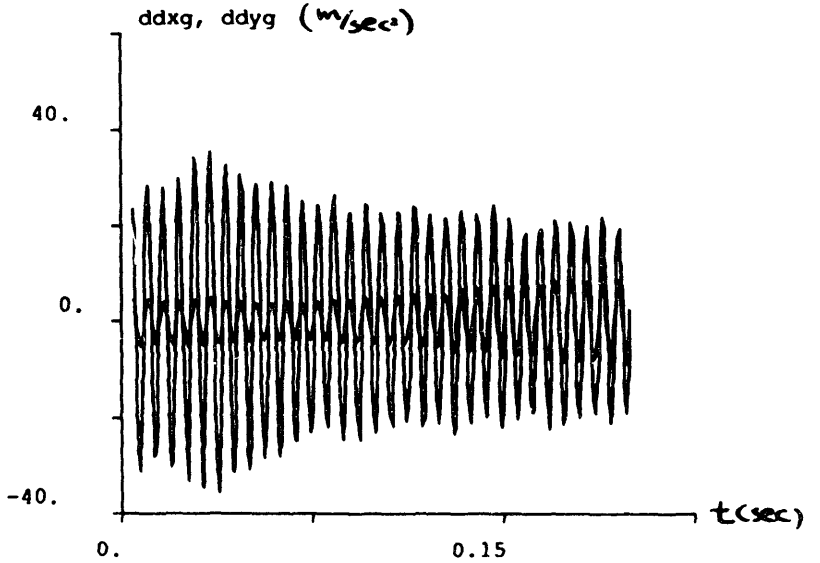


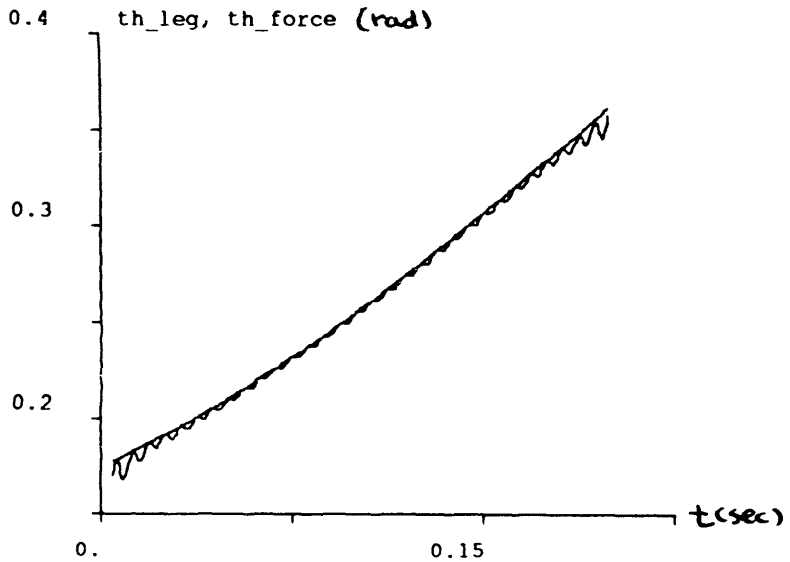


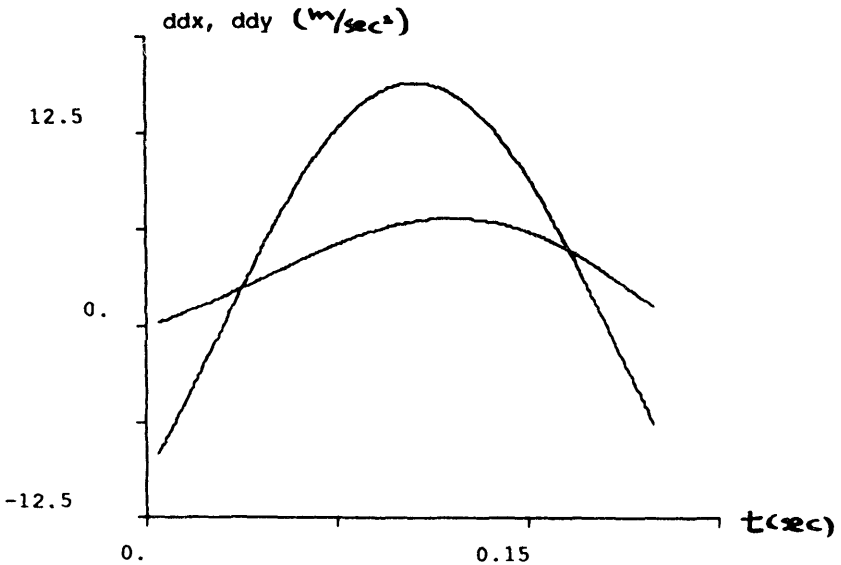


$\theta_{leg-0} = 10°$, $\theta_{td} = -30°$

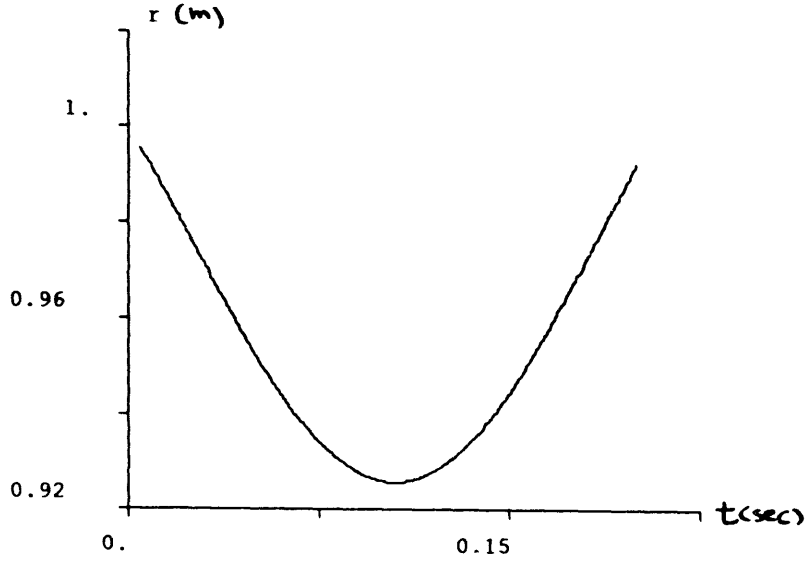


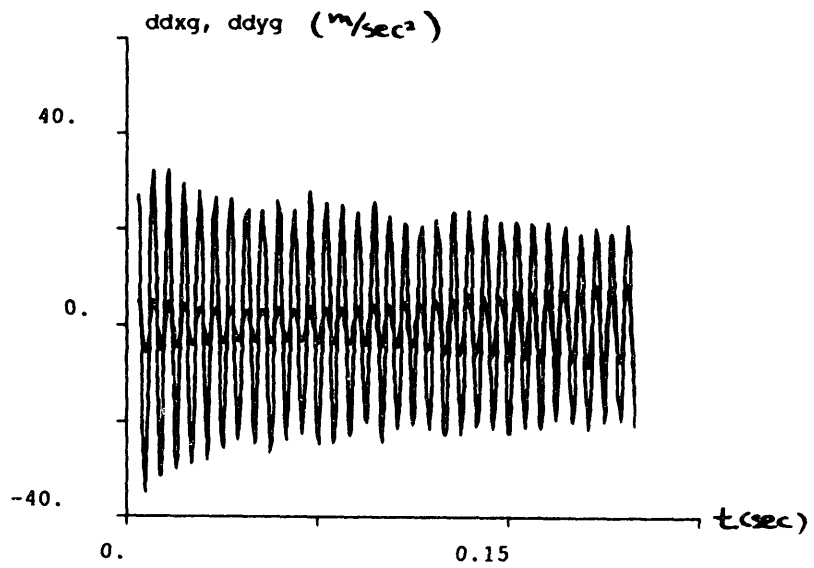


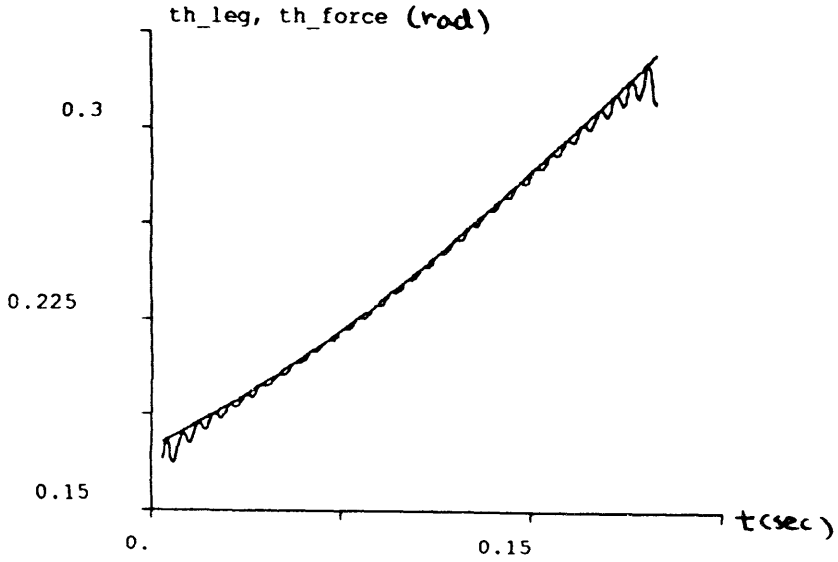


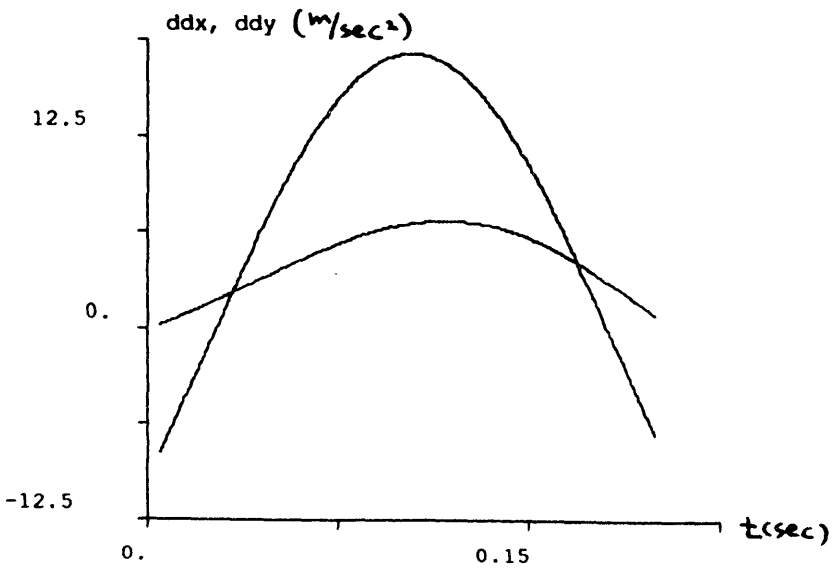


$\theta_{leg\_0} = 10°$, $\theta_{td} = -20°$

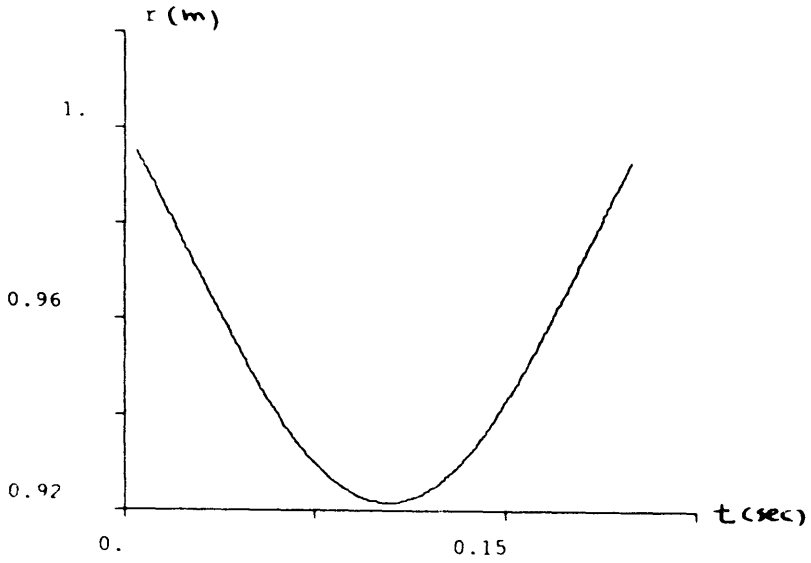


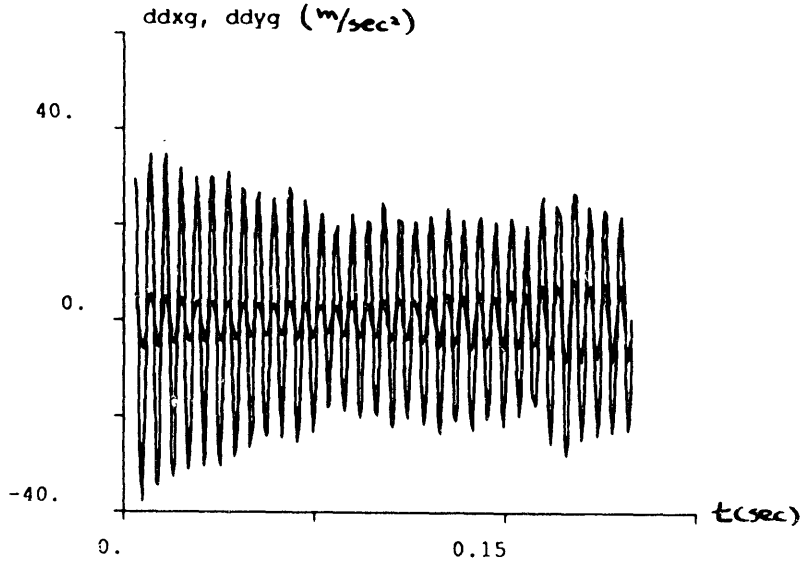


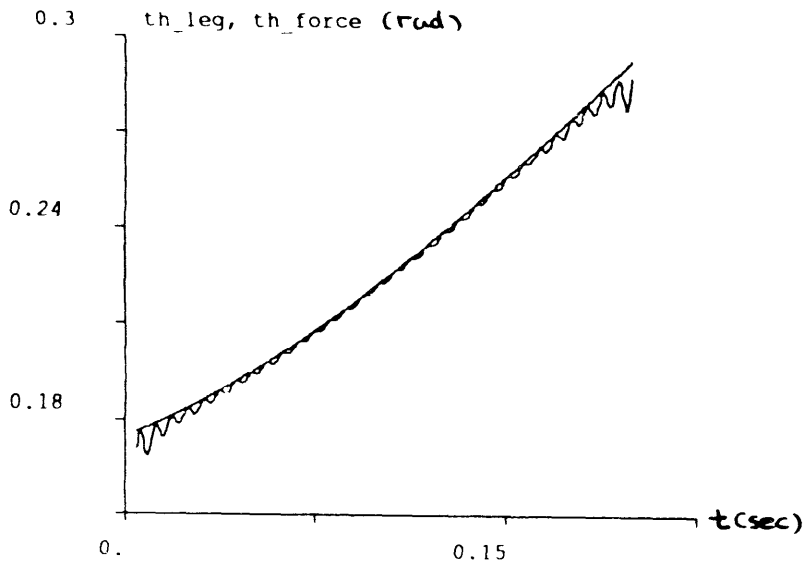


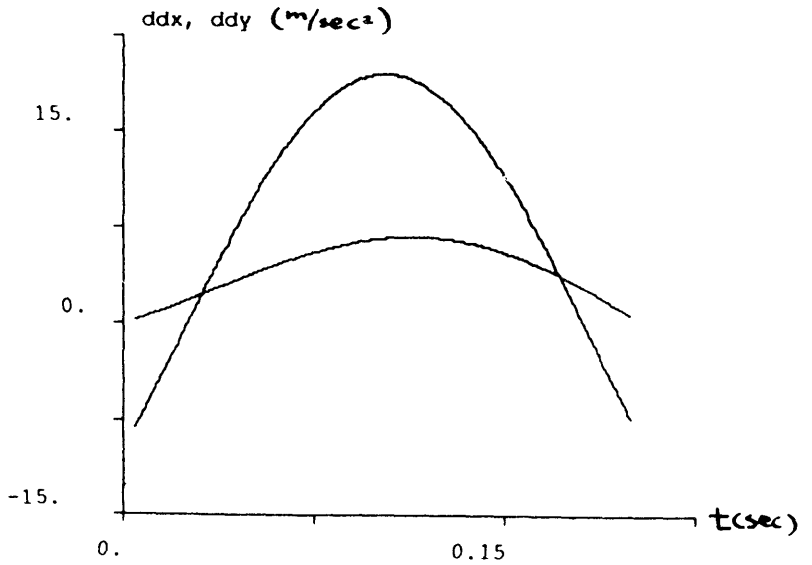


$\theta_{leg_0} = 10°$, $\theta_{ft} = -10°$

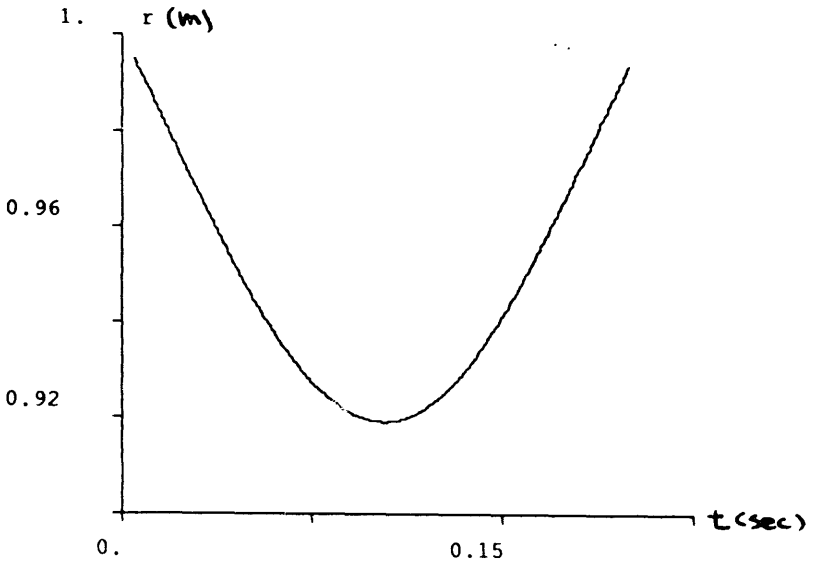


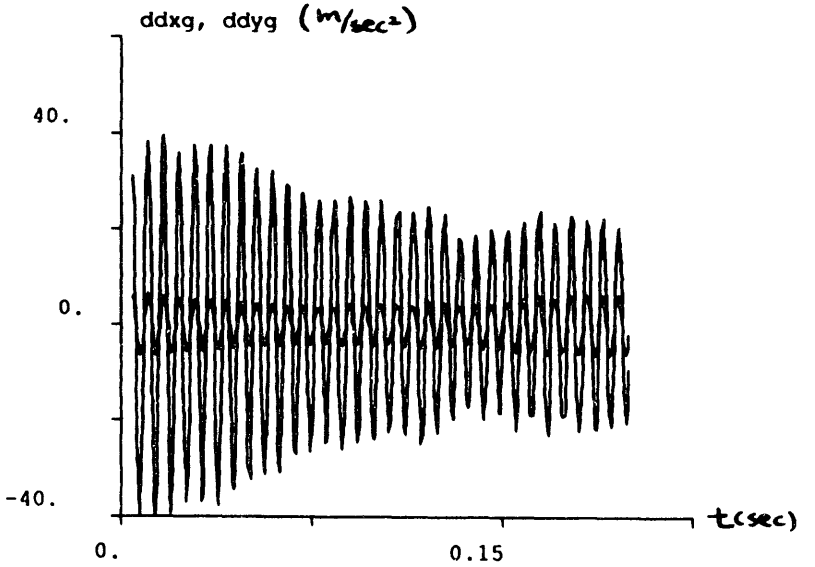


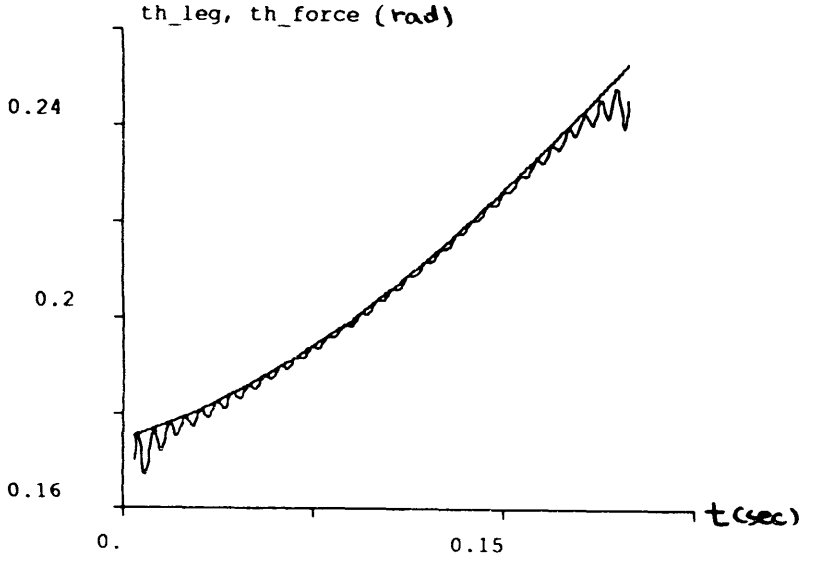


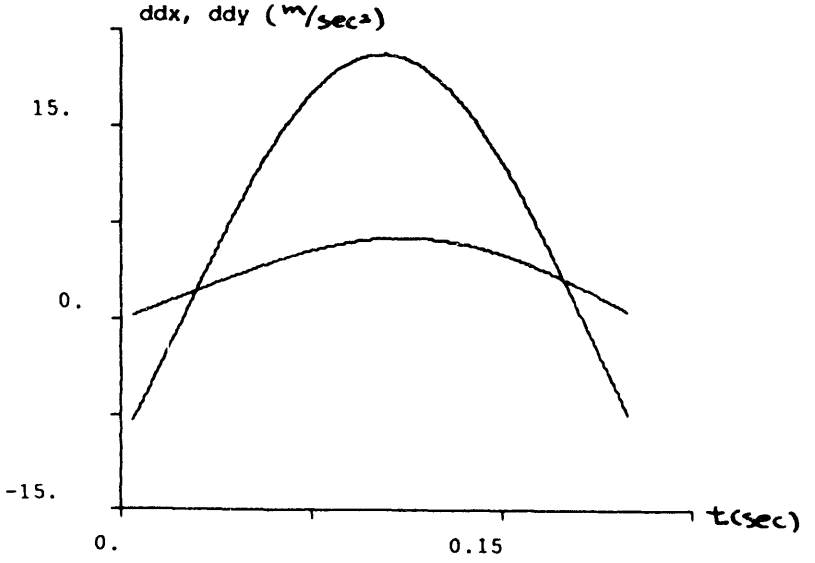


$\theta_{leg_0} = 10°$, $\theta_{tA} = 0$

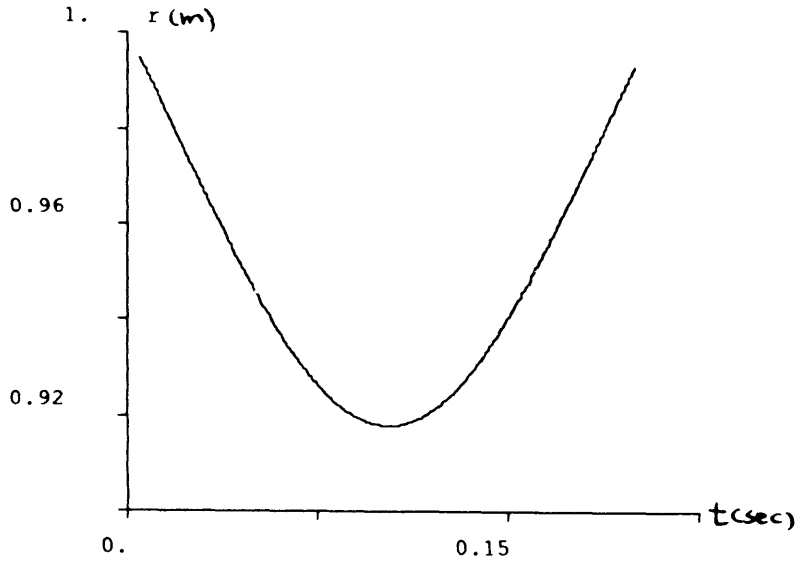


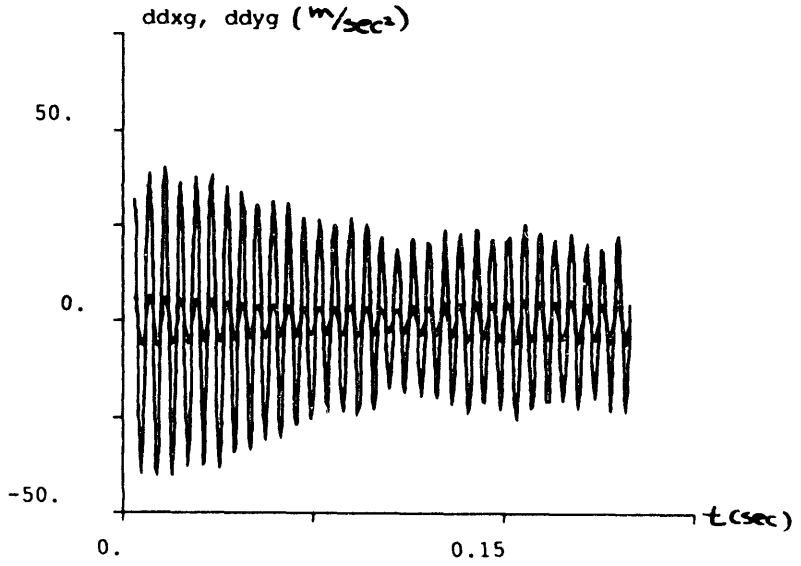


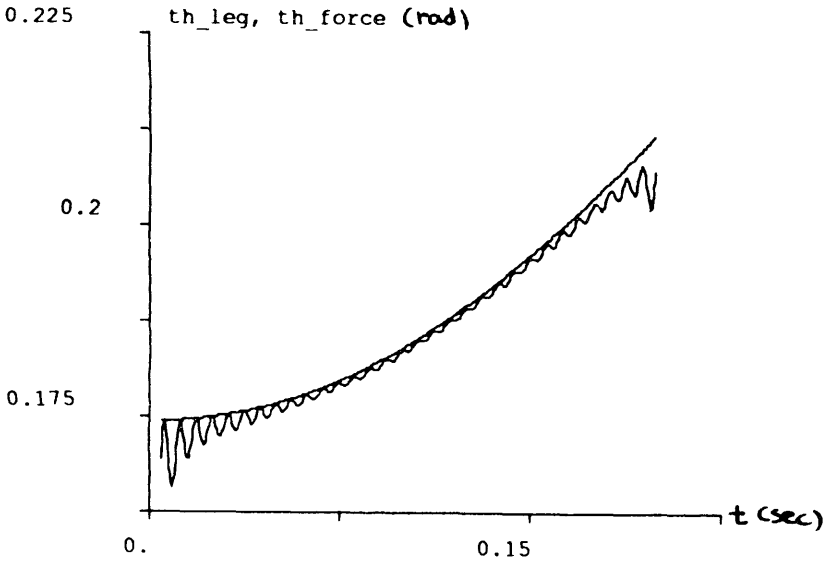


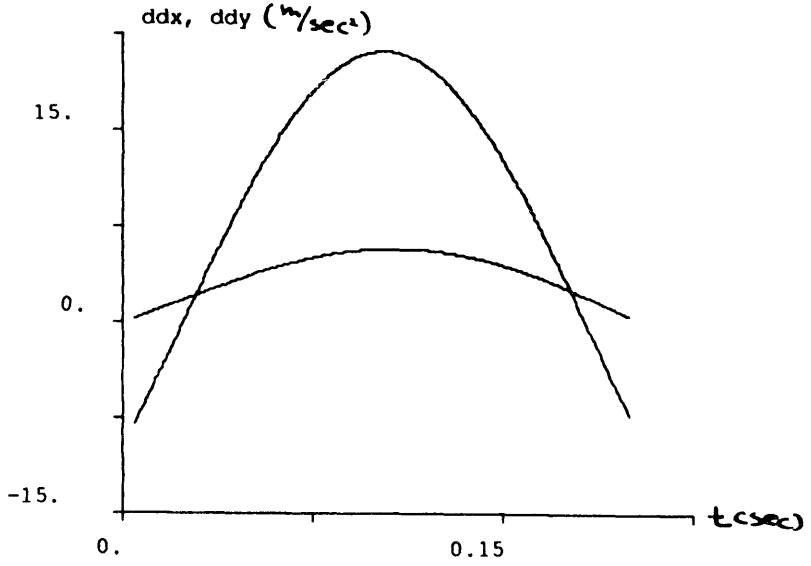


$\theta_{leg_0} = 10°$, $\theta_{td} = 10°$

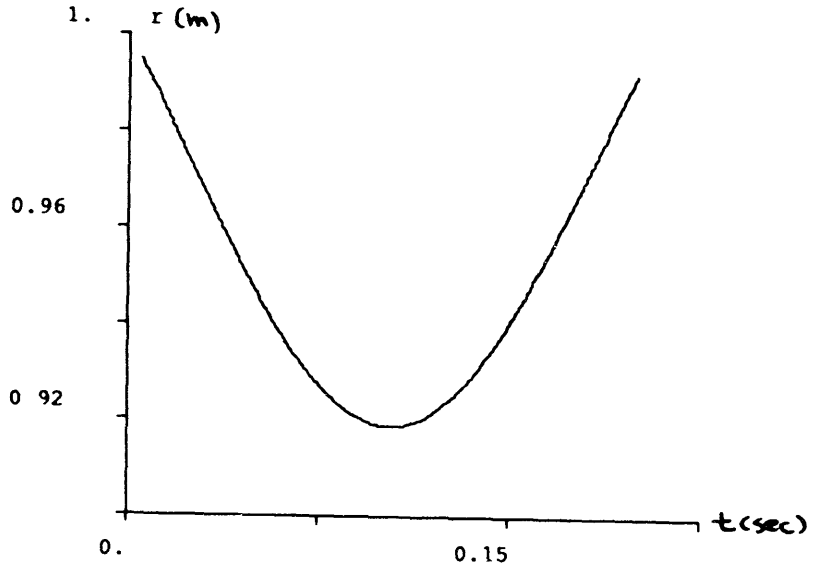


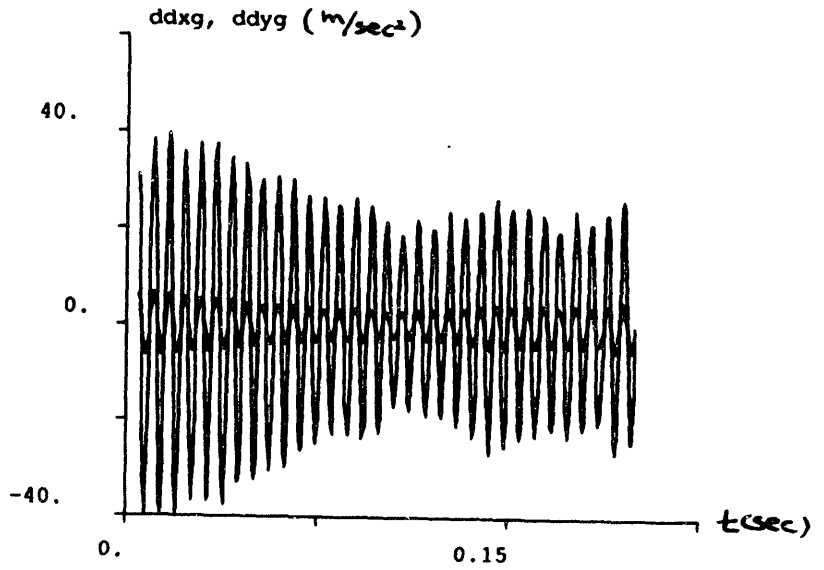


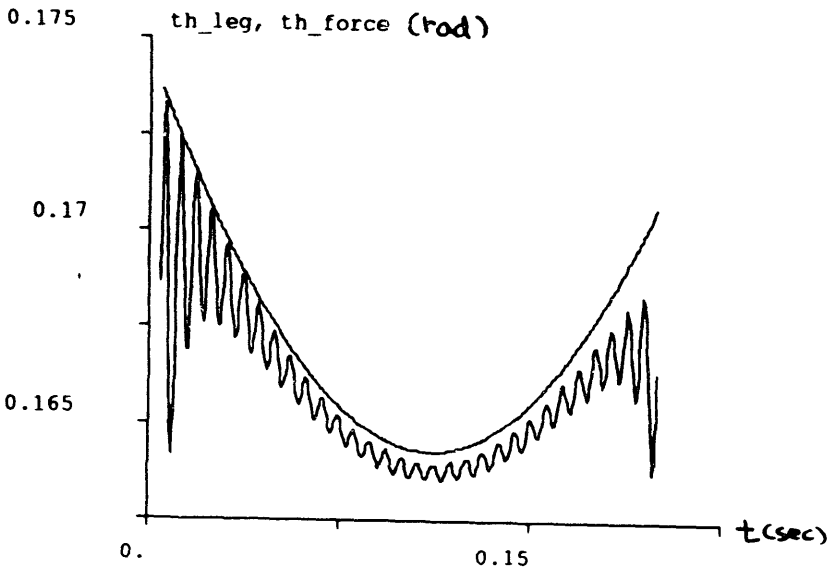


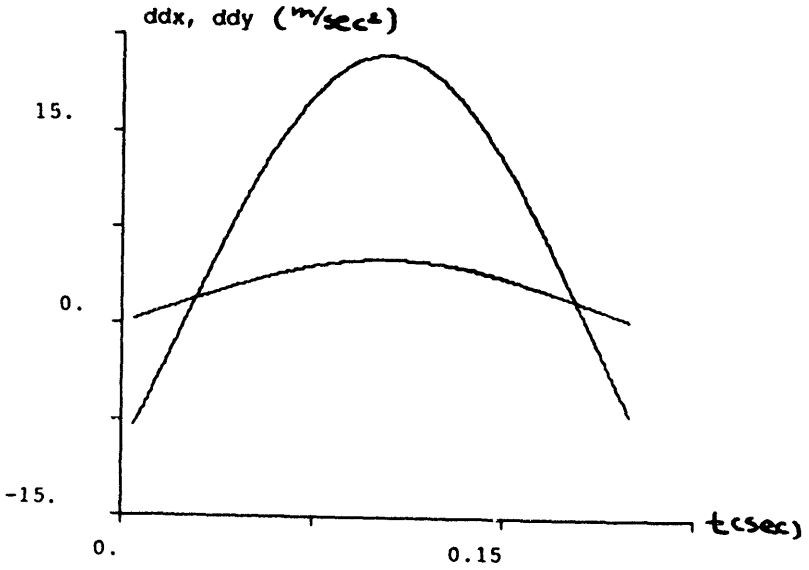


$\theta_{leg-0} = 10°$, $\theta_{rd} = 20°$

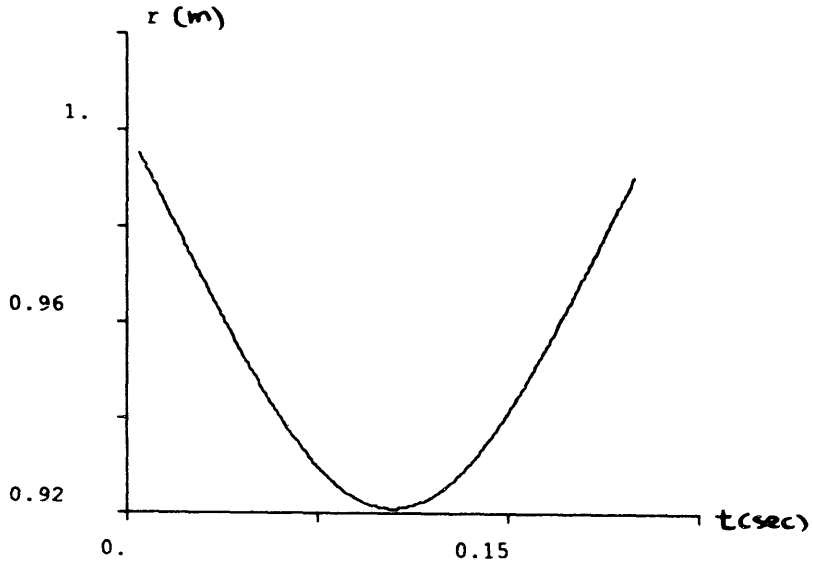


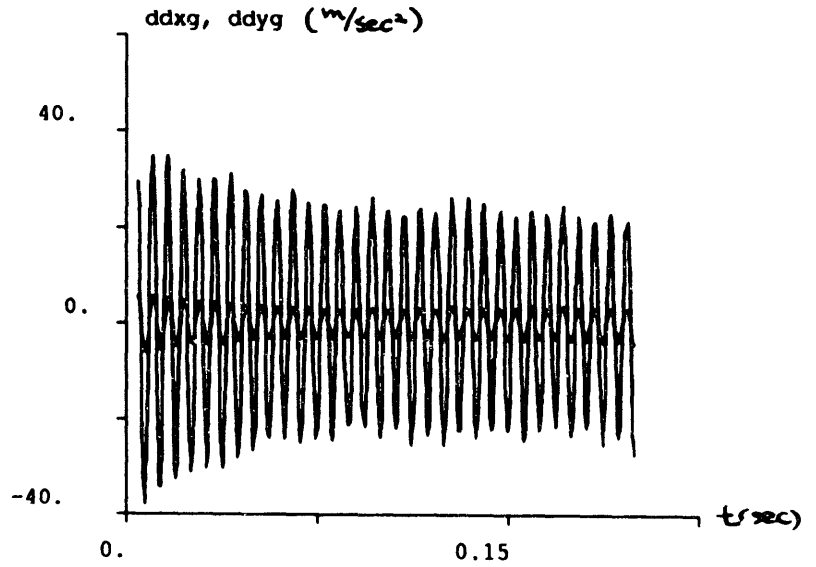


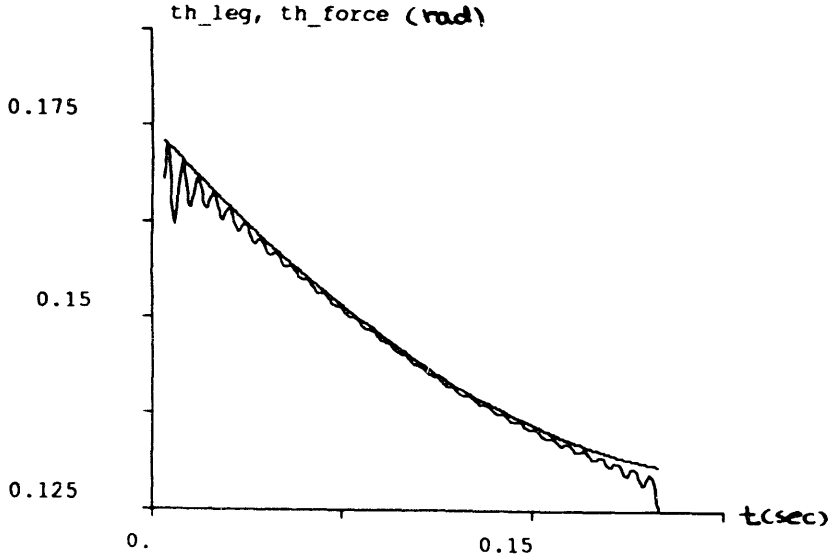


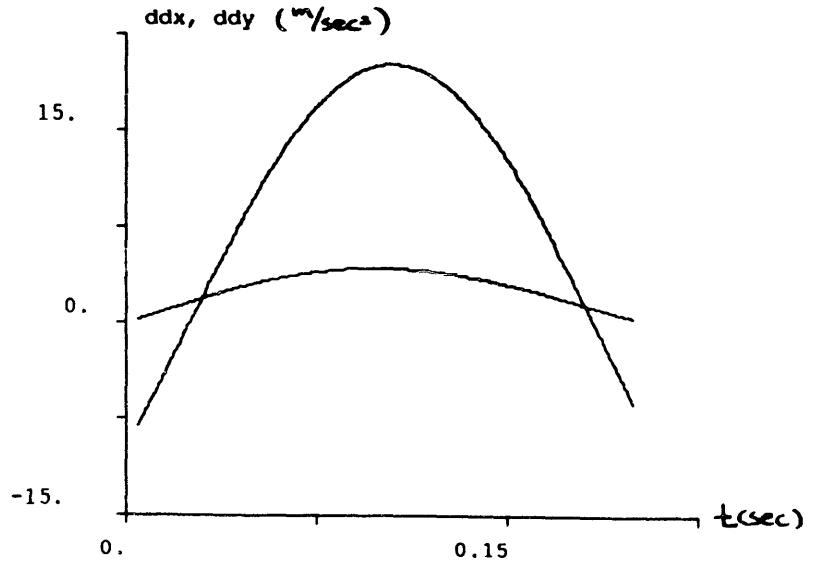


$\theta_{leg,0} = 10°$, $\theta_{td} = 30°$

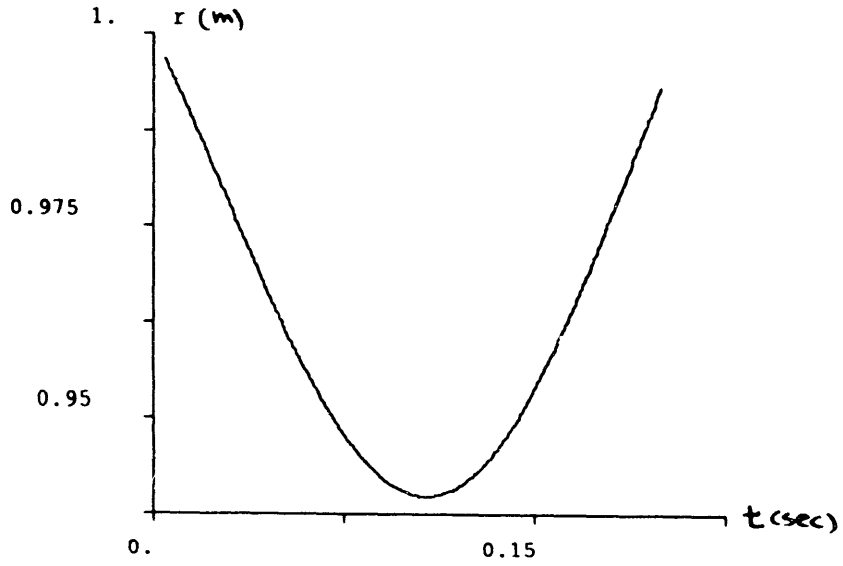


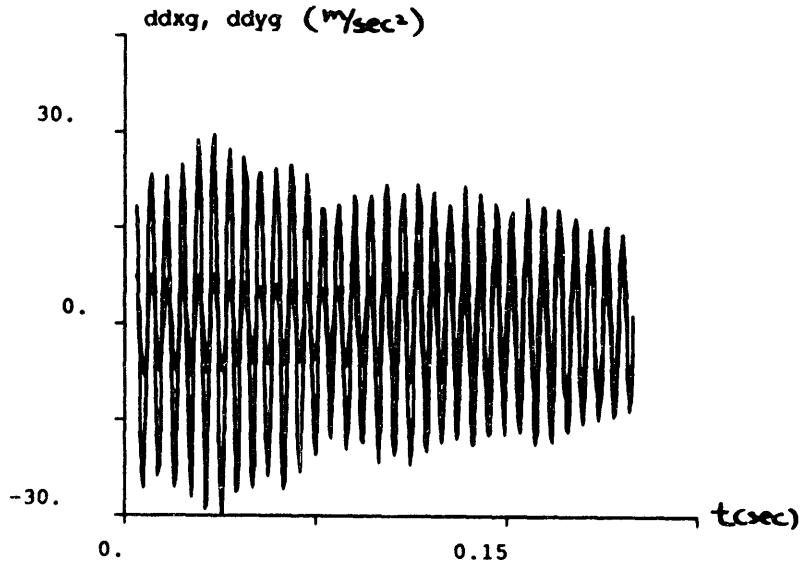


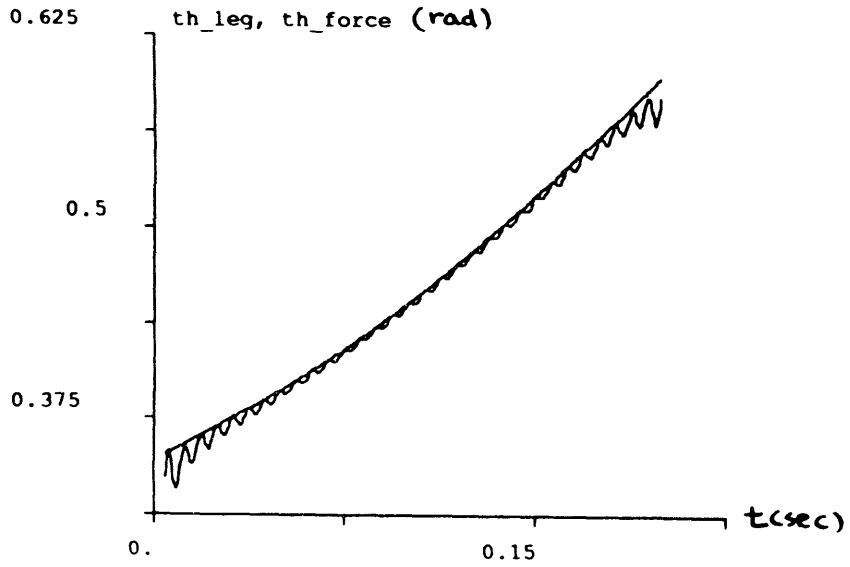


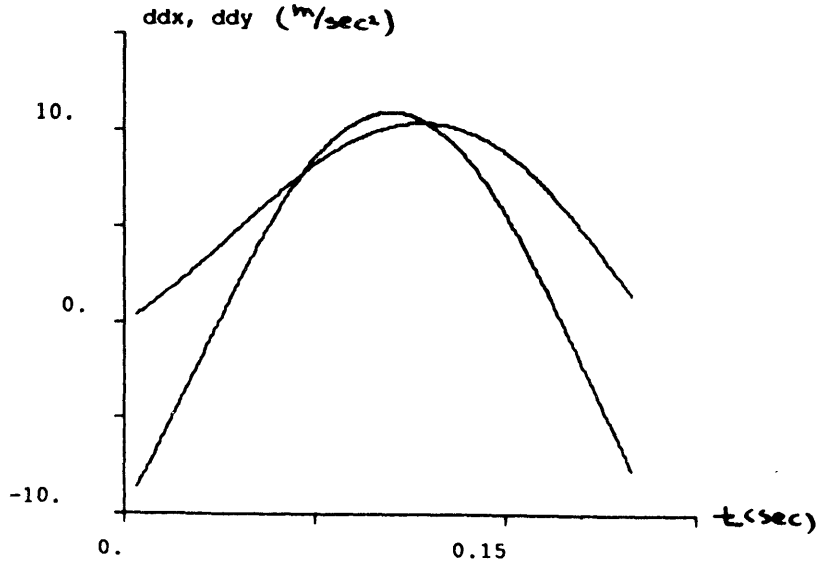


$\theta_{leg_0} = 20°$, $\theta_{fd} = -30°$

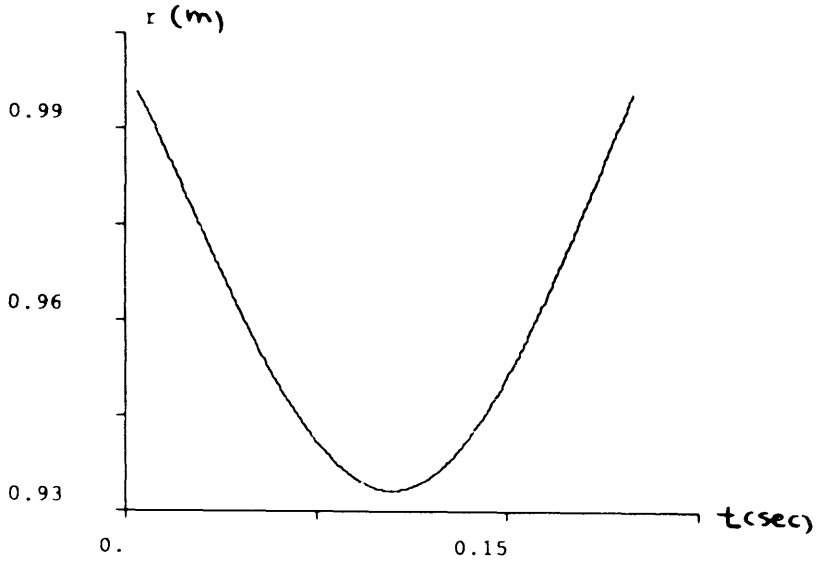


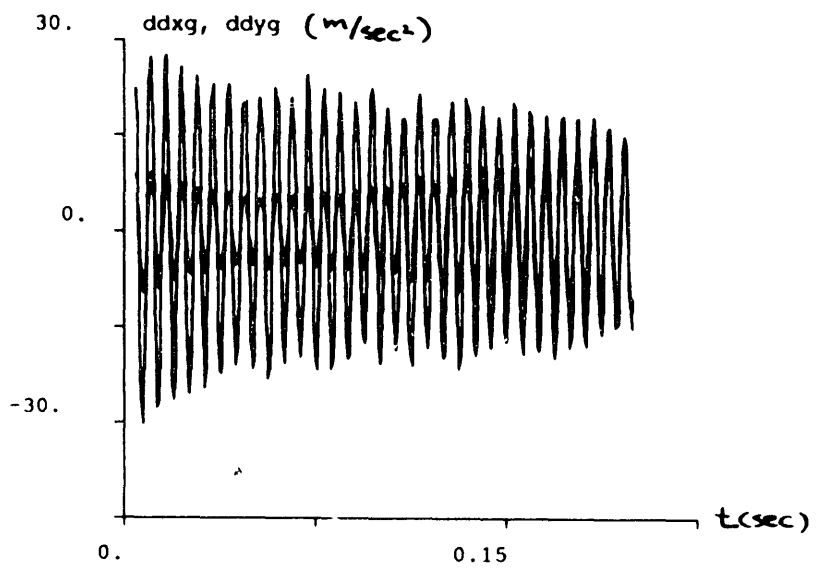


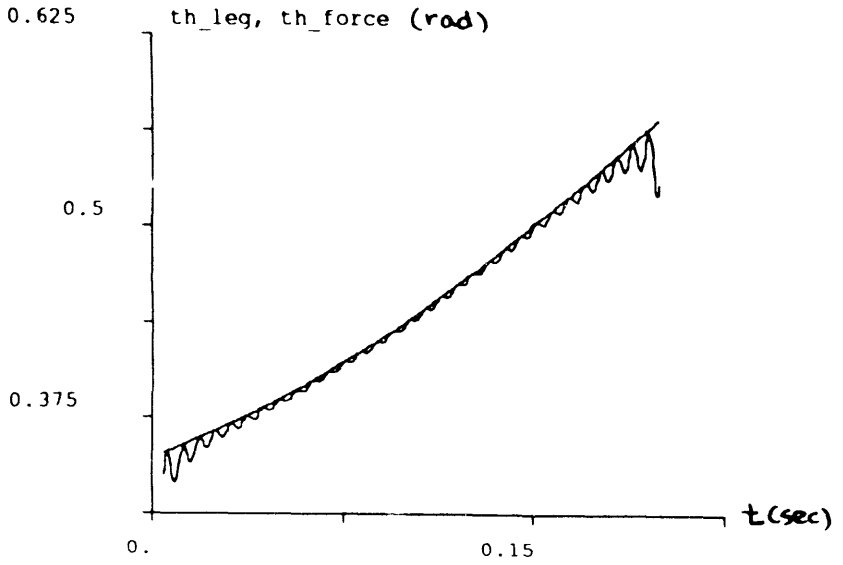


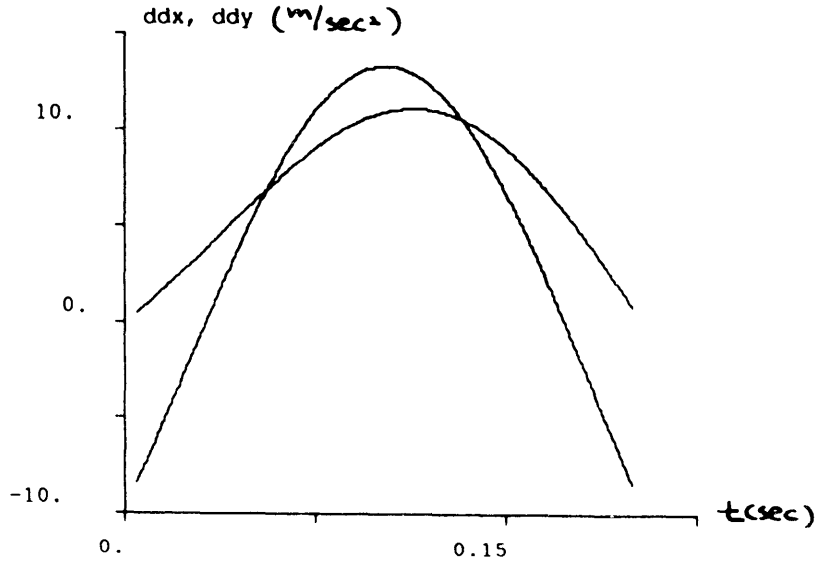


$\theta_{leg-0} = 20°$, $\theta_{td} = -20°$

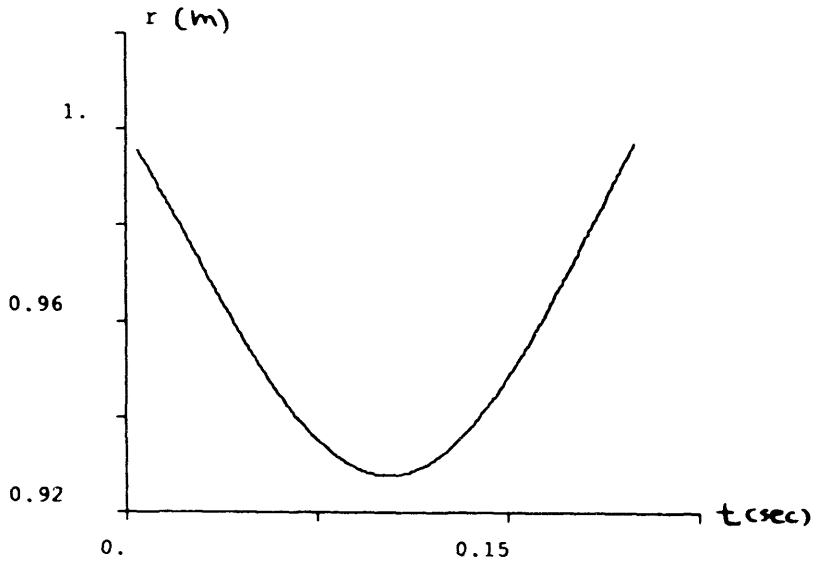


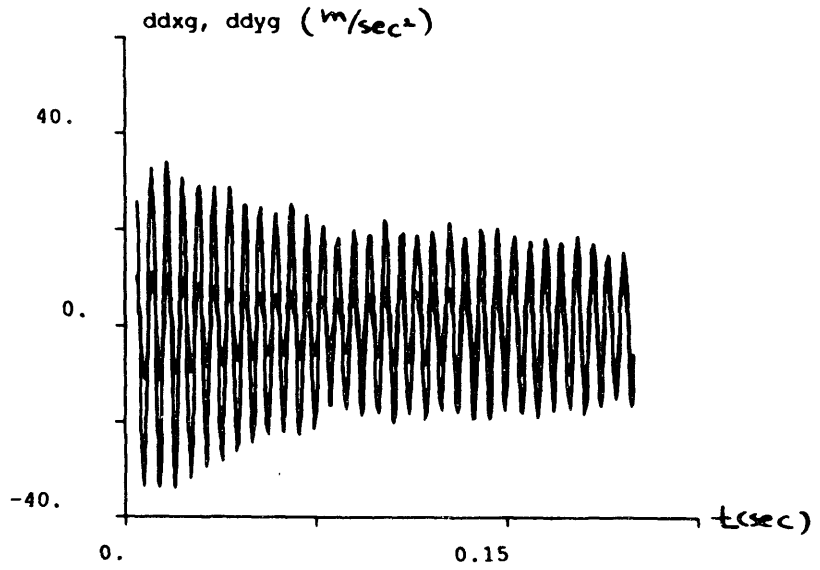


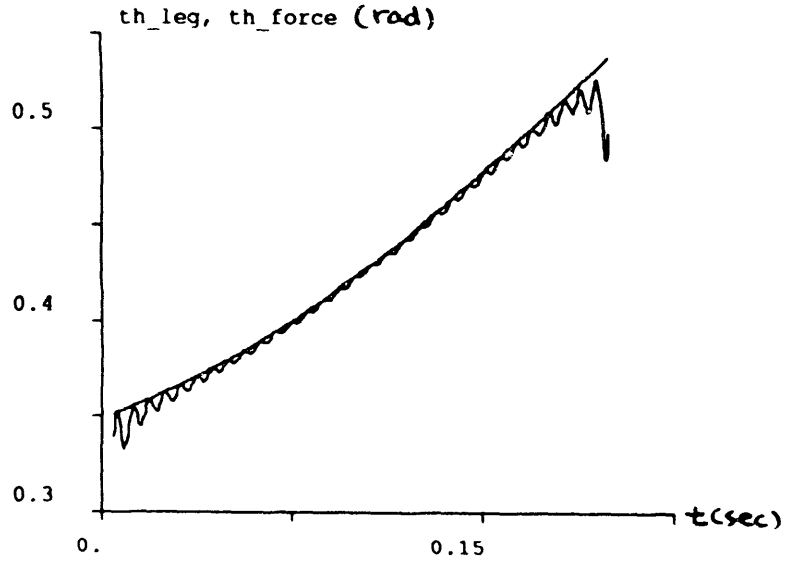


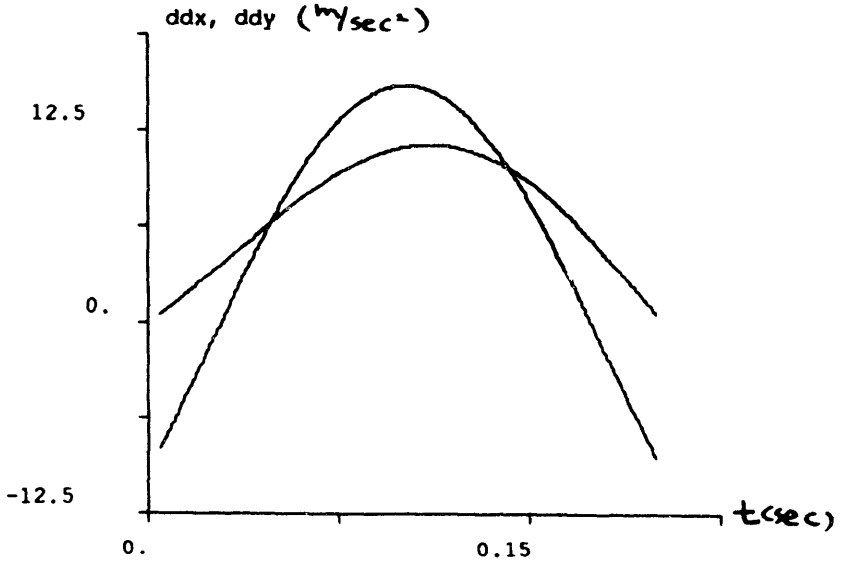


$\theta_{leg_0} = 20°$, $\theta_{ts} = -10°$

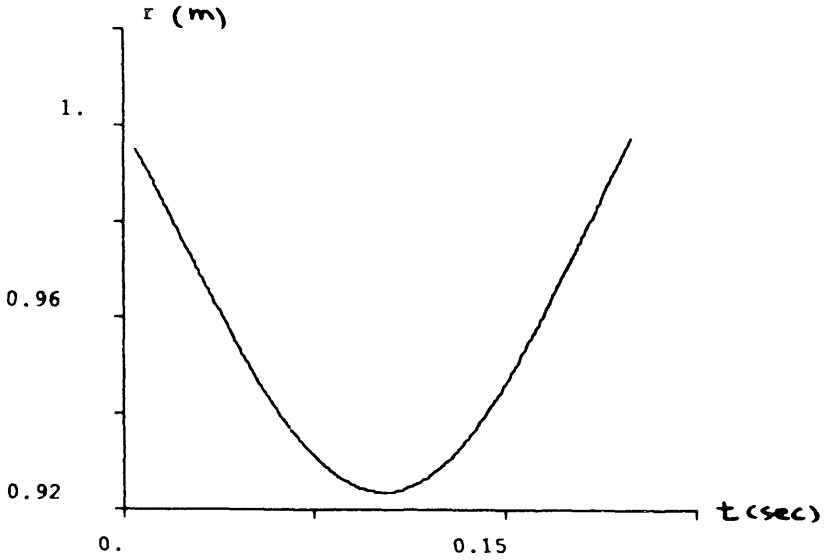


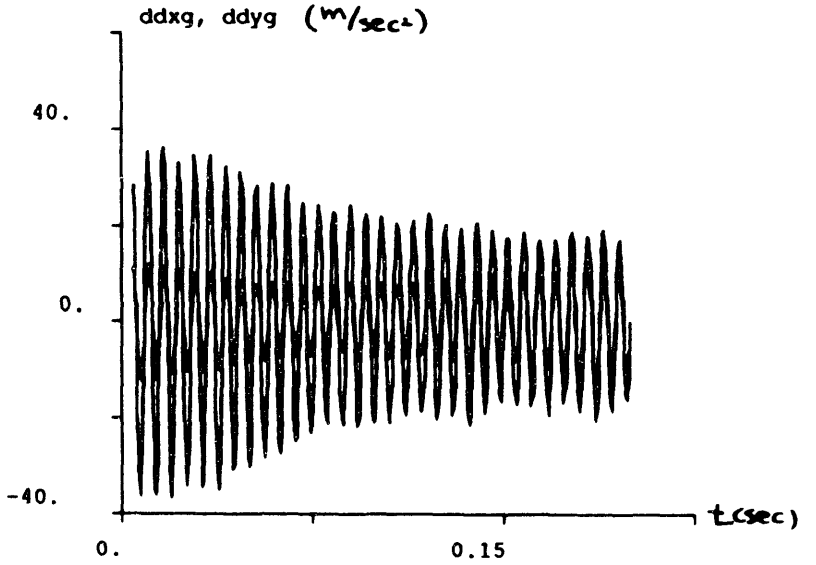


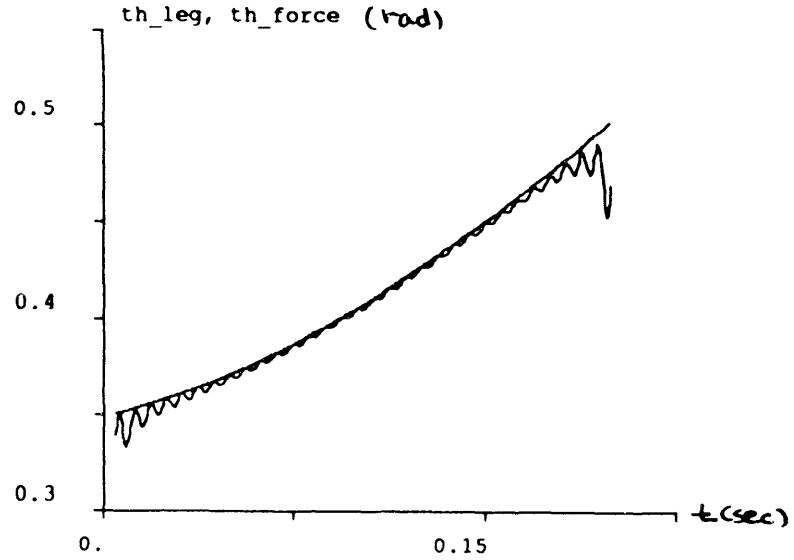


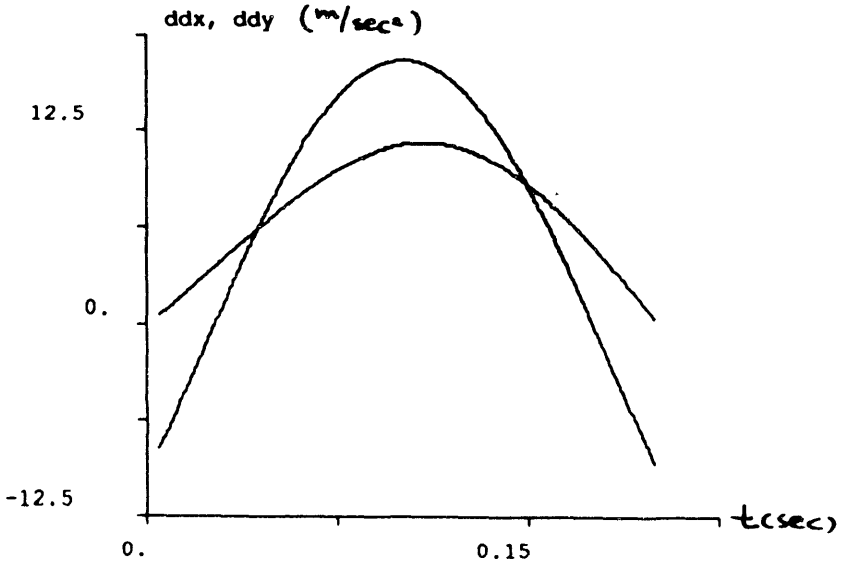


$\theta_{leg-0} = 20°$, $\theta_{td} = 0°$

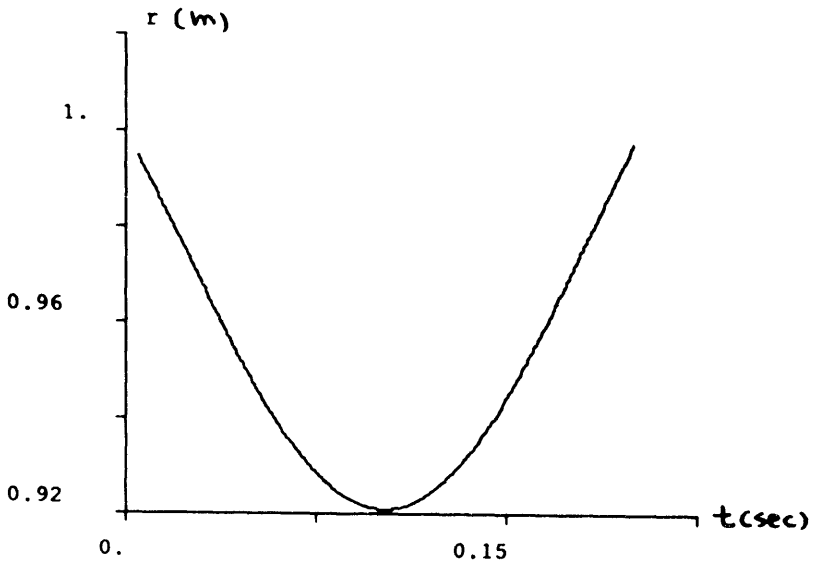


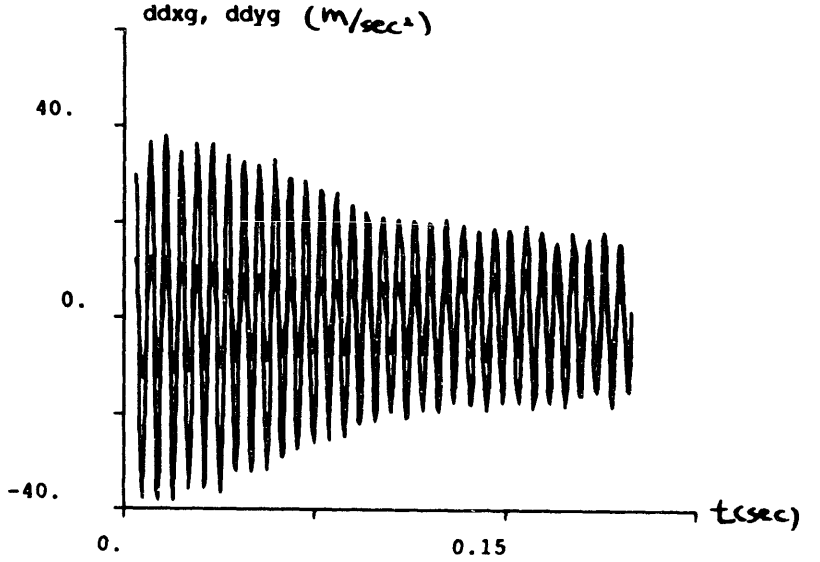


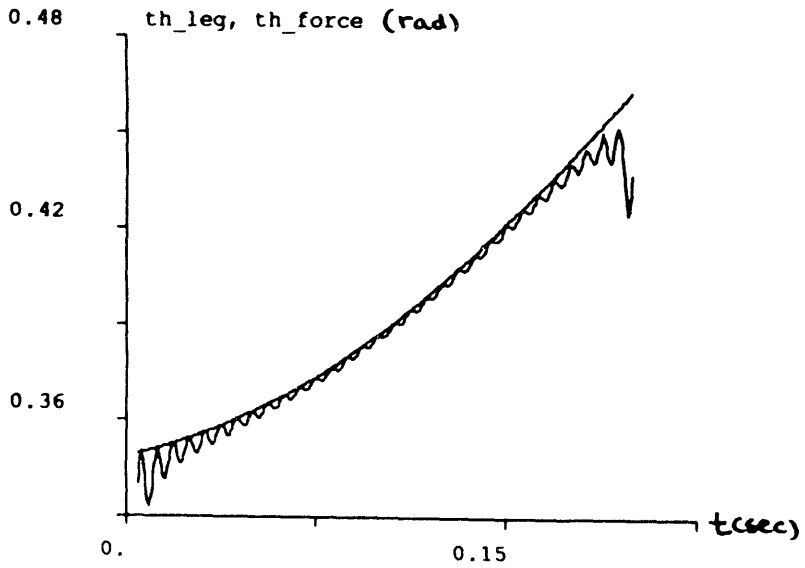


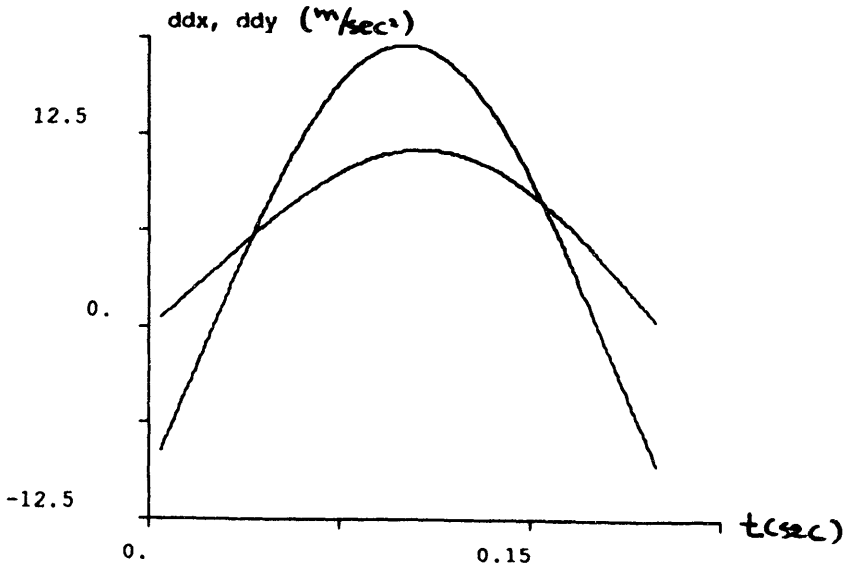


$\theta_{leg\text{-}o} = 20°$, $\theta_{td} = 10°$

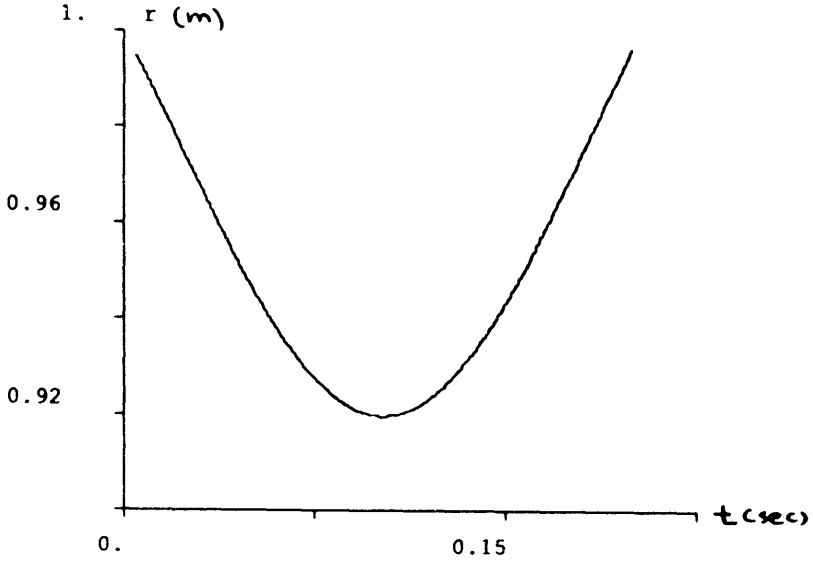


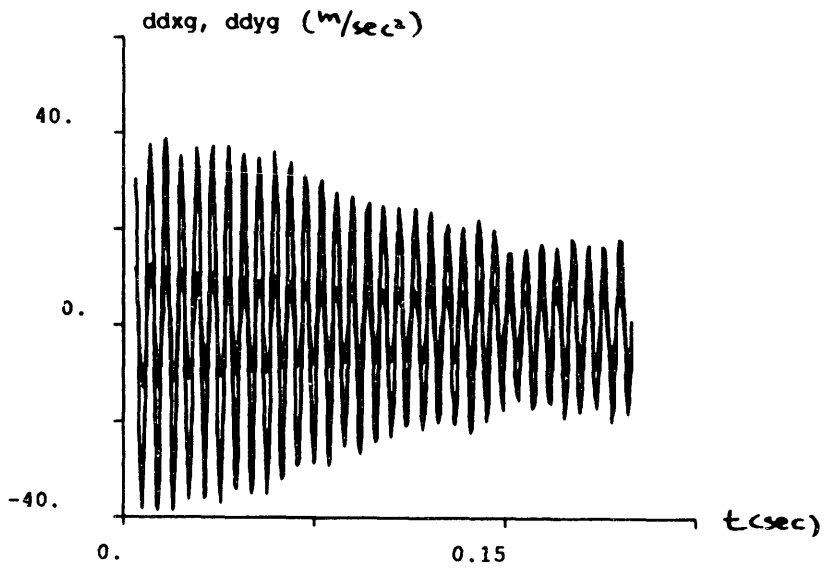


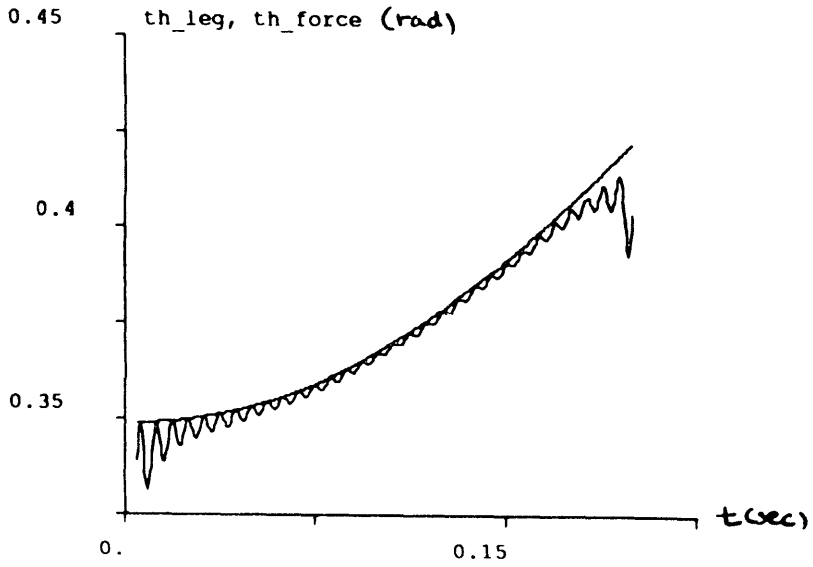


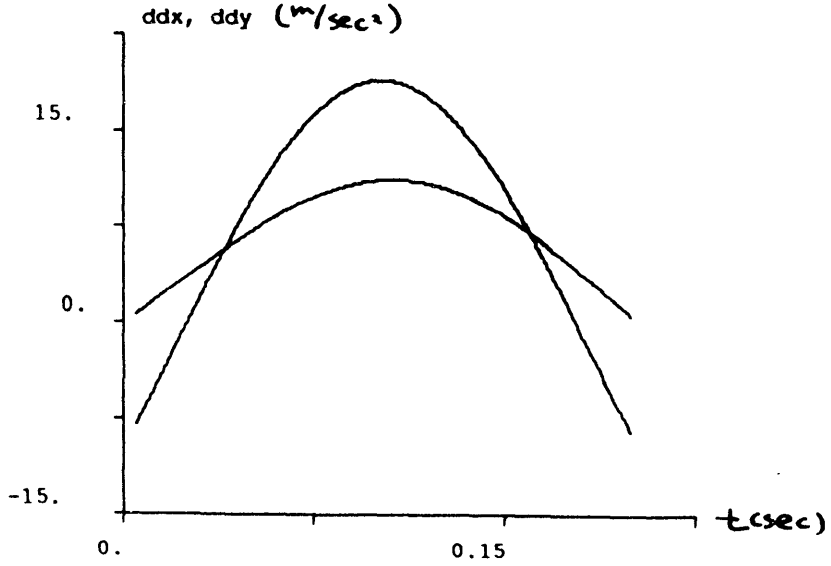


$\theta_{leg,0} = 20°$, $\theta_{ft} = 20°$

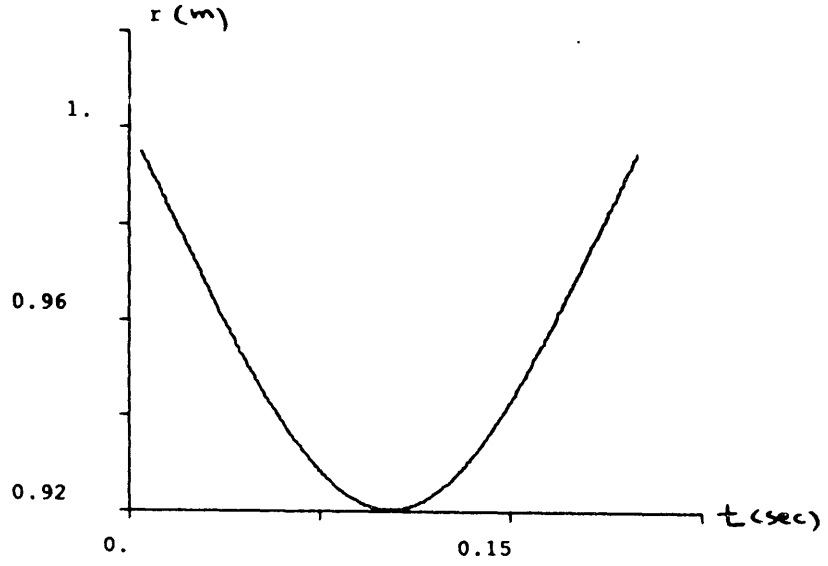


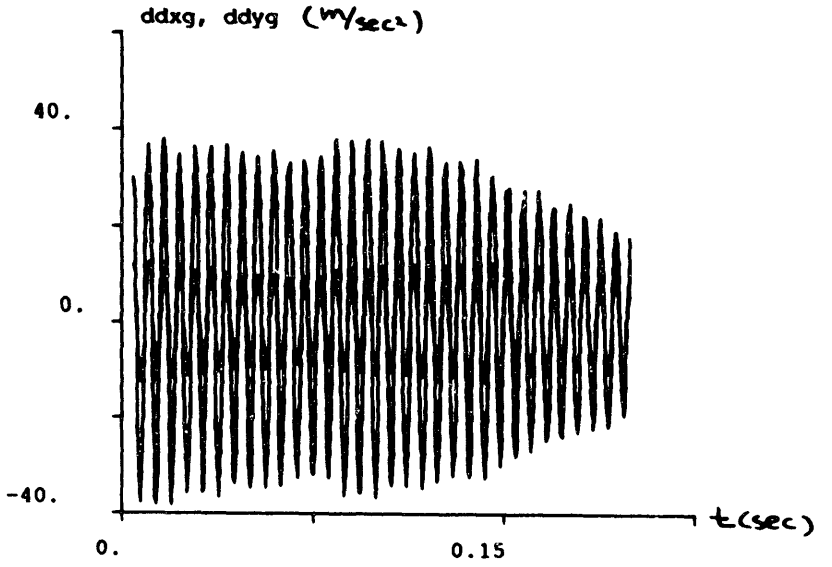


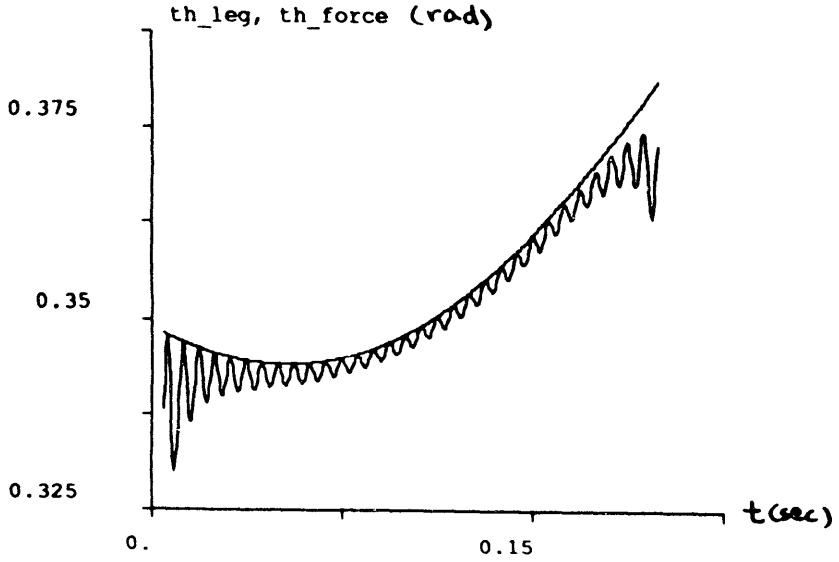


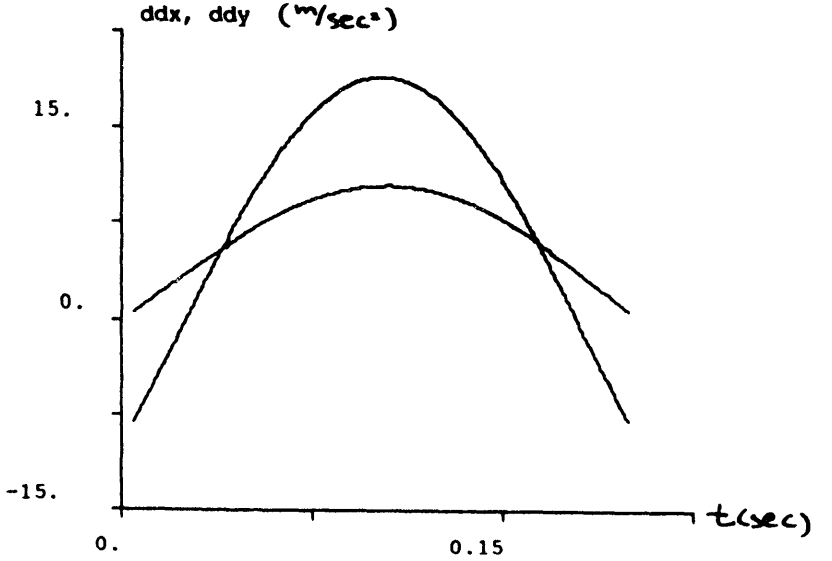


$\theta_{Leg-0} = 20°$, $\theta_{[illegible]} = 30°$









Appendix H:
Real Time Control Algorithm for the Monopod

```
/*****************************************************************/
/* kwp_mint.c - Monopod Controller code.                         */
/*****************************************************************/
/*
 * HISTORY
 *
 * 24-Jul-89  Woojin Lee (wlee) at MIT AI Lab added the x & z
 *      sensors on monopod's planerizer, and got rid of the
 *      x & z estimation scheme. Usage of x & z sensors requires
 *      16 more words. Thus, monopod uses total 32 words on vax.
 *
 * 05-Jul-89  Woojin Lee (wlee) at MIT AI Lab added the EWAIT
 *      which waits for the stable encoder data.
 *
 * 30-Jun-89  Woojin Lee (wlee) at MIT AI Lab changed the
 *      Monopod io_board A/D conversion waiting time from
 *      4.67 usec to 9.66 usec to help eliminate glitches.
 *
 * 26-Jun-89  Woojin Lee (wlee) at MIT AI Lab modified the
 *      Monopod Controller code to estimate the x and z
 *      positions of the center of mass of Monopod.
 *
 * 19-Jun-89  Woojin Lee (wlee) at MIT AI Lab
 *      Replaced the pitch sensor from the potentiometer to the
 *      optical encoder, and modified the code for its usage.
 *
 * 26-Sep-86  Michael Chepponis (mac) at Carnegie-Mellon University
 *          Add reading of channels 11 and 15 into variables "ch11" and
 *          "ch15" - they are not scaled, are strictly raw sensor readings.
 *          Only in fast sensor read mode (gait == 100)
 *          Also add variable "bits" to hold contents of the I/O board's
 *          digital inputs.
 *
 * 18-Sep-86 Jeff Koechling (jck) at Carnegie-Mellon University
 *  Added Gait100 to read sensors faster. Tried to fix things so
 *  that it will all work when clock is less than 10 or not
 *  divisible by ten. Replace local_dt and record_timer with
 *  mop.dt and mop.rec_tim.
 *
 * 28-Jul-86  Jessica Hodgins (jkh) and Ben Brown (hbb) at CMU
 *  Changed a whole bunch of stuff (<- great comment!)
 *  (Yeah, I like that comment, too! -mac)
 *
 * 28-Jul-86 Jeff Koechling (jck) at Carnegie-Mellon University
 *  Installed new xpot code and removed obsolete assembly language
 *  Added usec clock reads to time interrupt routine.
 *
 * 10-Jul-86  Jessica Hodgins (jkh) and Ben Brown (hbb) at CMU
 *  Added set button
```

```
*
* 08-Jul-86 Jessica Hodgins (jkh) and Ben Brown (hbb) at CMU
*  added gait 1
*
* 04-Jun-86 Jessica Hodgins (jkh) and Ben Brown (hbb) at CMU
*  Got rid of micro-second clock, added separate ringlens for
*  x,z,pitch,theta's. Recomputed thetaheel to model geometry of
*  machine more accurately. Added theta2d_nom to pitch servo
*  during stance. Added zd.
*
* 08-Apr-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Add velocity feedback to the servos. Introduce "mop.thetaheeld".
*
* 07-Apr-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Convert angles to degrees (actually, a calibration problem), and
*  add some new angle definitions. Now servo thetaheel instead of
*  theta1.
*
* 31-Mar-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Add flight servo for theta2. Requires a computation of x, xd,
*  ts and xd_d. Kxd also required.
*
* 27-Mar-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Try to get a ground pitch servo installed; add another state,
*  300: Compression. Gnd servo in states 300 and 100 (thrust).
*
* 26-Mar-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Add simple hopping algorithm, in use when switch 0x100 off.
*
* 26-Mar-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Add simple servos for slider setpoints for theta1 and theta2.
*  Also, T1 is for theta1, and T2 is for Hip (theta2). Compute
*  deltafd, theta1d, theta2d.
*
* 19-Mar-86 Michael Chepponis (mac) at Carnegie-Mellon University
*  Created from kwp_qint.c
*/
```

```
/************************************************************************/
/* Include files                                                        */
/************************************************************************/
#include "../h/ioctl.h"
#include "../h/param.h"
#include "../h/dir.h"
#include "../h/user.h"
#include "../h/buf.h"
#include "../h/tty.h"
#include "../h/systm.h"
#include "../h/map.h"
#include "../machine/pte.h"
#include "../machine/mtpr.h"
#include "../vaxuba/pdma.h"
#include "../vaxuba/ubavar.h"
#include "../vaxuba/ubareg.h"
#include "../vaxuba/kwpreg.h"
#include "../vaxuba/kwp_m_vars.h"
#include <math.h>
```

```
/*****************************************************************/
/* external variable declarations:                               */
/*****************************************************************/
/*
 * Declaration of homemade trig, etc. routines
 */
extern float mcos(), msin(), mtan(), macos(), masin(), matan(),
   msqrt01(), msqrt();


#define ZBASE        (0166740)
#define ZRST   ZBASE + (000)
#define ZSTART  ZBASE + (002)
#define ZLOWD   ZBASE + (004)
#define ZHIGHD  ZBASE + (006)


#define XBASE        (0166620)
#define XRST   XBASE + (000)
#define XSTART  XBASE + (002)
#define XLOWD   XBASE + (004)
#define XHIGHD  XBASE + (006)


#define PBASE        (0166760)
#define PRST   PBASE + (000)
#define PSTART  PBASE + (002)
#define PLOWD   PBASE + (004)
#define PHIGHD  PBASE + (006)


#define gravity (384.0)

double sqrt ();
/*
 * stuff it needs to run:
 */
extern struct uba_device *kwpdinfo[];
extern struct kwp_struct kwp_desc;
extern struct record_control rr;
extern struct record_struct rs;
extern struct kwp_struct kwp_desc;
extern dev_t save_dev;

extern short Lap_Box_LEDs;

extern int old_clock;
extern int clock;

float pitch_ring[70];
float x_ring[70];
float z_ring[70];
float zd_ring[50];
float phi_ring[50];
float thetatoe_ring[50];
float deltaf_ring[50];
float theta1_ring[50];
```

```
float theta2_ring[50];
float thetaheel_ring[50];
float legxd_ring[50];
float legzd_ring[50];
/* float treadd_ring[50]; */
float ts_save;   /* For computing Ts, time of stance */
extern float *var_addresses[];
extern float *src_addresses[];
extern float *dst_addresses[];

extern double accum_time;
extern double pitch;
extern double sp;
extern double cp;
extern float deg_to_rad;
extern float rad_to_deg;

/*************************************************************************/
/* 1l Monopod variables:                                                 */
/*************************************************************************/
extern struct m_variables mop;

int monopod_time;
```

```
/*****************************************************************/
/* Macros:                                                       */
/*****************************************************************/
/*
 * Unibus addressing macro:
 */
#define U(Dev_Addr) \
   (*((short *) ((char *) kwpaddr - KWP_OFFSET + (Dev_Addr))))
/*
 * Macros to read lapbox analog channels:
 */
/*
 * There should be 5.6 us of time waiting for A->D converter
 * on the lapbox I/O board.  This macro is
 * 24 NOPs * .333 usec each --> 8.00 us waiting time
 */
#define WAIT asm("nop"); /* 1 */\
        asm("nop"); /* 2 */\
        asm("nop"); /* 3 */\
        asm("nop"); /* 4 */\
        asm("nop"); /* 5 */\
        asm("nop"); /* 6 */\
        asm("nop"); /* 7 */\
        asm("nop"); /* 8 */\
        asm("nop"); /* 9 */\
        asm("nop"); /* 10 */\
        asm("nop"); /* 11 */\
        asm("nop"); /* 12 */\
        asm("nop"); /* 13 */\
        asm("nop"); /* 14 */\
        asm("nop"); /* 15 */\
        asm("nop"); /* 16 */\
        asm("nop"); /* 17 */\
        asm("nop"); /* 18 */\
        asm("nop"); /* 19 */\
        asm("nop"); /* 20 */\
        asm("nop"); /* 21 */\
        asm("nop"); /* 22 */\
        asm("nop"); /* 23 */\
        asm("nop")  /* 24 */

#define LAPBOX(a,b,c) \
       U(IO1ADR) = 0; \
       WAIT; \
       a = ((float) U(IO1ADR) * b + c)
```

```
/*
 * There should be 4.0 us of time waiting for A->D converter
 * on the monopod I/O board.  This macro is
 * 14 NOPs * .333 usec each --> 4.67 us waiting time
 * 15 NOPs * .333 usec each --> 5.00 us waiting time
 * 16 NOPs * .333 usec each --> 5.33 us waiting time
 * 18 NOPs * .333 usec each --> 6.00 us waiting time
 * 29 NOPs * .333 usec each --> 9.66 us waiting time
 */
#define MWAIT asm("nop"); /* 1 */\
        asm("nop"); /* 2 */\
        asm("nop"); /* 3 */\
        asm("nop"); /* 4 */\
        asm("nop"); /* 5 */\
        asm("nop"); /* 6 */\
        asm("nop"); /* 7 */\
        asm("nop"); /* 8 */\
        asm("nop"); /* 9 */\
        asm("nop"); /* 10 */\
        asm("nop"); /* 11 */\
        asm("nop"); /* 12 */\
        asm("nop"); /* 13 */\
        asm("nop"); /* 14 */\
        asm("nop"); /* 15 */\
        asm("nop"); /* 16 */\
        asm("nop"); /* 17 */\
        asm("nop"); /* 18 */\
        asm("nop"); /* 19 */\
        asm("nop"); /* 20 */\
        asm("nop"); /* 21 */\
        asm("nop"); /* 22 */\
        asm("nop"); /* 23 */\
        asm("nop"); /* 24 */\
        asm("nop"); /* 25 */\
        asm("nop"); /* 26 */\
        asm("nop"); /* 27 */\
        asm("nop"); /* 28 */\
        asm("nop")  /* 29 */

/*
 * There should be 8.33 us of time waiting for encoder data
 * to be stable.  This macro is
 * 14 NOPs * .333 usec each --> 4.67 us waiting time
 * 15 NOPs * .333 usec each --> 5.00 us waiting time
 * 16 NOPs * .333 usec each --> 5.33 us waiting time
 * 24 NOPs * .333 usec each --> 8.00 us waiting time
 */

#define EWAIT asm("nop"); /* 1 */\
        asm("nop"); /* 2 */\
        asm("nop"); /* 3 */\
        asm("nop"); /* 4 */\
        asm("nop"); /* 5 */\
        asm("nop"); /* 6 */\
        asm("nop"); /* 7 */\
        asm("nop"); /* 8 */\
        asm("nop"); /* 9 */\
```

```
        asm("nop"); /* 10 */\
        asm("nop"); /* 11 */\
        asm("nop"); /* 12 */\
        asm("nop"); /* 13 */\
        asm("nop"); /* 14 */\
        asm("nop"); /* 15 */\
        asm("nop"); /* 16 */\
        asm("nop"); /* 17 */\
        asm("nop"); /* 18 */\
        asm("nop"); /* 19 */\
        asm("nop"); /* 20 */\
        asm("nop"); /* 21 */\
        asm("nop"); /* 22 */\
        asm("nop"); /* 23 */\
        asm("nop")  /* 24 */


#define M(a,b,c) \
        U(MADR) = 0; \
        MWAIT; \
        a = ((float) U(MADR) * b + c)

#define CLAMP(v,l,h) if ((v) < (l)) (v) = (l); \
             else if ((v) > (h)) (v) = (h); \
           else

#define is :{
#define esac    break;}

/*****************************************************************/
/* Symbolic Constants:                                           */
/*****************************************************************/
/*
 * Conversion factors:
 */
#define PI      (3.1415926535897262846)
#define HALFPI (3.1415926535897262846 / 2.0)     /* PI / 2.0 */
#define TWOPI  (2.0 * 3.1415926535897262846)     /* 2.0 * PI */
#define DTOR   (3.1415926535897262846 / 180.0)   /* PI / 180.0 */
#define RTOD   (180.0 / 3.1415926535897262846)   /* 180.0 / PI */
#define USEC_WRAP 10000.0
```

```
/*****************************************************************/
/* mint - the interrupt routine to control the monopod           */
/*****************************************************************/
mint() {
   register struct uba_device *ui;
   register struct kwpdevice *kwpaddr =
      (struct kwpdevice *) kwpdinfo[minor(save_dev)]->ui_addr;

   float *rsptr;       /* Pointer into big array (rs[][]) */
   float **vaptr;      /* Pointer into address table */
   float **srcptr; /* Pointer into address table (src_addresses) */
   float **dstptr; /* Pointer into address table (dst_addresses) */

   int Lap_Box_Switches; /* The Digital Inputs from the Lap Box   */
   unsigned short start_encoder, low, high;
   long int result;
   double arg;
/*
 * Lap Box Switches:
 *
 *    0x0800(sw)   0x0400(sw)   0x0200(sw)   0x0100(sw)
 *    Halt Driver            Slider Servo
 *
 *    0x0008(pb)   0x0004(pb)   0x0002(pb)   0x0001(pb)
 *    Record
 *
 *    0x8000(sw)   0x4000(sw)   0x2000(sw)   0x1000(sw)
 *
 *
 *    0x0080(pb)   0x0040(pb)   0x0020(pb)   0x0010(pb)
 *
 *
 * Center LEDs:
 *
 *    0x0008      0x0004      0x0002      0x0001
 *    Heartbeat                 Active
 *
 *    0x0080      0x0040      0x0020      0x0010
 *
 * Right side LEDs:
 *
 *    0x8000      0x4000      0x2000      0x1000
 *    (100)       (40)        (20)        (10)
 *
 *    0x0800      0x0400      0x0200      0x0100
 */
```

```
/*****************************************************************/
/* Begin executable code:                                        */
/*****************************************************************/
/*
 * Record time and turn on "driver active" light:
 */
   mop.t_in = (float) mfpr (ICR);
   mop.t_delay = (float) U(KWPCNT);
   Lap_Box_LEDs |= 0x1;
   U(LBDO) = (short)Lap_Box_LEDs;
/*
 * See how long things are taking:
 */
   mop.t_dt = mop.t_in - mop.t_save;
   if (mop.t_dt < 0.0) mop.t_dt += USEC_WRAP;
   mop.t_used = mop.t_out - mop.t_save;
   if (mop.t_used < 0.0) mop.t_used += USEC_WRAP;
   mop.t_save = mop.t_in;
/*
 * If the driver is not open, turn off interrupts and return:
 */
   if (!kwp_desc.kwp_open) {
      U(KWPCSR) &= ~KWPCSR_RUN;
      return;
   }
/*
 * If the clock rate has changed, (and is nonzero) reset the clock:
 * mop.dt is in milliseconds, mop.clock is in tenths of msecs
 */
   if (mop.clock <= 0.0) mop.clock = 1000.0;
   if ((int)mop.clock != old_clock) {
      old_clock = (int)mop.clock;
      U(KWPBUF) = (short)old_clock;
      mop.dt = 0.1 * mop.clock;
   }
/*
 * If the "stop driver" lap box switch is up
 * turn off the "driver active" light, and return:
 */
   Lap_Box_Switches = U(LBDI);
   if (Lap_Box_Switches & 0x800) {
      Lap_Box_LEDs &= ~(0x1);
      U(LBDO) = (short)Lap_Box_LEDs;
      mop.t_out = (float) mfpr (ICR);
      return;
   }
/*
 * Keep track of time and blink "Heartbeat" light:
 */
   accum_time += (double)mop.clock;   /* Total time, milliseconds*/
   if (((long) accum_time) & 0x1000) Lap_Box_LEDs |= 0x8;
   else Lap_Box_LEDs &= ~(0x8);
   monopod_time++;   /* OK, do a clock tick */
```

```
/*****************************************************************/
/* Read Pitch Angle From Encoder                                 */
/*****************************************************************/

/*
 * Start Setup for Reading Switches by selecting MUX1
 */
   start_encoder = (unsigned int) U(PSTART);
/*
 * Wait for stable data
 */
         EWAIT;
/*
 * Read Low Byte Data
 */
   low = (unsigned int) U(PLOWD),
/*
 * Read High Byte Data
 */
   high = (unsigned int) U(PHIGHD),

   (long int) result = (long int) low + (long int) 65536*high,
   if ((long int) result >=8300000) {
     (long int) result = (long int) result - 16777216;
  }
   mop pitch = -1.0 * (long int) result * 2.0 *
         3.1415926535897262846 / 4000.0;

/*****************************************************************/
/* Read Z Position From Encoder                                  */
/*****************************************************************/


/*
 * Start Setup for Reading Switches by selecting MUX1
 */
   start_encoder = (unsigned int) U(ZSTART);
/*
 * Wait for stable data
 */
         EWAIT;
/*
 * Read Low Byte Data
 */
   low = (unsigned int) U(ZLOWD);
/*
 * Read High Byte Data
 */
   high = (unsigned int) U(ZHIGHD);

   (long int) result = (long int) low + (long int) 65536*high;
   if ((long int) result >=8300000) {
     (long int) result = (long int) result - 16777216;
  }
   mop.z = (long int) result * 8.18 / 1024.0 + mop.z_offset;
```

```
/*****************************************************************/
/* Read X Position From Encoder                                  */
/*****************************************************************/

/*
 * Start Setup for Reading Switches by selecting MUX1
 */
   start_encoder = (unsigned int) U(XSTART);
/*
 * Wait for stable data
 */
         EWAIT;
/*
 * Read Low Byte Data
 */
   low = (unsigned int) U(XLOWD);
/*
 * Read High Byte Data
 */
   high = (unsigned int) U(XHIGHD);

   (long int) result = (long int) low + (long int) 65536*high;
   if ((long int) result >=8300000) {
    (long int) result = (long int) result - 16777216;
  }
   mop.x = (long int) result * 8.08 / 1024.0;


/*****************************************************************/
/* Fast sensor read mode                                         */
/*****************************************************************/
        if ((int) mop.gait == 100) {
/*
 * Read the Monopod analog inputs:
 */

          U(MADC) = 1;          /* Begin at channel 1 of monopod */
          MWAIT;
          M(mop.theta1,  mop.theta1f,  mop.theta1zo);  /*  1 */
          M(mop.deltaf,  mop.deltaff,  mop.deltafzo);  /*  2 */
          U(MADC) = 5;          /* Begin at channel 5 of monopod */
          MWAIT;
          M(mop.theta2,  mop.theta2f,  mop.theta2zo);  /*  5 */
          M(mop.p2a,     mop.p2af,     mop.p2azo);     /*  6 */
          M(mop.p2b,     mop.p2bf,     mop.p2bzo);     /*  7 */


/*        U(MADC) = 0; */
/*        MWAIT; */
/*        M(mop.ch0, mop.ch0f, mop.ch0fzo); */
/*        M(mop.ch1, mop.ch1f, mop.ch1fzo); */
/*        M(mop.ch2, mop.ch2f, mop.ch2fzo); */
/*        M(mop.ch3, mop.ch3f, mop.ch3fzo); */
/*        M(mop.ch4, mop.ch4f, mop.ch4fzo); */
/*        M(mop.ch5, mop.ch5f, mop.ch5fzo); */
/*        M(mop.ch6, mop.ch6f, mop.ch6fzo); */
/*        M(mop.ch7, mop.ch7f, mop.ch7fzo); */
/*        M(mop.ch8, mop.ch8f, mop.ch8fzo); */
```

```
/*          M(mop.ch9, mop.ch9f, mop.ch9fzo); */
/*          M(mop.ch10, mop.ch10f, mop.ch10fzo); */
/*          M(mop.ch11, mop.ch11f, mop.ch11fzo); */
/*          M(mop.ch12, mop.ch12f, mop.ch12fzo); */
/*          M(mop.ch13, mop.ch13f, mop.ch13fzo); */
/*          M(mop.ch14, mop.ch14f, mop.ch14fzo); */
/*          M(mop.ch15, mop.ch15f, mop.ch15fzo); */

          mop.bits = (float)U(MDI);          /* Read the digital inputs */

/*
 * Increment the timer and see if it is time to save data:
 */

           mop.rec_tim += mop.dt;
           if (rr.record_go && (mop.rec_tim >= mop.rec_dt)) {
/*
 * Set the slot list pointer and the record structure pointer:
 */
              rsptr = (float *) rs.r[rr.ring_p];
              vaptr = var_addresses;
/*
 * Save some variables for recording or plotting:
 */
              while (*vaptr) *rsptr++ = **vaptr++;
/*
 * Reset the timer and increment the ring pointer:
 *   (rr.ring_p points to oldest data)
 */
              mop.rec_tim = 0.001;
              rr.ring_p++;
              if (rr.ring_p >= MAX_REC) rr.ring_p = 0;
/*
 * Check lapbox switches to see if we should stop recording:
 */
              if (Lap_Box_Switches & 0x08) rr.record_go = 0;
           } /* end of recording code */
/*
 * Turn off driver active light, record time out, and return:
 */
     Lap_Box_LEDs &= ~(0x1);
     U(LBDO) = (short)Lap_Box_LEDs;
     mop.t_out = (float) mfpr (ICR);
     return;
         }
```

```
/*****************************************************************/
/* Read the Thumbwheel Switches:                                 */
/*****************************************************************/
   mop.sw0  = U(SW0) * mop.sw0f;
   mop.sw1  = U(SW1) * mop.sw1f;
   mop.sw2  = U(SW2) * mop.sw2f;
   mop.sw3  = U(SW3) * mop.sw3f;
   mop.sw4  = U(SW4) * mop.sw4f;
   mop.sw5  = U(SW5) * mop.sw5f;
   mop.sw6  = U(SW6) * mop.sw6f;
   mop.sw7  = U(SW7) * mop.sw7f;
   mop.sw8  = U(SW8) * mop.sw8f;
   mop.sw9  = U(SW9) * mop.sw9f;
   mop.sw10 = U(SW10) * mop.sw10f;
   mop.sw11 = U(SW11) * mop.sw11f;
   mop.sw12 = U(SW12) * mop.sw12f;
   mop.sw13 = U(SW13) * mop.sw13f;
   mop.sw14 = U(SW14) * mop.sw14f;
   mop.sw15 = U(SW15) * mop.sw15f;

/*****************************************************************/
/* Read the Lapbox analog inputs:                                */
/*****************************************************************/

/* Read treadmill speed */
  U(LBADC) = 3;      /* Channel 10 on Lap Box, treadd        */
  WAIT;
  LAPBOX(mop.treadd, mop.treaddf, mop.treaddzo); /* Channel 3   */

/*
 * Read the sliders and the joystick:
 */
  U(LBADC) = 10;   /* Channel 10 on Lap Box, slide4 */
  WAIT;
  LAPBOX(mop.slide4, mop.slide4f, mop.slide4zo);   /* Channel 10 */
  LAPBOX(mop.joyx, mop.joyxf, mop.joyxzo);      /* Channel 11 */
  LAPBOX(mop.joyy, mop.joyyf, mop.joyyzo);      /* Channel 12 */
  LAPBOX(mop.slide1, mop.slide1f, mop.slide1zo);   /* Channel 13 */
  LAPBOX(mop.slide2, mop.slide2f, mop.slide2zo);   /* Channel 14 */
  LAPBOX(mop.slide3, mop.slide3f, mop.slide3zo);   /* Channel 15 */

/*****************************************************************/
/* Read the Monopod analog inputs: (aaaa)                        */
/*****************************************************************/


  U(MADC) = 1;         /* Begin at channel 1 of monopod */
  MWAIT;
  M(mop.theta1,  mop.theta1f,  mop.theta1zo);  /*  1 */
  M(mop.deltaf,  mop.deltaff,  mop.deltafzo);  /*  2 */
  U(MADC) = 5;         /* Begin at channel 4 of monopod */
  MWAIT;
  M(mop.theta2,  mop.theta2f,  mop.theta2zo);  /*  5 */
  M(mop.p2a,     mop.p2af,     mop.p2azo);     /*  6 */
  M(mop.p2b,     mop.p2bf,     mop.p2bzo);     /*  7 */
```

```
/*   U(MADC) = 0; */
/*   MWAIT; */
/*   M(mop.ch0, mop.ch0f, mop.ch0fzo); */
/*   M(mop.ch1, mop.ch1f, mop.ch1fzo); */
/*   M(mop.ch2, mop.ch2f, mop.ch2fzo); */
/*   M(mop.ch3, mop.ch3f, mop.ch3fzo); */
/*   M(mop.ch4, mop.ch4f, mop.ch4fzo); */
/*   M(mop.ch5, mop.ch5f, mop.ch5fzo); */
/*   M(mop.ch6, mop.ch6f, mop.ch6fzo); */
/*   M(mop.ch7, mop.ch7f, mop.ch7fzo); */
/*   M(mop.ch8, mop.ch8f, mop.ch8fzo); */
/*   M(mop.ch9, mop.ch9f, mop.ch9fzo); */
/*   M(mop.ch10, mop.ch10f, mop.ch10fzo); */
/*   M(mop.ch11, mop.ch11f, mop.ch11fzo); */
/*   M(mop.ch12, mop.ch12f, mop.ch12fzo); */
/*   M(mop.ch13, mop.ch13f, mop.ch13fzo); */
/*   M(mop.ch14, mop.ch14f, mop.ch14fzo); */
/*   M(mop.ch15, mop.ch15f, mop.ch15fzo); */
```

```
    mop.w1 = 4.0 - 1.5 * sin12;
    mop.dw1 = (mop.w1 - 4.) * mop.dw1f + mop.dw1zo;
    mop.w2 = 4.295 * sqrt(1.122 - .698 * mcos(mop.theta2 + 1.2141));
    mop.dw2 = (mop.w2 - 4.025) * mop.dw2f + mop.dw2zo;
    mop.alpha1 = (mcos(mop.theta1) - mcos(mop.theta2)) / (2.667 - sin12);
    mop.beta2 = macos((mop.w2 * mop.w2 - 16.197)/ (3.0 * mop.w2));
    mop.r1 = 1.5 * mcos(mop.alpha1 + mop.theta2);
    mop.r2 = 1.5 * msin(mop.beta2);
    mop.phi = (1.0 - mop.jleg) * mop.pitch + mop.jleg * mop.thetaleg;
  }
}
/*****************************************************************/
```

```
                    /*
                     * Rings:
                     */
                    /*
                     * Do not divide by zero:
                     */
                      if (mop.dt == 0) mop.dt = 1;
                      if (mop.ringlen_encoder == 0) mop.ringlen_encoder = 1;
                      if (mop.ringlen_zd == 0) mop.ringlen_zd = 1;
                        if (mop.ringlen_theta == 0) mop.ringlen_theta = 1;
                    /*  if (mop.ringleg_treadd == 0) mop.ringleg_treadd = 1; */

                      mop.recip_dt_encoder = 1000.0/(mop.dt * mop.ringlen_encoder);
                      mop.recip_dt_zd = 1000.0/(mop.dt * mop.ringlen_zd);
                      mop.recip_dt_theta = 1000.0/(mop.dt * mop.ringlen_theta);
                    /*  mop.recip_dt_treadd = 1000.0/(mop.dt * mop.ringleg_treadd); */

                      mop.deltafd = (mop.deltaf-deltaf_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      deltaf_ring[(int)mop.ringn_p_theta] = mop.deltaf;

                      mop.pitchd = (mop.pitch-pitch_ring[(int)mop.ringn_p_encoder])
                        * mop.recip_dt_encoder;
                      pitch_ring[(int)mop.ringn_p_encoder] = mop.pitch;

                      mop.phid = (mop.phi-phi_ring[(int)mop.ringn_p_encoder])
                        * mop.recip_dt_encoder;
                      phi_ring[(int)mop.ringn_p_encoder] = mop.phi;

                      mop.xd = (mop.x - x_ring[(int)mop.ringn_p_encoder]) *
mop.recip_dt_encoder;
                      x_ring[(int)mop.ringn_p_encoder] = mop.x;

                      mop.zd = (mop.z - z_ring[(int)mop.ringn_p_encoder]) *
mop.recip_dt_encoder;
                      z_ring[(int)mop.ringn_p_encoder] = mop.z;

                      mop.zdd = (mop.zd - zd_ring[(int)mop.ringn_p_zd]) * mop.recip_dt_zd;
                      zd_ring[(int)mop.ringn_p_zd] = mop.zd;

                      mop.theta1d = (mop.theta1-theta1_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      mop.theta2d = (mop.theta2-theta2_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      mop.thetaheeld = (mop.thetaheel-thetaheel_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      mop.thetatoed = (mop.thetatoe-thetatoe_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      mop.legzd = (mop.legz-legzd_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                      mop.legxd = (mop.legx-legxd_ring[(int)mop.ringn_p_theta])
                        * mop.recip_dt_theta;
                    /*  mop.treaddd = (mop.treadd-treadd_ring[(int)mop.ringn_p_treadd])
                        * mop.recip_dt_treadd; */
                      theta1_ring[(int)mop.ringn_p_theta] = mop.theta1;
                      theta2_ring[(int)mop.ringn_p_theta] = mop.theta2;
                      thetaheel_ring[(int)mop.ringn_p_theta] = mop.thetaheel;
```

```
    thetatoe_ring[(int)mop.ringn_p_theta] = mop.thetatoe;
    legzd_ring[(int)mop.ringn_p_theta] = mop.legz;
    legxd_ring[(int)mop.ringn_p_theta] = mop.legx;
/*    treadd_ring[(int)mop.ringn_p_treadd] = mop.treadd;  */

/*
 * End of rings.  Update ring pointer.
 */
    mop.ringn_p_theta++;
    mop.ringn_p_encoder++;
/*    mop.ringn_p_treadd++;  */
/*
 * Reset if we run off the end:
 */
    if ((int)mop.ringn_p_theta >= (int)mop.ringlen_theta)
       mop.ringn_p_theta = 0;
    if ((int)mop.ringn_p_encoder >= (int)mop.ringlen_encoder)
       mop.ringn_p_encoder = 0;
    if ((int)mop.ringn_p_zd >= (int)mop.ringlen_zd)
       mop.ringn_p_zd = 0;
/*    if ((int)mop.ringn_p_treadd >= (int)mop.ringlen_treadd)
       mop.ringn_p_treadd = 0;  */

   mop.foot_f = mop.foot_k * mop.deltaf; /* vertical force in lbf */
   mop.zforce = mop.foot_f * cos(mop.thetaheel);
   mop.force_a = mop.p2a * mop.area_a;
   mop.force_b = mop.p2b * mop.area_b;
   mop.delf = mop.force_a - mop.force_b;
   mop.toef = mop.delf * 1.5 * cos(mop.theta2) / 24.0;

   mop.downf = mop.zforce - (mop.toef * sin(mop.thetaleg));

   mop.sidef = (mop.toef*cos(mop.thetaleg))+
          (mop.foot_f*sin(mop.thetaheel));

   mop.sidef_abs = sqrt(mop.sidef * mop.sidef);

   mop.xd = mop.xd + mop.treadd;



/*****************************************************************/
/* Automatic assignment of variables                             */
/*****************************************************************/
/*
 * Set the source and destination address pointers:
 */
     srcptr = (float **) src_addresses;
     dstptr = (float **) dst_addresses;
/*
 * Do the assignments:
 */
     while (*srcptr) **dstptr++ = **srcptr++;
```

```
/*****************************************************************/
/* Demo mode:                                                    */
/*****************************************************************/
   if (Lap_Box_Switches & 0x100) {
/*
 * - .5 radians < thetaheel_sp < 1.5 radians:
 */
      mop.thetaheel_sp = (2.0 * mop.slide1) - .5;
/*
 * - .785 radians < theta2_sp < .785 radians
 */
      mop.theta2_sp = (mop.slide2 - 0.5) * 1.57;

      mop.t1 = mop.sw0/10 * (mop.thetaheel - mop.thetaheel_sp)
        + mop.sw1/100 * mop.thetaheeld;
      mop.t2 = mop.sw4/10 * (mop.theta2 - mop.theta2_sp)
        + mop.sw5/100 * mop.theta2d;
  }
   else { /* Go into simple hopping mode */
      if (Lap_Box_Switches & 0x10) { /* Check SET button */
        mop.inx = 300;
        mop.ts = .150;
    }

/*****************************************************************/
/* Gait 2 - this is the gait with timed thrust:                  */
/*****************************************************************/
     if ((int)mop.gait == 2) {
        switch ((int)mop.inx) {

     case 50 is   /* Pre-stance */
         mop.thetaheel_sp = mop.thetaheel_min;

         mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw1 * mop.thetaheeld;
         mop.t2 = - mop.theta2d_nom * mop.ksweep;

         if (mop.deltaf > mop.deltaf_th1){
           mop.inx = 150;
           ts_save = 0.0;   /* Start our time of stance counter */
           mop.zerr = mop.zerr + mop.zerr_wt * (mop.legz_cal - mop.z);
     }
    esac

     case 150 is   /* Stance */

        mop.t_start = mop.slide2 * 0.1;
        mop.t_thrust = mop.slide1 * 0.2;

        if ((ts_save > mop.t_start)
           && (ts_save < mop.t_start + mop.t_thrust)) {
           mop.t1 = - mop.slide3 * 2048.0;
     }
        else mop.t1 = 0.0;

        mop.t2 = -mop.theta2d_nom * mop.ksweep
          - mop.sw12 * mop.phi
```

```
            - mop.sw13 * mop.phid
            - mop.sw14 * mop.pitchdd
            + mop.sw15 * (mop.p2a - mop.arat * mop.p2b);

          ts_save += mop.dt * 0.001;   /* on ground now */
          if (mop.deltaf < mop.deltaf_th2) {  /* Liftoff */
            mop.inx = 250;
            mop.ts = ts_save;
            mop.z_lo = mop.z;
        }
      esac

      case 250 is
/*
 * Twilight Zone, shorten the foot, and sweep leg at nominal velocity
 */
          mop.thetaheel_sp = mop.thetaheel_min;
          mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
            + mop.sw1 * mop.thetaheeld;
          mop.t2 = -mop.theta2d_nom * mop.ksweep;
          if ((mop.z - mop.z_lo) > mop.thresh_deltaz)
            mop.inx = 300;
      esac

      case 300 is   /* Flight */

            mop.sin_cg = ((mop.xd * mop.ts) / (mop.cg_factor * L));
            if (mop.sin_cg > 1.0) mop.sin_cg = 1.0;
            if (mop.sin_cg < -1.0) mop.sin_cg = -1.0;

          mop.thetaheel_sp = mop.thetaheel_min;
          mop.theta2_sp = -mop.pitch -
            (masin(mop.sin_cg)
            + mop.kxd * (mop.xd - mop.xd_d));

          mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
            + mop.sw1 * mop.thetaheeld;
          mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
            + mop.sw5 * mop.theta2d;

          if ((mop.zd < mop.thresh_zd)
            && ((mop.z - mop.legz_cal) < -mop.zd * mop.t50)) {
            mop.inx = 50;
            mop.xd_td = mop.xd;
        }
      esac

      default is  /* Nothing happening, so Assume flight */
        mop.inx = 300;
        ts_save = 0.0;
      esac
        } /* end of SWITCH */
      } /* end of gait 2 */

/*****************************************************************/
/* Gait 1 - this is the new spiffy gait:                         */
/*****************************************************************/
      if ((int)mop.gait == 1) {
```

```
        switch ((int)mop.inx) {

      case 50 is   /* Pre-stance */
         mop.thetaheel_sp = mop.thetaheel_min;

         mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw1 * mop.thetaheeld;
         mop.t2 = - mop.theta2d_nom * mop.ksweep;

         if (mop.deltaf > mop.deltaf_th1){
           mop.inx = 100;
            ts_save = 0.0;   /* Start our time of stance counter */
            mop.zerr = mop.zerr + mop.zerr_wt * (mop.legz_cal - mop.z);
       }
     esac

      case 100 is   /* Compression */
         mop.thetaheel_sp = mop.thetaheel_min;

         mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw1 * mop.thetaheeld;
         mop.t2 = -mop.theta2d_nom * mop.ksweep
           - mop.sw12 * mop.phi
           - mop.sw13 * mop.phid
           - mop.sw14 * mop.pitchdd
           + mop.sw15 * (mop.p2a - mop.arat * mop.p2b);

         ts_save += mop.dt * 0.001;   /* on ground now */
         if (mop.deltafd < mop.deltafd_th1) {
           mop.inx = 200;   /* If we are beginning to de-compress */
       }
     esac

      case 200 is   /* Thrust */
         mop.thetaheel_sp = mop.thetaheel_max;  /* Fully extend thetaheel */

         mop.t1 = mop.sw8 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw9 * mop.thetaheeld;
         mop.t2 = -mop.theta2d_nom * mop.ksweep
           - mop.sw12 * mop.phi
           - mop.sw13 * mop.phid
           - mop.sw14 * mop.pitchdd
           + mop.sw15 * (mop.p2a - mop.arat * mop.p2b);
         ts_save += mop.dt * 0.001;   /* on ground now */
         if (mop.deltaf < mop.deltaf_th2) {  /* Liftoff */
           mop.inx = 250;
           mop.ts = ts_save;
           mop.z_lo = mop.z;
       }
     esac

     case 250 is
/*
 * Twilight Zone, shorten the foot, and sweep leg at nominal velocity
 */
         mop.thetaheel_sp = mop.thetaheel_min;
         mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw1 * mop.thetaheeld;
```

```
        mop.t2 = -mop.theta2d_nom * mop.ksweep;
        if ((mop.z - mop.z_lo) > mop.thresh_deltaz)
          mop.inx = 300;
      esac

      case 300 is   /* Flight */

        mop.sin_cg = ((mop.xd * mop.ts) / (mop.cg_factor * L));
        if (mop.sin_cg > 1.0) mop.sin_cg = 1.0;
        if (mop.sin_cg < -1.0) mop.sin_cg = -1.0;

        mop.thetaheel_sp = mop.thetaheel_min;
        mop.theta2_sp = -mop.pitch -
          (masin(mop.sin_cg)
          + mop.kxd * (mop.xd - mop.xd_d));

        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
          + mop.sw5 * mop.theta2d;

        if ((mop.zd < mop.thresh_zd)
          && ((mop.z - mop.legz_cal) < -mop.zd * mop.t50)) {
          mop.inx = 50;
          mop.xd_td = mop.xd;
      }
    esac

      default is  /* Nothing happening, so Assume flight */
        mop.inx = 300;
        ts_save = 0.0;
    esac
      } /* end of SWITCH */
    } /* end of gait 1 */
^L
/*************************************************************************/
/* Gait 4 - hoof-roll prevention w/ variable heel angle  gait:     */
/*************************************************************************/
    if ((int)mop.gait == 4) {
      switch ((int)mop.inx ) {

    case 100 is /* Compression */


      mop.thetaheel_sp = mop.thetaheel_min;

/*          if (mop.roll_counter != 0) {
            mop.pitch_des = mop.pitch_des_o -
                (mop.roll_th1/mop.roll_counter) * mop.pitch_peak;
            mop.roll_counter = mop.roll_counter + 1;
            if (mop.roll_counter == mop.roll_th1)
                mop.roll_counter = 0;
          }
          else if ((mop.downf > (mop.sidef_abs + mop.sidef_th1)) ||
                (mop.downf <= mop.sidef_th1)) {
            mop.pitch_des = mop.pitch_des_o;
          }
```

```
                    else {
                      mop.sidef_sign = mop.sidef / mop.sidef_abs;
                      mop.pitch_peak = (mop.kt2 * mop.sidef_sign *
                                        (mop.downf - mop.sidef_abs -
mop.sidef_th1));
                      mop.pitch_des = mop.pitch_des_o - mop.pitch_peak;
                      mop.roll_counter = 1;
                    } */

                    mop.ktp = (mop.sw12 /80.0)*mop.xd + (mop.sw12/2.0);
                    mop.ktv = (mop.sw13 /80.0)*mop.xd + (mop.sw13/2.0);

                  mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
                    + mop.sw1 * mop.thetaheeld;
                  mop.t2 =  -mop.theta2d_nom * mop.ksweep
                    - mop.ktp * (mop.pitch - mop.pitch_des)
                    - mop.ktv * mop.pitchd;


                  ts_save += mop.dt * 0.001;   /* on ground now */

                  if (mop.deltafd < mop.deltafd_th1) {
                    mop.inx = 200;   /* If we are beginning to de-compress */
                  }
                esac

                case 200 is   /* Thrust */
                  mop.thetaheel_max = (2.0 * mop.slide1) - .5;

                    mop.thetaheel_sp = mop.thetaheel_min + mop.thetaheel_max;

                    mop.t1 = mop.sw8 * (mop.thetaheel - mop.thetaheel_sp)
                    + mop.sw9 * mop.thetaheeld;

          /*          if (mop.roll_counter != 0) {
                      mop.pitch_des_o -
                          (mop.roll_th1/mop.roll_counter) * mop.pitch_peak;
                      mop.roll_counter = mop.roll_counter + 1;
                      if (mop.roll_counter == mop.roll_th1)
                          mop.roll_counter = 0;
                    }
                     else if ((mop.downf > (mop.sidef_abs + mop.sidef_th1)) ||
                          (mop.downf <= mop.sidef_th1)) {
                      mop.pitch_des = mop.pitch_des_o;
                    }

                    else {
                      mop.sidef_sign = mop.sidef / mop.sidef_abs;
                      mop.pitch_peak = (mop.kt2 * mop.sidef_sign *
                                        (mop.downf - mop.sidef_abs -
mop.sidef_th1));
                      mop.pitch_des = mop.pitch_des_o - mop.pitch_peak;
                      mop.roll_counter = 1;
                    } */


                    mop.ktp = (mop.sw12 /80.0)*mop.xd + (mop.sw12/2.0);
                    mop.ktv = (mop.sw13 /80.0)*mop.xd + (mop.sw13/2.0);
```

```
        mop.t2 = -mop.theta2d_nom * mop.ksweep
          - mop.ktp * (mop.pitch - mop.pitch_des)
          - mop.ktv * mop.pitchd;

            ts_save += mop.dt * 0.001;    /* on ground now */

        if (mop.deltaf < mop.deltaf_th2) {  /* Liftoff */
          mop.inx = 250;
          mop.ts = ts_save;
          mop.z_lo = mop.z;
                  mop.roll_counter = 0;
      }
    esac

    case 250 is
/*
 * Twilight Zone, shorten the foot, do not swing the leg
 */
        mop.thetaheel_sp = mop.thetaheel_min;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
       mop.t2 = 0;

            mop.xd_lo = mop.xd;
       if ((mop.z - mop.z_lo) > mop.thresh_deltaz)
            mop.inx = 300;
   esac

    case 300 is /* Flight */

          mop.sin_cg = ((mop.xd * mop.ts) / (mop.cg_factor * L));
          if (mop.sin_cg > 1.0) mop.sin_cg = 1.0;
          if (mop.sin_cg < -1.0) mop.sin_cg = -1.0;

          mop.cg_print = masin(mop.sin_cg);
          mop.accel_d = mop.kxd * (mop.xd - mop.xd_d);

       if (mop.xd_lo < 20.0) mop.accel_th1 = mop.accel_th1_o;
          else {
            mop.accel_th1 = (1/117.6) * ((40.0 - mop.xd_lo) - 0.64213)
                - mop.accel_th1_offset;
            if (mop.accel_th1 < 0.0) mop.accel_th1 = 0.0;
          }
           if (mop.accel_d > mop.accel_th1)
            mop.accel = mop.accel_th1;
           else if (mop.accel_d < -mop.accel_th1)
            mop.accel = -mop.accel_th1;
           else mop.accel = mop.accel_d;

           mop.theta2_sp = -mop.pitch -
         (mop.cg_print + mop.accel) + mop.cg_offset;

       mop.thetaheel_min = mop.kh*(- mop.accel) +
                    mop.thetaheel_sp_o;
          mop.thetaheel_sp = mop.thetaheel_min;
```

```
    mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
      + mop.sw1 * mop.thetaheeld;
        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
      + mop.sw5 * mop.theta2d;

        if (mop.zd < 0) mop.theta_cg = matan(mop.xd / mop.zd);

              if ((mop.deltaf > mop.deltaf_th1) && (mop.zd < - 15.0) &&
                 (mop.deltaf_previous > mop.deltaf_th1)) {
              mop.xd_td = mop.xd;
              mop.thetaleg_accel =
                mop.thetaleg + mop.cg_print - mop.cg_offset;
              mop.inx = 100;
      ts_save = 0.0;   /* Start our time of stance counter */
      mop.xd_td = mop.xd;
  }
esac

 default is          /* Nothing happening, so... */
   mop.inx = 300;          /* Assume flight */
   ts_save = 0.0;
esac
   } /* end of SWITCH */
} /* end of gait 4 */
```

```
/*************************************************************************/
/* Gait 5 - hoof-roll prevention w/ variable heel angle  gait:     */
/*************************************************************************/
      if ((int)mop.gait == 5) {
        switch ((int)mop.inx ) {

      case 100 is /* Compression */


          mop.thetaheel_sp = mop.thetaheel_min;

              mop.ktp = (mop.sw12 /80.0)*mop.xd + (mop.sw12/2.0);
              mop.ktv = (mop.sw13 /80.0)*mop.xd + (mop.sw13/2.0);

          mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
            + mop.sw1 * mop.thetaheeld;
          mop.t2 =  -mop.theta2d_nom * mop.ksweep
            - mop.sw12 * (mop.pitch - mop.pitch_des)
            - mop.sw13 * mop.pitchd;


          ts_save += mop.dt * 0.001;   /* on ground now */

          if (mop.deltafd < mop.deltafd_th1) {
            mop.inx = 200;   /* If we are beginning to de-compress */
       }
      esac

      case 200 is   /* Thrust */
          mop.thetaheel_max = (2.0 * mop.slide1) - .5;

              mop.thetaheel_sp = mop.thetaheel_min + mop.thetaheel_max;

              mop.t1 = mop.sw8 * (mop.thetaheel - mop.thetaheel_sp)
            + mop.sw9 * mop.thetaheeld;

              mop.ktp = (mop.sw12 /80.0)*mop.xd + (mop.sw12/2.0);
              mop.ktv = (mop.sw13 /80.0)*mop.xd + (mop.sw13/2.0);


          mop.t2 =  -mop.theta2d_nom * mop.ksweep
            - mop.sw12 * (mop.pitch - mop.pitch_des)
            - mop.sw13 * mop.pitchd;

              ts_save += mop.dt * 0.001;   /* on ground now */

          if (mop.deltaf < mop.deltaf_th2) {  /* Liftoff */
            mop.inx = 250;
            mop.ts = ts_save;
            mop.z_lo = mop.z;
                    mop.roll_counter = 0;
       }
      esac

      case 250 is
/*
```

```
* Twilight Zone, shorten the foot, do not swing the leg
*/
        mop.thetaheel_sp = mop.thetaheel_min;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
       mop.t2 = 0;

            mop.xd_lo = mop.xd;
        if ((mop.z - mop.z_lo) > mop.thresh_deltaz)
            mop.inx = 300;
    esac

    case 300 is /* Flight */

            mop.sin_cg = ((mop.xd * mop.ts) / (mop.cg_factor * L));
            if (mop.sin_cg > 1.0) mop.sin_cg = 1.0;
            if (mop.sin_cg < -1.0) mop.sin_cg = -1.0;

            mop.cg_print = masin(mop.sin_cg);
            mop.accel_d = mop.kxd * (mop.xd - mop.xd_d);



        if (mop.xd_lo < 20.0) mop.accel_th1 = mop.accel_th1_o;
            else {
              mop.accel_th1 = (1/117.6) * ((40.0 - mop.xd_lo) - 0.64213)
                   - mop.accel_th1_offset;
              if (mop.accel_th1 < 0.0) mop.accel_th1 = 0.0;
            }
            if (mop.accel_d > mop.accel_th1)
              mop.accel = mop.accel_th1;
            else if (mop.accel_d < -mop.accel_th1)
              mop.accel = -mop.accel_th1;
            else mop.accel = mop.accel_d;

            mop.theta2_sp = -mop.pitch -
           (mop.cg_print + mop.accel) + mop.cg_offset;
            mop.thetaheel_min = mop.kh*(- mop.accel) +
                       mop.thetaheel_sp_o;
            mop.thetaheel_sp = mop.thetaheel_min;

       mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
         + mop.sw1 * mop.thetaheeld;
           mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
         + mop.sw5 * mop.theta2d;

           if (mop.zd < 0) mop.theta_cg = matan(mop.xd / mop.zd);

                  if ((mop.deltaf > mop.deltaf_th1) && (mop.zd < - 15.0) &&
                     (mop.deltaf_previous > mop.deltaf_th1)) {
                  mop.xd_td = mop.xd;
                  mop.thetaleg_accel =
                    mop.thetaleg + mop.cg_print - mop.cg_offset;
                  mop.inx = 100;
          ts_save = 0.0;   /* Start our time of stance counter */
          mop.xd_td = mop.xd;
     }
    esac
```

```
 default is          /* Nothing happening, so... */
    mop.inx = 300;          /* Assume flight */
    ts_save = 0.0;
esac
    } /* end of SWITCH */
 } /* end of gait 4 */
```

```
/*************************************************************************/
/* Gait 6 - actuator response test                                       */
/*************************************************************************/
        if ((int)mop.gait == 6) {
           switch ((int)mop.inx) {

        case 100 is /* actuator #1 test; square wave */

          mop.wave = sin(accum_time / mop.factor);
          if (mop.wave > 0.0)
            mop.square = 1.0;
        else if (mop.wave < 0.0)
          mop.square = -1.0;
        else if (mop.wave = 0.0)
          mop.square = 0.0;

        mop.thetaheel_sp = (mop.square + 1.0)/mop.factor1 ;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
         + mop.sw1 * mop.thetaheeld;

        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
         + mop.sw5 * mop.theta2d;

        esac

        case 150 is /* actuator #1 test; sinusoidal wave */

          mop.wave = sin(accum_time / mop.factor);
        mop.thetaheel_sp = (mop.wave + 1.0)/mop.factor1;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;

        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
         + mop.sw5 * mop.theta2d;

        esac

        case 200 is /* actuator #2 test; square wave */

          mop.wave = sin(accum_time / mop.factor);
          if (mop.wave > 0.0)
            mop.square = 1.0;
        else if (mop.wave < 0.0)
          mop.square = -1.0;
        else if (mop.wave = 0.0)
          mop.square = 0.0;

        mop.thetaheel_sp = 0.0;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
         + mop.sw1 * mop.thetaheeld;
        mop.theta2_sp = mop.square/mop.factor1;
        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
         + mop.sw5 * mop.theta2d;

        esac
```

```
        case 250 is /* actuator #2 tese; sinusoidal wave */

          mop.wave = sin(accum_time / mop.factor);
/*        mop.thetaheel_sp = 0.0; */
          mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
           + mop.sw1 * mop.thetaheeld;
          mop.theta2_sp = mop.wave/mop.factor1 + mop.theta2_offset;
          mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
           + mop.sw5 * mop.theta2d;

          esac

          case 300 is /* sinusoidal signal to actuators */
           mop.wave = sin(accum_time / mop.factor);
           mop.t1 = mop.amplitude * mop.wave;
           mop.t2 = mop.amplitude * mop.wave;

          esac
          } /* end of SWITCH */
           }

/*************************************************************************/
/* Gait 0 - old default gait:                                            */
/*************************************************************************/
     if ((int)mop.gait == 0) {
       switch ((int)mop.inx ) {

     case 100 is /* Compression */
        mop.thetaheel_sp = mop.thetaheel_min;

        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
            mop.t2 = - mop.sw12 * (mop.pitch - mop.pitch_des)
          - mop.sw13 * mop.pitchd;
            ts_save += mop.dt * 0.001;   /* on ground now */
       if (mop.deltafd < mop.deltafd_th1) {
         mop.inx = 200;   /* If we are beginning to de-compress */
     }
    esac

     case 200 is   /* Thrust */
       mop.thetaheel_max = (2.0 * mop.slide1) - .5;

            mop.thetaheel_sp = mop.thetaheel_max;

            mop.t1 = mop.sw8 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw9 * mop.thetaheeld;

            mop.t2 =  -mop.theta2d_nom * mop.ksweep
             - mop.sw12 * (mop.pitch - mop.pitch_des)
                  - mop.sw13 * mop.pitchd;

           ts_save += mop.dt * 0.001;   /* on ground now */
       if (mop.deltaf < mop.deltaf_th2) {  /* Liftoff */
         mop.inx = 250;
         mop.ts = ts_save;
         mop.z_lo = mop.z;
     }
```

```
    esac

    case 250 is
/*
 * Twilight Zone, shorten the foot, do not swing the leg
 */
        mop.thetaheel_sp = mop.thetaheel_min;
        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
        mop.t2 = 0;
        if ((mop.z - mop.z_lo) > mop.thresh_deltaz)
          mop.inx = 300;
    esac

    case 300 is /* Flight */

          mop.sin_cg = ((mop.xd * mop.ts) / (mop.cg_factor * L));
          if (mop.sin_cg > 1.0) mop.sin_cg = 1.0;
          if (mop.sin_cg < -1.0) mop.sin_cg = -1.0;

        mop.thetaheel_sp = mop.thetaheel_min;
        mop.theta2_sp = -mop.pitch -
          (masin(mop.sin_cg)
           + mop.kxd * (mop.xd - mop.xd_d) ) + mop.cg_offset;

        mop.t1 = mop.sw0 * (mop.thetaheel - mop.thetaheel_sp)
          + mop.sw1 * mop.thetaheeld;
        mop.t2 = mop.sw4 * (mop.theta2 - mop.theta2_sp)
          + mop.sw5 * mop.theta2d;

                if ((mop.deltaf > mop.deltaf_th1) && (mop.zd < - 15.0) &&
                   (mop.deltaf_previous > mop.deltaf_th1)) {
          mop.inx = 100;
          ts_save = 0.0;   /* Start our time of stance counter */
          mop.xd_td = mop.xd;
      }
    esac

    default is          /* Nothing happening, so... */
      mop.inx = 300;          /* Assume flight */
      ts_save = 0.0;
    esac
      } /* end of SWITCH */
    } /* end of gait 0 */
  } /* ELSE of go into simple hopping mode */
```

```
mop.deltaf_previous = mop.deltaf;

/*************************************************************************/
/* Send the signals to the valves:                                       */
/*************************************************************************/
/*
 * Limit the signals (to try to stay away from the stops):
 */
   mop.t2 = mop.t2 + mop.t2zo;
   mop.t2high = mop.t2_max * (1.0 + (mop.dw2 / mop.dw2_max));
   mop.t2low = -mop.t2_max * (1.0 - (mop.dw2 / mop.dw2_max));
   if (mop.t2high > 2047.0) mop.t2high = 2047.0;
   if (mop.t2low < -2048.0) mop.t2low = -2048.0;
   if (mop.t2 > mop.t2high) mop.t2 = mop.t2high;
   else if (mop.t2 < mop.t2low) mop.t2 = mop.t2low;

   mop.t1 = mop.t1 + mop.t1zo;
   if (mop.t1 > 2047.0) mop.t1 = 2047.0;
   else if (mop.t1 < -2048.0) mop.t1 = -2048.0;
/*
 * Try not to hit the stops:
 */
   if ((mop.dw1 > mop.dw1_max) && (mop.t1 > 0.0)) mop.t1 = 0.0;
   if ((mop.theta1 < mop.theta1_min) && (mop.t1 > 0.0)) mop.t1 = 0.0;
   if ((mop.dw1 < -mop.dw1_max) && (mop.t1 < 0.0)) mop.t1 = 0.0;
   if ((mop.theta1 > mop.theta1_max) && (mop.t1 < 0.0)) mop.t1 = 0.0;
/*
   if ((mop.dw2 > mop.dw2_max) && (mop.t2 < 0.0)) mop.t2 = 0.0;
   if ((mop.dw2 < -mop.dw2_max) && (mop.t2 > 0.0)) mop.t2 = 0.0;
 */
/*
 * Output to the valves:
 */
   U(MDA2) = (short)mop.t1;
   U(MDA0) = (short)mop.t2;
```

```
/*************************************************************************/
/* Save some variables for recording or plotting:                        */
/*************************************************************************/
/*
 * Increment the timer and see if it is time to save data:
 */
   mop.rec_tim += mop.dt;
   if (rr.record_go && (mop.rec_tim >= mop.rec_dt)) {
/*
 * Set the slot list pointer and the record structure pointer:
 */
      rsptr = (float *) rs.r[rr.ring_p];
      vaptr = var_addresses;
/*
 * Save the variables:
 */
      while (*vaptr) *rsptr++ = **vaptr++;
/*
 * Reset the timer and increment the ring pointer:
 *   (rr.ring_p points to oldest data)
 */
      mop.rec_tim = 0;
      rr.ring_p++;
      if (rr.ring_p >= MAX_REC) rr.ring_p = 0;
/*
 * Check lapbox switches to see if we should stop recording:
 */
      if (Lap_Box_Switches & 0x08) rr.record_go = 0;
   }  /* end of recording code */
/*
 * Turn off driver-active light:
 */
   Lap_Box_LEDs &= ~(0x1);
/*
 * Write the lapbox LEDs:
 */
   U(LBDO) = (short)Lap_Box_LEDs;
   mop.t_out = (float) mfpr (ICR);
/*
 * End of the interrupt routine:
 */
   return;
} /* end of kwp_mint */
```