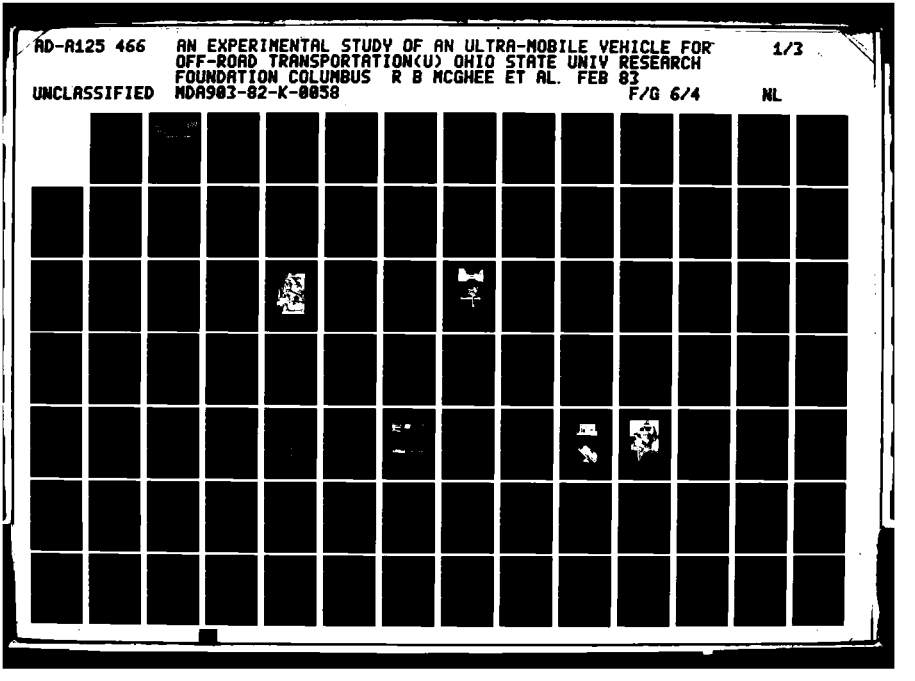
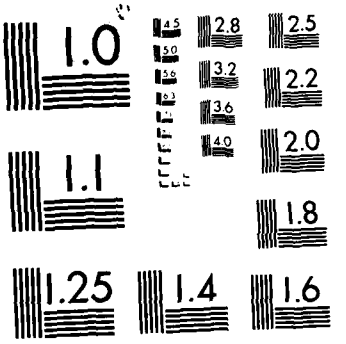
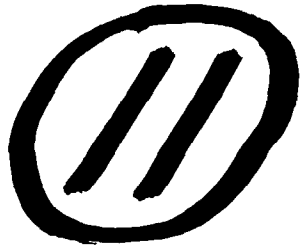
AD-A125 466 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. FEB 83 1/3

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL





RF Project 762945/714250
Semi-Annual Report



the
ohio
state
university

research foundation
1314 kinnear road
columbus, ohio
43212

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

Robert B. McGhee and Kenneth J. Waldron
Department of Mechanical Engineering

For the Period
July 1, 1982 - September 30, 1982

DEFENSE SUPPLY SERVICE
Washington, D.C. 20310

Contract No. MDA903-82-K-0058

February, 1983

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER / 2. GOVT ACCESSION NO. AD-A125 460	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION	5. TYPE OF REPORT & PERIOD COVERED Semi-Annual Technical July 1, 1982 - Sept. 30, 1982
	6. PERFORMING ORG. REPORT NUMBER 762945/714250
7. AUTHOR(s) Robert B. McGhee and Kenneth J. Waldron	8. CONTRACT OR GRANT NUMBER(s) Contract No. MDA903-82-K-0058
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Ohio State University Research Foundation, 1314 Kinnear Road Columbus, Ohio 43212	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Supply Service - Washington Room ID-245, The Pentagon Washington, D.C. 20310	12. REPORT DATE February, 1983 13. NUMBER OF PAGES 177
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Off-road vehicles
Suspension systems
Walking machines
Legged locomotion

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This research is concerned with computer control of terrain-adaptive suspension systems for off-road vehicles. A small-scale laboratory model walking machine has demonstrated an ability to overcome sizeable obstacles by varying limb cycles to conform to terrain. This action is achieved through the use of feedback from ground-reaction force transducers mounted at the end of each of its six legs operating in conjunction with body attitude sensing from a vertical gyro. A preliminary design of a larger vehicle suitable for outdoor

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Item 20. - continued:

testing has been completed. This vehicle will use an optical radar system to obtain terrain preview information in order to increase permissible speed of motion in rough-terrain locomotion. Power will be distributed hydraulically under computer control using an internal combustion engine as a prime mover. A human operator will ride in the vehicle cab, and will provide steering and speed commands via controls similar to those used in aircraft. This report includes a definition of six distinct operational modes for this vehicle, corresponding to differing terrain types and various mission objectives.

Semi-Annual Technical Report

for

DARPA Contract MDA903-82-K-0058

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

prepared by

Robert B. McGhee and Kenneth J. Waldron

covering the period

July 1, 1982, through September 30, 1982

College of Engineering
The Ohio State University
Columbus, Ohio 43210

INTRODUCTION

Research under this contract began as scheduled on October 1, 1981. During the first twelve months, no significant deviations from the plan contained in our proposal dated May 13, 1981, occurred. Research performed relative to each of the tasks listed in our Statement of Work, dated September 21, 1981, is summarized in the following paragraphs. Further details regarding some of these topics are provided in the attached appendices, in our Semi-Annual Report dated August, 1982, and in videotapes previously furnished to DARPA.

RESEARCH ACCOMPLISHED

Task 1: Vertical Sensor Evaluation

An aircraft-quality vertical gyro has been purchased and evaluated using the OSU Hexapod as a test bed. Test results are documented in the attached Appendix 3. This gyro has been judged to be fully satisfactory and is a candidate for the ASV-84 vertical sensor.

Task 2: Control Mode Studies

As a result of research coordination meetings held during the second half of this contract year, a total of six control modes have been defined for the ASV-84 vehicle. These modes were formalized in an internal memorandum entitled "Operational Modes and Computer Architecture for ASV-84 Vehicle," dated July, 1982. Copies of this memorandum were distributed to DARPA and all ASV-84 contractors during a meeting at the University of Wisconsin in August. Appendix 2 provides a summary of the projected characteristics of each control mode. It was agreed during the August meeting that highest priority for software development would be attached to the terrain-following mode.

Task 3: Insect Control Studies

Subcontracted work at the University of Alberta to analyze climbing and ditch-crossing strategies used by locusts has been completed. A copy of the subcontractor's report and a videotape illustrating major findings of this research have been provided to DARPA. One of the most important outcomes of this work was the discovery that locusts abandon wave gaits for rough-terrain locomotion and use instead a class of gaits more suited to traversing large obstacles. These gaits will be investigated for application to the OSU Hexapod and the ASV-84 during the coming contract year.

Task 4: Fault Tolerant Software

A first version of the safety software for the ASV-84 vehicle has been tested on the OSU Hexapod. This software correctly detects imminent collisions of joints with their limits, of limbs with other limbs, and of the body with the ground (resulting from static instability). When such conditions occur, the Hexapod is stopped automatically and an error message is provided to the operator. Details of this work are provided in an M.S. thesis previously furnished to DARPA, dated August, 1982, and entitled "Safety Checking System with Voice Response for the OSU Hexapod."

Task 5: Laser Foothold Designator

Hardware and software for triangulation by the OSU Hexapod on optically-designated footholds has been completed. The accuracy of this system exceeds that of the leg servos. Closed loop testing using follow-the-leader gaits with a hand-held Helium-Neon laser foothold designator is scheduled for the coming quarter of this research program.

Task 6: Leg Geometry Design

There have been a number of important developments in this area during this contract year. The DUWE vehicle (Dynamic Unpowered Walking Experiment) was successfully operated on a tilted plane with a slope of 5 percent, corresponding to a specific resistance of only .05. This demonstrates that there is no intrinsic mechanical obstacle to operation of suitably designed walking machines at power levels competitive with automotive vehicles. Agreement with computer simulation results was also satisfactory. A complete discussion of this experiment is given in the thesis "Dynamic Study of a Four-Bar Linkage Walking Machine Leg" by T. Frank Brown, Jr., a copy of which has been supplied to DARPA. A video-tape of the DUWE in operation has also been provided.

Dr. Shigeo Hirose was appointed as Visiting Associate Professor in the Department of Mechanical Engineering from April 1st to May 31st and was supported by this project. He attended the contractors' meeting in May and assisted with all aspects of the project but, particularly, leg geometry design. Dr. Hirose was unable to accept the budgeted subcontract because of differences in the organization and philosophy of research sponsorship in Japan. In lieu of that, it was agreed that he should be invited to make a similar extended visit next year.

Strength and deflection studies of the candidate leg designs have progressed substantially and a detailed structural design of a breadboard leg has been completed. It was found that the leg configuration with a four-bar mechanism and sliding shank could not be satisfactory embodied in hardware. This configuration has, therefore, been abandoned. The first configuration to be tested in the breadboard leg rig will be a panto-graph. All structural members for this leg have been completed. It is

anticipated that final assembly and initial testing will take place early in the next quarter of this project.

A fully serviced laboratory space has been commissioned for ASV hydraulic system testing. It is a ground-floor level bay with 440 square feet of area and a large roller door to the outside of the building. It is equipped with an overhead hoist. This space will be used initially for the breadboard leg tests and subsequently for the assembly of ASV subsystems.

Task 7: Power Package Specifications

Discussions continued with the University of Wisconsin on the specifications for this unit. Motor, flywheel, and alternator capacities have now been settled. Final pump specifications await completion of operational mode definitions for the hydraulic circuits. The overall configuration of an interim system for use in preliminary tests has been decided on. A lightweight, high performance motorcycle engine will be used.

Task 8: Power Transmission and Actuator Servo System Design

Simulation studies of proposed circuit configurations are continuing. Modifications of the drive and lift circuit configurations to deal with problems of operation in the close maneuvering mode and of supplying leg return energy are being studied.

The major hydraulic components of the breadboard leg system have been delivered and tested. They have been found satisfactory and will be mounted on the breadboard leg for functional testing during the coming quarter.

The design of the lift circuit for the breadboard leg is being done by Battelle in coordination with OSU. A regenerative circuit arrangement to give rapid leg extension if footing is lost is being studied.

While on a visit to Australia for other purposes in July, Dr. Waldron also visited the University of Western Australia at which a hydraulically actuated robot sheep-shearing system is being designed and tested. Observations of that system were also relevant to design of the ASV system. Dr. Waldron also visited Ifield Corporation, Dural, New South Wales, Australia, a manufacturer of very high efficiency servo-hydraulic pumps. Verified performance figures are superior to those of U.S. suppliers presently under consideration. Delivery time and costs would probably preclude use of Ifield pumps for the 1984 ASV, but the possibility is being further explored.

An Intellec Model 286 Series III Microcomputer Development System has been donated to the project by Intel for use in the Mechanical Engineering Department for actuator servo design and development of the breadboard leg controller. The breadboard leg computer has been delivered and control programs are currently under development.

Task 9: Overall Structural Design

Field tests to study mobility in forest and steep terrain conditions have been caried out using a light weight mock-up frame. The testing was performed on private property near Stockdale in Southern Ohio. A videotape showing the results of these tests has been supplied to DARPA.

Finite element studies of strength and vibrational modes of the vehicle frame were concluded using selected structural cross-sections. Earlier work of this type is shown on the videotape accompanying the final

report on contract number MDA903-81-C-0138. Detailed manufacturing drawings of the frame have been completed. A copy of these drawings has been forwarded to the University of Wisconsin to permit construction of a cockpit mockup for control studies. In addition, a decision was made to construct a full-scale wooden mockup of the entire frame at OSU to support studies of internal system packaging, wiring layout, etc. Construction of this frame will be completed early in the coming quarter.

Task 10: On-Board Computer Design

The Intel 8086 with the 8087 co-processor has been selected as the most suitable microprocessor for the on-board computer. These two chips are available along with appropriate analog-to-digital and digital-to-analog converters on the Intel 8630 single-board computer. A first-cut multiprocessor computer architectural design has been completed and documented in the internal memorandum referenced under Task 2. An Intellec Series III Microprocessor Development System suitable for programming this computer has been delivered. Construction of a breadboard version of this computer is scheduled to begin during the coming quarter.

Task 11: Software Design

Software for terrain-following locomotion by the OSU Hexapod has been completed and tested. Details are presented in the attached Appendix 3. The greater part of this software is equally suited to the ASV-84 vehicle. Conversion to ASV-84 control and transfer of the software to the breadboard multiprocessor computer will begin next quarter.

Task 12: Electronic Subsystem Design

Load cells for the ASV-84 foot-force sensors have been designed. Transducers and amplifiers have been selected. Component orders have been

placed for one cell to be tested on the ASV-84 breadboard leg. A servo-valve controller for the breadboard leg has been completed and tested. In addition to this work, the construction and bench testing of an ultrasonic ranging system has been completed. This system will be replicated and tested as a foot proximity sensor on the OSU Hexapod during the next contract year. Optical alternatives to ultrasonic proximity sensing will also be studied with the assistance of CEN, Saclay, France, under an existing subcontract.

Appendix 1

List of Publications

10/1/82

RESEARCH PUBLICATIONS SUPPORTED BY CONTRACT MDA903-82-K-0058

College of Engineering
The Ohio State University, Columbus, Ohio 43210

The following theses, dissertations, and papers have been produced with the support of DARPA Contract MDA903-82-K-0058 since its inception on October 1, 1981. Copies of most of these documents are available upon request.

1. Waldron, K.J., Frank, A.A., and Srinivasan, K., "The Use of Mechanical Energy Storage in an Unconventional, Rough-Terrain Vehicle," 17th Intersociety Energy Conversion Engineering Conference, Los Angeles, California, August 8-13, 1982.
2. Brown, F.T., Dynamic Study of a Four-Bar Linkage Walking Machine Leg, M.S. thesis, The Ohio State University, August, 1982.
3. Ju, J.T., Safety Checking System with Voice Response for the OSU Hexapod, M.S. thesis, The Ohio State University, August, 1982.
4. Pugh, D.R., An Autopilot for a Terrain-Adaptive Hexapod Vehicle, M.S. thesis, The Ohio State University, August, 1982.
5. Srinivasan, K., Waldron, K.J., and Dworak, J.A., "The Design and Evaluation of a Hydraulic Actuation System for a Legged Rough-Terrain Vehicle," ASME Winter Annual Meeting, Phoenix, Arizona, November 14-16, 1982.
6. McGhee, R.B., "Vehicular Legged Locomotion," to appear in Advances in Automation and Robotics, ed. by G.N. Saridis, Jai Press, Inc., 1983.

Appendix 2

ASV-84 Operational Mode Definitions

	CRUISE MODE	DASH MODE
SPEED:	5 MPH	8 MPH
FUEL ECONOMY:	HIGHEST	LOW
MANEUVERABILITY:	CRAB-ANGLE LIMITED TO ±10 DEGREES; MINIMUM TURNING RADIUS IS 3 BODY LENGTHS; SCANNER DETERMINES AVERAGE TERRAIN BODY SLOPE	TURNING & CRAB LIMITED BY OVERSTRIDING
USE OF SCANNER/ GUIDANCE SYSTEM:	USED IN CONTROL OF BODY ATTITUDE & FOOT-LIFT HEIGHT	USED IN CONTROL OF BODY ATTITUDE
GAIT:	TRIPOD	OVERSTRIDE TRIPOD
DISPLAYS USED:	SYSTEM STATUS	SYSTEM STATUS
OPERATOR CONTROLS:	JOYSTICK VELOCITY CONTROL & TURNING RADIUS	JOYSTICK VELOCITY CONTROL & TURNING RADIUS
USE OF FORCE SENSORS:	NONE	NONE
USE OF PROXIMITY SENSORS:	IMPACT CONTROL	IMPACT CONTROL

	UTILITY MODE	PRECISION FOOTING MODE
SPEED:	0	VERY SLOW
FUEL ECONOMY:	N/A	LOW
MANEUVERABILITY:	LEG TEST	HIGHEST--FULL RANGE OF MOTION FOR EACH LEG
USE OF SCANNER/ GUIDANCE SYSTEM:	CHECKOUT & CALIBRATION	NONE
GAIT:	N/A	MANUAL
DISPLAYS USED:	ALL	SUPPORT POLYGON STABILITY MARGIN LIFTABLE FEET
OPERATOR CONTROLS:	ALL	FOOT SELECTION SWITCHES JOYSTICK VELOCITY CONTROL ATTITUDE TRIM BUTTON ALTITUDE LEVER
USE OF FORCE SENSORS:	CHECKOUT & CALIBRATION	TO DETERMINE STABILITY MARGIN & CONTROL FOOT LOADING
USE OF PROXIMITY SENSORS:	CHECKOUT & CALIBRATION	FOOT ALTITUDE CONTROL

	CLOSE MANEUVERING MODE	TERRAIN FOLLOWING MODE
SPEED:	1 MPH	2-3 MPH
FUEL ECONOMY:	LOW	MODERATE
MANEUVERABILITY:	6 DOF BODY MOTION	FORWARD DIRECTED-- FOOTHOLD MUST BE VISIBLE TO SCANNER
USE OF SCANNER/ GUIDANCE SYSTEM:	NONE	USED IN CONTROL OF BODY ATTITUDE & LEG SEQUENCING & FOOT LIFT
GAIT:	FREE	FREE OR PARAMETRIC
DISPLAYS USED:	SYSTEM STATUS, TURNING CENTER	TERRAIN MAP STABILITY MARGIN
OPERATOR CONTROLS:	JOYSTICK VELOCITY CONTROL & TURNING RADIUS, ATTITUDE TRIM BUTTON, ALTITUDE LEVER	JOYSTICK VELOCITY CONTROL & TURNING RADIUS
USE OF FORCE SENSORS:	TO DETERMINE STABILITY MARGIN & CONTROL FOOT LOADING	TO DETERMINE STABILITY MARGIN & CONTROL FOOT LOADING
USE OF PROXIMITY SENSORS:	FOOT ALTITUDE CONTROL	FOOT ALTITUDE CONTROL

Appendix 3

An Autopilot for Terrain-Adaptive Hexapod Vehicle

AN AUTOPILOT FOR A TERRAIN-ADAPTIVE HEXAPOD VEHICLE

A Thesis

Presented in Partial Fulfillment of the Requirements
for the Degree Master of Science

by

Dennis Ray Pugh, B.S.E.E.

The Ohio State University
1982

Approved by

Advisor
Department of Electrical Engineering

To my parents

ACKNOWLEDGEMENTS

I would like to express my appreciation to the many people who have contributed to this work. Special thanks go to my advisor, Professor Karl W. Olson, for his many hours of consultation, and to Professor Charles A. Klein, whose assistance was invaluable throughout the test and documentation phases of the research. I am grateful to Professor David E. Orin for his valuable advice regarding software structure. Thanks are also due to Professor Robert B. McGhee for his support and guidance.

Others who have made important contributions include Mr. Carlo E. Barrientos, with whom I spent a great deal of time discussing control algorithms. Thanks are due to Mr. Vincent J. Vohnout for the design and fabrication of the mechanical components of the force sensors. I would also like to express my appreciation to Mr. Ronald W. Ventola for his work in fabricating circuits for this research, and to Ms. Joan A. Marn for her excellent work in the typing of this manuscript.

Finally, I thank my parents for their constant support and encouragement, which has made my academic career both fruitful and enjoyable.

This work was supported in part by the National Science Foundation under Grant ECS-7818957 and in part by the Defense Advanced Research Projects Agency under Contracts MDA903-81-C-0138 and MDA903-82-K-0058.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

Throughout history, man has continually developed better transportation systems. In the beginning he had to rely on his own two legs for transportation. As he became the master of his environment, he domesticated animals and invented the boat and wheel. Today, man can fly aircraft faster than sound, take a submarine under polar icecaps, and fly spacecraft to the moon. For personal transportation, the automobile has given him mobility never before possible. Transportation over very rough terran, however, is one area in which little progress has been made.

Various schemes for locomotion over highly irregular surfaces have been tried. Perhaps the most common off-road machines are multi-wheel drive vehicles and tracked vehicles. These are quite successful for moderately rough terrain. However, even the most versatile of these machines is vastly inferior to natural legged systems (animals) on very difficult surfaces [1].

The primary difficulty with designing highly terrain-adaptive vehicles lies in the complexity of the control task. While a vehicle such as an automobile possesses only two controlled degrees of freedom,

the simplest model of a natural quadraped requires twelve degrees of freedom. It has been shown that controlling twelve degrees of freedom is too difficult a task for a human [2]. Until recently, this represented an insurmountable obstacle to the development of highly adaptive vehicles. Now, however, the advent of miniature, low-cost digital computers has made it possible to relieve the operator of the coordination task.

In the early 70's, researchers at The Ohio State University began studying the problem of computer control of a multi-jointed vehicle [3]. To aid in this study, the OSU Hexapod Vehicle was constructed as a laboratory testbed. This legged vehicle possesses six legs with three degrees of freedom per leg, for a total of eighteen degrees of freedom. Prior to the work on this thesis, solutions to the fundamental control problems had been found, including leg gait specification, foot position specification to achieve the desired gait, and joint coordination to achieve the desired foot position. The Hexapod had the ability to walk forward, backward, sideways, turn, or execute a combination of these maneuvers simultaneously. However, it could walk only on level surfaces, due to the fact that it lacked the sensors to detect surface irregularities.

The problem of locomotion over irregular terrain had been studied in simulation, and one leg had been equipped with vector force sensors [4]. A method of obstacle accommodation had been developed for that only leg, enabling obstacles to be placed under the leg without impeding vehicle motion. However, the level of sophistication of

the control software and the amount of sensor hardware was still inadequate to allow locomotion over random terrain. The objective of the work presented in this thesis is to fully equip the OSU Hexapod Vehicle with the necessary sensors and to develop the necessary control software to achieve locomotion over random terrain with only directional commands being provided by the human operator.

1.2 Organization

A description of the existing systems of the OSU Hexapod Vehicle is given in Chapter 2, including the mechanical hardware, electronics, and control software. Chapter 3 details the improvements made to the vehicle system in the areas of electronics, force sensors, attitude sensors, and utility software. The control algorithms necessary for adaptive locomotion are developed in Chapter 4. Results of locomotion tests are given in Chapter 5. Chapter 6 summarizes the contributions made in this work and suggests areas where further work is needed.

Chapter 2

REVIEW OF EXISTING SYSTEMS OF THE OSU HEXAPOD VEHICLE

2.1 Introduction

The OSU Hexapod Vehicle is a complex system consisting of mechanical hardware, electronic hardware, and software control algorithms. In this chapter, all aspects of the vehicle which relate to autopilot design will be discussed. The systems are described as they existed prior to the modifications and additions which are detailed in subsequent chapters. Complete details of the vehicle design can be found in the referenced literature.

2.2 Mechanical Hardware

A photograph of the OSU Hexapod Vehicle is shown in Figure 2.1. The vehicle structure consists of six legs mounted on an aluminum frame. Each leg possesses three independently powered joints arranged in an arthropod configuration. Each joint actuator consists of an industrial grade series-wound electric drill motor and a gear reduction unit. Figure 2.2 is a diagram of the actuator system. The second stage of the gear reduction is a non-backdriveable worm gear. This insures that the joints lock into position when power is removed, but makes the relationship between motor shaft torque and applied foot

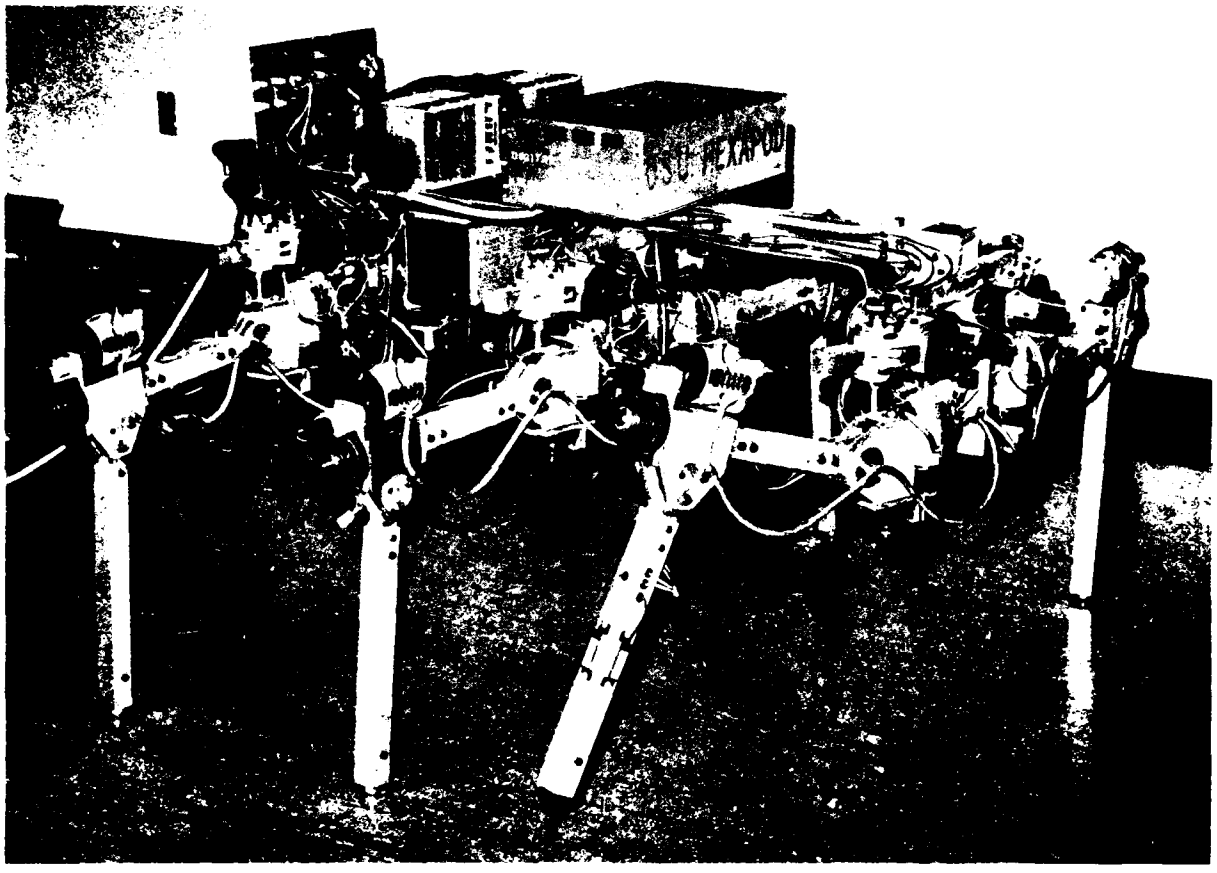


Figure 2.1. The OSU Hexapod Vehicle.

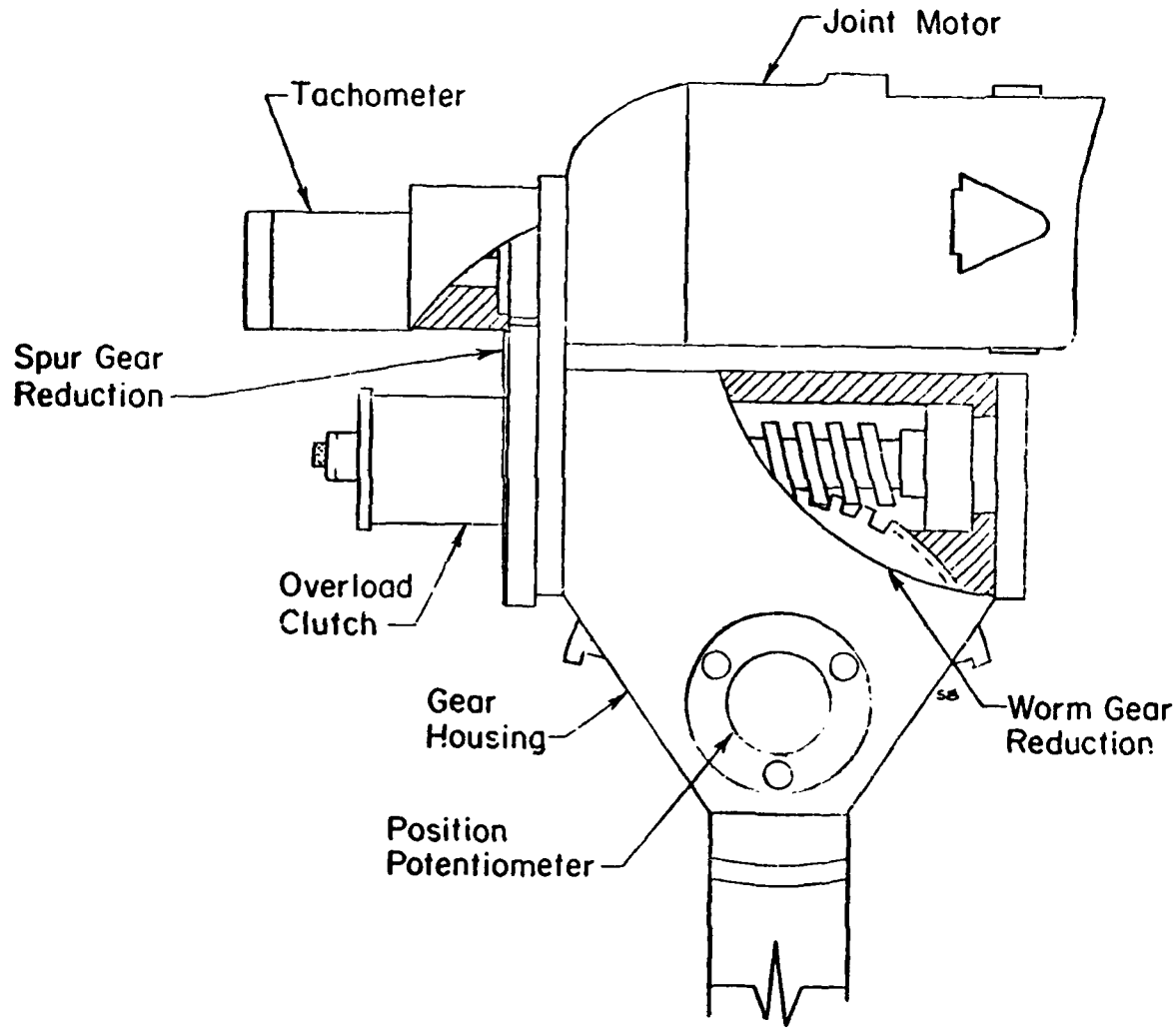


Figure 2.2 The Joint Actuator [5].

force highly nonlinear. Complete specifications on the joint actuator system can be found in [5].

2.3 Instrumentation

Each joint is instrumented with a potentiometer and a tachometer to measure position and rate, respectively. The tachometer is mounted directly to the motor output shaft, while the potentiometer is mounted at the output of the gear reduction unit. The tachometer location insures that the rate information is not affected by gear backlash. The position feedback loop is affected by gear backlash, however, so care must be taken to avoid limit cycling when designing the servo loop.

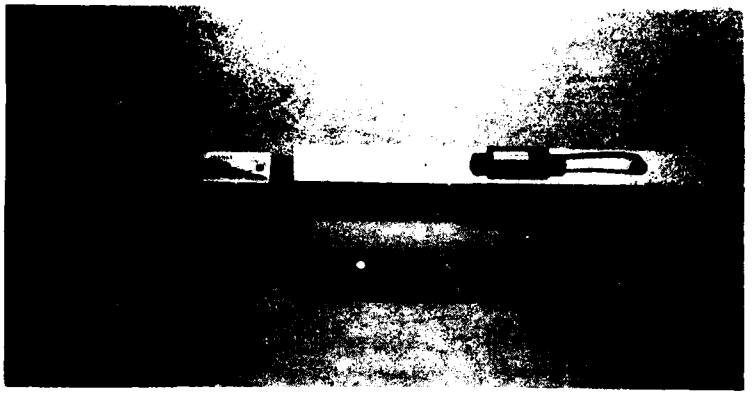
In 1977, one leg of the vehicle was equipped with vector force sensors. The lateral forces are measured by semiconductor strain gauges mounted to the sides of the lower leg segment. The axial force component is measured by a semiconductor load cell mounted in the foot. Amplification circuitry is located inside the leg segment. The force sensor design is illustrated in Figure 2.3, and is described in detail in [6].

2.4 Electronics

2.4.1 Motor Controller Circuitry

The motors are operated using half-wave AC phase control. The power control circuitry consists of a bridge rectifier and a triac for each motor. Each triac is controlled by an analog trigger generator circuit which compares the input voltage signal with a reference waveform and sends a turn-on pulse to the triac gate when the line voltage

(A)



(B)

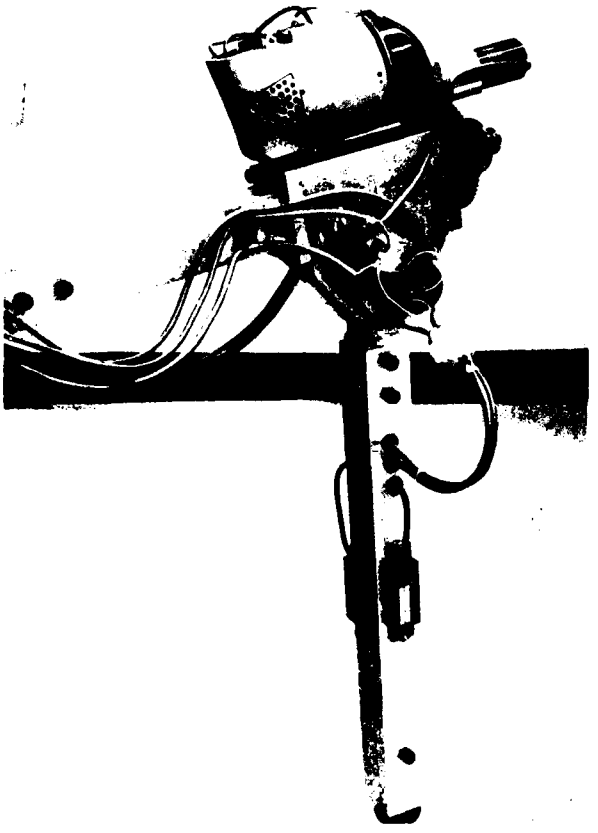


Figure 2.3 Existing Force Sensor Design [6].

is at the proper phase. The reference waveform is generated by a separate circuit, and removes the nonlinearity associated with AC phase control. Complete schematics for the motor controller circuitry can be found in [5]. The power circuitry is located on panels mounted between leg pairs; the trigger and reference circuitry is located in the card cage at the rear of the vehicle.

2.4.2 Instrumentation Electronics

The sensors installed on the Hexapod all require signal conditioning before undergoing analog-to-digital conversion. The potentiometer outputs are scaled by operational amplifiers to give a ten volt signal at full mechanical deflection. The tachometer outputs are scaled and low-pass filtered to eliminate the commutator noise. The potentiometer and tachometer signal conditioning circuitry is detailed in [5]. This circuitry is located in the card cage.

The first-stage amplification circuitry for the force sensors is located inside the sensor-equipped leg. A second-stage amplifier is located in the card cage, where the signal is filtered and the gain may be adjusted. Force amplifier schematics can be found in [6].

2.4.3 Computer Interface

To expedite communication between the Hexapod and its control computer, a special purpose digital data link has been constructed. The data link is composed of two primary segments: the vehicle interface and the computer interface. There is no direct electrical connection between the two interfaces; all lines are optically isolated to

protect the computer in the event of an electrical malfunction of the Hexapod. The data link operation is outlined in the following paragraphs, and complete specifications can be found in [7].

The data path from the computer to the Hexapod is defined as the feedforward path, while the feedback path is directed from the Hexapod to the computer. Only the actuator input voltages need to be transmitted over the feedforward path. Thus, there are eighteen words of data to be transmitted each time the servo loop is executed by the computer. Each word consists of eight bits, and is transmitted in parallel along with five bits of address information. The data link addresses are memory mapped into the PDP-11/70 control computer, and transmission is initiated when the computer performs a write to one of the data link addresses. After allowing adequate set-up time for the optical isolators, the data is strobed into one of eighteen registers in the vehicle interface, as selected by the address information. This process requires a total of 40 microseconds, after which time the computer may transmit another word. A digital-to-analog converter associated with each of the data registers in the vehicle interface converts the data to an analog voltage, which is then used as the input voltage to a triac trigger generator. A block diagram of the feedforward circuitry is shown in Figure 2.4.

The feedback portion of the data link originally handled 54 channels of information, including 18 channels each of position and rate information. In addition, provision was made for 18 channel of force information, although only three were used in the initial system

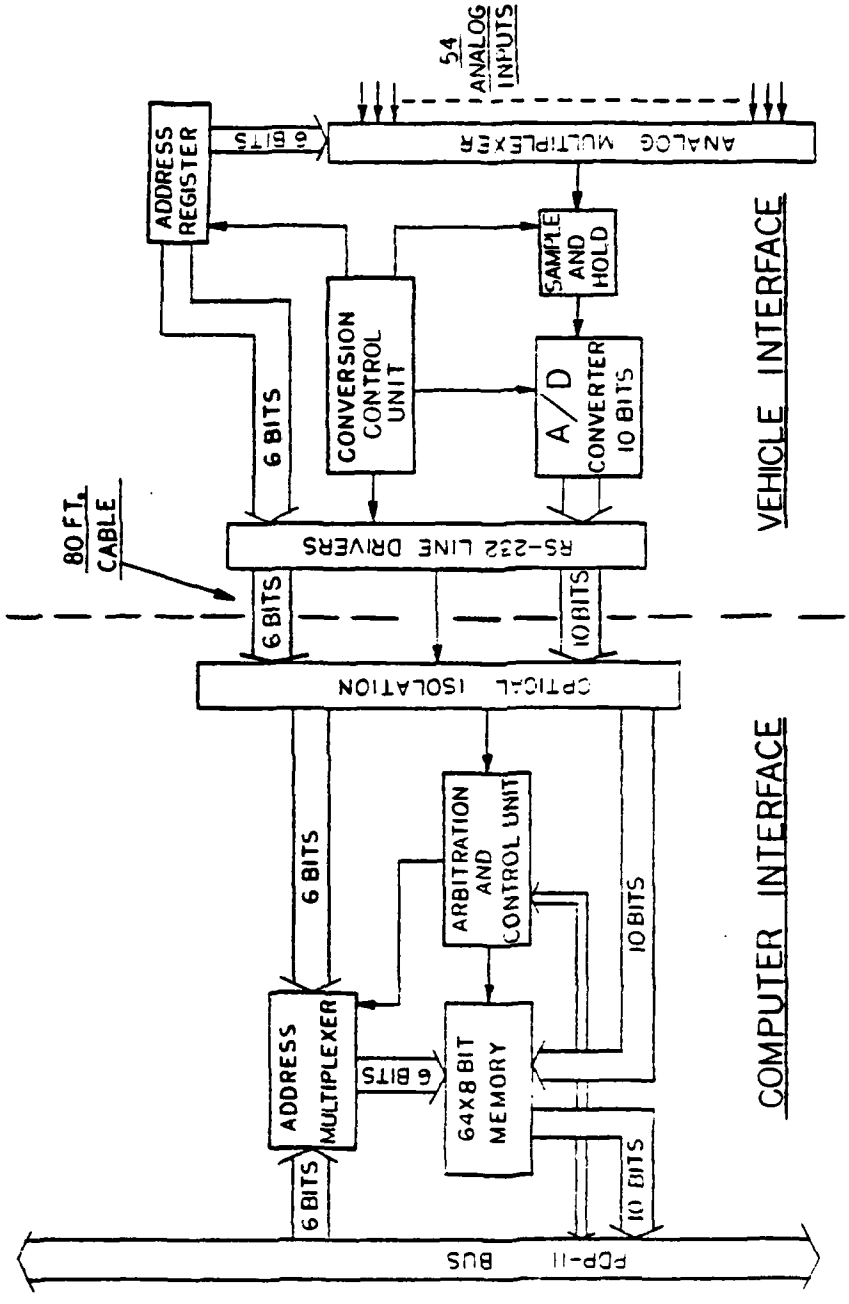


Figure 2.4 Functional Block Diagram of the Forward Portion of the Digital Data Link [7].

configuration. To implement the feedback without causing the computer to wait for conversion, a shared memory approach has been employed. A 64 word shared memory, located in the computer interface, is continually updated by the vehicle interface. Thus, the computer can obtain current information by simply performing a memory read operation. The feedback portion of the vehicle interface contains an internal clock and a modulus 54 counter to generate addresses. The address information selects one channel of a 54 channel analog multiplexer. The multiplexer selects one of the analog feedback signals, which is then input to an analog-to-digital converter. The resulting 10 bit digital word is then transmitted to the computer interface along with the 6 address bits. An arbitration and control circuit in the computer interface then resolves any memory access conflicts and directs the writing of the information into the shared memory. One data transmission requires 100 microseconds; thus each location in the shared memory is updated every 5.4 milliseconds. A Block diagram of the feedback circuitry is shown in Figure 2.5.

2.5 Software

2.5.1 Software Organization

The control software for the OSU Hexapod Vehicle is arguably the most complex part of the vehicle system. The control algorithm design is simplified by partitioning the task into well-defined functional blocks, or subtasks, with a minimal amount of communication between the subtasks. Although the current control program (Hexapod Control Program version 3.0) is the result of years of work by many researchers,

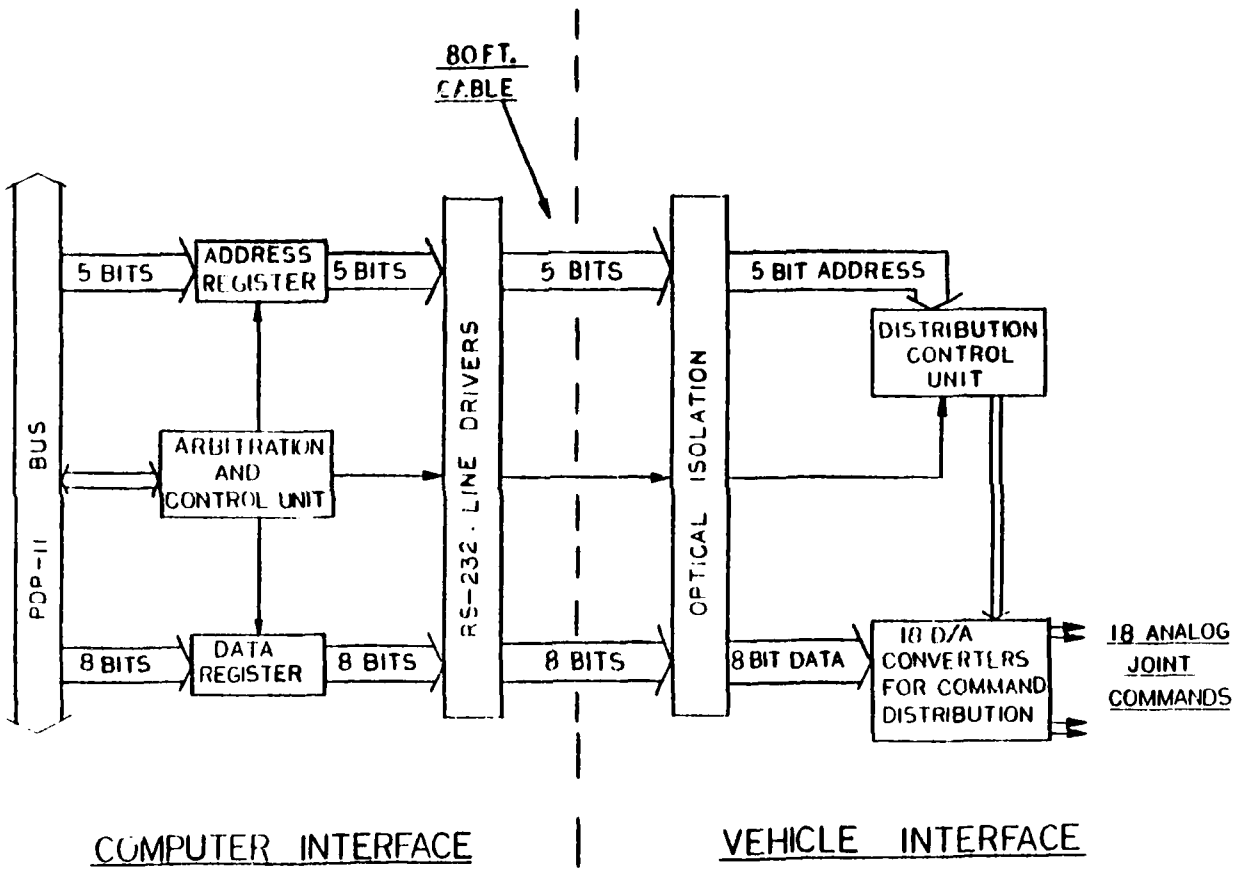


Figure 2.5 Functional Block Diagram of the Feedback Portion of the Digital Data Link [7].

the partitioning problem is still not perfectly understood. Version 3.0 software represents a continuing effort to define a structured algorithm which can be reflected in the software architecture.

The control task partitioning of Hexapod Control Program version 3.0 is illustrated in Figure 2.6. The figure shows the general direction of information flow among the subtasks. The particular algorithm implemented in a given functional block can vary, depending on the type of motion to be performed by the vehicle. The parameters which are passed between the functional blocks are not, in general, precisely specified, but rather vary somewhat depending on which algorithms are being executed.

Figure 2.7 shows the major routines of version 3.0 and their relationship to the control structure of Figure 2.6. Each block in Figure 2.7 is implemented as a subroutine, with the exception of the executive software. The software is largely self-documenting; the parameters required by a given subroutine are identified in the subroutine header.

2.5.2 Executive Software

The executive software provides the interface between the human operator and the vehicle system via a CRT terminal. The operator enters commands through the keyboard, specifying parameters such as vehicle speed, direction, operating mode, etc. The current vehicle status is displayed on the CRT, along with a list of valid operator commands. Program flow is established by the executive software as a function of the operator inputs.

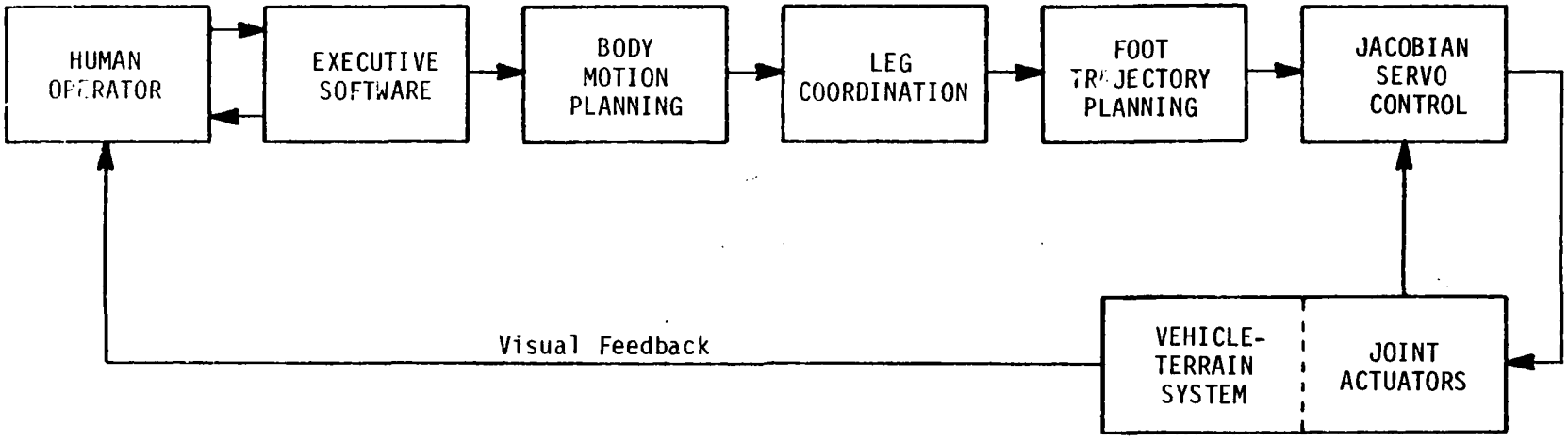


Figure 2.6 Control Task Partitioning of Hexapod Control Program Version 3.0.

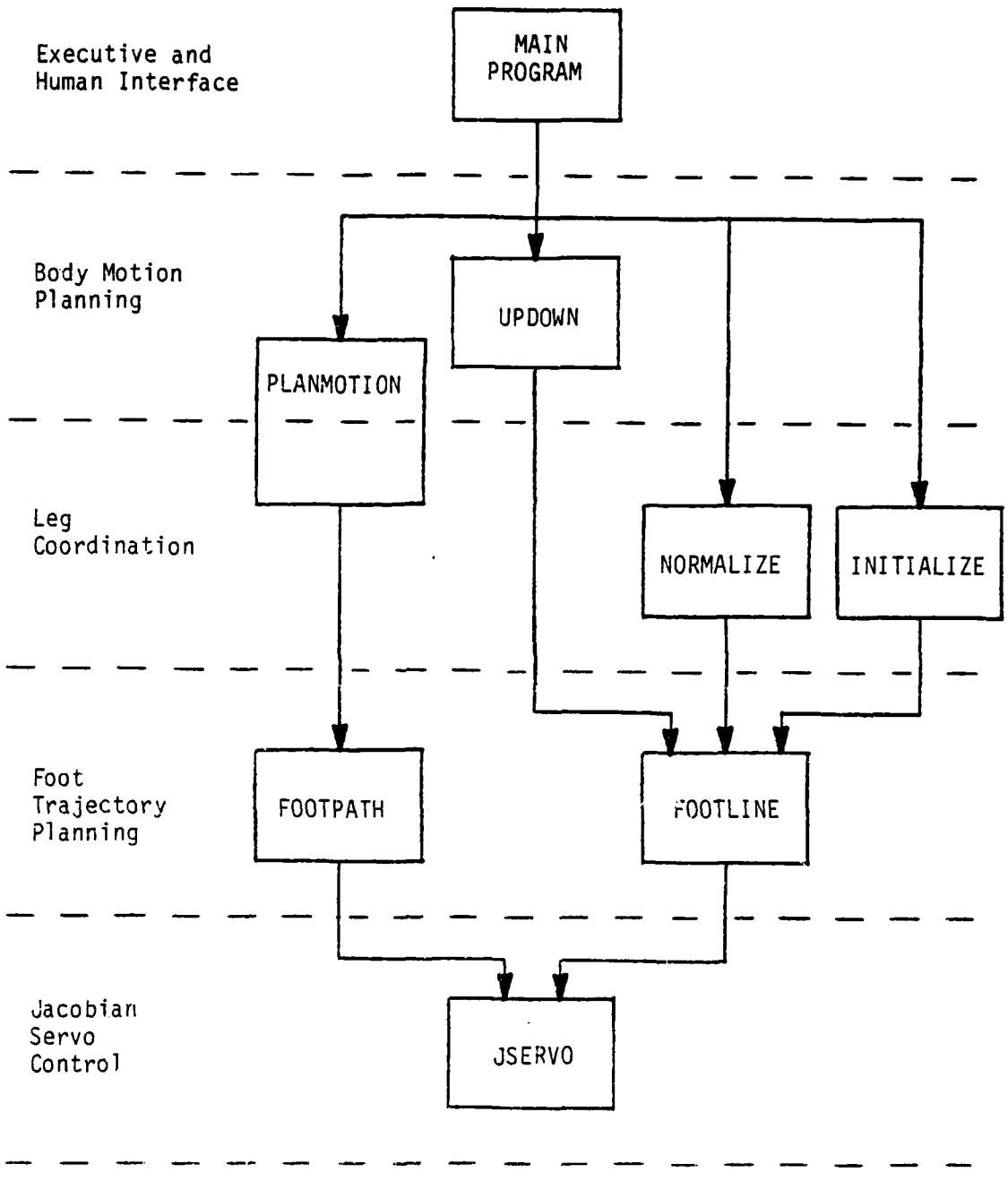


Figure 2.7 Software Organization of Hexapod Control Program Version 3.0.

2.5.3 Body Motion Planning

The body motion planning functional block is responsible for conditioning operator speed and direction commands so that they can be implemented by the lower level routines. This includes low-pass filtering the input velocities and limiting the velocities to realizable values. For some simple motion classes, this functional block can be eliminated.

2.5.4 Leg Coordination

The leg coordination algorithms are designed to insure that the vehicle is always supported by three or more legs. In addition, the legs must not collide with other legs. For ambulatory motion, these conditions are satisfied implicitly by implementing periodic gaits [8]. Other algorithms, such as the normalization routine, satisfy these conditions explicitly.

2.5.5 Foot Trajectory Planning

Hexapod Control Program version 3.0 contains two foot trajectory planning routines, FOOTPATH and FOOTLINE. The most important of these, FOOTPATH, is used during wave gait implementation. In this routine, the trajectory of a given foot is divided into two phases: support phase and transfer phase. A foot is in support phase when it is on the ground supporting a portion of the vehicle weight. Vehicle direction and speed are determined by the trajectories of the feet in support phase. Algorithms have been implemented in version 3.0 software which allow the vehicle to exhibit three ambulatory modes [9,10]. In cruise

mode, the vehicle may walk forward or backward, turn with a minimum radius of 60 inches, crab at up to a 45 degree angle from the vehicle heading, or exhibit all of these motions simultaneously. Turn-in-place mode allows the vehicle to perform a pure rotational motion about its geometrical center, and sidestep mode implements pure sideways locomotion. With the system configured as it was prior to the work of this thesis, locomotion in any of these modes is possible only over relatively smooth, level ground.

When a leg reaches the end of its support phase, it enters transfer phase. In transfer phase, the foot lifts off of the ground and moves to the point, predicted by an algorithm in [10], where it will next enter support phase. The foot follows a half-sine wave trajectory through the air in transfer phase. Switching from transfer phase to support phase is done as a function of the kinematic cycle phase variable.

The second foot trajectory planning routine, FOOTLINE, moves all six feet along arbitrary and independent straight-line trajectories. This is useful for simple procedures such as the initializing of leg positions. In version 3.0, FOOTLINE is used to implement all motions except ambulatory motion. The output of both foot trajectory routines is the desired position and desired rate of a foot, expressed in body coordinates.

2.5.6 Jacobian Servo Control

The function of the Jacobian servo routine is to cause the vehicle legs to follow the trajectories specified by the foot trajectory planning routines. The Jacobian servo routine implemented in version 3.0 has two servo levels. The inner loop is a rate feedback loop which controls the angular rate of a given actuator. The outer loop is closed in rectangular body coordinates and controls the position and velocity of a foot tip. A block diagram of the control structure is shown in Figure 2.8, and definitions of the control parameters are given in Table 2.1.

The Jacobian servo derives its name from the use of the inverse Jacobian matrix to transform between rectilinear rates and joint rates, where the Jacobian matrix is the matrix of partial derivatives of rectilinear foot position coordinates as functions of joint angles. The general Jacobian control problem is discussed in [11], and the specific implementation used for control of the OSU Hexapod Vehicle is well documented in [12].

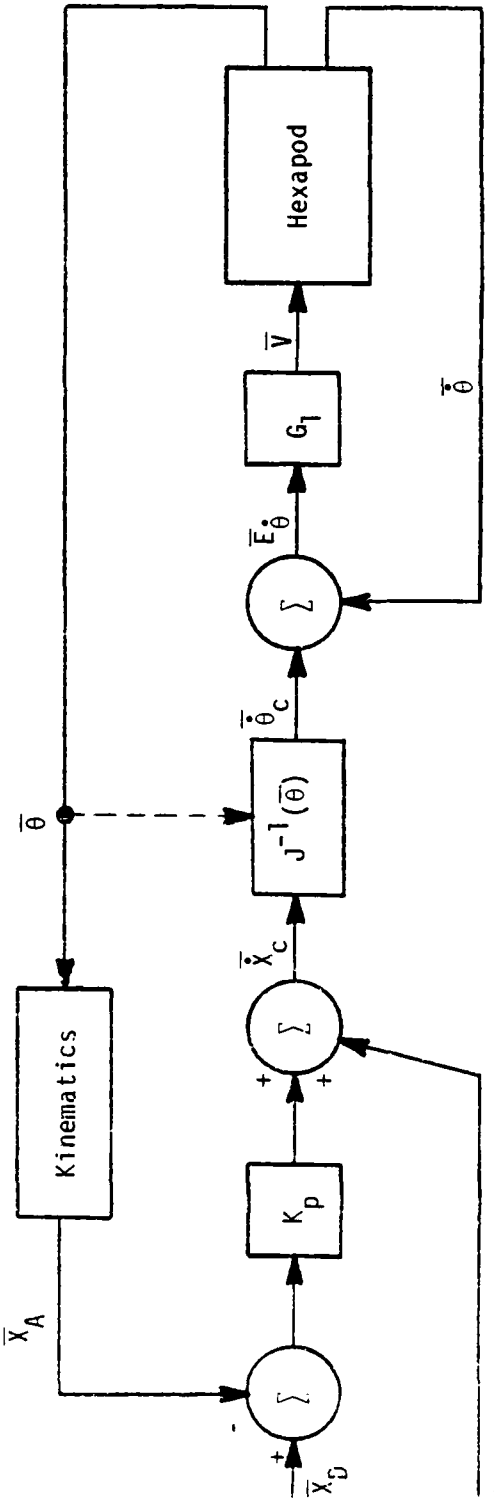


Figure 2.8 Jacobian Control Structure for One Leg of the OSU Hexapod Vehicle.

TABLE 2.1

JACOBIAN CONTROL PARAMETER DEFINITIONS

$\overline{X}_D \equiv [X_D \; Y_D \; Z_D]^T$	(desired foot position expressed in body coordinates)
$\overline{X}_A \equiv [X_A \; Y_A \; Z_A]^T$	(actual foot position expressed in body coordinates)
$\dot{\overline{X}}_D \equiv [\dot{X}_D \; \dot{Y}_D \; \dot{Z}_D]^T$	(desired foot velocity expressed in body coordinates)
$\dot{\overline{X}}_A \equiv [\dot{X}_A \; \dot{Y}_A \; \dot{Z}_A]^T$	(actual foot velocity expressed in body coordinates)
$\dot{\overline{X}}_C \equiv [\dot{X}_C \; \dot{Y}_C \; \dot{Z}_C]^T$	(commanded foot velocity expressed in body coordinates)
$\dot{\overline{\theta}}_C \equiv [\dot{\psi}_C \; \dot{\theta}_{1C} \; \dot{\theta}_{2C}]^T$	(vector of joint rate commands)
$\overline{E}_{\dot{\theta}} \equiv [e_\psi \; e_{\theta 1} \; e_{\theta 2}]^T$	(vector of joint rate errors)
$\overline{V} \equiv [V_\psi \; V_{\theta 1} \; V_{\theta 2}]^T$	(vector of joint actuator input voltages)
$\overline{\theta} \equiv [\psi \; \theta_1 \; \theta_2]^T$	(vector of actual joint angles)
$\dot{\overline{\theta}} \equiv [\dot{\psi} \; \dot{\theta}_1 \; \dot{\theta}_2]^T$	(vector of actual joint rates)

$$K_p \equiv \begin{bmatrix} k_{pX} & 0 & 0 \\ 0 & k_{pY} & 0 \\ 0 & 0 & k_{pZ} \end{bmatrix}$$ (position gain matrix)

$J(\overline{\theta})$ ≡ Jacobian matrix which converts from joint rates to rectilinear foot velocity

Kinematics ≡ equations which convert joint angles to rectilinear foot coordinates

Chapter 3

HEXAPOD SYSTEM MODIFICATIONS AND ADDITIONS

3.1 Introduction

The OSU Hexapod Vehicle, as described in Chapter 2, did not have the capability of walking on uneven terrain. The vehicle configuration was designed for the purpose of rough-terrain locomotion, but was unable to utilize its highly flexible geometry for this purpose due to a lack of sensor hardware and control software. The proposed vehicle autopilot must allow the Hexapod to traverse irregular terrain while maintaining the vehicle pitch and roll attitudes within acceptable limits with only directional inputs being provided by a human operator.

To implement the autopilot function, the Hexapod system required new software control algorithms and additional sensing hardware. The addition of vector force sensors on all legs of the Hexapod provides ground reaction force information, and a vehicle attitude sensor provides the needed information about body pitch and roll. The sensor hardware design is detailed in this chapter, along with system hardware modifications necessary to accommodate the sensors. Additions made in the interest of improving system reliability are also documented in this chapter. A discussion of the autopilot control algorithms is presented in Chapter 4.

3.2 Force Sensors

As noted in section 2.3, one leg of the Hexapod was equipped with vector force sensors in 1977. This sensor design, which is described in section 2.3 and detailed in [6], was evaluated as a basis for the new sensors. The design had proven to be reliable over the three years it was in operation, and the sensor performance was demonstrated to be adequate in [6] and [7]. As a result, the lateral force sensor design was retained, and consists of two semiconductor strain gauges mounted on adjacent faces of the lower limb segment of the Hexapod leg.

The amplifier circuit configuration used with the semiconductor strain gauges is identical to that used in [6]. It consists of a Wheatstone bridge composed of a strain guage plus two external resistors and a trim potentiometer. The bridge circuit is followed by a differential amplifier which is located inside the vehicle leg. A second amplification stage which provides manual gain adjustment and low-pass filtering is located in the card cage at the rear of the vehicle. Schematics for the strain gauge amplification circuitry are shown in Figure 3.1.

The axial force sensor in the previous design consisted of a semiconductor load cell mounted in an aluminum housing, with the ground reaction force being transmitted to the load cell via a steel piston. The load cell used in [6] was no longer available at the time of the redesign, and thus a search of comparable commercial sensors was undertaken. Three types of load cells were investigated: semiconductor,

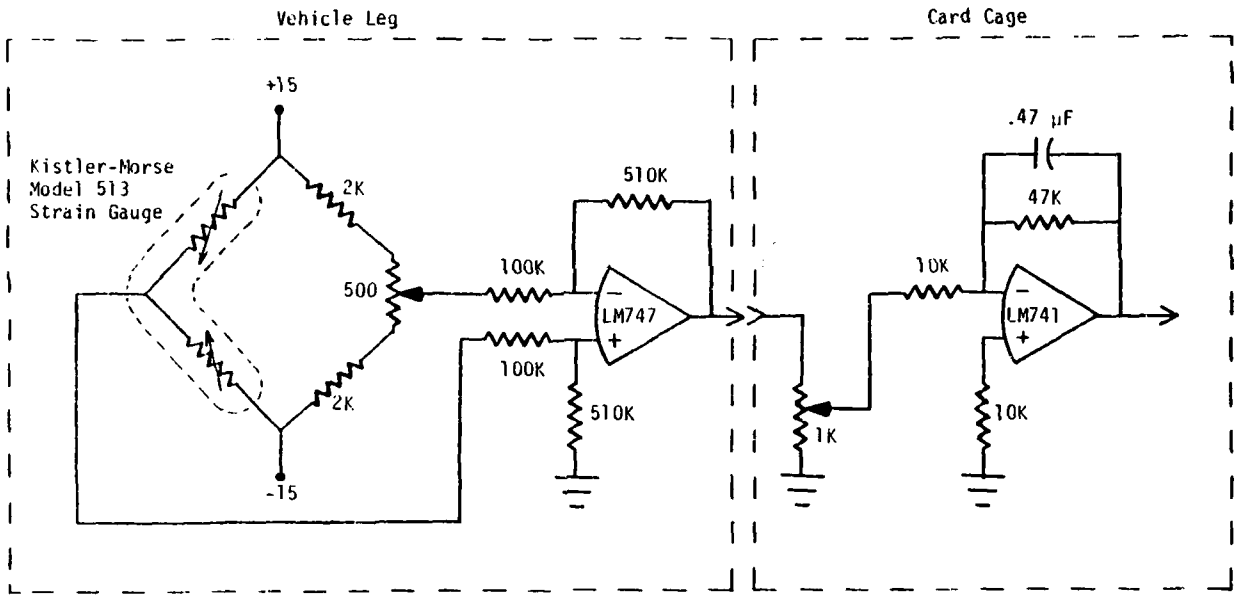


Figure 3.1 Strain Gauge Amplification Circuitry.

strain gauge, and piezoelectric. The characteristics of these devices are listed in Table 3.1.

TABLE 3.1

LOAD CELL CHARACTERISTICS

Strain Guage:	Low output
	Shear loadings not tolerated
	Poor dynamic range
	Large size
	Usable for static applications
Semiconductor:	High output
	Shear loadings not tolerated
	Fair dynamic range
	Small size
	Usable for static applications
Piezoelectric:	Selectable output
	Small shear loadings tolerated
	Excellent dynamic range
	Small size
	Not for static applications

The strain gauge type of load cell was rejected as a design solution primarily on the basis of its physical size and poor dynamic range. The semiconductor type of load cell had been shown to be a workable solution. However, it was believed that the Hexapod, weighing 285 pounds, could exceed the 1000 pound impact rating of the load cell used previously if it were to slip off of an obstacle. Semiconductor load cells with an impact load rating of more than 1000 pounds could not be located.

The major difficulty with the use of piezoelectric load cells in this application is the fact that they require charge amplifiers, which drift over time. Hexapod locomotion, however, gives an opportunity to recalibrate the amplifier outputs when the foot is in the air during transfer phase. This fact makes the piezoelectric load cell a workable solution when coupled with a low drift rate charge amplifier, and this type of load cell was therefore chosen as the axial force transducer.

Consideration of the design factors discussed above led to the selection of a Kistler Instrument Corporation model 912 load cell as the axial force sensor. The relevant specifications for this device are given in Table 3.2.

TABLE 3.2

LOAD CELL SPECIFICATIONS

Manufacturer:	Kistler Instrument Corp.
Model:	No. 912
Range (compression):	5000 lbs.
Resolution:	0.002 lbs.
Overload:	20 percent
Sensitivity:	50 picocoulombs/lb.
Rigidity:	0.2 microinches/lb.
Linearity:	+/- 1 percent
Capacitance:	58 picofarads
Insulation resistance:	10,000,000 megohms
Shear force (maximum):	100 lbs.
Shock and vibration:	10,000 g's
Size:	5/8 in. hexagonal x 1/2 in.

With the piezoelectric transducers specified, attention was turned to charge amplifier selection. It was desired to mount the charge amplifiers inside the lower leg segments of the Hexapod. The commercially available units were too large to allow internal mounting. This fact, together with the expense of the units (about $500.00 each), leg to a decision to design custom amplifiers.

The output of a piezoelectric crystal is an electrical charge which is proportional to the applied force, or

$$Q = kF \tag{3.1}$$

where k is the sensitivity of the crystal. When the charge is allowed to flow into an amplifier, the resulting current is given by

$$i = \frac{dQ}{dt} = k \frac{dF}{dt} \tag{3.2}$$

If the amplifier input current is integrated, the resulting output voltage is then proportional to the applied force. A circuit which performs this integration is shown in Figure 3.2. If the circuit components are ideal, the circuit transfer function is

$$V = - \frac{1}{C} \int i \, dt \tag{3.3}$$

where C is the value of the capacitor in farads. Substituting equation 3.2 into 3.3 gives

$$V = - \frac{1}{C} \int k \, dF \tag{3.4}$$

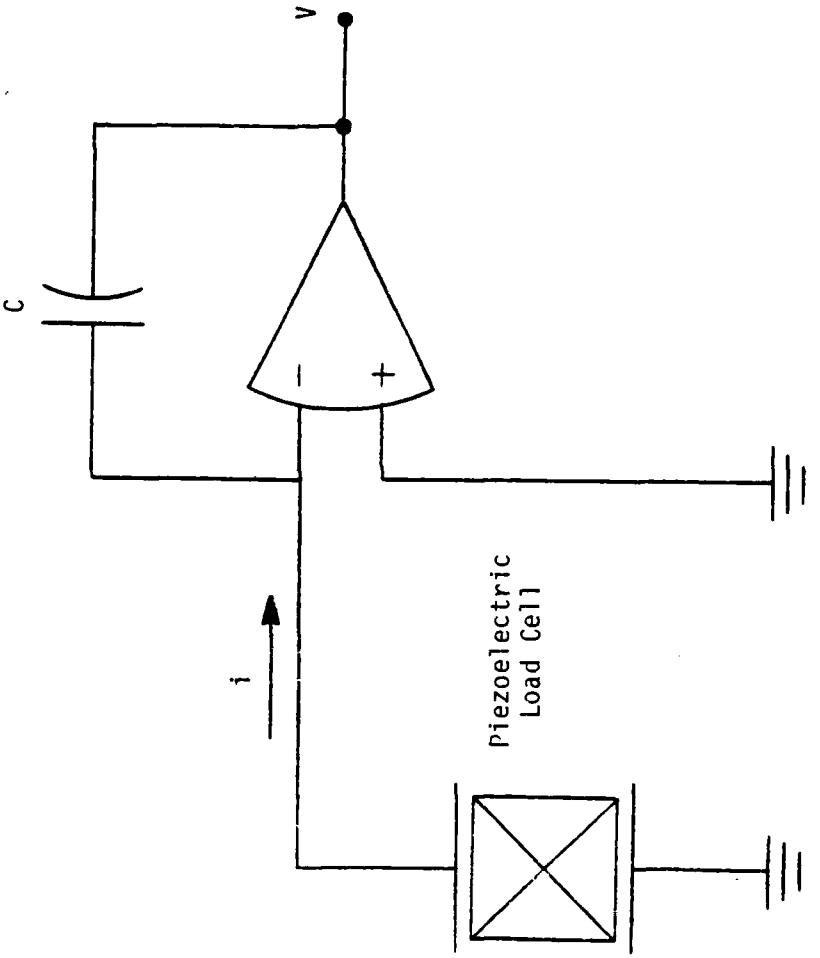


Figure 3.2 Ideal Charge Amplifier.

or

$$V = -\frac{K}{C} F \tag{3.5}$$

which is the desired result.

It was desired that the charge amplifier circuitry should not be damaged when the full 5000 pound rated load was applied to the load cell. The feedback capacitor must therefore be large enough to store the output charge of the crystal at full load without exceeding the input voltage rating of the operational amplifier, which is approximately 12 volts when operated with a 15 volt supply. The maximum output charge of the crystal is equal to the maximum load multiplied by the crystal sensitivity, or 0.25 microcoulombs. A 0.022 microfarad capacitor stores this amount of charge with a 12 volt output, and is therefore the value used in the circuit. From equation 3.5, this gives an output voltage of

$$V = 2.5 \frac{Mv}{lb.} \tag{3.6}$$

To minimize capacitor leakage, a polystyrene capacitor was employed with an insulation resistance of 10,000,000 megohms.

In order to minimize integrator drift, an operational amplifier with a low input bias current was needed. Referring to Figure 3.2, it is seen that any current flowing into the inverting input of the operational amplifier is directly subtracted from the current being integrated, and would eventually cause the integrator to drift into saturation. The operational amplifier chosen to minimize this problem

was the National Semiconductor model LH0052D. This J-FET input amplifier has a typical input bias current of 0.5 picoamps at room temperature, and an input resistance of 1,000,000 megohms. Using the value of capacitance chosen, the bias current results in an output drift rate of 81 mV/hr. In terms of force, the drift rate is 32.5 lbs./hr. from equation 3.6.

Although the charge amplification circuitry was designed to be undamaged by a 5000 pound input, the maximum measurable force does not need to be this large. The design value used for maximum measurable force was 200 pounds, or slightly more than half the weight of the vehicle. Forces exceeding this value may cause saturation of the final amplification stages.

To prevent the charge amplifier from drifting into saturation, it was decided to place a large resistor in parallel with the feedback capacitor, making the circuit an extremely low-pass filter. The value of resistance chosen reflects a compromise between a large time constant and a small maximum voltage drift. It was decided to choose the resistor such that the maximum output voltage due to drift is equal to the voltage due to the maximum measurable force intput. A 200 pound input results in an output voltage of 500 mV, from equation 3.6. Rather than designing with a drift current equal to the 0.5 picoamp input bias current, it was decided to use a value of 5 picoamps to allow for current leakage through the printed circuit board and variations in operational amplifiers. To satisfy the above condition the feedback resistor must realize a voltage drop of 500 mV when the entire

5 picoamp drift current is passing through it. This dictates the selection of a 100,000 megohm resistor. Neglecting insulation resistances, the circuit time constant is then 37 minutes.

Due to the relatively low output of the charge amplifier, it was necessary to provide additional voltage amplification. It was desired that the maximum measurable force input of 200 pounds should cause a 5 volt swing at the circuit output, leaving sufficient headroom for the integrator drift. Thus, a gain of 100 was needed in the voltage amplification circuit. The voltage amplifier was constructed in two stages. The first stage is located in the vehicle leg near the charge amplifier, and raises the signal to a level safe from electrical noise. The second stage is located in the card cage at the rear of the vehicle. It low-pass filters the signal and provides a manual gain adjustment. Schematics of the complete charge amplifier circuit are shown in Figure 3.3.

To use the model 912 quartz load cell as a foot force sensor, it was necessary to construct a special housing for the load cell. The housing, which was designed in the Department of Mechanical Engineering, is illustrated in Figure 3.4. A steel piston is employed to transmit the ground reaction forces to the load cell. Teflon bushings minimize friction between the piston and the aluminum case, and also provide electrical insulation between the vehicle leg and the load cell. The point at which the load cell is mounted to the case is insulated with a high-impact plastic insert. The assembly was designed to withstand the 5000 pound rating of the load cell. Additional

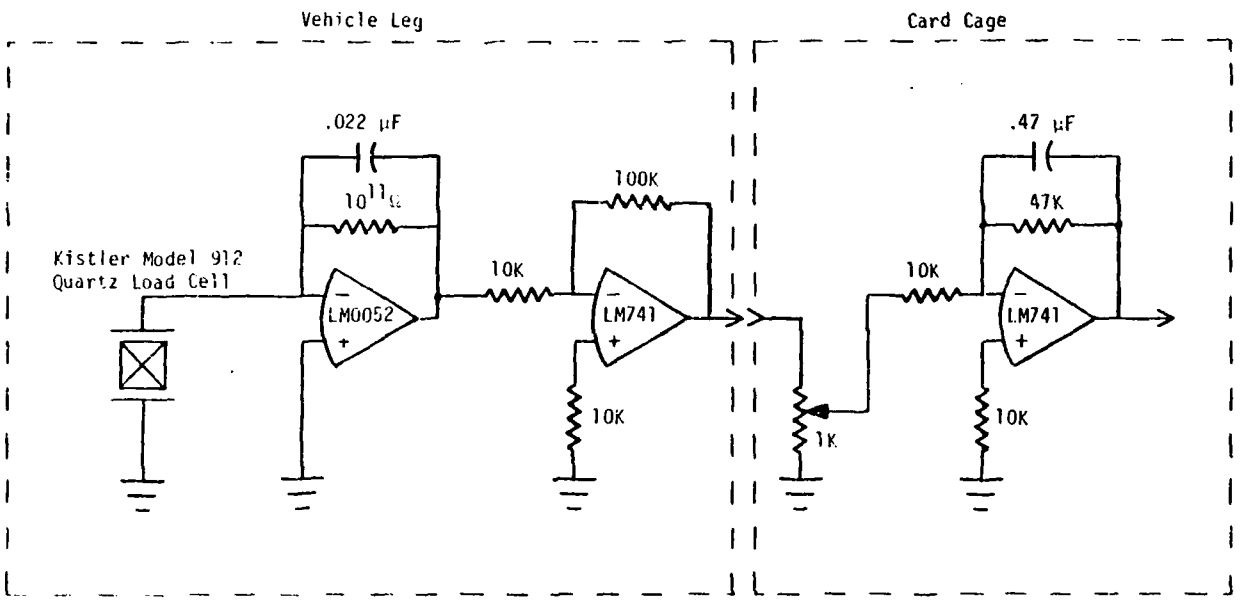


Figure 3.3 Piezoelectric Charge Amplification Circuitry.

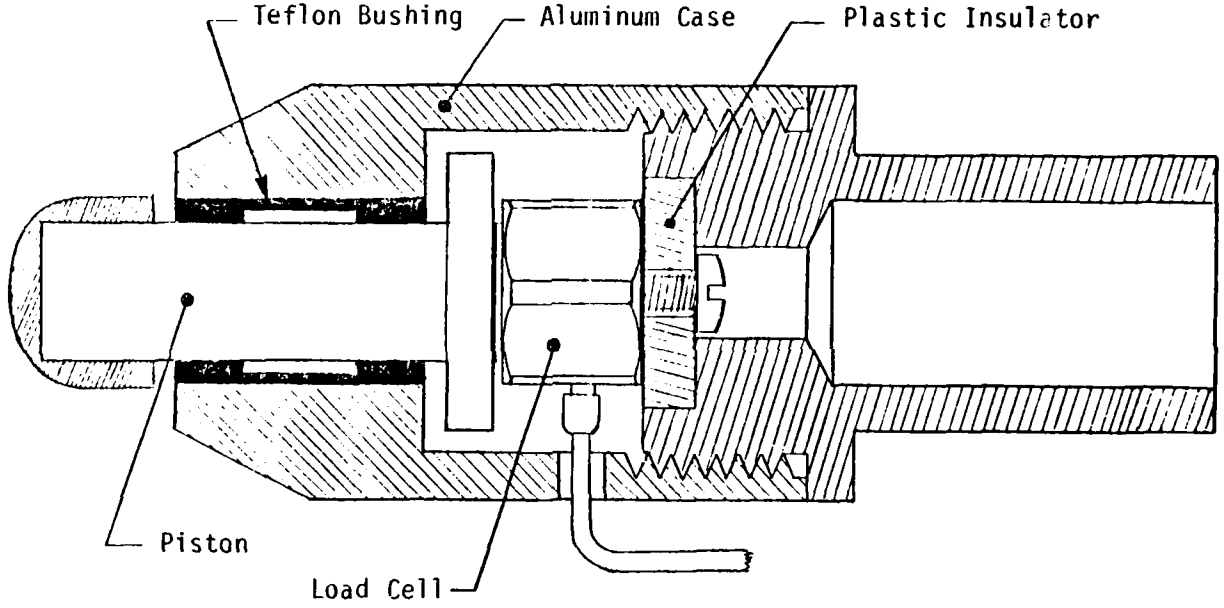


Figure 3.4 Load Cell Housing.

details can be found in [13]. Tests have been performed which show that the piston stiction never exceeds 10 percent of the applied side loading on the piston under static conditions. Since Hexapod locomotion is a dynamic process, even better results should be obtained in actual use.

A photograph of a force sensor equipped leg is shown in Figure 3.5. The unit is shown disassembled in Figure 3.6.

3.3 Attitude Sensors

Two types of attitude sensors were investigated for use on the OSU Hexapod Vehicle: gravitational pendulums and vertical gyroscopes. The characteristics of each are listed in Table 3.3.

TABLE 3.3
ATTITUDE SENSOR CHARACTERISTICS

Pendulums:	Small size and weight
	Low cost
	Immediate operation
	Sensitive to lateral acceleration
	Limited bandwidth
	High reliability
Gyroscopes:	Larger size and weight
	High cost
	Requires time to "spin up" and erect
	Insensitive to lateral acceleration
	Infinite bandwidth
	Limited lifetime

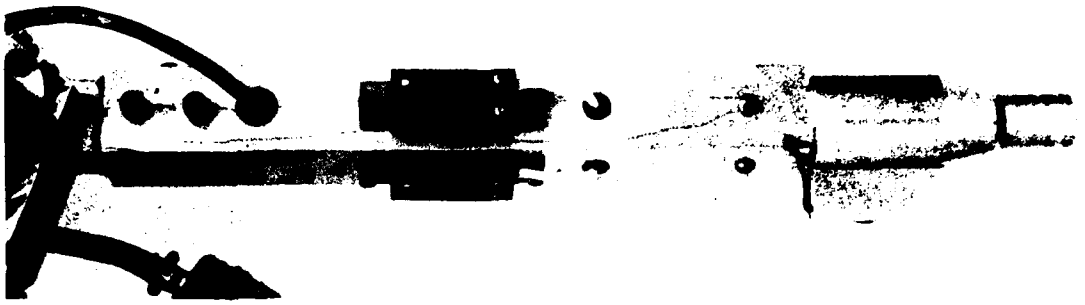


Figure 3.5 OSU Hexapod Leg with Force Sensors Installed.

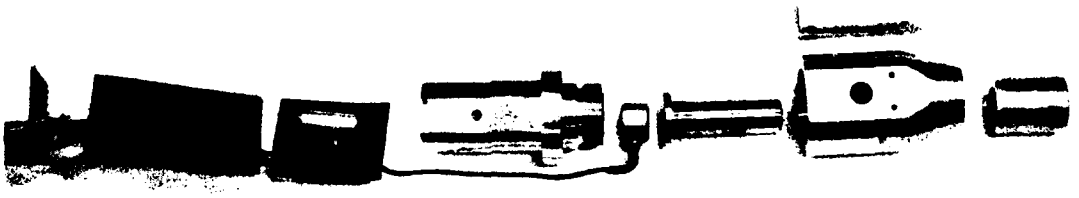


Figure 3.6 Vertical Force Sensor Unit Disassembled.

The pendulums are seen to be desirable for this application in terms of their size, reliability, ease of operation, and cost. At the time of sensor selection, however, it was not known whether their limited bandwidth and sensitivity to acceleration would represent a serious problem. Since a vertical gyroscope does not exhibit these shortcomings, it was decided that one should be installed on the vehicle as the primary attitude sensor. The problem of high cost was avoided by the use of a surplus gyroscope. A pair of pendulums was also installed as a backup sensor and for evaluation as the primary attitude sensor. The specifications for the sensors chosen are given in Table 3.4. A photograph of the vertical gyroscope is shown in Figure 3.7, and the pendulum installation is shown in Figure 3.8. The Hexapod is shown with all sensors installed in Figure 3.9.

3.4 Data Link Expansion

All sensor outputs on the OSU Hexapod Vehicle are transmitted to the control computer via the digital data link which was described in section 2.4.3. It will be recalled that the feedback portion of the data link was originally designed with a 54 channel capability, allowing for 18 channels each of rate, position, and force information. Since the attitude sensors require 4 channels, it was necessary to expand the data link to allow for the additional sensors.

It was decided to use the full data link address space of 64 channels to provide for future sensor additions. The analog multiplexer, however, was expanded to a 72 channel capability and is

TABLE 3.4

ATTITUDE SENSOR SPECIFICATIONS

GYROSCOPE	Manufacturer:	Electronic Specialty Co.
	Model:	N3200
	Size:	8" × 7.5" × 6.75"
	Weight:	6.25 lb.
	Erection system:	Gravity controlled air jets
	Erection rate:	3 deg./min. average
	Caging time:	1 minute maximum
	Output:	2000 Ohm potentiometer
	Accuracy:	+/- 1.5 deg. maximum error
	Shock rating:	30 G's
PENDULUMS	Manufacturer:	Humphrey, Inc.
	Model:	CP17-0601-2
	Size:	2.5" dia. × 1.3" deep
	Weight:	.75 lb.
	Natural frequency:	2 Hz. minimum
	Damping ratio:	.7 nominal
	Output:	2000 Ohm potentiometer
	Accuracy:	+/- 1% (static conditions)
	Shock rating:	100 G's for 10 milliseconds

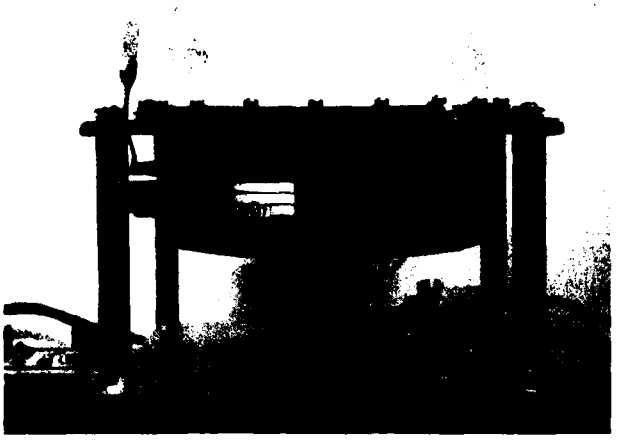


Figure 3.7 Vertical Gyroscope Installation.

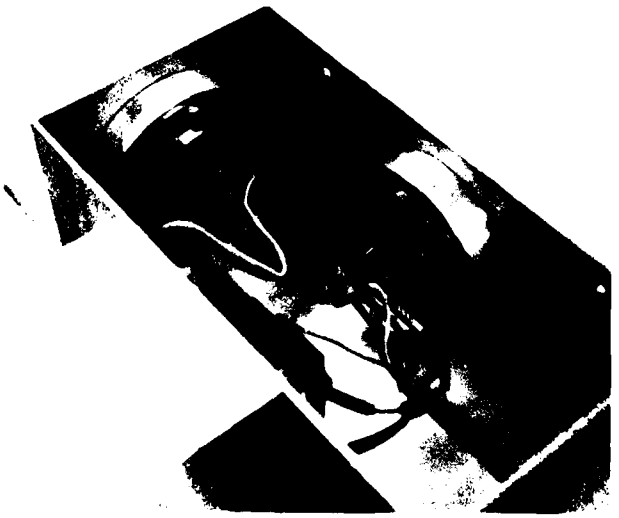


Figure 3.8 Gravitational Pendulum Installation.

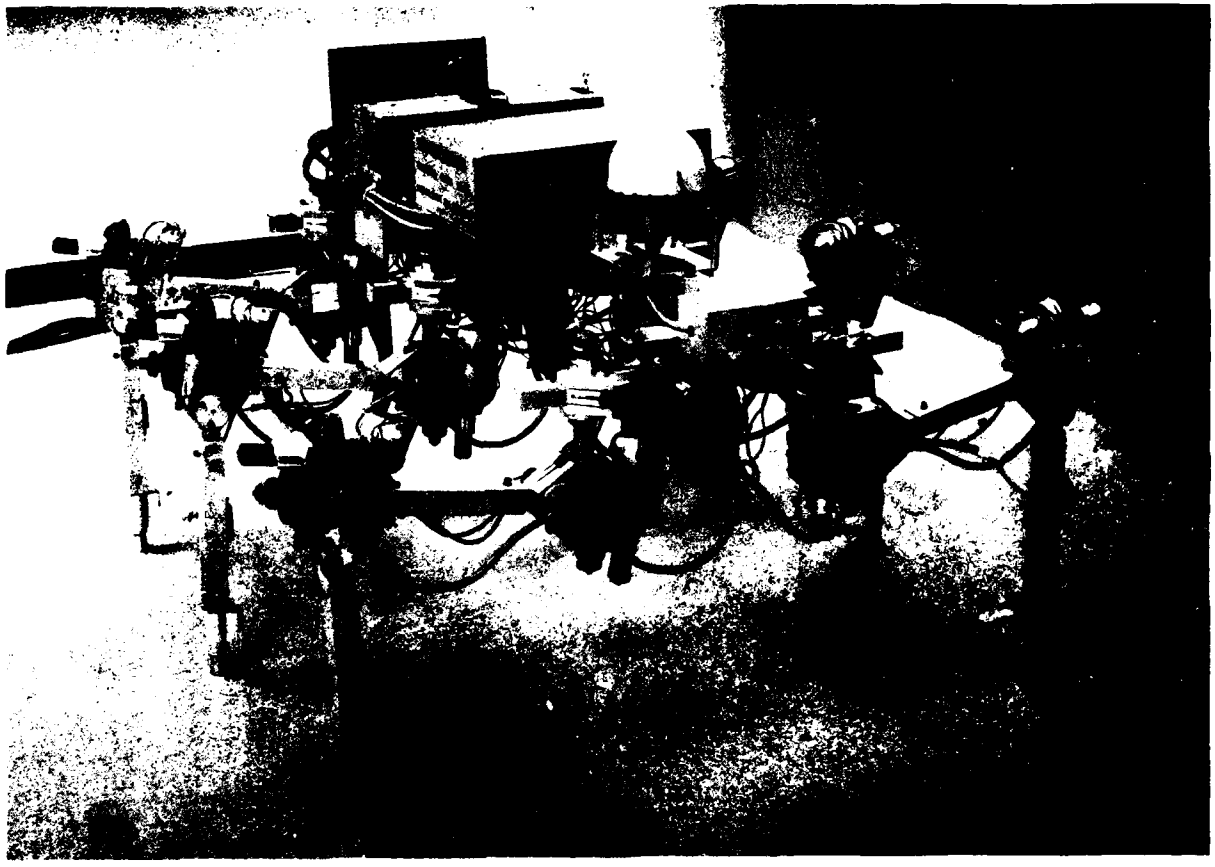


Figure 3.9 OSU Hexapod Fully Equipped with Sensors.

organized in nine groups of eight channels each. Group 1 through group 7 are always active. The last two groups, 8A and 8B, occupy the same address space and are selected by a specific command from the control computer. To provide this command capability, an additional data register was added to the data link feedforward circuitry. This command register is memory mapped to the control computer at address 166244 octal. The command register organization is illustrated in Figure 3.10.

In addition to the analog channel expansion of the data link, it was decided to include features to improve the realiability of the system. To reduce the chance of human error, a circuit was constructed which allows motor supply power and instrumentation electronics power to be controlled directly by the computer. Another data register was added to the data link feedforward circuitry for communication with the power controller. The power register is memory mapped to the control computer at address 166246 octal. The power register organization is illustrated in Figure 3.10.

A four channel digital multiplexer was added to the data link feedback circuitry to increase the flexibility of the system. Channel 0 is connected to the output of the analog-to-digital converter, and is selected in normal operation. Channel 1 provides digital status information about the Hexapod, and is selected when data link address 63 is decoded. Thus, address 63 is seen externally as a status word and not as a sensor output, which reduces the number of available analog channels to 70. The status word is defined in Figure 3.10.

COMMAND REGISTER (address 166244_8)

7	6	5	4	3	2	1	0
X	X	X	X	X	ANALOG SELECT	FORWARD TEST	FEEDBACK TEST
					0→Group 8A 1→Group 8B	0→OFF 1→ ON	0→OFF 1→ ON

POWER REGISTER (address 166246_8)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	ELECTRONICS POWER	MOTOR POWER
						0→OFF 1→ ON	0→OFF 1→ ON

STATUS WORD (address 166176_8)

7	6	5	4	3	2	1	0
POWER CONTROLLER MODE	X	X	X	X	EMERGENCY KILL CIRCUIT	ELECTRONICS POWER	MOTOR POWER
0 →MANUAL 1→COMPUTER					0→INACTIVE 1→ ACTIVE	0→OFF 1→ ON	0→OFF 1→ ON

Figure 3.10 Data Link Control Registers. Bits 8 through 15 are not used.

The remaining two channels of the digital multiplexer are used to implement data link self-diagnostics. Channel 2 is used to perform a feedforward circuit test, and is connected to the output of one of the feedforward data registers. Channel 3 is connected to the data link address lines, and is used in feedback circuit diagnostics. Channels 2 and 3 are selected by the command register, as illustrated in Figure 3.10.

Due to the modularity of the original design, it was possible to make the necessary data link modifications by redesigning only the feedback circuits which are located in the vehicle interface. A block diagram of the redesigned feedback circuitry is shown in Figure 3.11. In addition to the digital multiplexer, a pipeline register was added at the circuit output which allows data conversion to be performed while the last channel is being transmitted to the computer interface. The pipeline register allows the data conversion frequency to be doubled from 10 kHz to 20 kHz. Thus, all 64 channels are updated in 3.2 milliseconds.

3.5 Utility Software

To realize the system reliability improvements made possible by the hardware additions described in the last section, software was written which tests specific Hexapod systems automatically. To provide all Hexapod users with this capability, subroutines were written which can be linked with any control program. These subroutines are contained in file DLNK35.PAS of Hexapod Control Program version 3.5, which is listed in Appendix C.

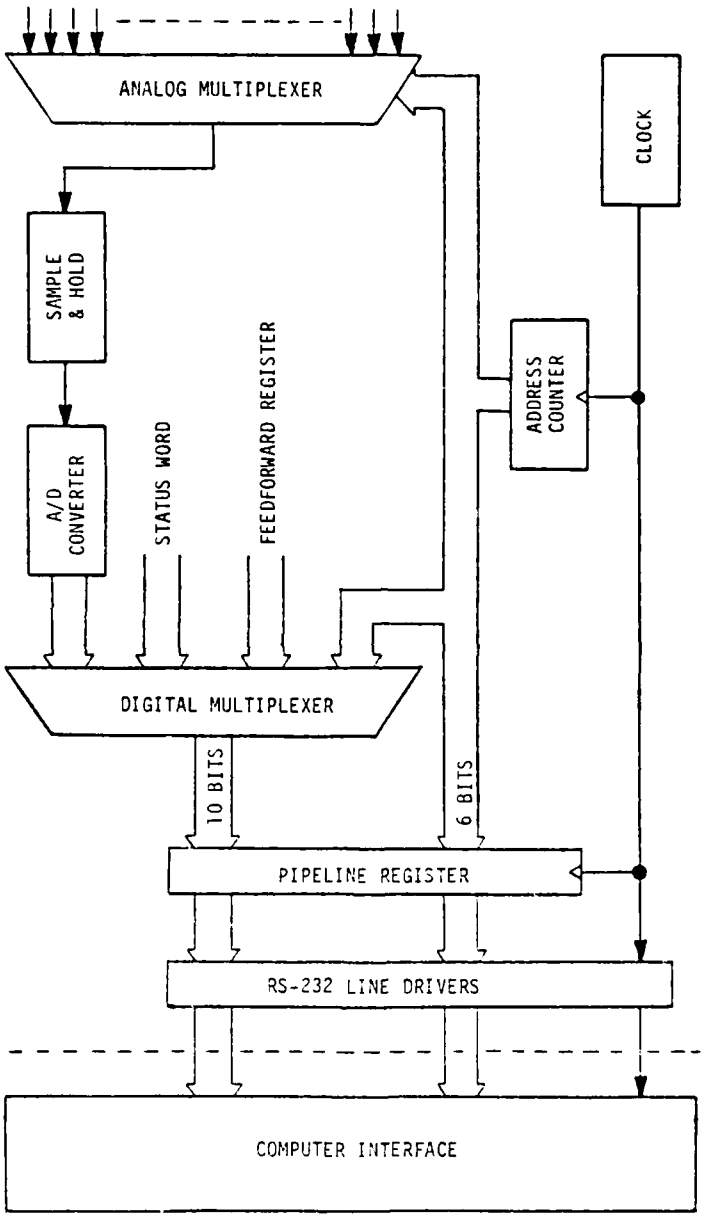


Figure 3.11 Functional Block Diagram of the Feedback Section of the Digital Data Link Located in the Vehicle Interface.

The data link diagnostics are performed in two steps. First, the feedback portion of the data link is tested independently. The feedback test mode is selected by setting bit 0 of the command register. This causes the data link address lines to be multiplexed onto the data lines. The contents of a location in the shared memory can then be predicted from the address of that memory location. If the data link is operating properly, the predicted contents will agree with the actual contents. If not, a Boolean equivalence operation will identify which bits are incorrect. Note that this test will not detect every possible malfunction of the feedback circuits, but does thoroughly test the line drivers, transmission cable, and optical isolators.

When proper operation of the feedback circuitry is verified, the forward path may be tested. By setting bit 1 of the command register, the output of the feedforward register is multiplexed onto the feedback data lines. Binary values are then written to that register which test all bits of the forward path. The contents of the shared memory are then compared to the value written, and bad bits can be detected as before.

In addition to data link diagnostics, routines were written which test and calibrate all sensors on the vehicle. To test the force and attitude sensors, the measured quantities are output on a CRT display, where they can be compared with an external reference. The potentiometers are tested by servoing on their output with a fixed reference input. The leg positions are then visually calibrated by utilizing the position offset adjustment potentiometers located in

the card cage. To calibrate the tachometers, their outputs are numerically integrated while the joints are moved through known angular displacements. The tachometer gains are then adjusted such that the rate integral agrees with the known displacement.

Chapter 4

DEVELOPMENT OF CONTROL ALGORITHMS FOR AUTOPILOT IMPLEMENTATION

4.1 Introduction

In this chapter, the control algorithms necessary to utilize the sensor hardware presented in Chapter 3 will be developed. In section 4.2, a force control law which allows active force accommodation [14] is presented. The control law is identical to the active compliance algorithm used by Klein and Briggs [4], but the control law constants are specified as functions of parameters which predict the response of the entire vehicle. Section 4.3 deals with the problem of force set-point specification. A solution is presented based on the pseudo-inverse algorithm used by Klein and Wahawisan [15]. In section 4.4, the complete control structure of the compliance servo is designed. An algorithm for controlling the body attitude of the Hexapod is developed in section 4.6. These algorithms together constitute the software portion of the autopilot design.

4.2 Force Control Law Design

The problem of designing a control law to utilize the foot force information is complicated by the fact that the Hexapod is operating in a relatively unstructured environment. The control law

must combine force information along with position and velocity information in a manner which enhances the ability of the Hexapod to negotiate rough terrain, even if little is known about a particular terrain point. The control law must therefore be flexible enough to operate successfully on terrain points with a wide range of values for elevation, compliance, coefficient of friction, etc.

The state of a foot shall be defined by vectors describing the position, velocity, and applied force of that foot along each axis of the body coordinate system.

$$\bar{S}_{XA} \equiv [X_A \; \dot{X}_A \; f_{XA}]^T \tag{4.1}$$

$$\bar{S}_{YA} \equiv [Y_A \; \dot{Y}_A \; f_{YA}]^T \tag{4.2}$$

$$\bar{S}_{ZA} \equiv [Z_A \; \dot{Z}_A \; f_{ZA}]^T \tag{4.3}$$

The subscript 'A' denotes an actual value as opposed to a desired value.

The output of the force sensing circuitry is a vector with components along each axis of the foot coordinate system, which is described in Appendix A. In general, the foot coordinate system is physically rotated from the body coordinate system. The force information can, however, be mathematically transformed into the body coordinate system by the relation

$$\bar{F}_B = R(\bar{\theta}) \; \bar{F}_f \tag{4.4}$$

where $\overline{F}_f$ is the force vector expressed in the foot coordinate system, $\overline{F}_B$ is the force vector expressed in the body coordinate system, and $R(\overline{\theta})$ is the required rotation matrix. The rotation matrix is derived in Appendix A.

The control law design is simplified by considering the state of a foot in only one spatial dimension. The z-axis is chosen for discussion; the following development is, however, equally applicable to the x and y axes.

Suppose that a desired state has been specified for a foot, and is given by

$$\overline{S}_{ZD} \equiv [Z_D \ \dot{Z}_D \ f_{ZD}]^T \tag{4.5}$$

where the subscript 'D' denotes a desired value. Since the inner loop of the existing Jacobian servo is a rate loop, it is reasonable to specify a rectilinear velocity error signal e_v such that

$$e_v = k_{pz}(Z_D - Z_A) + k_{vz}(\dot{Z}_D - \dot{Z}_A) + k_{fz}(f_{ZD} - f_{ZA}) \ . \tag{4.6}$$

To simplify analysis of this control law, it shall be assumed that the rate servo loop is ideal, and therefore that the error signal e_v is always zero. The ideal rate servo assumption is used in much of this work. The actual servo model will be included in a later section, and it will be shown that the results obtained using this assumption are valid. Equation 4.6 then reduces to

$$k_{pz}(Z_D - Z_A) + k_{vz}(\dot{Z}_D - \dot{Z}_A) + k_{fz}(f_{ZD} - f_{ZA}) = 0 \ . \tag{4.7}$$

For a given state setpoint $\bar{S}_{ZD}$, equation 4.7 constrains the actual state $\bar{S}_{ZA}$ to lie within a two-dimensional space. This is illustrated in Figure 4.1, where plane I represents this two-dimensional space. The plane passes through point D, which represents the desired state $\bar{S}_{ZD}$. Note that the actual state is not forced to point D, but rather can exist anywhere on plane I.

It is reasonable to expect that the external environment will impose additional constraints on the system. To illustrate, consider that a foot in support phase is in contact with the terrain. The state of a given terrain point can be defined in the same manner as the state of a foot. Remaining in the body coordinate system and using the subscript 'T' to denote a terrain point yields

$$\bar{S}_{ZT} \equiv [Z_T \; \dot{Z}_T \; f_{ZT}]^T \tag{4.8}$$

Any terrain point will exhibit some amount of compliance. In addition, the passive compliance of the hexapod structure can be lumped with the terrain compliance. For the sake of discussion, this compliance will be assumed to be linear, and can be described by

$$f_{ZT} = k_T(Z_T - Z_0) + \alpha_T(\dot{Z}_T) \tag{4.9}$$

where k_T is the terrain spring constant, Z_0 is the terrain height with no force applied, and α_T is the terrain viscous damping constant. Note that for a constant Z_0, the vehicle coordinate system is at a fixed height above the nominal terrain surface, and thus the hexapod mass

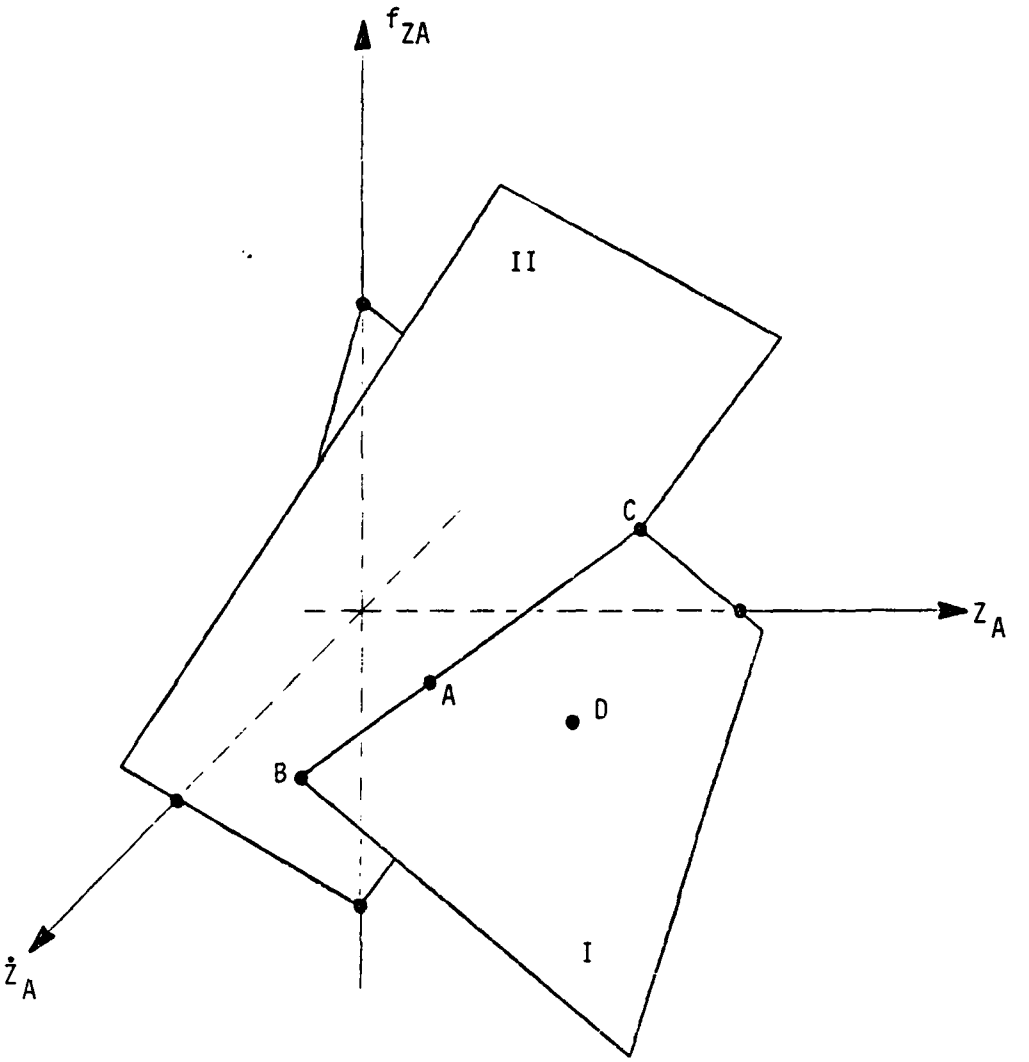


Figure 4.1. State-space Model of the Force Control Problem.

does not undergo acceleration.

When a foot is in support phase, it is assumed to be in firm contact with the terrain; i.e., the relative velocity between the foot and the terrain point is zero. If the foot force f_{ZA} and the ground reaction force f_{ZT} are defined is opposite directions, then their values must be exactly equal. The state of a foot in support phase is thus exactly equal to the state of the supporting terrain point, or

$$\bar{S}_{ZA} = \bar{S}_{ZT} \ . \tag{4.10}$$

Substituting into equation 4.9 then yields

$$f_{ZA} = k_T(Z_A - Z_0) + \alpha_T \dot{Z}_A \ . \tag{4.11}$$

Equation 4.11 is the constraint imposed on the system by the external environment, and is represented by plane II in Figure 4.1. Plane I and plane II intersect at line $\overline{BC}$. The actual system state is therefore constrained to remain within the one-dimensional space represented by $\overline{BC}$ if the constant vehicle height condition is imposed.

Note that the previous discussion considered only one possible constraint on the system. When the foot is not in contact with the terrain, the system is constrained to lie on the plane $f_{ZA} = 0$. The environmental constraints on the system will in general be time-varying and nonlinear.

Additional insight into the control law can be obtained by modeling it as a physical system. Equation 4.7 can be solved for f_{ZA},

giving

$$f_{ZA} = -\frac{k_p}{k_F}\left(Z_A - \left[Z_D + \frac{k_F}{k_p} f_{ZD}\right]\right) - \frac{k_v}{k_F}(\dot{Z}_A - \dot{Z}_D) \quad . \tag{4.12}$$

Comparing this to the expression for a mechanical spring-damper system,

$$f_{ZA} = -k_s(Z_A - Z_0) - \alpha(\dot{Z}_A - \dot{Z}_0) \tag{4.13}$$

it can be seen immediately that

$$s = \frac{k_p}{k_F} \tag{4.14}$$

$$\alpha = \frac{k_v}{k_F} \tag{4.15}$$

$$Z_0 = Z_D + \frac{k_F}{k_p} F_{ZD} \tag{4.16}$$

$$\dot{Z}_0 = \dot{Z}_D \quad . \tag{4.17}$$

Figure 4.2 shows a model of equation 4.13 if Z_0 is defined as

$$Z_0 \equiv Z_\ell + \ell_0 \tag{4.18}$$

where $Z_\ell \equiv$ length of the sliding links L_1 and L_2,

$\ell_0 \equiv$ relaxed length of the spring-damper system.

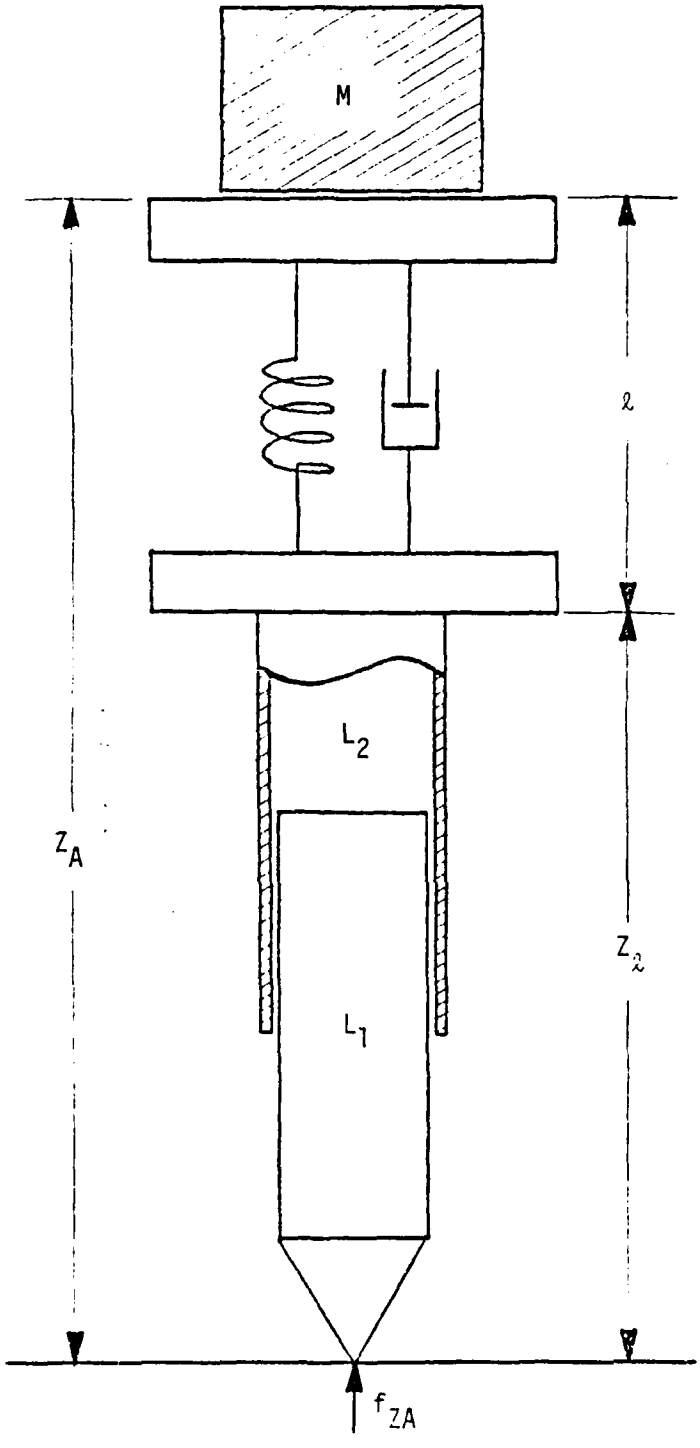


Figure 4.2. Physical Model of the Force Control Law.

Note that the block 'M' is not part of this model, but is included for later use. The unfamiliar $\dot{Z}_0$ term in equation 4.13 is due to the sliding joint between links L_1 and L_2 which allows the overall length to be adjusted without exercising the spring-damper system. Note that ℓ_0 is a constant, giving

$$\dot{Z}_0 = \dot{Z}_\ell \quad . \tag{4.19}$$

from equation 4.18. Thus, if L_1 and L_2 comprise a rigid link, $\dot{Z}_0$ is seen to be zero and equation 4.13 reduces to standard form.

The physical system model of Figure 4.2 is helpful in understanding the characteristics of the force control law, and will be referred to in later sections. It also illustrates why the control law of equation 4.6 is referred to as an 'active compliance' algorithm.

Before investigating the effects of a time-varying state set-point $\overline{S}_{ZD}$, it must be emphasized that the desired velocity $\dot{Z}_D$ is not necessarily equal to the derivative of the desired position Z_D. To avoid notation problems, the temporary change of variables

$$r_D \equiv \dot{Z}_D \equiv \left(\frac{dZ}{dt}\right)_D \tag{4.20}$$

shall be made.

Consider the case where $r_D = 0$ and Z_D varies. The $\dot{Z}_D$ term in equation 4.12 is then zero, and the equation has the form of a simple spring-damper system. With reference to the physical model, this means that the sliding joint is locked. Thus any change in Z_A is

effected by the spring-damper system only.

Next consider the case where the desired rate is equal to the derivative of the desired position, or

$$r_D = \frac{d}{dt}(Z_D) \ . \tag{4.21}$$

If the force error is assumed to be constant and the change of variables is made, equation 4.7 reduces to

$$k_p(Z_A - Z_D) + k_v(\dot{Z}_A - r_D) = C_1 \tag{4.22}$$

where C_1 is a constant. Substituting equation 4.21 into 4.22 gives

$$k_p\left(\int_{t_0}^{t} \dot{Z}_A dt - \int_{t_0}^{t} r_D dt\right) + k_v(\dot{Z}_A - r_D) = C_2 \tag{4.23}$$

where C_2 is constant and includes the position terms at time t_0. If the velocity error term at time t_0 is zero, then it is seen that

$$C_2 = 0 \tag{4.24}$$

and

$$\dot{Z}_A = r_D \ , \qquad t > t_0 \ . \tag{4.25}$$

In words, the actual rate is exactly equal to the desired rate if the following conditions are met: the desired rate is the derivative of the desired position, the force error is constant, and the initial rate error is zero. It is seen intuitively that for a non-zero initial rate error, the rate error approaches zero exponentially.

It is informative to interpret the implications of equation 4.21 in terms of the physical model. Reverting to the original notation, equation 4.25 becomes

$$\dot{Z}_A = \dot{Z}_D \ . \tag{4.26}$$

Equations 4.17 and 4.19 may be used to write equation 4.26 in terms of the physical model parameters, giving

$$\dot{Z}_A = \dot{Z}_\ell \ . \tag{4.27}$$

Referring to Figure 4.2, it is seen that

$$Z_A = Z_\ell + \ell \ , \tag{4.28}$$

giving

$$\dot{Z}_A = \dot{Z}_\ell + \dot{\ell} \ . \tag{4.29}$$

Combining 4.27 and 4.29 gives

$$\dot{\ell} = 0 \ . \tag{4.30}$$

Thus, all system motion occurs at the sliding joint and the spring-damper system is not excited given that equation 4.21 is imposed, the force error is constant, and the initial velocity error is zero.

The force control law design can now be completed by specifying the constants k_p, k_v, and k_F in terms of desired system parameters. For the purposes of vehicle suspension design, the obvious ones are

the spring constant k_s and the system damping ratio ζ. The approach used to solve for the constants is to write the system dynamic equations using the physical model and then transform between physical model parameters and control law constants. To study the system dynamics, the mass of the hexapod supported by the leg must be included in the physical model, and is represented by the block 'M' in Figure 4.2. For the purpose of studying system dynamics, the sliding joint is assumed to be locked.

Applying the laws of physics to Figure 4.2, one can obtain

$$-f_{ZA} = -M(\ddot{Z}_A - g) \tag{4.31}$$

where 'g' is the acceleration of gravity. Summing all forces on the sliding link assembly gives

$$-f_{ZA} - k_s(Z_A - Z_0) - \alpha(\dot{Z}_A - \dot{Z}_0) = 0 \ . \tag{4.32}$$

Combining equations 4.31 and 4.32 gives

$$k_s(Z_A - Z_0) + \alpha(\dot{Z}_A - \dot{Z}_0) + M(\ddot{Z}_A - g) = 0 \ . \tag{4.33}$$

With the sliding joint locked, $\dot{Z}_0$ is zero and Z_0 is constant, as is the acceleration of gravity. Thus, the homogeneous part of equation 4.33 is

$$k_s Z_A + \alpha \dot{Z}_A + M \ddot{Z}_A = 0 \ . \tag{4.34}$$

Using differential operator notation on equation 4.34 gives

$$(D^2 + \frac{\alpha}{M} D + \frac{k_s}{M}) Z_A = 0 \qquad (4.35)$$

Comparing this to a general second-order system

$$(D^2 + 2\zeta\omega_n D + \omega_n^2) Z_A = 0 \ , \qquad (4.36)$$

it is apparent that

$$\omega_n = \sqrt{\frac{k_s}{M}} \qquad (4.37)$$

and

$$2\zeta\omega_n = \frac{\alpha}{M} \ . \qquad (4.38)$$

Combining equations 4.37 and 4.38 gives

$$\zeta = \frac{\alpha}{2\sqrt{k_s M}} \qquad (4.39)$$

or

$$1/\alpha = \frac{1}{2\zeta\sqrt{k_s M}} \ . \qquad (4.40)$$

The physical model parameters can now be replaced with control law constants. Substituting equation 4.15 into 4.40 results in

$$\frac{k_F}{k_v} = \frac{1}{2\zeta\sqrt{k_s M}} \ . \qquad (4.41)$$

Substituting equation 4.14 into 4.41 gives

$$\frac{k_p/k_s}{k_v} = \frac{1}{2\zeta\sqrt{k_s M}} \quad , \tag{4.42}$$

or

$$\frac{k_p}{k_v} = \frac{1}{2\zeta}\sqrt{k_s/M} \quad . \tag{4.43}$$

Since the control law of equation 4.6 includes three constants with which to specify two independent parameters, one of the constants can be specified arbitrarily. Thus, the following condition is imposed.

$$k_v = 1 \tag{4.44}$$

Equations 4.41, 4.43, and 4.44 complete the control law design.

The most general requirement for a control law utilizing force feedback was that it must improve the rough terrain capability of the hexapod. The linear control law just developed is expected to provide benefits similar to those of ordinary passive suspensions; in particular, it causes all legs in support phase to maintain contact with the ground and support a load approximately equal to a force setpoint specified by a higher level control algorithm. The control law is expected to have several advantages over a passive system. First, the spring stiffness and damping can be changed with no hardware modifications, or could even be changed dynamically. Secondly, the particular leg geometry used by a vehicle does not affect the algorithm, since

the algorithm is defined in body coordinates. Thirdly, the major compliant axis could, through the use of another rotation matrix, be maintained parallel to the gravitational vector at all times. This would allow the use of "soft" spring constants even with the body at extreme pitch and roll angles without affecting the vehicle's stability margin.

4.3 Force Setpoint Specification

Prior to implementing the control law of equation 4.6, the desired system state $\overline{S}_D$ must be defined. The parameters Z_D and $\dot{Z}_D$ are specified by the foot trajectory planning algorithm and by the body attitude regulation algorithm of section 4.5. It remains to specify the force setpoint $\overline{F}_D$.

For the purpose of force setpoint specification, the terrain is modeled as a flat, non-compliant surface. The problem then reduces to the determination of ground reaction forces which cause the vehicle body to maintain static equilibrium. Throughout this work, the vehicle body is assumed to be perpendicular to the gravitational vector, i.e., gravitational loading causes force components parallel to the body coordinate z-axis only. Since acclerations are quite small on this vehicle, zero force setpoints can be specified along the x and y components of the body coordinate system when active compliance is enabled, giving

$$f_{XD} = 0 \tag{4.45}$$

and

$$f_{YD} = 0 \ . \tag{4.46}$$

In addition, legs in transfer phase are not in contact with the ground, giving

$$f_{ZD} = 0 \quad , \quad \text{transfer phase} \ . \tag{4.47}$$

For the vehicle body to maintain static equilibrium, three conditions must be satisfied.

1. Sum of moments about the x-axis must equal zero.
2. Sum of moments about the y-axis must equal zero.
3. Sum of vertical ground reaction forces must equal the total weight of the vehicle.

Expressed as equations, these conditions become

$$\sum_i M_x = 0 \quad , \tag{4.48}$$

$$\sum_i M_y = 0 \quad , \tag{4.49}$$

and

$$\sum_i f_z = F_{total} \quad , \tag{4.50}$$

for leg i in support phase. There may be up to six legs in support phase, so in general the above system of equations is underspecified. A pseudo-inverse solution to these equations may be used to optimize the setpoints with respect to a minimum sum of squares of force. This

type of solution has been used previously by Klein and Wahawisan, who solved the system of equations numerically in a real-time computer program. In Appendix B, it is shown that a closed-form analytic solution can be found for the system. The solution is given by equations B.17, B.18, B.20, and B.26.

4.4 Force Control Law Implementation

In implementing the force control law, it is desired that as much as possible of the Jacobian control structure which is currently in use be retained. Figure 4.3 is a block diagram which implements the force control law in body coordinates and then uses a Jacobian control structure to convert between body and joint coordinates. The diagram shows the entire control structure for one leg. The parameters are defined in Table 2.1 and Table 4.1.

The implementation of Figure 4.3 satisfies the control law, but requires the use of the forward Jacobian matrix, resulting in increased complexity. However, the diagram can be reduced to the form of Figure 4.4 by making the following observations. Recall that when solving the control law for k_p, k_f, and k_v, it was chosen to set $k_v = 1$. Thus,

$$K_v = I \text{ , and}$$

$$J^{-1}(\theta)K_v\, J(\theta)\dot{\bar{e}} = J^{-1}(\theta)I\, J(\theta)\dot{\bar{e}} = \dot{\bar{e}} \text{ .} \qquad (4.51)$$

This allows the rate feedback loop to be reduced as shown in Figure 4.4. Comparison of this figure with the existing structure (Figure 2.8) shows

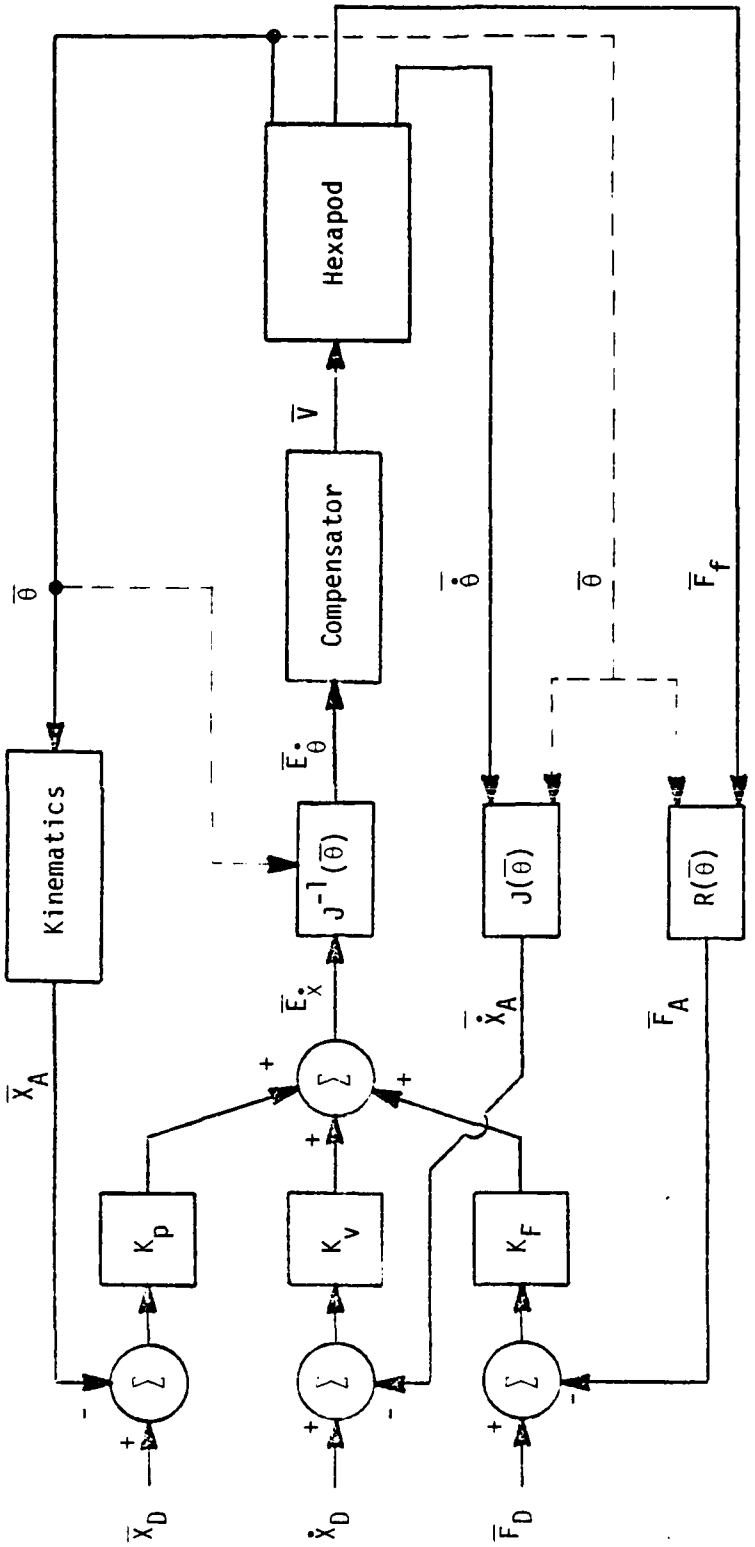


Figure 4.3. Full Jacobian Structure Implementing the Force Control Law.

Table 4.1

DEFINITIONS OF ADDITIONAL CONTROL PARAMETERS
FOR FORCE CONTROL LAW IMPLEMENTATION

$\bar{F}_D \equiv [f_{XD}\ f_{YD}\ f_{ZD}]^T$ (desired force expressed in body coordinates)

$\bar{F}_A \equiv [f_{XA}\ f_{YA}\ f_{ZA}]^T$ (actual force expressed in body coordinates)

$\bar{F}_f \equiv [f_{Xf}\ f_{Yf}\ f_{Zf}]^T$ (actual force expressed in foot coordinates)

$\bar{E}_{\dot{x}} \equiv [e_x\ \ e_y\ \ e_z\ ]^T$ (rectilinear rate error expressed in body coordinates)

$$K_V \equiv \begin{bmatrix} k_{vx} & 0 & 0 \\ 0 & k_{vy} & 0 \\ 0 & 0 & k_{vz} \end{bmatrix}$$ (rate gain matrix)

$$K_F \equiv \begin{bmatrix} k_{fx} & 0 & 0 \\ 0 & k_{fy} & 0 \\ 0 & 0 & k_{fz} \end{bmatrix}$$ (force gain matrix)

$R(\bar{\theta})$ ≡ foot force rotation matrix.

Compensator ≡ a functional block specifying the relationship between joint rate error and actuator input voltage.

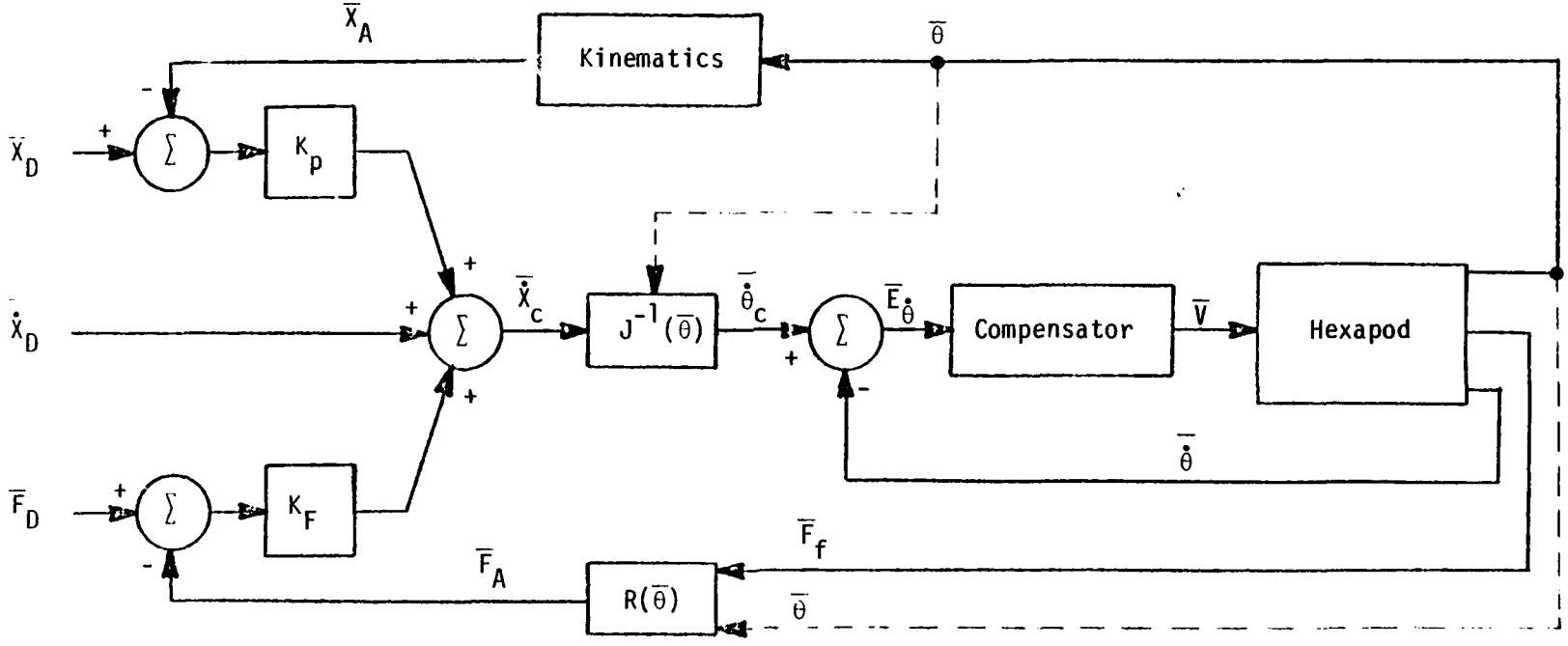


Figure 4.4. Reduced Jacobian Structure Implementing the Force Control Law (Compliance Servo).

that the force control law has been implemented with the simple addition of a force feedback loop to the existing structure. The parameters are defined as in Table 2.1 and Table 4.1.

4.5 Non-Ideal Servo Effects and Compensator Design

To study the effects of a non-ideal servo on system performance, the control block diagram of Figure 4.4 can be reduced to a single spatial dimension and linearized about an operating point. The operating point chosen is the normal or "square" position, which is shown in Figure A.1. In this position, the motion along each Cartesian axis is controlled primarily by a single joint actuator. Foot motion along the body coordinate x-axis is completely decoupled in the normal position, thus the simplified control model uses this axis only.

Referring to Figure A.1, it is seen that

$$X = (\ell_1 + \ell_4) \sin \psi \tag{4.52}$$

where X is the distance from the origin of the hip coordinate system to the foot along the body coordinate x-axis, and the offset ℓ_3 has been neglected. If ℓ is defined as

$$\ell \equiv \ell_1 + \ell_4 \ , \tag{4.53}$$

linearizing the system yields

$$X = \ell\psi \tag{4.54}$$

and

$$\dot{X} = \ell\dot{\psi} \ . \tag{4.55}$$

With respect to Figure 4.4, the kinematic equations reduce to simple multiplication by ℓ, and the inverse Jacobian matrix becomes the scalar $1/\ell$. The force feedback loop is eliminated, since the relationship between input voltage and applied force is not well defined. The reduced model is intended to study system dynamics only; the reference inputs may therefore be eliminated. The hexapod itself is reduced to the single ψ-axis joint actuator. A simple linear transfer function which has been found to give accurate predictions of actuator performance is

$$\frac{\psi(D)}{V(D)} = \frac{.14}{D(D+3)} \ . \tag{4.56}$$

When all of the preceeding simplifications are made to Figure 4.4, the block diagram of Figure 4.5 results.

The control system external to the actual actuator hardware is implemented on a digital computer; the system is therefore a discrete time system. It shall be assumed, however, to be a continuous-time system for the purpose of analysis. To insure the validity of this assumption, all time constants will be chosen to be at least four times as long as the time between servo computations. The control program executes at approximately 40 Hz, thus the time constants are to be chosen such that

$$\tau \geq 1/10 \text{ sec} \ . \tag{4.57}$$

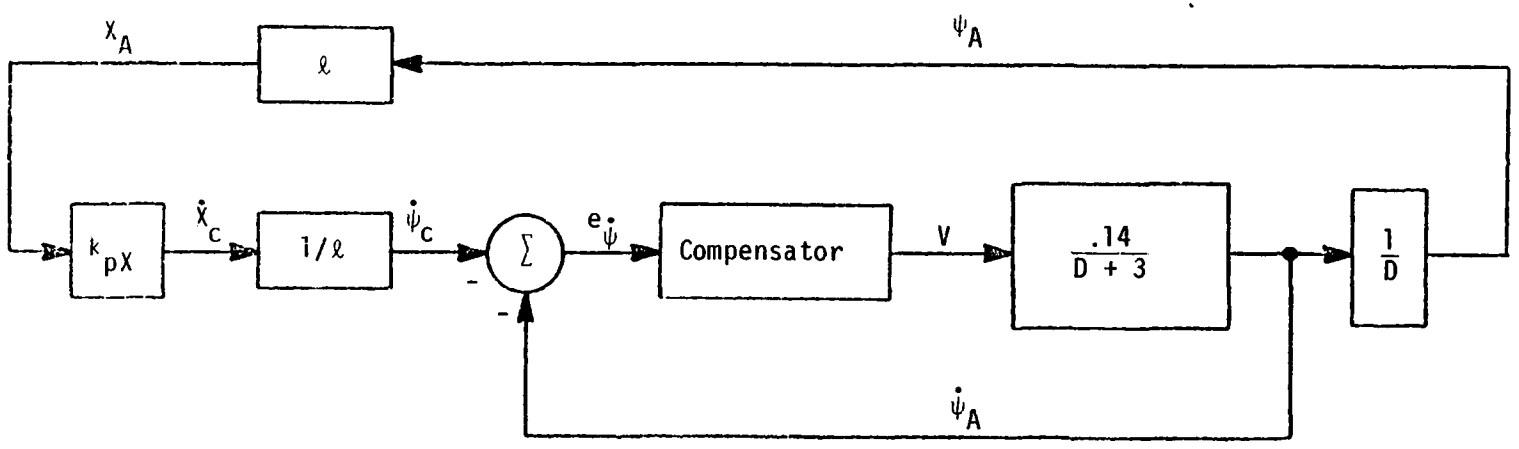


Figure 4.5. One-dimensional Linearized Control Structure.

The compensator block provides opportunities to minimize the effects of servo nonlinearities and disturbance inputs. Klein and Briggs [4], for example, used "variable structure system" control to eliminate motor stalling. A linear gain, however, is known to result in satisfactory performance and is therefore chosen for the sake of simplicity. When a compensator gain G_1 is substituted for the compensator block, Figure 4.5 can be reduced to the form shown in Figure 4.6, which shall be used for the system analysis. If the substitution

$$G = .14\ G_1 \tag{4.58}$$

is made, the system can be described mathematically by

$$\begin{bmatrix} \dot{\psi}_A \\ \ddot{\psi}_A \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_p G & -3-G \end{bmatrix} \begin{bmatrix} \psi_A \\ \dot{\psi}_A \end{bmatrix} \tag{4.59}$$

The eigenvalues of the system are the roots of the characteristic equation

$$\lambda^2 + (3+G)\lambda + Gk_p = 0\ . \tag{4.60}$$

Since k_p is already specified by equation 4.43, only one eigenvalue can be assigned by specifying a value for G. The associated time constant should be as small as possible to avoid invalidating the ideal servo assumption. The system should thus have two real eigenvalues, giving a characteristic equation of the form

$$(\lambda+a)(\lambda+b) = 0 \tag{4.61}$$

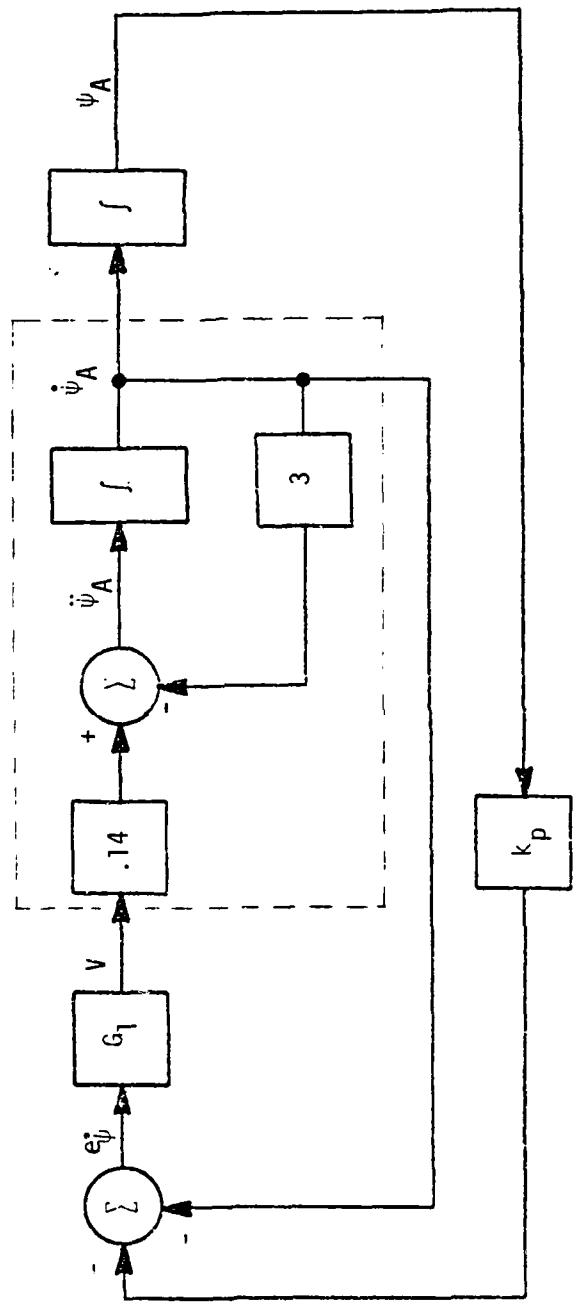
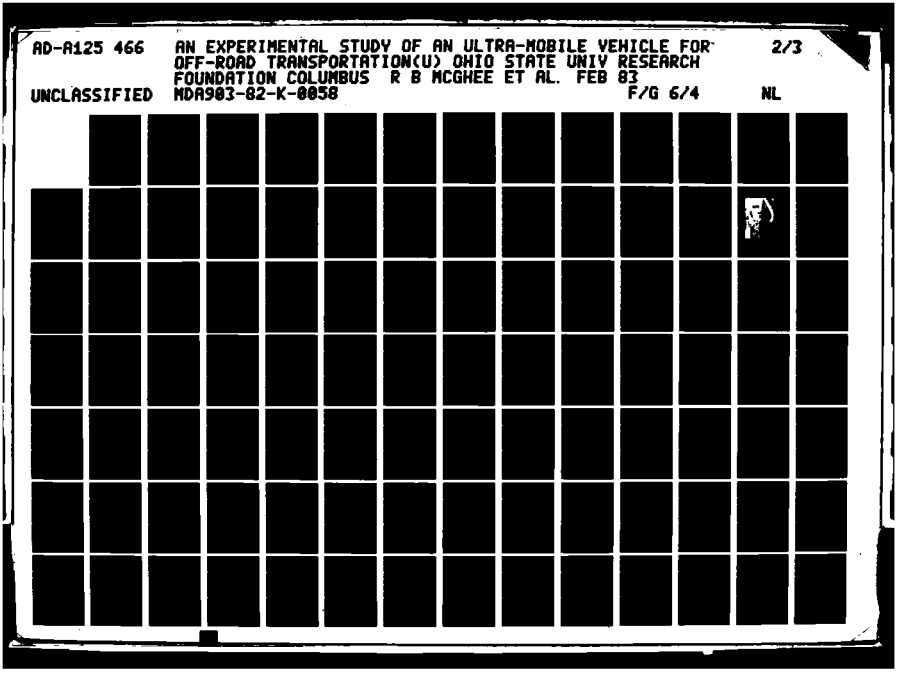
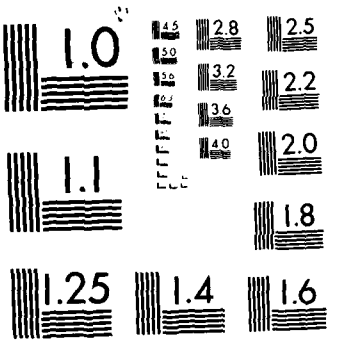


Figure 4.6. One-dimensional Control Structure with Simple Compensator.

AD-A125 466 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. FEB 83 2/3

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL





or

$$\lambda^2 + (a+b)\lambda + ab = 0 \ . \tag{4.62}$$

Equating the coefficients of equations 4.60 and 4.62 gives the system of equations

$$3+G = a+b \tag{4.63}$$

and

$$Gk_p = ab \ . \tag{4.64}$$

Solving equations 4.63 and 4.64 yields

$$a = \frac{k_p(b-3)}{b-k_p} \tag{4.65}$$

and

$$G = \frac{ab}{k_p} \ . \tag{4.66}$$

For purposes of comparison, consider again the ideal servo assumption. If the inner rate loop of Figure 4.5 were ideal, then the actual velocity would equal the commanded velocity, or

$$\dot{\psi}_A = \dot{\psi}_C \ . \tag{4.67}$$

From the block diagram, equation 4.67 results in

$$\dot{\psi}_A = -k_p\psi_A \ . \tag{4.68}$$

Solving equation 4.68 yields the ideal system response

$$\psi_A = \psi_0 e^{-k_p t} . \tag{4.69}$$

Thus, the ideal system is seen to be first order with an eigenvalue at

$$\lambda_{ideal} = -k_p . \tag{4.70}$$

This eigenvalue has a special significance with respect to the active compliance algorithm. Consider equation 4.12, which is the force control law of equation 4.7 rearranged to describe a spring-damper system. The homogeneous response may be obtained by solving the equation with zero setpoints and zero actual force, giving

$$0 = - \frac{k_p}{k_F} Z_A - \frac{k_v}{k_F} \dot{Z}_A \tag{4.71}$$

or

$$\dot{Z}_A = - \frac{k_p}{k_v} Z_A . \tag{4.72}$$

Making use of equation 4.44, the solution of 4.72 is

$$Z_A = Z_0 e^{-k_p t} . \tag{4.73}$$

Thus, the ideal system pole gives the homogeneous response of the spring-damper system.

The actual system eigenvalues can now be compared with the ideal system eigenvalue. From equation 4.61, the actual system has eigenvalues at

$$\lambda_1 = -a \tag{4.74}$$

and

$$\lambda_2 = -b \ . \tag{4.75}$$

Taking the limit of equation 4.65 results in

$$\lim_{b\to\infty} a = k_p \tag{4.76}$$

or

$$\lim_{b\to\infty} \lambda_1 = -k_p \ . \tag{4.77}$$

Comparing equations 4.70 and 4.77, it is seen that if the second eigenvalue λ_2 is very large, the actual system response is the same as the ideal system response. The value of λ_2 is limited by equation 4.57, however, giving

$$\lambda_2 = -\frac{1}{\tau_{min}} = -10 \tag{4.78}$$

or

$$b = 10 \ . \tag{4.79}$$

There now exist analytic expressions for all system gains as functions of the spring stiffness and damping parameters k_s and ζ.

The values used to obtain the experimental results presented in Chapter 5 are

$$k_s = 8 \text{ lb/in} \tag{4.80}$$

and

$$\zeta = 2 \ . \tag{4.81}$$

The largest mass which will be supported by one leg is 1/2 of the total mass of the hexapod. The effective mass is then

$$M = \frac{285 \text{ lbm}}{2} = .368 \ \frac{\text{lbf sec}^2}{\text{in}} \ . \tag{4.82}$$

Substituting these values into equations 4.41, 4.43, and 4.44 yields

$$k_F = .146 \tag{4.83}$$

and

$$k_p = 1.166 \ . \tag{4.84}$$

The system pole λ_1 associated with the spring damping is given by equations 4.74, 4.65, and 4.79 as

$$\lambda_1 = -.924 \ . \tag{4.85}$$

Comparing this value with equation 4.70 shows that there exists a twenty percent difference in the eigenvalues predicted by the ideal and the non-ideal servo models. Thus, the force control law is expected to approximate a spring-damper system with reasonable accuracy. If

more accuracy is needed, a compensator design using integral error feedback can be employed which reduces the eigenvalue discrepancy to under five percent. This compensator, however, has not yet been tested experimentally.

To finish the design, the system forward gain is obtained from equation 4.66 as

$$G = 7.92 \ . \tag{4.86}$$

The compensator gain is obtained from equation 4.58, giving

$$G_1 = 56.6 \ . \tag{4.87}$$

The constants specified in equations 4.83 and 4.84 are for use as z-axis parameters. The x and y axis parameters are determined similarly. The compensator gain of equation 4.87 is, however, used for all actuators.

4.6 Attitude Control

The primary function of the active compliance algorithm is to provide a suspension system for the hexapod; it causes all legs in support phase to maintain contact with an uneven surface and to support a proportionate amount of the vehicle weight. To insure that the vehicle can accommodate large terrain height variations under the supporting feet, fairly "soft" spring constants were chosen. This, however, causes the vehicle body attitude to be less restrained than is desirable, i.e., the body can pitch and roll excessively. Even

with "firm" spring constants, the terrain itself could cause the vehicle to enter an undesirable attitude. To avoid those problems and to generally increase the control capabilities of the hexapod, an algorithm for attitude control is seen to be desirable.

In Chapter 3, hardware was implemented which measures the angular displacement of the vehicle body from the vertical with respect to both the pitch and roll axes. As was stated earlier, the vehicle body is assumed to be maintained in a level attitude through this work. It is therefore desired that the attitude control system maintain the pitch and roll displacements as close as possible to zero. To accomplish this, a linear feedback control law may be implemented with a zero reference input. The attitude error is then simply the output of the vertical sensor. The mechanism by which body attitude is changed is the displacement of feet in support phase with respect to the body coordinate system. Figure 4.7 shows the relationship between foot displacement and body attitude in two dimensions. The actual body pitch, γ_A, is defined as the body angular displacement (positive counterclockwise) from the vertical about the positive y axis and is expressed in radians. Figure A shows the vehicle with an attitude error, while Figure B shows the vehicle after the attitude has been corrected. The z-axis position displacement of leg i necessary to accomplish the correction is

$$\Delta Z_i = Z_i' - Z_i = [Z_i - X_i \sin(\phi_2 - \phi_1)] - Z_i \qquad (4.88)$$

or

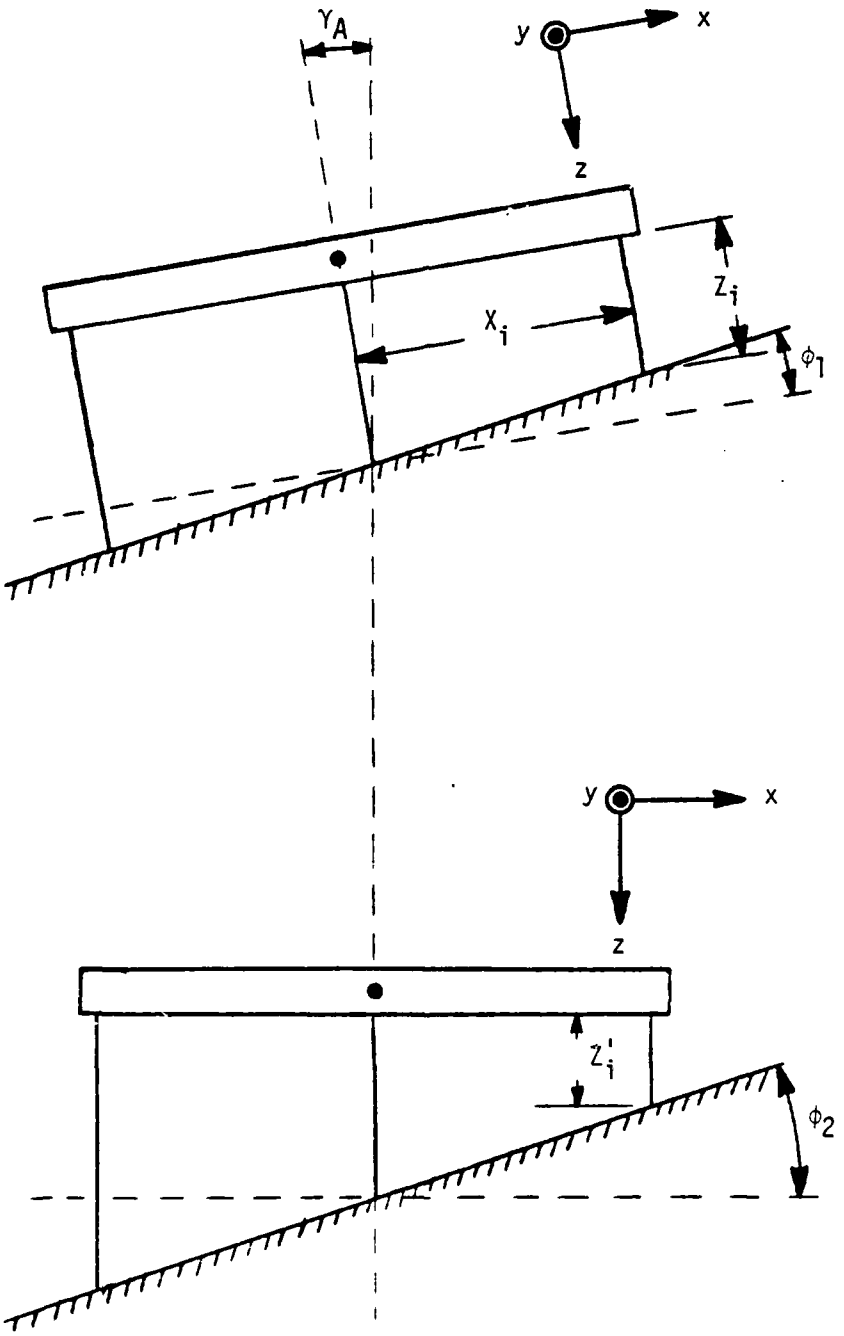


Figure 4.7. Relationship between Foot Position and Body Attitude.

$$\Delta Z_i = -X_i \sin(\phi_2 - \phi_1) \ . \tag{4.89}$$

where ϕ is defined as the angle between the vehicle body and the terrain. The term $(\phi_2 - \phi_1)$ is seen to have the same value as γ_A in Figure A. Since the feedback control will be forcing γ_A to be near zero, the approximation

$$\sin \gamma_A \approx \gamma_A \tag{4.90}$$

may be used. Then the relation

$$\Delta Z_i = -X_i \gamma_A \tag{4.91}$$

can be used in the control algorithm to convert from attitude error to position displacement.

The actual body roll, δ_A, is now defined as the body angular displacement (positive counterclockwise) from the vertical about the positive x axis. For a body roll error, the foot displacement necessary for correction is found to be

$$\Delta Z_i = Y_i \delta_A \ . \tag{4.92}$$

In general, the vehicle may exhibit pitch and roll errors simultaneously. It can be shown that for small errors, the foot correction necessary for an error about a given axis is independent of the error about the other axis, i.e., the axes are decoupled. The general expression for foot displacement is then

$$\Delta Z_i = -X_i \gamma_A + Y_i \delta_A \ . \tag{4.93}$$

It should be noted that a change in the vehicle body attitude necessitates a change not only in the z-coordinate of the foot, but also in the x and y coordinates. Since the attitude is expected to remain almost constant, however, these x and y displacements are assumed to be zero. If it were desired to implement variable attitude setpoints, a rotation matrix could be found which would give exact values for displacements in all dimensions. Equation 4.93 is actually a small-signal linearization of that matrix.

The required foot motions can be effected by varying the reference inputs to the compliance servo. Referring to the physical model of Figure 4.2, it is seen that a straightforward mechanism for implementing the attitude correction term, ΔZ_i, is to simply adjust the sliding joint without directly invoking any spring-damper response. Recall that this is achieved by specifying the reference inputs such that

$$\dot{Z}_D = \frac{d}{dt}(Z_D) \ . \tag{4.94}$$

In a hexapod control scheme without attitude feedback, the compliance servo inputs are specified directly by the foot trajectory planning algorithm; these inputs are

$$Z_D = Z_D^T \tag{4.95}$$

and

$$\dot{Z}_D = \dot{Z}_D^T \ . \tag{4.96}$$

where the parameters with the superscript 'T' are the outputs of the foot trajectory algorithm and satisfy equation 4.94. If attitude control is to be implemented, the control loop must be able to access the compliance servo inputs. The following equations provide that access.

$$Z_D = Z_D^T + Z_D^A \tag{4.97}$$

$$\dot{Z}_D = \dot{Z}_D^T + \dot{Z}_D^A \ . \tag{4.98}$$

Here the superscript 'A' denotes parameters generated by the attitude control algorithm.

To close the attitude control loop, the parameters Z_D^A and $\dot{Z}_D^A$ must be specified in terms of the attitude correction displacement ΔZ_i, and they must be specified such that they satisfy equation 4.94. If ideal rate servos are assumed, then the compliance servo for each leg is ideal, and therefore the entire vehicle can be modeled as an ideal system, such that

$$\dot{\gamma}_A = \dot{\gamma}_D \tag{4.99}$$

and

$$\dot{\delta}_A = \dot{\delta}_D \ . \tag{4.100}$$

The control laws

$$\dot{\gamma}_D = -k_A \gamma_A \tag{4.101}$$

$$\dot{\delta}_D = -k_A \delta_A \tag{4.102}$$

then cause the pure exponential responses

$$\gamma_A = \gamma_0 e^{-k_A(t-t_0)} \tag{4.103}$$

and

$$\delta_A = \delta_0 e^{-k_A(t-t_0)} \tag{4.104}$$

if the hexapod mass is neglected, and the system time constant is seen to be

$$\tau_A = 1/k_A \ . \tag{4.105}$$

The control laws of equations 4.101 and 4.102 cannot be implemented as written, however, since there exists no direct attitude inputs. Differentiating equation 4.93 gives

$$\Delta\dot{Z}_i = -X_i\dot{\gamma}_A + Y_i\dot{\delta}_A \ . \tag{4.106}$$

Since this relation was obtained from geometrical considerations only, it is valid as a transformation between desired rates as well as actual rates. Then

$$\dot{Z}_D^A = -X_i\dot{\gamma}_D + Y_i\dot{\delta}_D \ . \tag{4.107}$$

Substituting equations 4.101 and 4.102 into 4.107 gives

$$\dot{Z}_D^A = X_i k_A \gamma - Y_i k_A \delta \tag{4.108}$$

or

$$\dot{Z}_D^A = k_A(X_i\gamma - Y_i\delta) \ . \tag{4.109}$$

Equation 4.109 specifies the foot rate required to implement the control laws 4.101 and 4.102. The position input to the compliance servo can be obtained by numerically integrating the rate input, giving

$$Z_D^A = \int_{t_s}^{t} \dot{Z}_D^A \, dt \tag{4.110}$$

where t_s is the time at which the foot entered support phase. Equation 4.110 insures that equation 4.94 is satisfied, and therefore the active spring damping is not invoked.

To complete the design, a value for the attitude system pole k_A must be chosen. Recall that the compliance servo has two poles, one of which provides the active spring damping, and another which was assigned at 10 radians/second. The attitude control loop does not excite the pole associated with the damping due to the fact that equation 4.94 is satisfied. For the ideal servo assumption to be valid, however, the attitude control pole must be smaller than the other servo pole. Fast response is desired of the attitude control loop, so the attitude pole is specified at

$$k_A = 8 \ . \tag{4.111}$$

A block diagram of the attitude control system is shown in Figure 4.8.

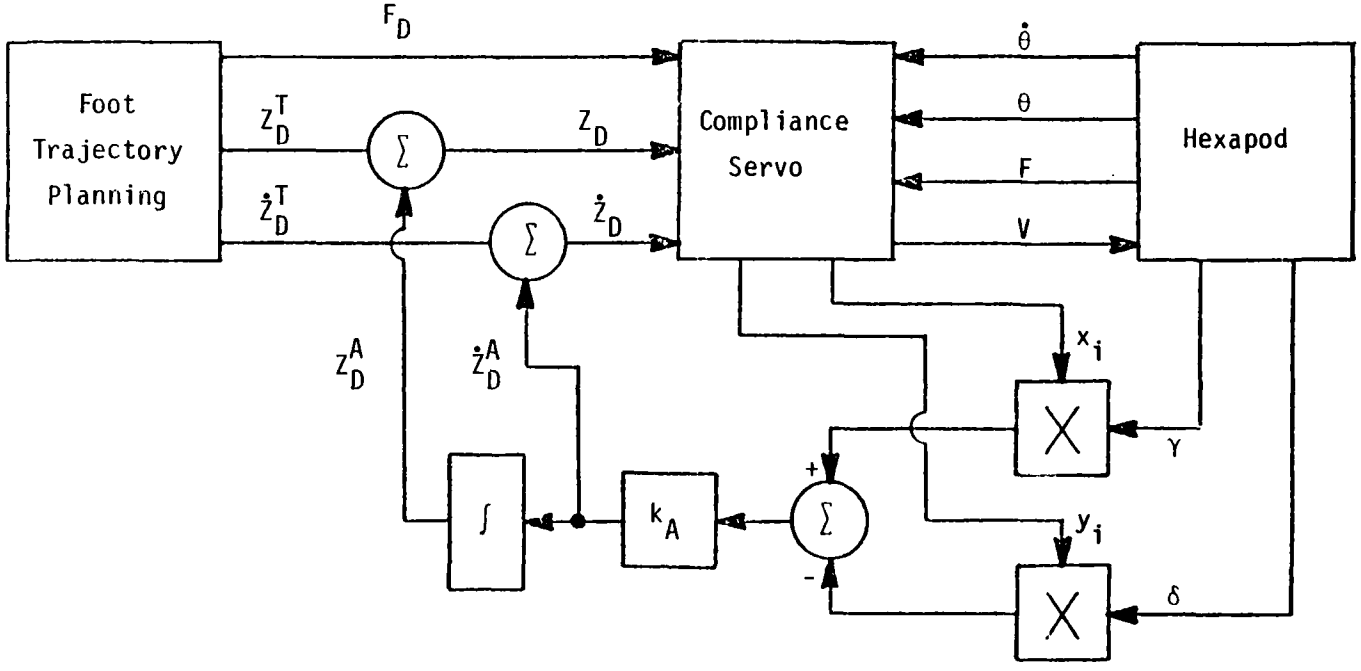


Figure 4.8. Implementation of the Attitude Control System.

Chapter 5

EXPERIMENTAL RESULTS OF AUTOPILOT PERFORMANCE

5.1 Introduction

Experimental results are presented in this chapter which document the performance of the control algorithms developed in Chapter 4. Modifications, based on experimental results, to the basic control algorithms are also explained and documented. The algorithms are tested both individually and in combination.

Hexapod Control Program version 3.0 was used as the base-line program for this work. The force accommodation and attitude control algorithms were added to the program, with software switches allowing them to be bypassed if desired. The resulting program is Hexapod Control Program version 3.4. Appendix C is a listing of this program.

The primary form of documentation in this chapter is plots of various vehicle parameters which indicate overall performance. To obtain these plots, additional code was added to version 3.4 which directed the writing of vehicle parameters to magnetic disk. Data acquisition occurred in real-time during Hexapod locomotion. The data was then retrieved from disk, processed if necessary, and output on a plotter interfaced to the PDP-11/70 control computer.

5.2 Active Compliance

The compliance servo, shown in Figure 4.4, was slightly modified prior to implementation in Hexapod Control Program version 3.4. In the event of force transducer failure, the unmodified control structure could lead to a position offset of about 15 inches. To avoid this potential problem, a limiter was added which clips the force error at a value which results in a four inch position offset. When the system is operating with normal force distributions, the force error is small, and the limiter does not affect the system at all. When, however, a foot contacts the ground at the end of transfer phase, a large amplitude pulse of measured force can result. The limiter keeps this pulse from exciting the system unnecessarily.

The compliance servo was implemented with provision for enabling active compliance along each of the three body axes, as described in Chapter 4. The Hexapod feet, however, exhibit a large amount of passive compliance along the x and y axes, due primarily to the compliance of the Hexapod frame. To make the experimental results easier to interpret, active compliance was enabled along the body z-axis only.

The first on-ground test of the active compliance algorithm was performed with the Hexapod stationary; the control loop stability was tested, as was the response to external force inputs. When a force was applied, the vehicle did exhibit active compliance. However, the interaction between the force feedback and the passively compliant frame of the vehicle resulted in small oscillations of the joint actuators and of the vehicle frame. The oscillation frequency was approximately

the resonant frequency of the frame. To eliminate these oscillations, it was decided to include a first-order, low-pass filter in the force feedback loop. The filter break frequency was determined experimentally by starting with a large break frequency and then reducing it until the oscillations were eliminated. The value required was found to be two radians per second. A block diagram of the compliance servo which includes force error limiting and filtering is shown in Figure 5.1.

Before evaluating the performance of the compliance servo, it was desired to establish a base-line servo performance reference for purposes of comparison. The z-axis position accuracy of a foot using the Jacobian servo routine of Figure 2.8 supplies such a reference. Figure 5.2 shows the results obtained while cycling the legs in the air with only the leg mass loading the joint actuators. The plot shows a slight amount of phase lag between desired and actual positions, and also shows a steady-state position error of approximately 0.4 inches due to motor stalling. The exponential decays following the half-sine wave trajectories occur when the desired position reaches zero, at which time the desired velocity command is set to zero, resulting in a heavily damped decay.

The actual system response can now be compared to the ideal system response predicted in Chapter 4. To accomplish this, a computer modeling routine was written which simulates the compliance servo, modeling the Hexapod as an ideal system. A block diagram of the modeling routine is shown in Figure 5.3. The inputs to the model are the inputs to the real system, as recorded on magnetic disk. The

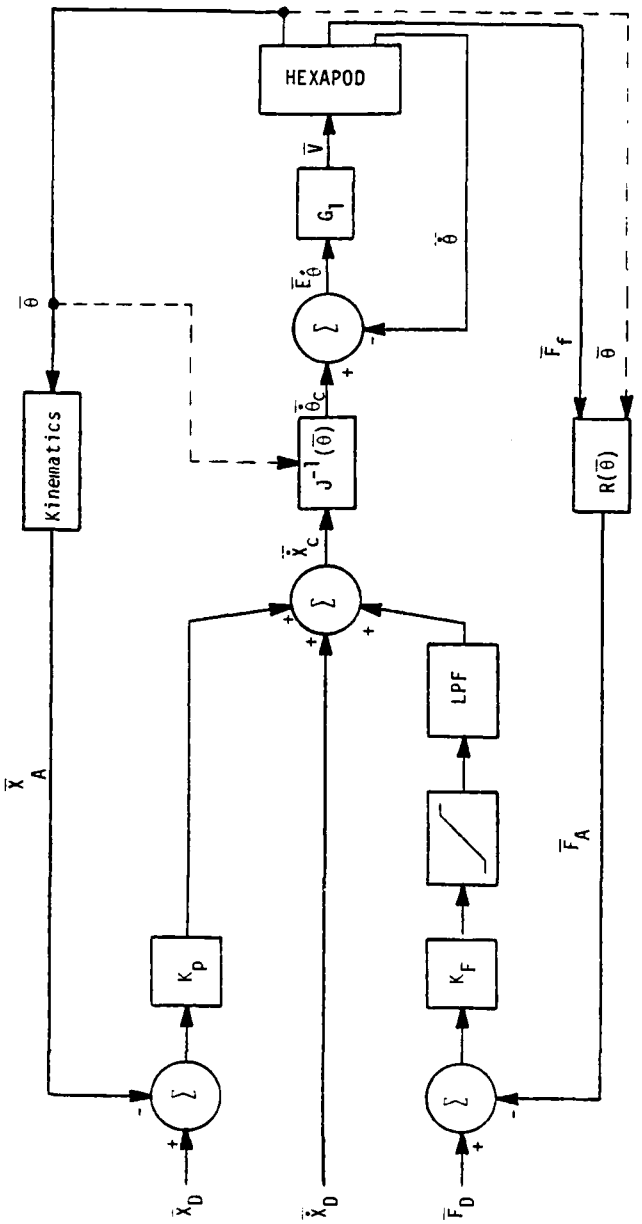


Figure 5.1 Modified Compliance Servo with Limiter and Low-Pass Filter.

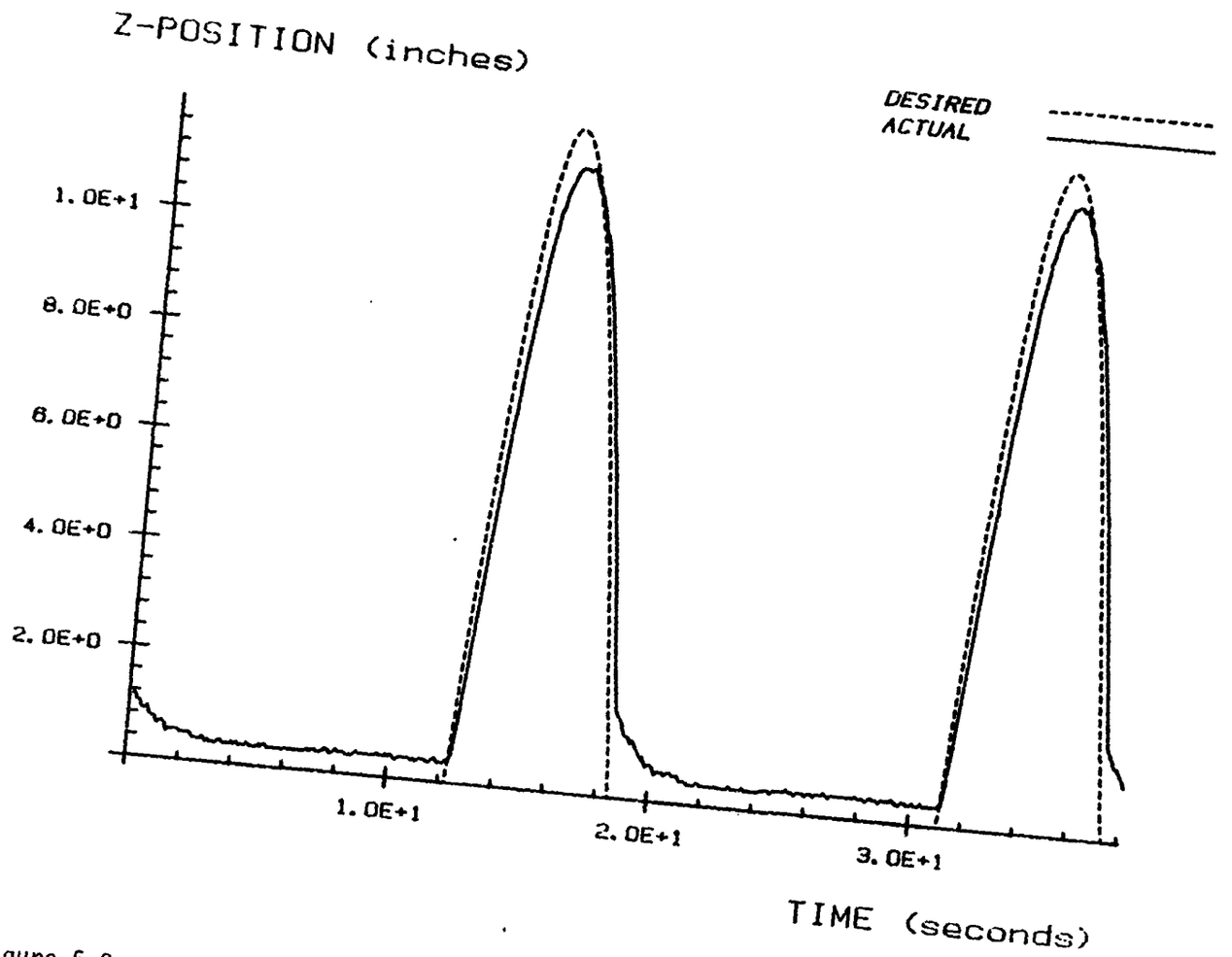


Figure 5.2 Base Servo Response. Plot obtained with legs cycling in the air.

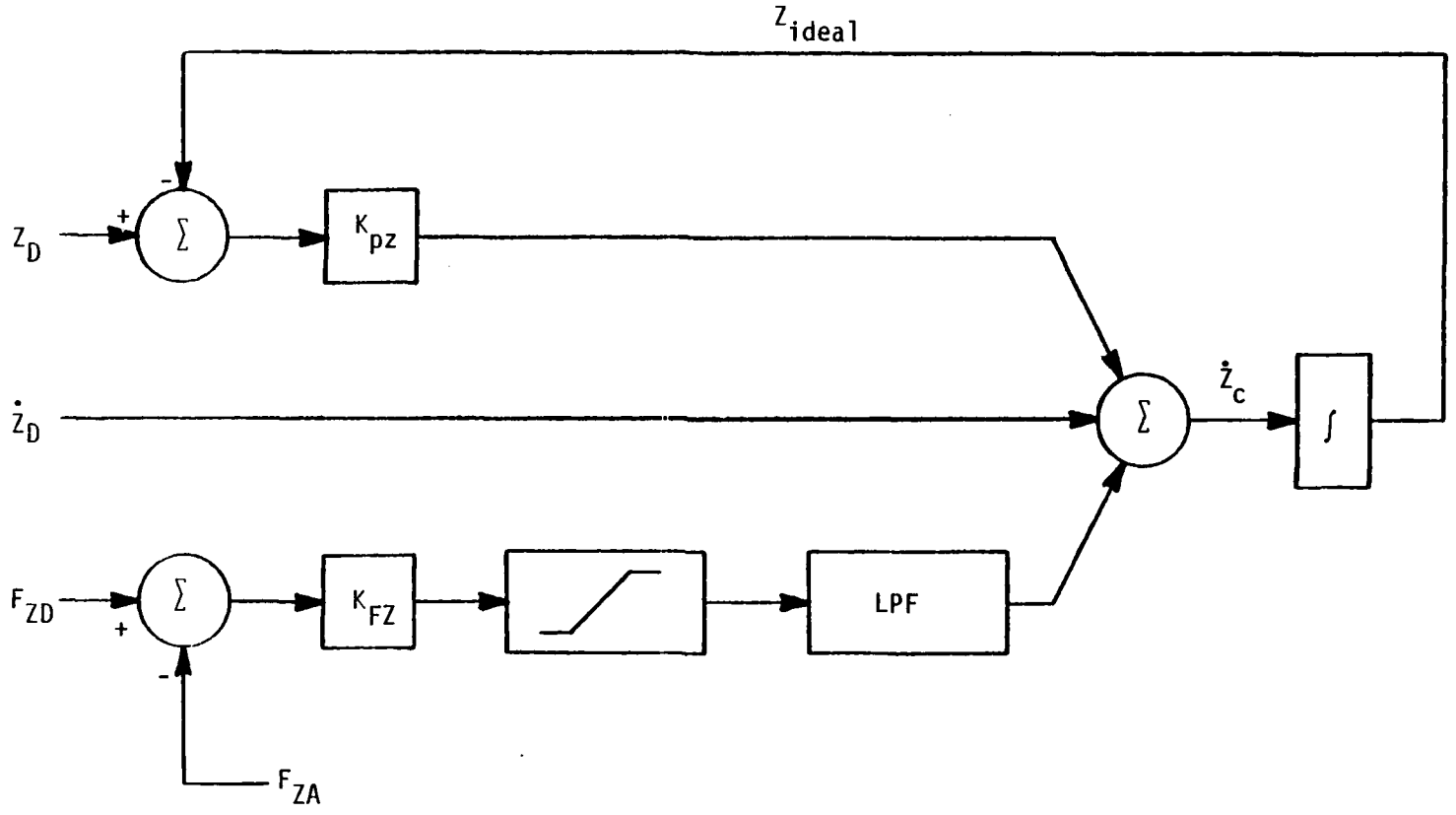


Figure 5.3 Compliance Servo Modeling Routine. Reference inputs and actual force are as recorded during locomotion tests.

recorded ground reaction forces are also used by the modeling routine, since these forces are impossible to model accurately. The solid trace in Figure 5.4 shows a foot trajectory of the actual system, obtained while the legs were cycling in the air with active compliance enabled. The dashed curve is the ideal system response as predicted by the modeling routine. The ideal trajectory agrees very closely with the actual results, differing only in the same respects as the base-line reference experiment. When this experiment was repeated on the ground, the results were comparable.

In order to test the force setpoint generation algorithm independently, the Hexapod was walked with the force feedback loop disabled. The resulting passive foot forces were compared with the output of the force setpoint algorithm. Figure 5.5 shows the results obtained for leg #1 while walking in a tripod gait. The close agreement between the actual force and the force setpoint should be expected; there are exactly three legs on the ground in a tripod gait, and thus equations 4.48 to 4.50 have a unique solution. In other gaits, however, this system of equations is underspecified, and has a infinite number of solutions. Thus, the setpoints generated by the pseudo-inverse algorithm may not agree well with the passive force distribution, since small uncertainties in leg positions and passive compliances determine which of the infinite number of solutions is valid. To observe the underspecified case, the experiment was repeated with a leg duty factor of 2/3, meaning that four legs were in support phase at all times. The results, plotted in Figure 5.6, show the presence of a component of the

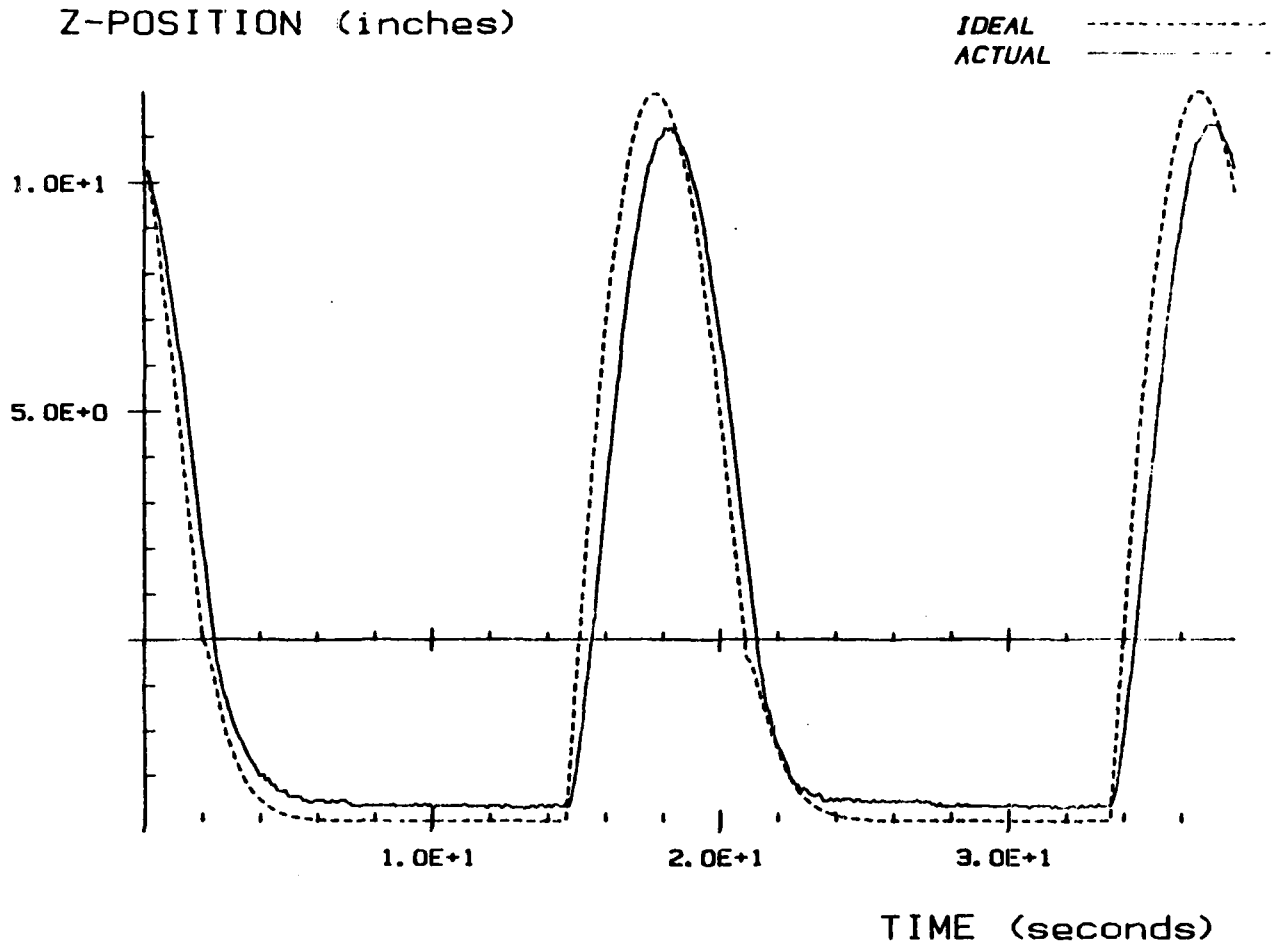


Figure 5.4 Foot Position with Active Compliance. Plot obtained with legs cycling in the air.

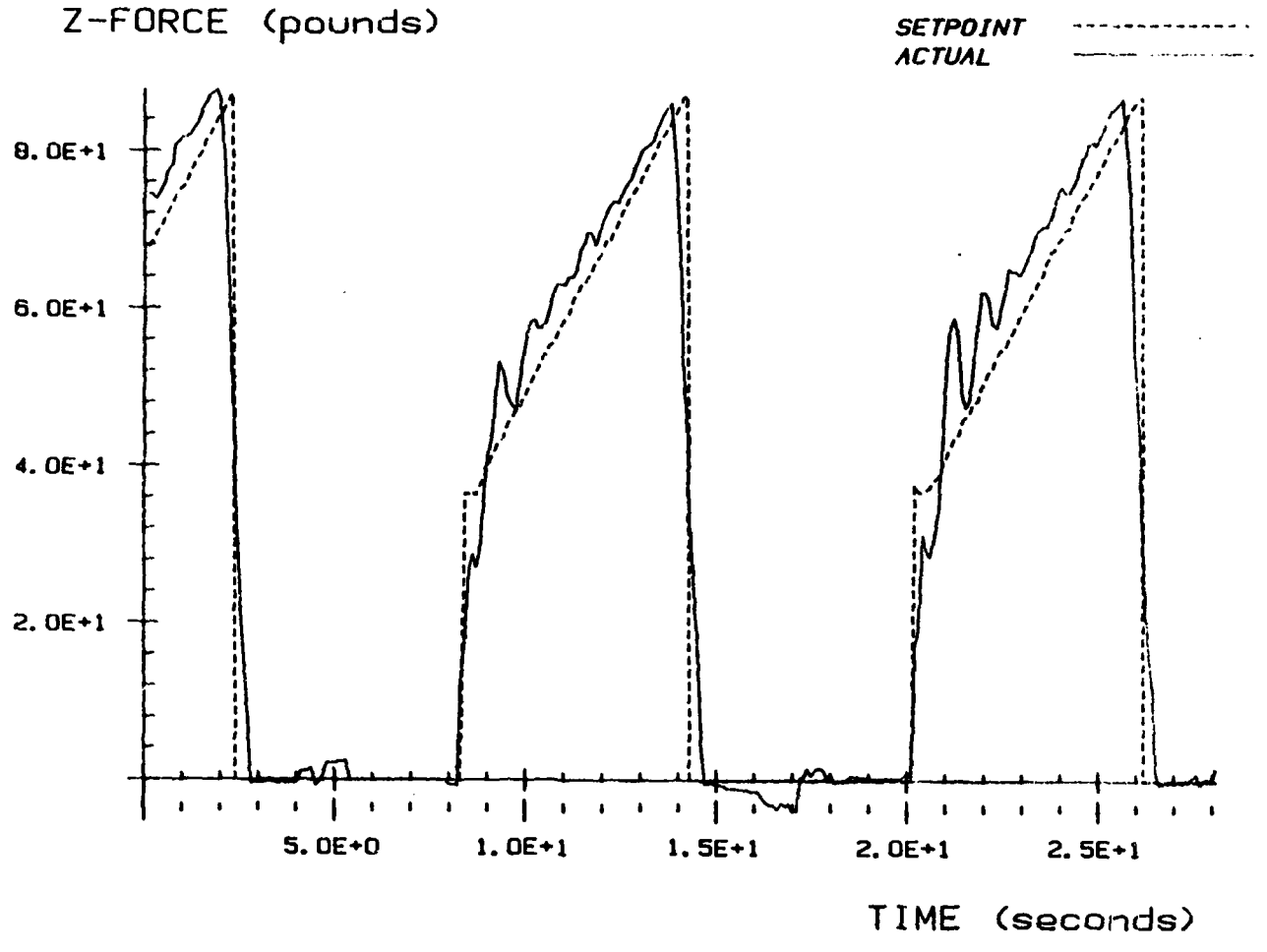


Figure 5.5 Passive Force Distribution in Tripod Gait. Plot obtained on level terrain.

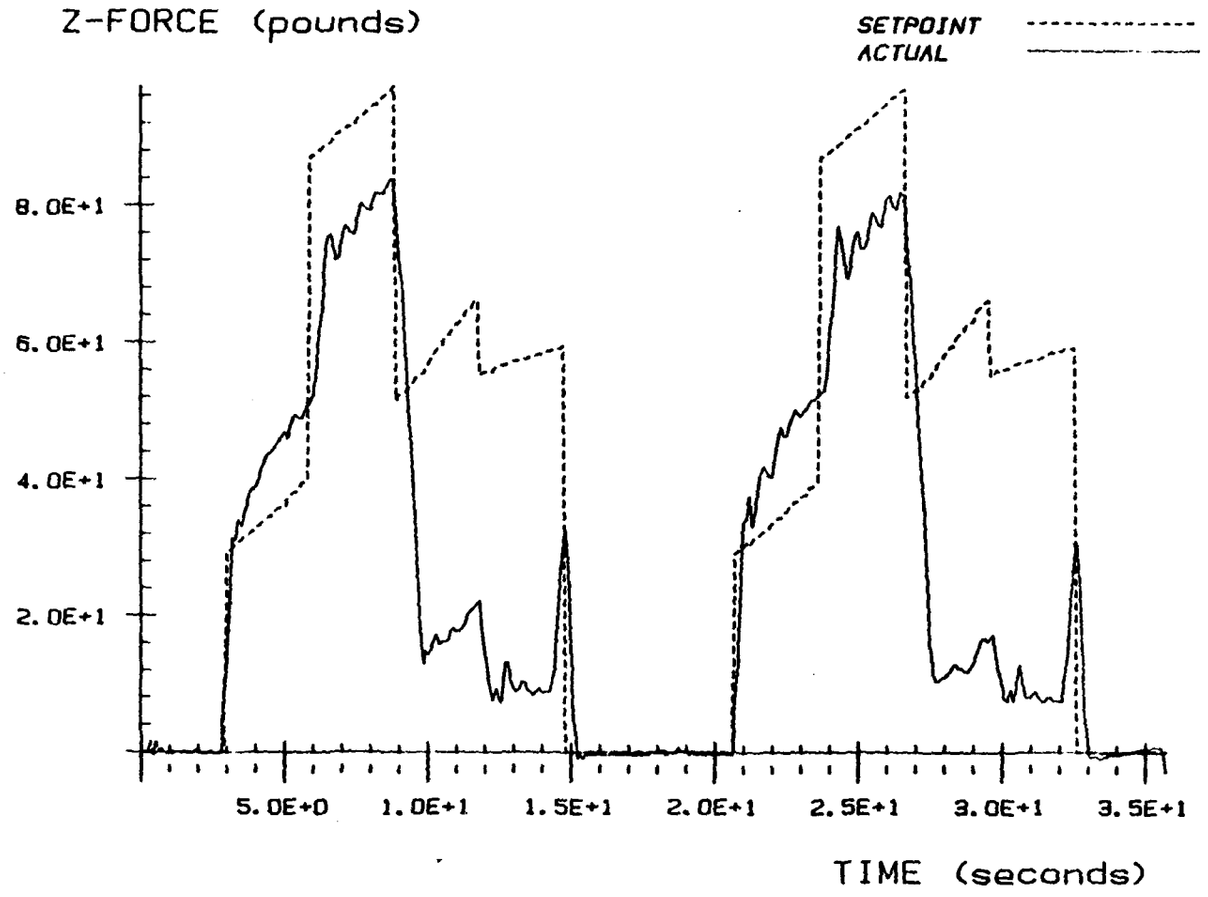


Figure 5.6 Passive Force Distribution with Leg Duty Factor of 2/3. Plot obtained on level terrain. Four legs are in support phase at all times.

homogeneous solution. Note that for a given set of feet in support phase, the homogeneous component is approximately constant, and the force tracks well.

5.3 Attitude Control

To test the performance of the attitude control algorithm over irregular terrain, an obstacle was constructed which was large enough to seriously obstruct the Hexapod's locomotion if non-adaptive control algorithms were used. The obstacle is constructed of 4" by 6" oak beams, and is two beams deep, giving a maximum height of 8 inches. A photograph of the Hexapod walking across the obstacle is shown in Figure 5.7.

In order to test the attitude control algorithm independently of the force control algorithm, the attitude control tests were performed using a tripod gait. This allows the force feedback loop to be disabled, since the foot force distribution is uniquely determined. To establish a reference for comparison for the performance of the attitude control algorithm, the Hexapod was walked across the obstacle with no attitude control. The resulting vehicle attitude traces are shown in Figure 5.8. The traces show a maximum vehicle pitch of 8 degrees, along with large angular rate transients. Note that the periodic nature of the traces in Figure 5.8 do not reflect any control loop instability, but rather is caused by locomotion across obstacles of increasing height while using a periodic gait. Although the vehicle did not become statically unstable, it has been observed to do so at pitch angles of less than 15 degrees. Thus, a slightly more formidable obstacle would prove

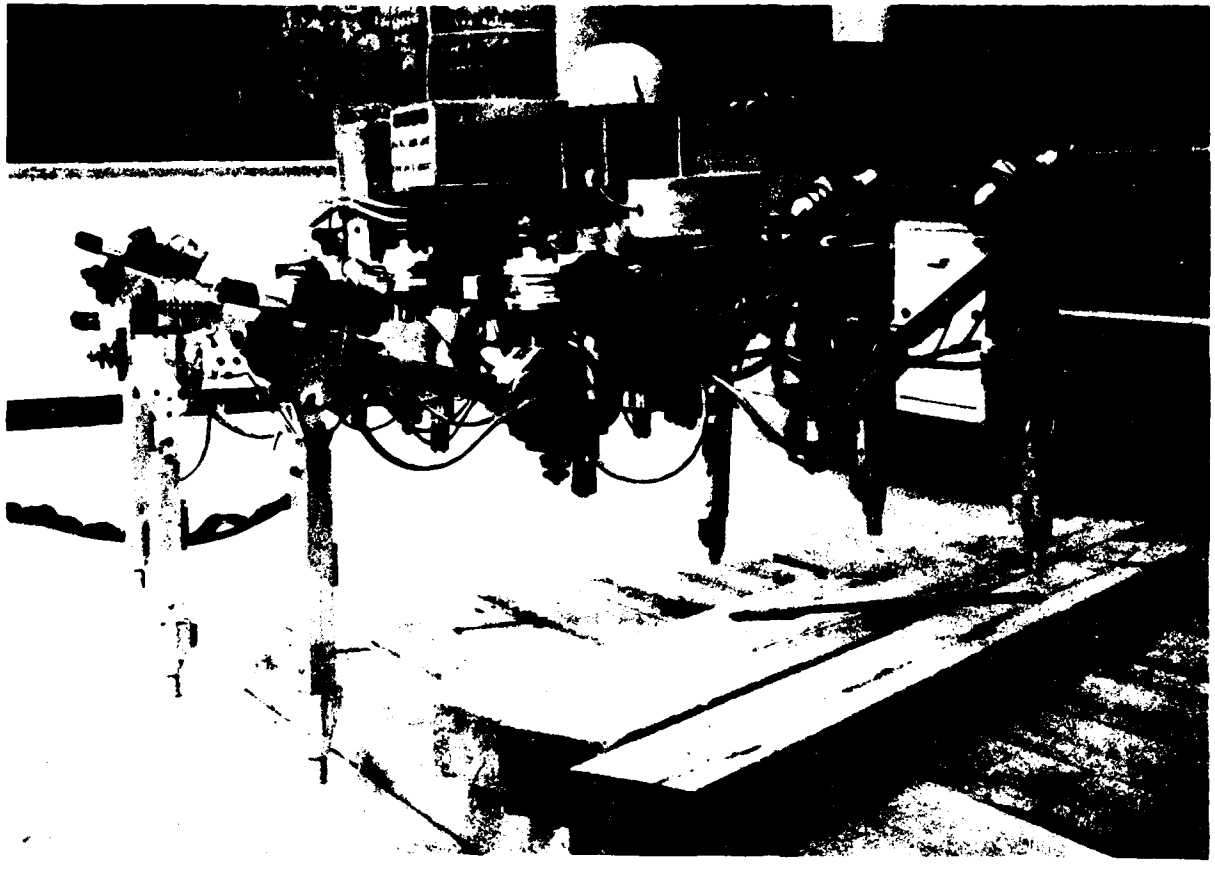


Figure 5.7 OSU Hexapod Traversing Obstacle.

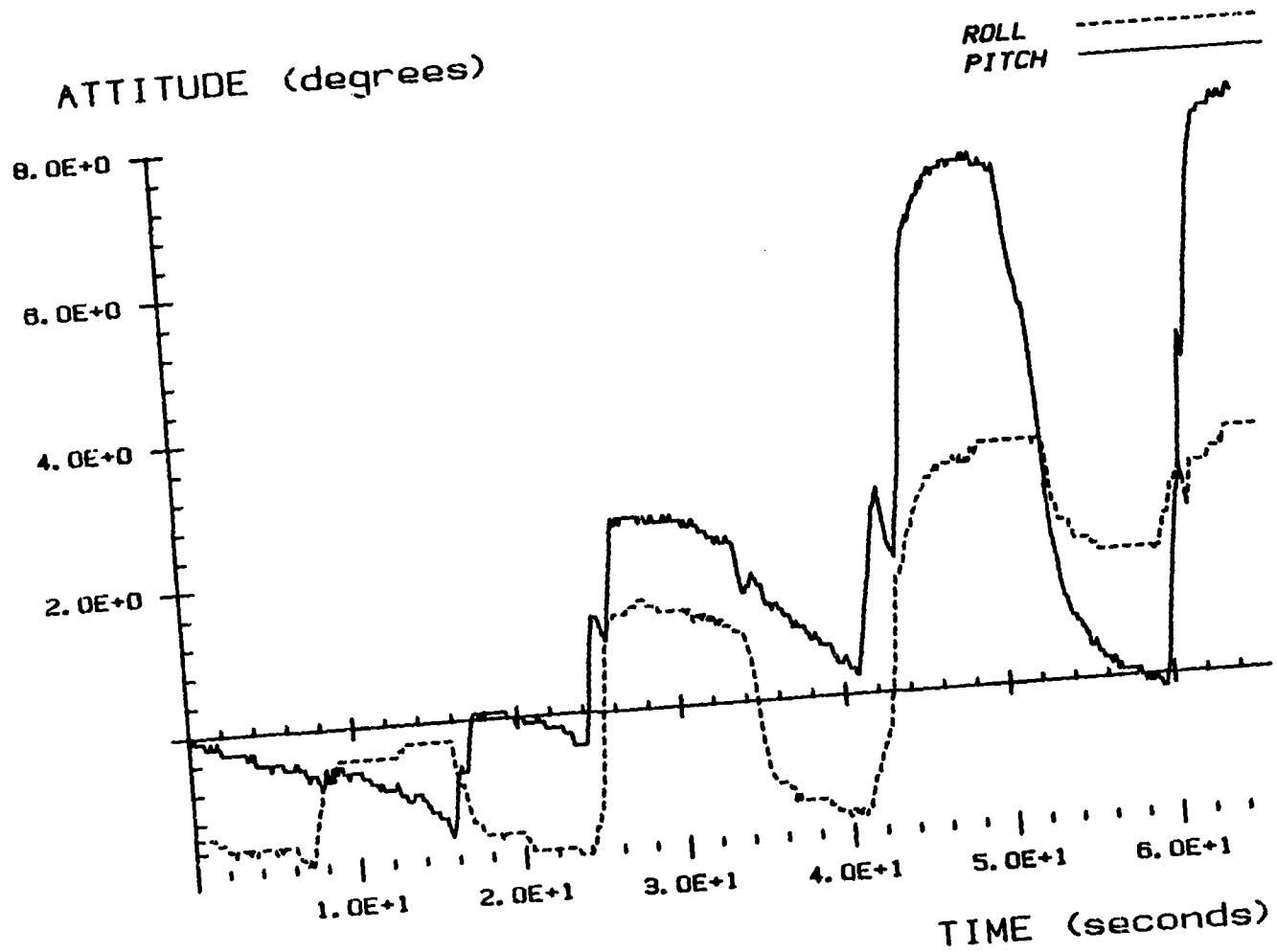


Figure 5.8 Vehicle Attitude Across Obstacle Using No Terrain-Adaptive Algorithms. Plot obtained using a tripod gait.

impossible to traverse without the use of terrain adaptive algorithms.

With an attitude control reference established, locomotion using attitude control can be evaluated. The results shown in Figure 5.9 were obtained while traversing the obstacle with the attitude control loop enabled. The plot shows a large improvement over the attitude reference test. The maximum angular displacement is two degrees. The attitude is maintained within 0.2 degrees of the vertical except during the transients caused by legs leaving and contacting the ground, and the maximum duration of a transient is one second.

5.4 Attitude Control with Active Compliance

The attitude control algorithm was seen in section 5.3 to be very effective using a tripod gait, but it remains to be evaluated in the case of gaits with non-unique foot force solutions. To study this more general case, the Hexapod was walked across the obstacle with the attitude loop enabled while using a leg duty factor of 2/3. The results of this locomotion test with the force feedback loop disabled are shown in Figure 5.10. When the test was repeated with the force feedback loop enabled, the plot in Figure 5.11 was obtained. Comparing the two figures, it is apparent that the force control algorithm reduced the size of most of the foot contact and liftoff transients, thus allowing the attitude control loop to maintain a more level body attitude.

5.5 Force Tracking

With the attitude control algorithm providing a stable body attitude, as was assumed in the development of the force control law, it is

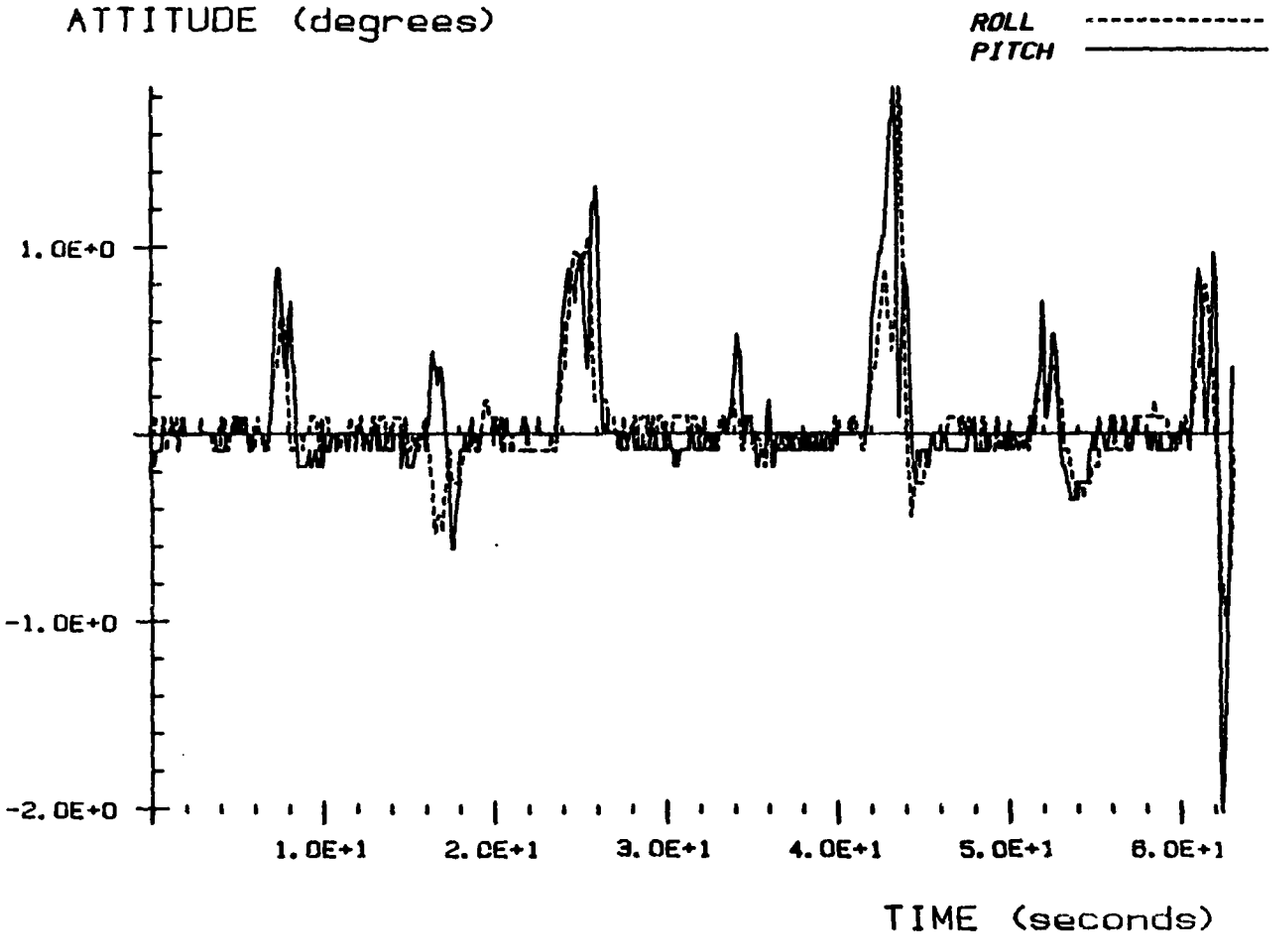


Figure 5.9 Vehicle Attitude Across Obstacle Using Attitude Control. Plot obtained using a tripod gait.

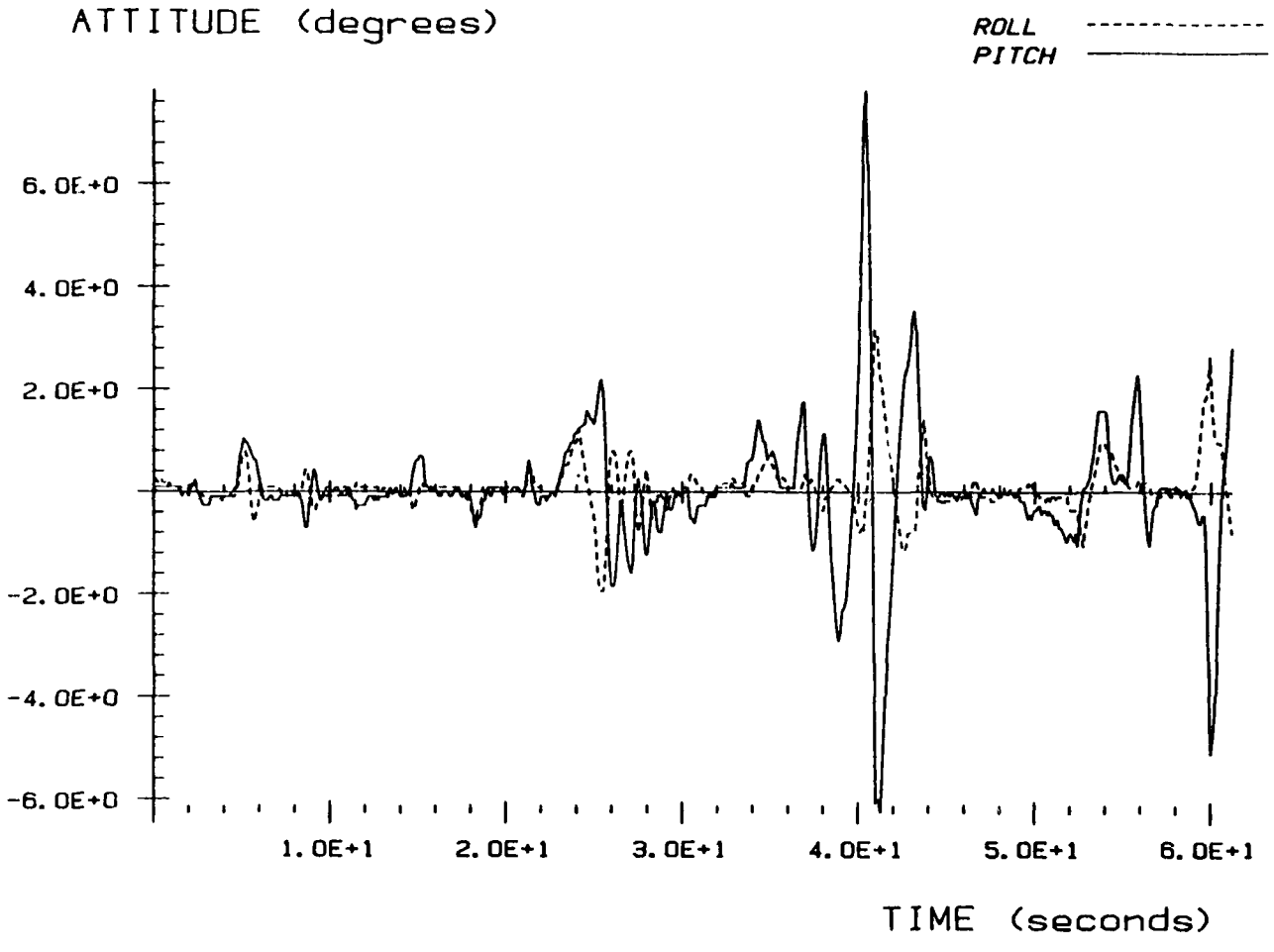


Figure 5.10 Vehicle Attitude Across Obstacle in Underspecified Gait Using Attitude Control. Plot was obtained using a leg duty factor of 2/3.

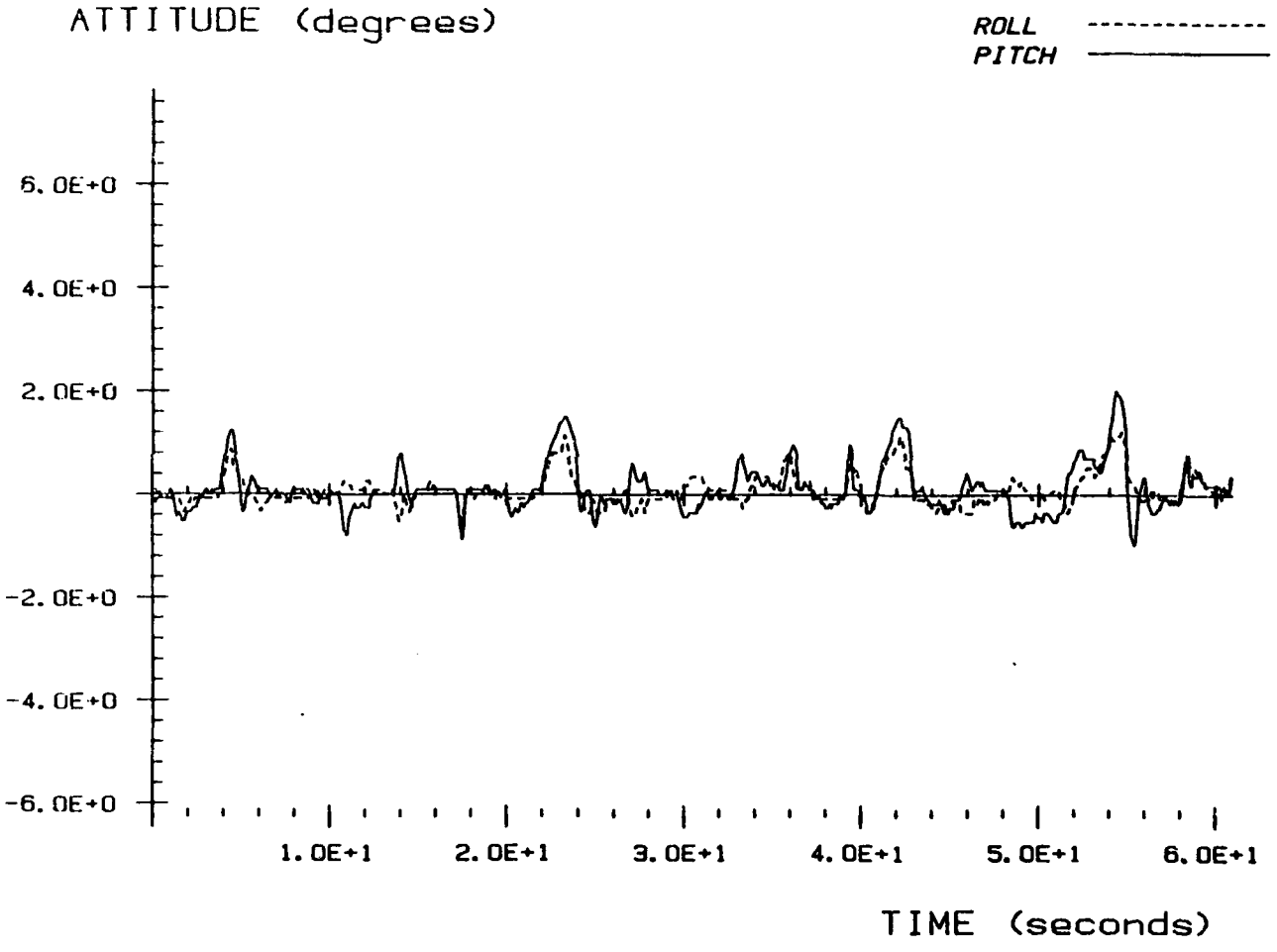


Figure 5.11 Vehicle Attitude Across Obstacle in Underspecified Gait Using Attitude Control and Active Compliance. Plot was obtained using a leg duty factor of 2/3.

informative to observe the accuracy with which the actual force tracks the desired force. From the last experiment of section 5.4 (leg duty factor = 2/3, attitude and force control loops enabled), the force tracking of leg #1 was plotted, and is shown in Figure 5.12. In general, the actual force tracks the desired force quite well. The ringing apparent in the actual force is due to underdamped passive compliances in the Hexapod structure. It is this frequency component that caused the instability described in section 5.2, and which was removed by the inclusion of the low-pass filter in the force feedback loop.

A second discrepancy in the force tracking is also apparent in Figure 5.12. Note that the actual force may become non-zero before the commanded force does so. This situation contributes to transients, as reflected in the vehicle attitude plots, and is due to the fact that a leg is switched into support phase as a function of foot trajectory timing and not as a function of ground contact. It is possible to use contact sensing [4] to minimize this effect, although vehicle height must then be regulated explicitly.

5.6 Attitude Sensor Evaluation

In Chapter 3, two separate attitude sensing systems were implemented: a vertical gyroscope and gravitational pendulums. In the previous sections of this chapter, all vehicle attitude information was obtained from the vertical gyroscope. To test the stability of the attitude control loop when using the pendulums as attitude transducers, an experiment was performed with a zero commanded rate for the Hexapod body and with the attitude control loop enabled. The vehicle became

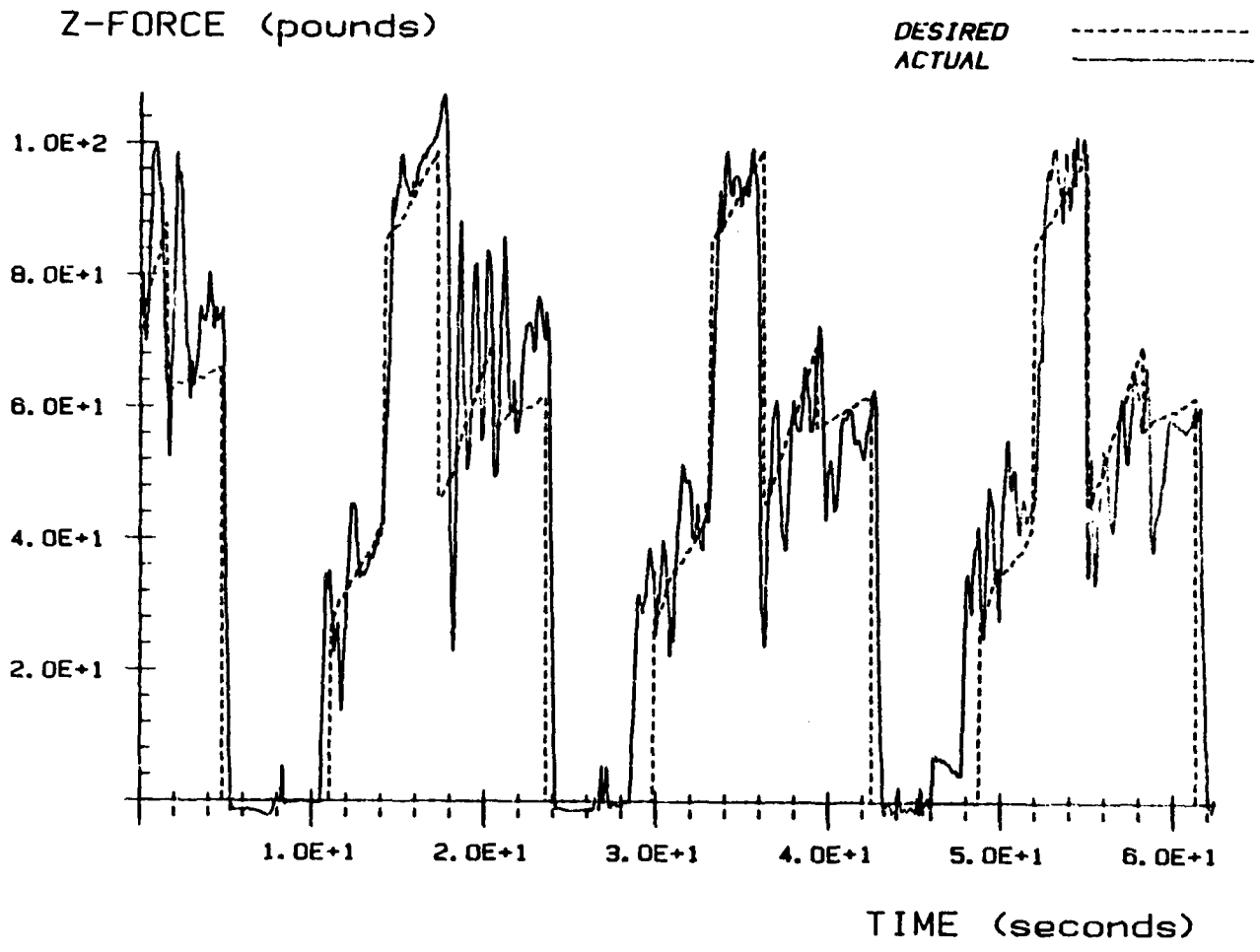


Figure 5.12 Foot Force Tracking Using Active Compliance in an Underspecified Gait. Plot obtained across obstacle using a leg duty factor of 2/3.

unstable with respect to the body roll axis, as shown in Figure 5.13. This instability is due to the fact that the pendulum is quite sensitive to lateral accelerations, which can be induced by the response of the attitude control system. The pendulum output could be low-pass filtered to eliminate the resonances, but this would preclude the high bandwidth response desired of the attitude control system. The pendulums are, however, useful as physically reliable backup, and can be used to check for proper gyroscope erection under static conditions.

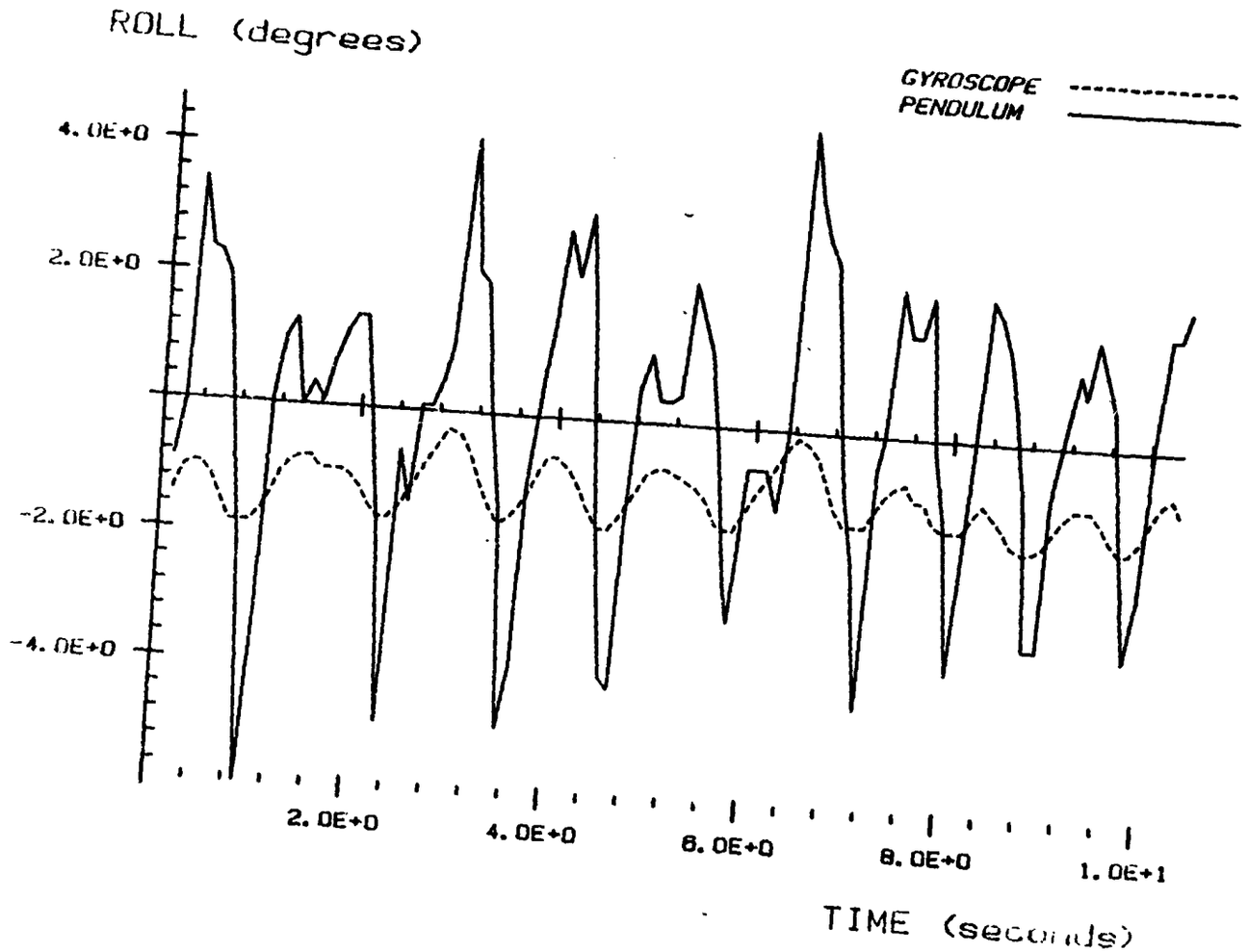


Figure 5.13 Attitude Control Performance Using Gravitational Pendulums.

Chapter 6

SUMMARY AND CONCLUSIONS

The problem of autopilot design for rough-terrain locomotion by a hexapod vehicle has been studied in this thesis. Hardware has been designed and control algorithms developed which allow rough-terrain locomotion with only directional inputs being provided by the operator. The results have shown that if the necessary sensors are present on a legged vehicle, the problems of joint coordination, rough-terrain accommodation, and body regulation can be solved by a digital computer, leaving the vehicle operator free to perform higher-level control functions.

6.1 Research Contributions

The implementation of the autopilot function has required that several well-defined problems be solved. The first of these tasks was the addition of vector force sensors to each leg. The lateral sensor design is identical to the previous design, but the vertical sensor design is similar only in the use of a piston to transmit the ground reaction force. The sensor performance is completely satisfactory, and there have been no reliability problems in the 10 months the sensors have been in operation.

Vehicle attitude sensors have also been added to enable automatic body attitude regulation. Two types of attitude sensors were installed on the Hexapod: gravitational pendulums and a vertical gyroscope. The vertical gyroscope has proven to be an acceptable design solution. It has been shown that pendulums are not an acceptable solution if rapid response is required of the attitude control system.

System reliability has been improved by implementing self-diagnostic capabilities in the digital interface between the vehicle and its control computer. By providing direct computer control over vehicle subsystems power, the chance of human error has been reduced. Since these additions have been made, vehicle readiness has been much improved and research has proceeded more rapidly.

To provide a suspension system for distributing force among all supporting legs on uneven terrain, linear force feedback has been used to implement active compliance. The active compliance algorithm has been examined in detail, and constraints on the system setpoints have been found which result in various types of system responses. Analytic expressions have been found for all system gains as functions of over-all system response. A closed-form solution has been found for the force setpoint problem based on a pseudo-inverse force minimization.

To provide automatic body regulation, a closed loop attitude regulation scheme has been designed. This represents a level of control never before implemented on this vehicle, and is easily expandable into a system which implements arbitrary body attitudes. The attitude regulation system has been shown to work well under all test conditions.

6.2 Research Extensions

Several system modifications have been seen to be desirable as a result of the experiments performed in this research. The modifications suggested below are direct extensions of the autopilot system developed in this work, and should result in improved performance and better control flexibility.

The most obvious of the proposed improvements is the structural stiffening of the OSU Hexapod Vehicle. In addition to making experimental results more difficult to interpret, the excessive compliance present in the vehicle frame contributes to both dynamic and static instability. A stiffer frame should decrease the amplitude and increase the frequency of structural resonances.

The Hexapod is affected by transients caused by foot placement and lifting even with the attitude control and force feedback loops enabled. It is believed that these transients could be greatly reduced by the use of ground contact sensing, where the foot height setpoint during support phase is determined by the height at which it made contact with the ground. This will require the addition of a vehicle altitude control loop due to cumulative errors in ground height sensing.

To properly implement contact sensing, the transfer phase trajectory must be modified to include a search phase in which the foot probes downward in search of the ground. In addition, the non-vertical foot velocity components should be zero with respect to the ground while the foot is below the maximum obstacle height. The foot

trajectory should not allow velocity discontinuities except at ground contact.

It should be possible to reduce the foot contact transients in underspecified gaits by using a different algorithm for force setpoint specification. The present algorithm minimizes the sum of squares of foot forces, but due to the non-backdriveable worm gear reductors, it is of debatable value for reducing energy consumption. An algorithm which minimizes the discontinuities in commanded force at foot contact and liftoff should reduce the actual force transients.

The software needs to be expanded and restructured to allow for arbitrary body attitudes. A "body servo" routine can be written which accepts attitude and altitude setpoints, compares them with the actual values, and modifies the foot state setpoints accordingly. The control loops can be closed in the same manner as the existing attitude control loop. The attitude and altitude setpoints could be specified either by a human operator or by a higher-level algorithm.

APPENDIX A

DERIVATION OF THE FOOT-FORCE ROTATION MATRIX

The geometry of a right-side (even numbered) hexapod leg is shown in Figure A.1. In deriving the rotation matrix $R(\vec{\theta})$, note that force is a sliding vector, and therefore no linear position offsets need to be considered. The right side rotation matrix $R_2(\vec{\theta})$ can be expressed as the produce of two simple rotation matrices:

$$R_2(\vec{\theta}) = [T_1(\psi)][T_2(\theta_1, \theta_2)] \qquad (A.1)$$

where T_2 rotates vectors from the foot coordinate system into the hip coordinate system, and T_1 rotates vectors from the hip coordinate system into the body coordinate system.

To derive $T_1(\theta_1, \theta_2)$, note that since linear offsets are neglected, θ_1 and θ_2 will appear only in the term $(\theta_1 + \theta_2)$. One can therefore define

$$\theta \equiv \theta_1 + \theta_2 \ . \qquad (A.2)$$

Figure A.2 is a simplified representation of foot force vectors, with the foot coordinate system rotated an arbitrary amount from the hip coordinate system. This figure is obtained by removing the linear offsets from Figure A.1 and then lumping the rotary displacements

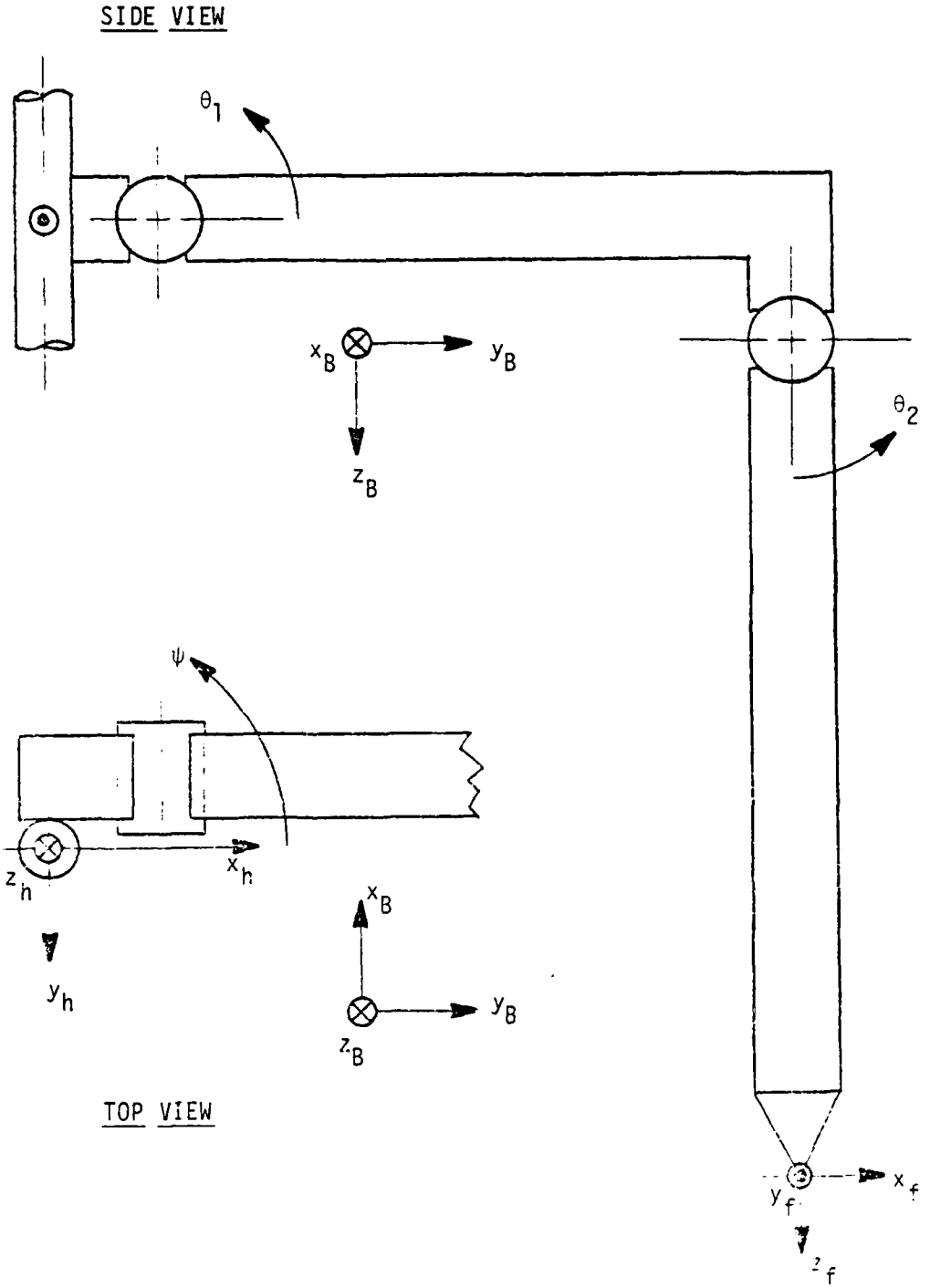


Figure A.1. Right-side Hexapod Leg Geometry.

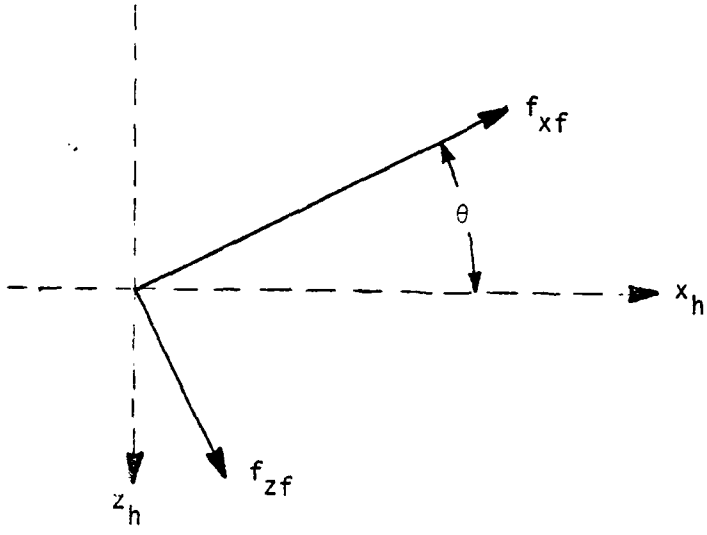


Figure A.2. Force Rotation with Respect to θ.

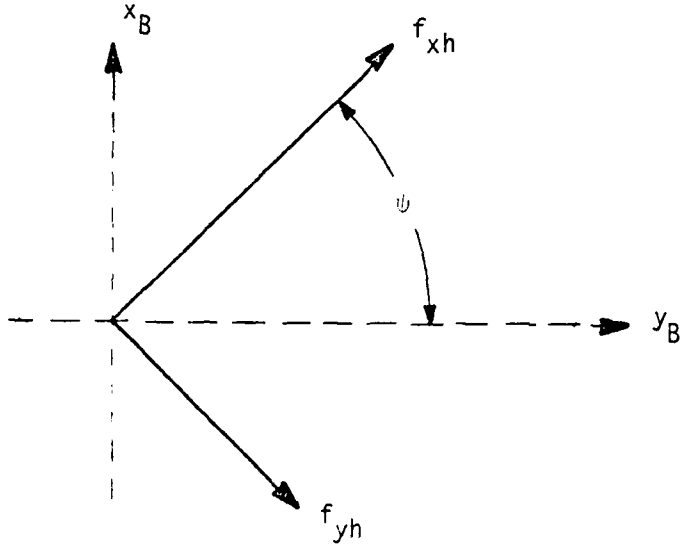


Figure A.3. Force Rotation with Respect to ψ.

θ_1 and θ_2. By defining

$$\vec{F}_f \equiv [f_{xf}\ f_{yf}\ f_{zf}]^T \tag{A.3}$$

as the actual force expressed in the foot coordinate system, and

$$\vec{F}_h \equiv [f_{xh}\ f_{yh}\ f_{zh}]^T \tag{A.4}$$

as the actual force expressed in the hip coordinate system, one can write

$$\vec{F}_h = [T_2(\theta)]\vec{F}_f \ . \tag{A.5}$$

Referring to Figure A.2, it is seen that

$$\begin{bmatrix} f_{xh} \\ f_{yh} \\ f_{zh} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} f_{xf} \\ f_{yf} \\ f_{zf} \end{bmatrix} \ .$$

Then

$$T_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{A.6}$$

By defining

$$\vec{F}_B \equiv [f_{xB}\ f_{yB}\ f_{zB}]^T \tag{A.7}$$

as the actual force expressed in the body coordinate system, one can write

$$\vec{F}_B = [T_1(\psi)]\vec{F}_h \tag{A.8}$$

Figure A.3 depicts this transformation, and is obtained in a manner analogous to Figure A.2. From this figure it is seen that

$$\begin{bmatrix} f_{xB} \\ f_{yB} \\ f_{zB} \end{bmatrix} = \begin{bmatrix} \sin\psi & -\cos\psi & 0 \\ \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{xh} \\ f_{yh} \\ f_{zh} \end{bmatrix} \tag{A.9}$$

and therefore

$$T_2(\psi) = \begin{bmatrix} \sin\psi & -\cos\psi & 0 \\ \cos\psi & \sin\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.10}$$

Combining equations A.1, A.2, A.6, and A.10, the final result

$$R_2(\vec{\theta}) = \begin{bmatrix} \sin\psi\cos(\theta_1+\theta_2) & -\cos\psi & \sin\psi\sin(\theta_1+\theta_2) \\ \cos\psi\cos(\theta_1+\theta_2) & \sin\psi & \cos\psi\sin(\theta_1+\theta_2) \\ -\sin(\theta_1+\theta_2) & 0 & \cos(\theta_1+\theta_2) \end{bmatrix} \tag{A.5}$$

is obtained.

The left side rotation matrix $R_1(\theta)$ is derived similarly, and is given by

$$R_1(\vec{\theta}) = \begin{bmatrix} \sin\psi\cos(\theta_1+\theta_2) & \cos\psi & \sin\psi\sin(\theta_1+\theta_2) \\ -\cos\psi\cos(\theta_1+\theta_2) & \sin\psi & -\cos\psi\sin(\theta_1+\theta_2) \\ -\sin(\theta_1+\theta_2) & 0 & \cos(\theta_1+\theta_2) \end{bmatrix} \qquad (A.6)$$

APPENDIX B

SOLUTION OF THE FOOT-FORCE SETPOINT PROBLEM FOR A MULTILEGGED VEHICLE

For a legged vehicle to maintain static equilibrium, the following equations must be satisfied.

$$\Sigma M_x = 0 \qquad \text{(sum of moments about x-axis)} \qquad \text{(B.1)}$$

$$\Sigma M_y = 0 \qquad \text{(sum of moments about y-axis)} \qquad \text{(B.2)}$$

$$\Sigma F_z = F_{total} \qquad \text{(sum of vertical forces)} \qquad \text{(B.3)}$$

On a hexapod vehicle, there will be from three to six legs in support phase. Numbering only legs in support phase, equations B.1, B.2, and B.3 become

$$f_1 y_1 + f_2 y_2 + \ldots + f_n y_n = 0 \qquad \text{(B.4)}$$

$$f_1 x_1 + f_2 x_2 + \ldots + f_n x_n = 0 \qquad 3 \leq n \leq 6 \qquad \text{(B.5)}$$

$$f_1 + f_2 + \ldots + f_n = F_{total} \qquad \text{(B.6)}$$

These can be written as

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ x_1 & x_2 & \cdots & x_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_{total} \end{bmatrix} \qquad \text{(B.7)}$$

This system is underspecified except for the special case $n = 3$, in which case the system is exactly specified. Thus, a pseudo-inverse-type solution may be applied to obtain a minimum sum of squares of forces.

Given the underspecified system

$$A \bar{z} = \bar{c} \tag{B.8}$$

it can be shown that a minimum norm solution for $\bar{z}$ must lie in the row space of A. This condition can be expressed as

$$\bar{z} = A^T \bar{w} \tag{B.9}$$

where $\bar{w}$ is a weighting vector for the rows of A. Substituting equation B.9 into B.8 gives

$$A A^T \bar{w} = \bar{c} \tag{B.10}$$

The matrix $A A^T$ is nonsingular if A has full row rank, and therefore Gaussian elimination may be employed to solve for $\bar{w}$. The value of $\bar{z}$ can then be obtained from equation B.9.

For the given system,

$$AA^T = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ x_1 & x_2 & \cdots & x_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 & x_1 & 1 \\ y_2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ y_n & x_n & 1 \end{bmatrix} \tag{B.11}$$

or

$$AA^T = \begin{bmatrix} \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} x_i & n \end{bmatrix} \tag{B.12}$$

Using Gaussian elimination and dropping subscripts, the following augmented matrices are obtained.

$$M_1 = \left[\begin{array}{ccc|c} \Sigma y^2 & \Sigma xy & \Sigma y & 0 \\ \Sigma xy & \Sigma x^2 & \Sigma x & 0 \\ \Sigma y & \Sigma x & n & F \end{array}\right] \tag{B.13}$$

$$M_2 = \left[\begin{array}{ccc|c} 1 & \frac{\Sigma xy}{\Sigma y^2} & \frac{\Sigma y}{\Sigma y^2} & 0 \\ 0 & \Sigma x^2 - \frac{(\Sigma xy)^2}{\Sigma y^2} & \Sigma x - \frac{\Sigma xy \Sigma y}{\Sigma y^2} & 0 \\ 0 & \Sigma x - \frac{\Sigma xy \Sigma y}{\Sigma y^2} & n - \frac{(\Sigma y)^2}{\Sigma y^2} & F \end{array}\right] \tag{B.14}$$

$$M_3 = \left[\begin{array}{ccc:c} 1 & 0 & \dfrac{\Sigma y}{\Sigma y^2} - \dfrac{\Sigma xy}{\Sigma y^2}\left(\dfrac{\Sigma x - \dfrac{\Sigma xy \Sigma y}{\Sigma y^2}}{\Sigma x^2 - \dfrac{(\Sigma xy)^2}{\Sigma y^2}}\right) & 0 \\ 0 & 1 & \dfrac{\Sigma x - \dfrac{\Sigma xy \Sigma y}{\Sigma y^2}}{\Sigma x^2 - \dfrac{(\Sigma xy)^2}{\Sigma y^2}} & 0 \\ 0 & 0 & n - \dfrac{(\Sigma y)^2}{\Sigma y^2} - \left(\Sigma x - \dfrac{\Sigma xy \Sigma y}{\Sigma y^2}\right)\left(\dfrac{\Sigma x - \dfrac{\Sigma xy \Sigma y}{\Sigma y^2}}{\Sigma x^2 - \dfrac{(\Sigma xy)^2}{\Sigma y^2}}\right) & F \end{array}\right] \tag{B.15}$$

$$M_4 = \left[\begin{array}{ccc:c} 1 & 0 & \dfrac{\Sigma y}{\Sigma y^2} - \dfrac{\Sigma xy}{\Sigma y^2}\left(\dfrac{\Sigma x \Sigma y^2 - \Sigma xy \Sigma y}{\Sigma x^2 \Sigma y^2 - (\Sigma xy)^2}\right) & 0 \\ 0 & 1 & \dfrac{\Sigma x \Sigma y^2 - \Sigma xy \Sigma y}{\Sigma x^2 \Sigma y^2 - (\Sigma xy)^2} & 0 \\ 0 & 0 & n - \dfrac{(\Sigma y)^2}{\Sigma y^2} - \left(\Sigma x - \dfrac{\Sigma xy \Sigma y}{\Sigma y^2}\right)\left(\dfrac{\Sigma x \Sigma y^2 - \Sigma xy \Sigma y}{\Sigma x^2 \Sigma y^2 - (\Sigma xy)^2}\right) & F \end{array}\right] \tag{B.16}$$

Before proceeding further, it must be shown that none of the above denominators are equal to zero. Obviously, Σy^2 is equal to zero only for the case where all legs are in a line under the longitudinal axis of the hexapod, and thus this condition is forbidden.

The other denominator term is

$$d = \Sigma x^2 \, \Sigma y^2 - (\Sigma xy)^2 \ .$$

Consider the n coordinates x_i to represent a vector $\bar{x}$ in R^n. Likewise, there exists a vector $\bar{y}$ in R^n defined by the n coordinates y_i.

Then

$$\Sigma x_i^2 = |\bar{x}|^2$$

$$\Sigma y_i^2 = |\bar{y}|^2$$

$$(\Sigma x_i y_i)^2 = (\bar{x} \cdot \bar{y})^2 = (|\bar{x}| \ |\bar{y}| \cos\theta)^2$$

$$= |\bar{x}|^2 \ |\bar{y}|^2 \cos^2\theta$$

Thus

$$d = |\bar{x}|^2 \ |\bar{y}|^2 - |\bar{x}|^2 \ |\bar{y}|^2 \cos^2\theta$$

$\rightarrow d \geq 0$, and $d = 0$ if $\theta = n\pi$, n any integer

$$\rightarrow \bar{y} = K \bar{x}$$

$$\rightarrow y_i = kx; \ i = 1 \ \ldots \ n$$

i.e., the denominator term is zero only if all the feet are in a straight line, which is again a forbidden condition.

The Gaussian elimination can now be continued.

Let

$$R = \frac{\Sigma x \Sigma y^2 - \Sigma xy \Sigma y}{\Sigma x^2 \Sigma y^2 - (\Sigma xy)^2} \qquad (B.17)$$

$$Q = \frac{\Sigma y}{\Sigma y^2} - \frac{\Sigma xy}{\Sigma y^2} R \qquad (B.18)$$

$$S = n - \frac{(\Sigma y)^2}{\Sigma y^2} - \left(\Sigma x - \frac{\Sigma xy \Sigma y}{\Sigma y^2}\right) R \tag{B.19}$$

$$S = n - \frac{1}{\Sigma y^2}\left[(\Sigma y)^2 + (\Sigma x \Sigma y^2 - \Sigma xy \Sigma y) R\right] \tag{B.20}$$

The augmented matrix M_4 from equation B.10 is then

$$M_4 = \left[\begin{array}{ccc:c} 1 & 0 & Q & 0 \\ 0 & 1 & R & 0 \\ 0 & 0 & S & F \end{array}\right] \tag{B.21}$$

From the physical description of the problem (given that all feet are not in a straight line) it is obvious that the system is solvable, and therefore that the matrix AA^T is nonsingular. Thus the pivot element S must be nonzero, and final augmented matrix is obtained as

$$M_5 = \left[\begin{array}{ccc:c} 1 & 0 & 0 & -\frac{Q}{S} F \\ 0 & 1 & 0 & -\frac{R}{S} F \\ 0 & 0 & 1 & \frac{1}{S} F \end{array}\right] \tag{B.22}$$

From equation B.25, the intermediate result $\bar{w}$ is seen to be

$$\bar{w} = \left[\begin{array}{c} -\frac{Q}{S} F \\ -\frac{R}{S} F \\ \frac{1}{S} F \end{array}\right] \tag{B.23}$$

The final solution is obtained from equations B.9 and B.23, giving

$$\bar{z} = \begin{bmatrix} y_1 & x_1 \\ y_2 & x_2 \\ \vdots & \vdots \\ y_n & x_n \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} -Q/S \\ -R/S \\ 1/S \end{bmatrix} F \qquad (B.24)$$

where $\bar{z}$ is the vector

$$\bar{z} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}, \text{ and } f_i \text{ is the optimal force for leg i.} \qquad (B.25)$$

Then for any leg i in support phase, the normalized result

$$\frac{f_i}{F_{total}} = \frac{1}{S}(1 - Qy_i - Rx_i) \qquad (B.26)$$

is obtained from equations B.24 and B.25.

APPENDIX C

HEXAPOD CONTROL PROGRAM VERSION 3.4

```
(*$A-*)
            /******  FILE:  GBLF34.PAS  ******/

/*************************************************************************/
/*                                                                       */
/* FUNCTION:     THIS FILE CONTAINS GLOBAL DECLARATIONS FOR              */
/*               ROBOT 3.4. THE EXECUTABLE FILES WHICH SHARE             */
/*               THESE GLOBALS ARE:                                      */
/*                                                                       */
/*               ROBT34.PAS      PLAN34.PAS      FOOT34.PAS              */
/*               INIT34.PAS      LINE34.PAS      SER 34.PAS              */
/*               DLNK34.PAS      LIBR34.PAS                              */
/*                                                                       */
/*************************************************************************/

CONST
        PI       = 3.14159;
        FSCALE   = -200.0;        /* FORCE SCALE FACTOR                  */
        MAXSTROKE = 12.0;         /* MAXIMUM FOOT STROKE                 */
        MOVE     = TRUE;          /* SWITCH TO ENABLE SERVOING           */
        NOMOVE   = FALSE;         /* SWITCH TO DISABLE SERVOING          */


TYPE
        ARRAY6  = ARRAY[1..6] OF REAL;
        ARRAY18 = ARRAY[0..17] OF REAL;
        MODETYPE = ( RANDOM, NEUTRAL, PREWALK,
                     CRUISE, SIDESTEP, TURN    );


 VAR
        MIDSTX, MIDSTY,           /* MIDSTANCE COORDINATES               */
        RPHASE,                   /* RELATIVE LEG PHASES                 */
        XFA,  YFA,  ZFA,          /* ACTUAL FOOT FORCES                  */
        XFD,  YFD,  ZFD,          /* DESIRED FOOT FORCES                 */
        XFF,  YFF,  ZFF,          /* FILTERED FOOT FORCES                */
        XPA,  YPA,  ZPA,          /* ACTUAL FOOT POSITIONS               */
        XPD,  YPD,  ZPD,          /* DESIRED FOOT POSITIONS              */
        XRD,  YRD,  ZRD,          /* DESIRED FOOT RATES                  */
        ZPTERM                    /* Z POS. TERM FOR ATT. CONTROL        */

                : ARRAY6;

        ZEROFORCE                 /* TRUE FORCE OFFSETS                  */

                : ARRAY18;

        FZERO                     /* FLAGS FOR FORCE ZEROING             */

                : ARRAY[1..6] OF BOOLEAN;
```

```
ATTITUDE,                   /* SWITCH FOR ATTITUDE CONTROL  */
COMPLIANCE,                 /* INDIRECT COMPLIANCE SWITCH   */
OPTIMIZATION,               /* SWITCH FOR OPTIMAL FORCE     */
PASS1,                      /* SWITCH TO INITIALIZE FILTER  */
SAVE,                       /* SWITCH FOR DATA AQUISITION   */
SPRING                      /* SWITCH FOR ACTIVE COMPLIANCE */

        : BOOLEAN;

COMMAND                     /* OPERATOR INPUT COMMAND       */

        : CHAR;

LASTCLOCK,                  /* STORAGE FOR CLOCK BUFFER     */
TOTALCLOCK                  /* CLOCK TICK ACCUMULATOR       */

        : INTEGER;

MODE                        /* HEXAPOD OPERATING MODE       */

        : MODETYPE;

BETA,                       /* LEG DUTY FACTOR              */
DPSI,                       /* FILTERED TURN RATE COMMAND   */
DT,                         /* DELTA TIME (SEC)             */
FOOTLIFT,                   /* FOOT LIFTING HEIGHT          */
MIDSTZ,                     /* MIDSTANCE Z COORDINATE       */
NVELX, NVELY,               /* OPERATOR VELOCITY COMMANDS   */
NDPSI,                      /* OPERATOR TURN RATE COMMAND   */
PERIOD,                     /* PERIOD OF KINEMATIC CYCLE    */
PHASE,                      /* KINEMATIC CYCLE PHASE        */
RADIUS,                     /* RADIUS FROM CG. TO MIDSTANCE */
SPERIOD,                    /* SUPPORT PERIOD (SEC)         */
VELX, VELY,                 /* FILTERED VELOCITY COMMANDS   */
VELMAX                      /* MAX FOOT VELOCITY COMPONENT  */

        : REAL;

SUPPORT                     /* SET OF LEGS IN SUPPORT PHASE */

        : SET OF 1..6;
```

```
                    /****** FILE: ROBT34.PAS ******/


                     /*********************************/
                     /*      PROGRAM: ROBOT 3.4       */
                     /*********************************/

/*************************************************************************/
/* PROGRAMMERS: TED CHANG, DENNIS PUGH                                   */
/* DATE:        30-MAR-82                                                */
/*                                                                       */
/* PROGRAMMER                                                            */
/* GUIDE:       >PAS ROBT34=GBLF34,ROBT34                                */
/*              >MAC ROBT34=ROBT34                                       */
/*              >TKB @ROBT34                                             */
/*                                                                       */
/* LINK FILES:  LIBR34                                                   */
/*              PLAN34                                                   */
/*              FOOT34                                                   */
/*              SERV34                                                   */
/*              DLNK34                                                   */
/*              INIT34                                                   */
/*              LINE34                                                   */
/*              KYCK34                                                   */
/*              POWR34                                                   */
/*                                                                       */
/* USER GUIDE:  >RUN ROBT34                                              */
/*              THE PROGRAM PROMPTS FOR INSTRUCTIONS                     */
/*                                                                       */
/*              (THIS PROGRAM IS TO RUN ON THE PDP 11/70)                */
/*                                                                       */
/*************************************************************************/


CONST   ZRMAX    = 6.0;          /* MAXIMUM  FOOT Z - VELOCITY   */

TYPE    CAPITALS = 'A'..'Z';
        CHARSET  = SET OF CAPITALS;
        SETARRAY = ARRAY[ MODETYPE ] OF CHARSET;


VAR     DPSIMAX   :REAL;         /* MAXIMUM  TURN RATE           */
        VTMAX     :REAL;         /* MAX. FOOT VELOCITY COMPONENT */

        BETAMODE  :INTEGER;      /* LEG DUTY FACTOR INDEX        */

        LETTER    :CHAR;         /* LOOP INDEX                   */
        HALTSET   :CHARSET;      /* SET OF COMMANDS FOR WHICH    */
                                 /* HEXAPOD HALTS AFTERWARD      */
        NEXTCOM   :SETARRAY;     /* SET OF VALID COMMANDS FOR    */
                                 /* FOR EACH MODE                */
```

```
PROCEDURE  NORMALIZE;  EXTERNAL;

PROCEDURE  INITIALIZE;  EXTERNAL;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE  UPDOWN( HEIGHT: REAL );  EXTERNAL;

PROCEDURE  PLANMOTION;  EXTERNAL;

PROCEDURE  CLOCKINIT;  EXTERNAL;

PROCEDURE  KEYINT;  FORTRAN;

PROCEDURE  KEYCK(VAR COMMAND: CHAR);  FORTRAN;

PROCEDURE  KYWAIT;  FORTRAN;

PROCEDURE  IOKILL;  FORTRAN;

PROCEDURE  INQUE;  FORTRAN;

PROCEDURE  TESTON;  EXTERNAL;

PROCEDURE  TURNON;  EXTERNAL;

PROCEDURE  TURNOF;  EXTERNAL;

FUNCTION  DELTATIME: REAL;  EXTERNAL;


BEGIN   /* START MAIN PROGRAM */

            /*******************************/
            /*  PARAMETER INITIALIZATION */
            /*******************************/

FOOTLIFT := 5.0;
BETAMODE := 4;

/***  ASSIGN MIDSTANCE POSITION ***/

MIDSTX[1] :=  22.75 + 1.436;     MIDSTX[2] :=  22.75 + 1.436;
MIDSTX[3] :=          1.436;     MIDSTX[4] :=          1.436;
MIDSTX[5] := -22.75 + 1.436;     MIDSTX[6] := -22.75 + 1.436;

MIDSTY[1] := -24.0;              MIDSTY[2] := 24.0;
MIDSTY[3] := -24.0;              MIDSTY[4] := 24.0;
MIDSTY[5] := -24.0;              MIDSTY[6] := 24.0;

MIDSTZ := 17.0;

RADIUS := SQRT( MIDSTX[1] * MIDSTX[1] + MIDSTY[1] * MIDSTY[1] );
```

```
/***  NEXTCOMMAND `ECLARATION  ***/

NEXTCOMC .ANDOM ]   := C'I','M','N','X']#
NEXTCOMC NEUTRAL ]  := C'D','I','M','N','U','X']#
NEXTCOMC PREWALK ]  := C'C','H','I','M','N','.','X','Z']#
NEXTCOMC CRUISE ]   := C'B','F','H','L','P','R','S','X']#
NEXTCOMC TURN ]     := C'H','P','S','X']#
NEXTCOMC SIDESTEP ] := C'H','L','R','X']#

HALTSET      := C'M','N','I','H','U','D','C','T','Z','W']#


WRITE( CHR(27), '[2J')#          /* ERASE SCREEN */
WRITE( CHR(27), '[1#1H')#        /* CURSOR TO HOME */

TESTON#     /* POWER UP THE HEXAPOD */
TURNOF#     /* SWITCH OFF MOTOR POWER UNTIL NEEDED */


            /************************/
            /*   DISPLAY  MENU      */
            /************************/

WRITE( CHR(27), '[2J')#          /* ERASE SCREEN */
WRITE( CHR(27), '[1#1H')#        /* CURSOR TO HOME */

WRITELN('ENTER N  TO NORMALIZE THE LEG POSITIONS              |   BETAMODE =   ')#
WRITELN('      U  TO MOVE THE BODY UP                         |                ')#
WRITELN('      D  TO MOVE THE BODY DOWN                       |   FOOTLIFT =   ')#
WRITELN('      I  TO INITIALIZE THE LEG POSITIONS             |----------------')#
WRITELN('      M  TO MODIFY THE PARAMETERS                    |                ')#
WRITELN('      X  TO EXIT THE PROGRAM EXECUTION               |   COMPLIANCE   ')#
WRITELN('                                                     |                ')#
WRITELN('      C  TO ENTER CRUISE MODE                        |   OPTIMIZATION ')#
WRITELN('      T  TO ENTER TURN-IN-PLACE MODE                 |                ')#
WRITELN('      Z  TO ENTER SIDESTEP MODE                      |   ATTITUDE     ')#
WRITELN('                                                     |----------------')#
WRITELN('      F  TO INCREASE FORWARD VELOCITY COMPONENT      |                ')#
WRITELN('      B  TO INCREASE REARWARD VELOCITY COMPONENT     |   AQUISITION   ')#
WRITELN('      S  TO INCREASE CLOCKWISE TURN RATE             |      TIME      ')#
WRITELN('      P  TO INCREASE COUNTERCLOCKWISE TURN RATE      |                ')#
WRITELN('      R  TO INCREASE RIGHTWARD VELOCITY COMPONENT    |                ')#
WRITELN('      L  TO INCREASE LEFTWARD VELOCITY COMPONENT     |                ')#
WRITELN('      H  TO HALT MOTION                              |                ')#
WRITELN#
```

```
/***  SET TERMINAL CHARACTERISTICS  ***/

WRITE( CHR(27), '[?41' );        /* SET TO NOSCROLL */
WRITE( CHR(27), '[20;22r' );     /* SET SCROLLING REGION */
WRITE( CHR(27), '[?61' );        /* SET ABSOLUTE ORIGIN MODE */
WRITE( CHR(27), '[20;1f' );      /* CURSOR TO SCROLLING REGION */

KEYINT;  { PLACE KEYBOARD INPUT REQUEST ON QUEUE }
CLOCKINIT;  { START PROGRAMMABLE CLOCK }


            /*********************************/
            /*   REAL - TIME  OPERATION   */
            /*********************************/

COMMAND := 'M';  { ENTER MODIFY SEQUENCE INITIALLY }
MODE := RANDOM;

REPEAT { UNTIL COMMAND = 'X' }
    WRITE( CHR(27), '[22;1H' );  { CURSOR TO SCREEN BOTTOM }
    WRITE( CHR(27), '[0m' );     { NORMAL CHARACTER ATTRIBUTES }
    WRITELN;
    WRITELN( COMMAND );

    IF NOT ( COMMAND IN NEXTCOM[ MODE ] )
        THEN { COMMAND IS INVALID }
            BEGIN
            WRITELN( CHR(13), CHR(27),'#6*** INVALID COMMAND ***');
            WRITE( 'VALID COMMANDS ARE: ' );
            FOR LETTER := 'A' TO 'Z' DO
                IF LETTER IN NEXTCOM[ MODE ] THEN WRITE(' ',LETTER);
            WRITELN;

            IF MODE IN [ RANDOM, NEUTRAL, PREWALK ]
                THEN COMMAND := 'W'    { WAIT }
                ELSE COMMAND := 'G';   { GO }
            END;

    { END IF }
```

```
IF COMMAND IN HALTSET
    THEN
        BEGIN
        CASE COMMAND OF

            'N':    BEGIN
                    NORMALIZE;
                    MODE := NEUTRAL;
                    END;

            'I':    BEGIN
                    NORMALIZE;
                    INITIALIZE;
                    MODE := PREWALK;
                    END;

            'U':    UPDOWN( 24.5 );

            'D':    UPDOWN( MIDSTZ );

            'H':    BEGIN
                    HALT;
                    MODE := PREWALK;
                    SAVE := FALSE;   { SWITCH OFF DATA AQUISITION }
                    END;

            'C':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '#6*** CRUISE MODE ***');
                    MODE := CRUISE;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;

            'T':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '#6*** TURN-IN-PLACE MODE ***');
                    MODE := TURN;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;

            'Z':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '#6*** SIDESTEP MODE ***');
                    MODE := SIDESTEP;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;
```

```
'M':    BEGIN
        IOKILL; { CANCEL INPUT REQUEST }
        WRITELN;
        WRITELN( CHR(27), '*6PARAMETER MODIFICATION');
        WRITELN;

{***}   WRITELN( 'CHANGE BETA?');
        READLN( LETTER );
        IF LETTER = 'Y' THEN
            REPEAT
            WRITELN('PLEASE ENTER DESIRED BETAMODE:');
            WRITELN(' 1:  BETA = 5/6      2:  BETA = 3/4');
            WRITELN(' 3:  BETA = 2/3      4:  BETA = 1/2');
            READLN(BETAMODE);
            UNTIL BETAMODE IN [1,2,3,4];
        WRITE( CHR(27),'7', CHR(27),'[1;67H', CHR(27),'[1m' );
        WRITE( BETAMODE:1, CHR(27), '8' );
        WRITELN;
        WRITELN;

        CASE BETAMODE OF
            1:   BETA := 0.8333;  { 5/6 }
            2:   BETA := 0.75;    { 3/4 }
            3:   BETA := 0.6667;  { 2/3 }
            4:   BETA := 0.5      { 1/2 }
        END;/* CASE */

        /***  ASSIGN RELATIVE LEG PHASES  ***/
        RPHASE[1] := 0.0;
        RPHASE[2] := 0.5;
        RPHASE[3] := BETA;
        RPHASE[4] := BETA - 0.5;
        RPHASE[5] := 2.0 * BETA - 1.0;
        RPHASE[6] := RPHASE[5] + 0.5;
        RPHASE[6] := RPHASE[6] - TRUNC(RPHASE[6]);

{***}   WRITELN( 'CHANGE FOOTLIFT?' );
        READLN( LETTER );
        IF LETTER = 'Y' THEN
            REPEAT
            WRITELN( 'ENTER NEW VALUE OF FOOTLIFT' );
            READLN( FOOTLIFT );
            UNTIL ( FOOTLIFT >= 1.0 ) AND ( FOOTLIFT <= 12.0 );
        WRITE( CHR(27),'7', CHR(27),'[3;67H', CHR(27),'[1m' );
        WRITE( FOOTLIFT:4:1, CHR(27), '8' );
        WRITELN;
        WRITELN;

        VTMAX := ZRMAX * MAXSTROKE / ( FOOTLIFT * PI );
        IF VTMAX > 4.0 THEN  VTMAX := 4.0; { IN. PER SEC }
        VELMAX := VTMAX * ( 1.0 - BETA ) / BETA;
        DPSIMAX := VELMAX / RADIUS;
```

```
{***}   WRITELN( 'TURN ACTIVE COMPLIANCE ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[6;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                COMPLIANCE := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                COMPLIANCE := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'COMPLIANCE', CHR(27), '8' );
        WRITELN;
        WRITELN;

{***}   WRITELN( 'TURN FORCE OPTIMIZATION ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[8;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                OPTIMIZATION := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                OPTIMIZATION := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'OPTIMIZATION', CHR(27), '8' );
        WRITELN;
        WRITELN;

{***}   WRITELN( 'TURN ATTITUDE CONTROL ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[10;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                ATTITUDE := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                ATTITUDE := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'ATTITUDE', CHR(27), '8' );
```

```
                        INQUE; { REASSERT INPUT REQUEST }
                        MODE := RANDOM;
                        WRITELN;
                        WRITELN;
                        WRITELN( CHR(27), '#6INITIALIZE HEXAPOD ');
                        END;

                    'W':

                    END; { CASE COMMAND }
                KYWAIT;
                KEYCK(COMMAND);
                DT := DELTATIME;  { UPDATE LASTCLOCK AFTER WAIT }
                END { THEN }

            ELSE
                BEGIN
                CASE COMMAND OF

                    'F':    IF NVELX < ( 0.9 * VELMAX )
                                THEN NVELX := NVELX + ( 0.1 * VELMAX )
                                ELSE NVELX := VELMAX;

                    'B':    IF NVELX > -( 0.9 * VELMAX )
                                THEN NVELX := NVELX - ( 0.1 * VELMAX )
                                ELSE NVELX := -VELMAX;

                    'S':    IF NDPSI < ( 0.9 * DPSIMAX )
                                THEN NDPSI := NDPSI + ( 0.1 * DPSIMAX )
                                ELSE NDPSI := DPSIMAX;

                    'P':    IF NDPSI > -( 0.9 * DPSIMAX )
                                THEN NDPSI := NDPSI - ( 0.1 * DPSIMAX )
                                ELSE NDPSI := -DPSIMAX;

                    'R':    IF NVELY < ( 0.9 * VELMAX )
                                THEN NVELY := NVELY + ( 0.1 * VELMAX )
                                ELSE NVELY := VELMAX;

                    'L':    IF NVELY > -( 0.9 * VELMAX )
                                THEN NVELY := NVELY - ( 0.1 * VELMAX )
                                ELSE NVELY := -VELMAX;

                    'G':

                    END; { CASE COMMAND }

                PLANMOTION;  { MOTION PLANNING & EXECUTION }
                             { UNTIL NEXT COMMAND INPUT    }

                END; { ELSE }
        { END IF COMMAND IN HALTSET }
    UNTIL COMMAND = 'X';
```

```
HALT;

IOKILL; { CANCEL KEYBOARD INPUT REQUEST }

WRITE( CHR(27), '[?4h');          /* SET TO SMOOTH SCROLL */
WRITE( CHR(27), '[1;22r');        /* RESTORE SCROLLING REGION */
WRITE( CHR(27), '[2J');           /* ERASE SCREEN */
WRITE( CHR(27), '[1;1H');         /* CURSOR TO HOME */
END.
```

```
(*$E+*)

          /*******    FILE:   PLAN35.PAS    *******/


                /*******************************/
                /*   PROCEDURE:  PLANMOTION    */
                /*******************************/

/*************************************************************************/
/*                                                                       */
/* PROGRAMMERS: TED CHANG, DENNIS PUGH                                   */
/* DATE:        30-MAR-82                                                */
/* FUNCTION:    PLAN BODY MOTION TO IMPLEMENT OPERATOR COMMANDS          */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              PLANMOTION;                                              */
/*                                                                       */
/* PROCEDURES CALLED:   FOOTLINE, KEYCK                                  */
/*                                                                       */
/* GLOBAL VARIABLES:                                                     */
/*      REFERENCED:     NVELX,  NVELY,  NDPSI                            */
/*                      BETA,   VELMAX                                   */
/*                      MODE                                             */
/*                                                                       */
/*      MODIFIED:       VELX,   VELY,   DPSI                             */
/*                      PERIOD, SPERIOD, PHASE                           */
/*                      DT                                               */
/*                                                                       */
/*************************************************************************/

PROCEDURE PLANMOTION;

CONST   TIMECONST = 0.5;        /* TIME CONST. FOR INPUT FILTER */
        MINSTROKE = 5.0;        /* MINIMUM FOOT STROKE          */
        RMIN = 60.0;            /* MIN. TURN RADIUS ( CRUISE )  */


VAR DVELX, DVELY, DDPSI,        /* VELOCITY DERIVATIVE TERMS    */
    NVEL,                       /* VELOCITY COMMAND MAGNITUDE   */
    LTIME,                      /* TEMPORARY STORAGE FOR TIME   */
    FOOTRATE,                   /* APPROX. FOOT RATE WRT BODY   */
    STROKE                      /* FOOT TRAVEL IN SUPPORT PHASE */
                        :REAL;


PROCEDURE FOOTPATH;  EXTERNAL;

PROCEDURE KEYCK( VAR COMMAND: CHAR );  FORTRAN;

FUNCTION SIGN( X: REAL ): REAL;  EXTERNAL;

FUNCTION DELTATIME: REAL;  EXTERNAL;
```

```
BEGIN  { PLANMOTION }

IF MODE = CRUISE  { SET LIMITS ON VELOCITY COMPONENTS }
    THEN
        BEGIN
        WRITELN( CHR(27), '[0m' ); { NORMAL CHARACTER ATTRIBUTES }

        /* SET LIMIT FOR CRAB ANGLE IN CRUISE MODE */
        IF ABS( NVELY ) > ABS( NVELX )
            THEN NVELY := SIGN( NVELY ) * ABS( NVELX );

        /* DETERMINE MAGNITUDE OF COMMANDED VELOCITY */
        NVEL := NVELX*NVELX + NVELY*NVELY;
        IF NVEL <> 0.0 THEN NVEL := SQRT( NVEL );

        /* LIMIT VELOCITY MAGNITUDE TO VELMAX */
        IF NVEL > VELMAX THEN
            BEGIN
            NVELX := NVELX / NVEL * VELMAX;
            NVELY := NVELY / NVEL * VELMAX;
            END;

        /* SET LIMIT FOR DPSI IN CRUISE MODE */
        IF ABS( NDPSI ) > ABS( NVEL / RMIN )
            THEN NDPSI := SIGN( NDPSI ) * ABS( NVEL / RMIN );

        WRITE( NVELX:8:3, NVELY:8:3, NDPSI:8:3 );
        IF ABS(NDPSI) > 0.00001
            THEN WRITE( NVEL / ABS( NDPSI ):8:2);

        END { THEN }

    ELSE
        WRITE( NVELX:8:3, NVELY:8:3, NDPSI:8:3 );

{ END IF MODE }

WRITE( CHR(27), '[1A' );  { ENABLE BOLD CHARACTERS }

REPEAT  { UNTIL A COMMAND IS INPUT }

    /*** GENERATE DT ***/
    DT := DELTATIME;

    /*** FILTER THE RATE COMMAND INPUTS ***/

    DVELX := (NVELX - VELX) / TIMECONST;  /* LONGITUDINAL ACCEL.  */
    DVELY := (NVELY - VELY) / TIMECONST;  /* LATERAL ACCELERATION */
    DDPSI := (NDPSI - DPSI) / TIMECONST;  /* TURNING ACCELATION   */

    VELX := DVELX*DT + VELX;              /* LONGITUDINAL VEL.    */
    VELY := DVELY*DT + VELY;              /* LATERAL VELOCITY     */
    DPSI := DDPSI*DT + DPSI;              /* TURNING RATE         */
```

```
        /*** CALCULATE STROKE & SUPPORT PERIOD ***/

        CASE MODE OF
            CRUISE:     FOOTRATE := VELX;

            SIDESTEP:   FOOTRATE := VELY;

            TURN:       FOOTRATE := RADIUS * DPSI;

            END; { CASE }

        IF ABS( FOOTRATE ) > 0.00001
            THEN
                BEGIN
                STROKE := MINSTROKE + ( MAXSTROKE - MINSTROKE )
                            * ABS( FOOTRATE ) / VELMAX;

                SPERIOD := STROKE / FOOTRATE;       { SUPPORT PERIOD }
                END
            ELSE
                SPERIOD := 10000.0;

        PERIOD := SPERIOD / BETA;           { TOTAL CYCLE PERIOD }


        /*** UPDATE PHASE VARIABLE ***/
        PHASE := PHASE + DT / PERIOD + 1;
        PHASE := PHASE - TRUNC(PHASE);

        /*** CALL FOOT TRAJECTORY GENERATION ROUTINE ***/
        FOOTPATH;

        /*** CHECK FOR OPERATOR INPUT ***/
        KEYCK(COMMAND);

    UNTIL ORD( COMMAND ) <> 0;  { UNTIL A COMMAND IS INPUT }

    END;
```

```
(*$E+*)

          /*******      FILE:  FOOT34.PAS    *******/


                /*******************************/
                /*   PROCEDURE:   FOOTPATH   */
                /*******************************/

/*********************************************************************/
/* PROGRAMMERS: DENNIS PUGH, TED CHANG                               */
/* DATE:        30-MAR-82                                            */
/*                                                                   */
/* FUNCTION:    CALCULATES FOOT TRAJECTORIES TO IMPLEMENT            */
/*              THE BODY RATES COMMANDED BY BODY MOTION              */
/*              PLANNING.                                            */
/*                                                                   */
/* USER GUIDE:  THE CALLING FORMAT IS:      FOOTPATH;                */
/*                                                                   */
/*                                                                   */
/* PROCEDURES CALLED:   JSERVO, RBADC                                */
/*                                                                   */
/* GLOBAL VARIABLES                                                  */
/*      REFERENCED:     VELX,   VELY,   DPSI                         */
/*                      MIDSTX, MIDSTY, MIDSTZ                       */
/*                      DT,     PERIOD, SPERIOD                      */
/*                      PHASE,  RPHASE, BETA                         */
/*                      FOOTLIFT                                     */
/*                                                                   */
/*      MODIFIED:       XPD,    YPD,    ZPD                          */
/*                      XRD,    YRD,    ZRD                          */
/*                      XFD,    YFD,    ZFD                          */
/*                      SUPPORT,   ZPTERM                            */
/*                                                                   */
/*********************************************************************/

PROCEDURE FOOTPATH;

CONST   ATTSCALE = -0.7854;     /* ATTITUDE SCALE FACTOR          */
        ATTPOLE = 8.0;          /* POLE FOR ATTITUDE CONTROL      */
        FTOTAL = 285.0;         /* TOTAL VEHICLE WEIGHT ( LBS.)   */
        PWRSCALE = -33333.3;    /* POWER SCALE FACTOR             */
```

```
VAR
        ALPHA,              /* CRAB ANGLE WRT LONGITUDINAL AXIS      */
        CGX, CGY,           /* COORDINATES OF CG. PROJECTION         */
        DT1,                /* BACKED UP TIME FROM MIDSTANCE         */
        DXB, DYB,           /* BODY DISPLACEMENTS IN BODY COORD.     */
        DXB1, DYB1,         /* TOUCHDOWN DISPL. FROM MIDSTANCE       */
        DXE, DYE,           /* BODY DISPLACEMENT IN EARTH COORD.     */
        LPHASE,             /* LEG PHASE VARIABLE                    */
        PITCH, ROLL,        /* ACTUAL ANGLES FROM GYRO               */
        PPITCH, PROLL,      /* ACTUAL ANGLES FROM PENDULUM           */
        POWER,              /* HEXAPOD INSTANTANEOUS POWER CONSUMP.  */
        PSIC, PSIC1,        /* BODY ANGULAR DISPLACEMENT OVER DT     */
        Q, R, S,            /* INTERMED. FORCE OPTIMIZATION TERMS    */
        SUMX,   SUMY,       /* SUM OF FOOT DISPLACEMENTS             */
        SUMX2,  SUMY2,      /* SUM OF SQUARES OF FOOT DISPLACEMENTS  */
        SUMXY,              /* SUM OF FOOT DISPL. CROSS-PRODUCTS     */
        TIMEFP,             /* REMAINING TIME IN TRANSFER PHASE      */
        TPHASE,             /* TRANSFER PHASE VARIABLE               */
        TRTIME,             /* TOTAL TIME IN TRANSFER PHASE          */
        VEL,                /* MAGNITUDE OF VELOCITY VECTOR          */
        ZOFF,               /* Z POSITION ERROR FOR ATT. CONTROL     */
        ZRTERM              /* Z RATE OFFSET FOR ATTITUDE CONTROL    */

                : REAL;

        FRSTCH,             /* FIRST A/D CHANNEL FOR DATA FETCH      */
        I,                  /* LOOP COUNTER                          */
        N                   /* SUPPORT PHASE LEG SET COUNTER         */

                : INTEGER;

        XFTHLD, YFTHLD  /* DESIRED FOOT TOUCHDOWN POINT              */

                : ARRAY6;


        TOPBLOCK            /* TOP A/D CHANNELS                      */

                : ARRAY18;

PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;

PROCEDURE R8ADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
        EXTERNAL;

FUNCTION  ATAN2(Y, X : REAL): REAL;  EXTERNAL;
```

```
BEGIN   /* FOOT34 */

/*********************************************************************/
/*  EXPRESS INCREMENTAL BODY DISPLACEMENT IN BODY COORDINATES  */
/*********************************************************************/

/*** CALCULATE INCREMENTAL BODY DISPLACEMENT WRT GROUND ***/
PSIC    := DPSI * DT;

/*** CALCULATE MAGNITUDE OF VELOCITY WRT GROUND ***/
VEL := VELX * VELX + VELY * VELY;
IF VEL <> 0.0 THEN VEL := SQRT(VEL);

IF ABS(DPSI) > 0.00001
    THEN
        BEGIN
        DXE     := VEL/DPSI * SIN(PSIC);
        DYE     := VEL/DPSI * (1.0 - COS(PSIC));
        END
    ELSE
        BEGIN
        DXE     := VEL * DT;
        DYE     := 0.0;
        END;

/*** ROTATE DISPLACEMENT VECTORS TO BODY COORDINATES ***/
ALPHA := ATAN2( VELY, VELX );

DXB :=   DXE * COS(ALPHA)
       - DYE * SIN(ALPHA);

DYB :=   DXE * SIN(ALPHA)
       + DYE * COS(ALPHA);



/*************************************/
/*  COMPUTE FOOT TOUCHDOWN OFFSETS  */
/*************************************/

/*** CALCULATE BODY DISPLACEMENT FROM MIDSTANCE TO TOUCHDOWN ***/
DT1     := -0.5 * ABS(SPERIOD);  { BACKED UP TIME FROM MIDSTANCE }
PSIC1   := DPSI * DT1;           { BACKED UP ANGLE }

IF ABS(DPSI) > 0.00001
    THEN
        BEGIN
        DXE     := VEL / DPSI * SIN(PSIC1);
        DYE     := VEL / DPSI * ( 1.0 - COS(PSIC1) );
        END
    ELSE
        BEGIN
        DXE     := VEL * DT1;
        DYE     := 0.0;
        END;
```

```
/*** ROTATE TOUCHDOWN OFFSET TO BODY COORDINATES ***/
DXB1 :=    DXE * COS(ALPHA)
         - DYE * SIN(ALPHA);

DYB1 :=    DXE * SIN(ALPHA)
         + DYE * COS(ALPHA);


RBADC(5,58,1,TOPBLOCK); { FETCH PITCH, ROLL, & POWER FROM VEHICLE }

PPITCH := TOPBLOCK[4] * ATTSCALE;
PROLL  := TOPBLOCK[5] * ATTSCALE;
POWER  := TOPBLOCK[6] * PWRSCALE;
PITCH  := TOPBLOCK[7] * ATTSCALE;
ROLL   := TOPBLOCK[8] * ATTSCALE;


/*********************************/
/*  GENERATE FOOT TRAJECTORIES   */
/*********************************/

FOR I := 1 TO 6 DO { GENERATE FOOT COORDINATE FOR LEG I }
    BEGIN

    /*** COMPUTE LEG PHASE VARIABLE ***/
    LPHASE := PHASE + RPHASE[I] + 1.0;
    LPHASE := LPHASE - TRUNC( LPHASE );

    IF LPHASE > BETA
        THEN { LEG IN TRANSFER PHASE }
            BEGIN

            TPHASE := (LPHASE - BETA) / (1.0 - BETA);
            SUPPORT := SUPPORT - [I];  { REMOVE LEG I FROM SUPPORT SET }

            IF { LEG AT TOP OF TRANSFER PHASE AND FORCE NOT YET ZEROED }
                (    ( (PERIOD > 0.0) AND (LPHASE > (BETA+1)/2.0) )
                  OR ( (PERIOD < 0.0) AND (LPHASE < (BETA+1)/2.0) ) )
                AND (FZERO[I] = FALSE)

                THEN  { UPDATE OFFSET FORCES FOR LEG I }
                    BEGIN
                    FRSTCH := ( I - 1 ) * 3;
                    RBADC( 3, FRSTCH, FSCALE, ZEROFORCE);
                    FZERO[I] := TRUE;  { FLAG THAT LEG I IS UPDATED }
                    ZPTERM[I] := 0.0;  { INITIALIZE ATTITUDE CORRECTION TERM }
                    END;

                { END IF }
```

```
        /*** CALCULATE TIME LEFT TILL TOUCHDOWN OF THE LEG ***/
        TRTIME := ( 1 - BETA ) * ABS(PERIOD);
        IF PERIOD > 0.0
            THEN TIMEFP := TRTIME * ( 1.0 - TPHASE)
            ELSE TIMEFP := TRTIME * TPHASE;

        /*** COMPUTE THE BEST TOUCHDOWN POINT ***/
        XFTHLD[I] :=    ( MIDSTX[I] - DXB1 ) * COS(PSIC1)
                      + ( MIDSTY[I] - DYB1 ) * SIN(PSIC1);

        YFTHLD[I] := - ( MIDSTX[I] - DXB1 ) * SIN(PSIC1)
                      + ( MIDSTY[I] - DYB1 ) * COS(PSIC1);

        /*** COMPUTE DESIRED FOOT POSITION ***/
        IF DT < TIMEFP
            THEN
                BEGIN
                XPD[I] := XPD[I] + (XFTHLD[I] - XPD[I]) * DT/TIMEFP;
                YPD[I] := YPD[I] + (YFTHLD[I] - YPD[I]) * DT/TIMEFP;
                ZPD[I] := MIDSTZ - FOOTLIFT * SIN( TPHASE * PI );
                END
            ELSE
                BEGIN
                XPD[I] := XFTHLD[I];
                YPD[I] := YFTHLD[I];
                ZPD[I] := MIDSTZ;
                END;

            { END IF DT }

        /*** COMPUTE DESIRED FOOT RATE ***/
        XRD[I] := ( XFTHLD[I] - XPD[I] ) / TIMEFP;
        YRD[I] := ( YFTHLD[I] - YPD[I] ) / TIMEFP;
        ZRD[I] := - FOOTLIFT * PI * COS( PI * TPHASE )
                    / ( PERIOD * ( 1 - BETA ) );

        END  { TRANSFER PHASE }

    ELSE  { FOOT IN SUPPORT PHASE }
        BEGIN

        SUPPORT := SUPPORT + [I];  { INCLUDE LEG I IN SUPPORT SET }
        FZERO[I] := FALSE;  { FLAG THAT FORCE NOT ZEROED THIS CYCLE }

        /*** COMPUTE ATTITUDE CONTROL VARIABLES ***/
        ZOFF := -PITCH  * XPA[I] + ROLL * YPA[I];
```

```
                IF ATTITUDE = TRUE
                    THEN
                        BEGIN
                        ZRTERM := ATTPOLE * ZOFF;
                        ZPTERM[I] := ZPTERM[I] + ZRTERM * DT;
                        END
                    ELSE
                        BEGIN
                        ZPTERM[I] := 0.0;
                        ZRTERM := 0.0;
                        END;

                    { END IF ATTITUDE }

                /*** COMPUTE DESIRED FOOT POSITION ***/
                XPD[I] :=    ( XPD[I] - DXB ) * COS(PSIC)
                           + ( YPD[I] - DYB ) * SIN(PSIC);

                YPD[I] := - ( XPD[I] - DXB ) * SIN(PSIC)
                           + ( YPD[I] - DYB ) * COS(PSIC);

                ZPD[I] := MIDSTZ + ZPTERM[I];

                /*** COMPUTE DESIRED FOOT RATE ***/
                XRD[I] := -VELX + DPSI * YPD[I];
                YRD[I] := -VELY - DPSI * XPD[I];
                ZRD[I] := ZRTERM;

                END; /* SUPPORT PHASE */

            /* END IF LPHASE */

        END;  /* FOR I */


/*************************************/
/*  COMPUTE OPTIMAL FORCE SETPOINTS  */
/*************************************/

N := 0;
SUMX := 0.0;
SUMY := 0.0;
SUMX2 := 0.0;
SUMY2 := 0.0;
SUMXY := 0.0;
```

```
FOR I := 1 TO 6 DO
    IF I IN SUPPORT THEN
        BEGIN
        SUMX := SUMX + XPA[I];
        SUMY := SUMY + YPA[I];
        SUMX2 := SUMX2 + XPA[I] * XPA[I];
        SUMY2 := SUMY2 + YPA[I] * YPA[I];
        SUMXY := SUMXY + XPA[I] * YPA[I];
        N := N + 1;
        END;  { IF I IN SUPPORT }

    { END FOR I }

R := (SUMX * SUMY2 - SUMXY * SUMY) / (SUMX2 * SUMY2 - SUMXY * SUMXY);
Q := (SUMY - SUMXY * R) / SUMY2;
S := N - ((SUMX * SUMY2 - SUMXY * SUMY) * R + SUMY * SUMY) / SUMY2;

CGX := -SIN( PITCH ) * MIDSTZ;
CGY := SIN( ROLL ) * MIDSTZ;

/*** COMPUTE FOOT FORCE SETPOINTS ***/

FOR I := 1 TO 6 DO
    BEGIN
    XFD[I] := 0.0;
    YFD[I] := 0.0;

    IF I IN SUPPORT
        THEN
            IF OPTIMIZATION = TRUE
                THEN ZFD[I] := ( 1 - Q * ( YPA[I] - CGY )
                                   - R * ( XPA[I] - CGX )  ) * FTOTAL / S
                ELSE ZFD[I] := FTOTAL / N
        ELSE
            ZFD[I] := 0.0;

    END; { FOR I }

JSERVO( MOVE );  { CALL SERVO ROUTINE }


END;  /* FOOTPATH */
```

```
(*$E+*)

            /******  FILE: SERV34.PAS  ******/


                /****************************/
                /*    PROCEDURE JSERVO      */
                /****************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        30-MAR-82                                                */
/* FUNCTION:    ACCOMPLISHES JACOBEAN SERVO CONTROL.                     */
/*              ACTIVE COMPLIANCE IS TURNED ON OR OFF                    */
/*              BY THE BOOLEAN SWITCH 'SPRING'                           */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              JSERVO( MOVE );                                          */
/*                                                                       */
/* PROCEDURES CALLED:   RBADC, RBDAC                                     */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*      REFERENCED:     XRD, YRD, ZRD                                    */
/*                      XPD, YPD, ZPD                                    */
/*                      XFD, YFD, ZFD                                    */
/*                      ZEROFORCE, SPRING                                */
/*                                                                       */
/*      MODIFIED:       XPA, YPA, ZPA                                    */
/*                      XFA, YFA, ZFA                                    */
/*                      XFF, YFF, ZFF                                    */
/*                      PASS1                                            */
/*                                                                       */
/*************************************************************************/

PROCEDURE  JSERVO( MOVE: BOOLEAN );


CONST   COMPGAIN = 57.4;        /* COMPENSATOR GAIN                      */

        PGAINX = 3.8;           /* RECTILINEAR POSITION GAIN             */
        PGAINY = 3.8;
        PGAINZ = 1.32;

        FGAINX = 0.0001;        /* RECTILINEAR FORCE GAIN                */
        FGAINY = 0.0001;
        FGAINZ = 0.165;

        FILTPOLE = 2.0;         /* POLE FOR FORCE TERM FILTER            */

        PSCALE = 1.571;         /* POSITION SCALE FACTOR                 */
        RSCALE = 0.61;          /* RATE SCALE FACTOR                     */
        ZFSCALE = -1.25;        /* Z-AXIS FORCE SCALE CHANGE             */
        VSCALE = 10.0;          /* VOLTAGE SCALE FACTOR                  */
```

```
        L1      = 12.564;       /* UPPER LIMB LENGTH            */
        L2      = 17.00;        /* LOWER LIMB LENGTH            */
        L3      = -1.436;       /* AZIMUTH JOINT OFFSET         */
        L4      = 2.5;          /* ELEVATION JOINT OFFSET       */
        L5      = 2.436;        /* KNEE JOINT OFFSET            */

        YHIP    = 8.562;        /* Y DISTANCE FROM HIP TO C.G.  */


VAR     PSI,    THETA,  THETA1, THETA2, /* JOINT ANGLES         */
        CPSI,   CTH,    CTH1,   CTH2,   /* COSINES OF ANGLES    */
        SPSI,   STH,    STH1,   STH2,   /* SINES OF ANGLES      */
        TN1,    TN2,    D1,     D2,     /* INTERMEDIATE TERMS   */

        A11,    A12,                    /* INVERSE              */
        A21,    A22,    A23,            /*      JACOBEAN        */
        A31,    A32,    A33,            /*               MATRIX */

        XRC,    YRC,    ZRC,            /* RECT. RATE COMMANDS  */
        VOUT,                           /* MOTOR OUTPUT VOLTAGE */
        YSIGN,                          /* LEFT SIDE CORRECTION */
        MAXTIME,                        /* MAX DT FOR FILTER    */
        DTF,                            /* FILTER DT            */
        XFE,    YFE,    ZFE,            /* RECT. FORCE ERROR    */

        XFORCE, YFORCE, ZFORCE          /* FOOT COORD. FORCES   */
                        : REAL;


        XHIP                            /* X COORD. HIP TO C.G. */
                        : ARRAY6;


        I,      J,                      /* LOOP INDEX           */
        CHAN0,  CHAN1,  CHAN2           /* A/D CHANNEL POINTERS */
                        : INTEGER;

        FORCE,                          /* FORCE INPUT ARRAY    */
        POSITION,                       /* ACT. JOINT POSITIONS */
        RATE,                           /* ACTUAL JOINT RATES   */
        RATECOM,                        /* COMMAND JOINT RATES  */
        VOLT                            /* MOTOR VOLTAGE OUTPUT */
                        : ARRAY18;


PROCEDURE  RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
        EXTERNAL;

PROCEDURE  RBDAC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VOLT: ARRAY18 );
        EXTERNAL;

FUNCTION SIGN( X: REAL): REAL;  EXTERNAL;
```

```
BEGIN  /* JSERVO */

XHIP[1] := 22.75;       XHIP[2] := 22.75;       XHIP[3] := 0.0;
XHIP[4] := 0.0;         XHIP[5] := -22.75;      XHIP[6] :=-22.75;

/* CONDITION DT FOR FAST FILTER */
MAXTIME := 1.0 / ( FILTPOLE * 4.0 );
IF DT > MAXTIME  { THEN FILTER DEGENERATES }
    THEN DTF := MAXTIME
    ELSE DTF := DT;

FOR I := 1 TO 6 DO
    BEGIN
    IF I IN [1,3,5]
        THEN YSIGN := -1.0
        ELSE YSIGN := 1.0;

    CHAN0 := (I-1) * 3;
    CHAN1 := CHAN0 + 1;
    CHAN2 := CHAN0 + 2;

    RBADC(3, 36+CHAN0, PSCALE, POSITION);  { READ JOINT POSITION }
    RBADC(3, CHAN0, FSCALE, FORCE);        { READ FOOT FORCES }

    THETA2 := POSITION[CHAN0];
    THETA1 := POSITION[CHAN1];
    PSI := POSITION[CHAN2];

    /* COMPUTE TRUE FORCES IN LEG COORDINATES */
    XFORCE := FORCE[CHAN0] - ZEROFORCE[CHAN0];
    YFORCE := -(FORCE[CHAN1] - ZEROFORCE[CHAN1]);
    ZFORCE := ZFSCALE * (FORCE[CHAN2] - ZEROFORCE[CHAN2]);

    /* COMPUTE & SAVE ALL NECESSARY TRIG. FUNCTIONS */
    CPSI := COS(PSI);
    SPSI := SIN(PSI);
    CTH1 := COS(THETA1);
    STH1 := SIN(THETA1);
    CTH2 := COS(THETA2);
    STH2 := SIN(THETA2);
    THETA := THETA1+THETA2;
    STH := SIN(THETA);
    CTH := COS(THETA);
    TN1 := L1*CTH1+L5*STH1+L2*STH;
    D1 := L4+TN1;
    D2 := L2*(L1*CTH2-L5*STH2);
    TN2 := TN1/D2;

    /* COMPUTE THE ACTUAL RECTILINEAR FOOT POSITION */
    XPA[I] :=  D1*SPSI - L3*CPSI + XHIP[I];
    YPA[I] := (D1*CPSI + L3*SPSI + YHIP) * YSIGN;
    ZPA[I] := -L1*STH1 + L2*CTH + L5*CTH1;
```

```
/* ROTATE FORCES TO BODY COORDINATES */
XFA[I] :=  SPSI*CTH          * XFORCE
         - YSIGN*CPSI        * YFORCE
         + SPSI*STH          * ZFORCE;

YFA[I] :=  YSIGN*CPSI*CTH    * XFORCE
         + STH               * YFORCE
         + YSIGN*CPSI*STH    * ZFORCE;

ZFA[I] := -STH               * XFORCE
         + CTH               * ZFORCE;


IF MOVE = TRUE THEN  { SERVO THE HEXAPOD }
    BEGIN

    IF SPRING = FALSE  { THE FORCE TERM MUST BE ZEROED }
        THEN
            BEGIN
            XFF[I] := 0.0;
            YFF[I] := 0.0;
            ZFF[I] := 0.0;
            END
        ELSE
            IF  PASS1 = TRUE   { THE FILTER MUST BE INITIALIZED }
            THEN
                BEGIN
                XFF[I] := 0.0;
                YFF[I] := 0.0;
                ZFF[I] := 0.0;
                END
            ELSE
                BEGIN
                /* COMPUTE THE FORCE ERRORS */
                XFE := XFD[I] - XFA[I];
                YFE := YFD[I] - YFA[I];
                ZFE := ZFD[I] - ZFA[I];

                /* LIMIT FORCE ERRORS TO GIVE 4° MAX. DEFLECTION */
                IF ABS(XFE) > 4.0 * PGAINX/FGAINX
                    THEN XFE := 4.0 * PGAINX/FGAINX * SIGN( XFE );

                IF ABS(YFE) > 4.0 * PGAINY/FGAINY
                    THEN YFE := 4.0 * PGAINY/FGAINY * SIGN( YFE );

                IF ABS(ZFE) > 4.0 * PGAINZ/FGAINZ
                    THEN ZFE := 4.0 * PGAINZ/FGAINZ * SIGN( ZFE );


                /* LOW-PASS FILTER THE FORCE ERRORS */
                XFF[I] := XFF[I] + FILTPOLE * (XFE - XFF[I]) * DTF;
                YFF[I] := YFF[I] + FILTPOLE * (YFE - YFF[I]) * DTF;
                ZFF[I] := ZFF[I] + FILTPOLE * (ZFE - ZFF[I]) * DTF;
```

```
                    END;  { ELSE }

                { END IF PASS1 }

            { END ELSE }

        { END IF SPRING }

    /* COMPUTE RECTILINEAR FOOT RATE COMMAND */
    XRC :=  XRD[I] + PGAINX * (XPD[I] - XPA[I]) + FGAINX * XFF[I];
    YRC := (YRD[I] + PGAINY * (YPD[I] - YPA[I]) + FGAINY * YFF[I]) * YSIGN;
    ZRC :=  ZRD[I] + PGAINZ * (ZPD[I] - ZPA[I]) + FGAINZ * ZFF[I];

    /* COMPUTE INVERSE JACOBIAN MATRIX */
    A11 := CPSI/D1;
    A12 := -SPSI/D1;
    A21 := -(L2*SPSI*STH)/D2+L2*L3*STH*CPSI/D1/D2;
    A22 := -L2*STH*CPSI/D2-L2*L3*STH*SPSI/D1/D2;
    A23 := -L2*CTH/D2;
    A31 := (SPSI-L3*CPSI/D1)*TN2;
    A32 := (CPSI+L3*SPSI/D1)*TN2;
    A33 := (-L1*STH1+L2*CTH+L5*CTH1)/D2;

    /* COMPUTE JOINT RATE COMMAND */
    RATECOM[CHAN2] := A11*XRC + A12*YRC;
    RATECOM[CHAN1] := A21*XRC + A22*YRC + A23*ZRC;
    RATECOM[CHAN0] := A31*XRC + A32*YRC + A33*ZRC;


    /*** RATE SERVO SECTION ***/

    RBADC(3,18+CHAN0,RSCALE,RATE);    { FETCH ACTUAL JOINT RATES }

    FOR J := CHAN0 TO CHAN2 DO
        BEGIN
        VOUT := COMPGAIN * (RATECOM[J] - RATE[J]);
        IF VOUT > 9.8 THEN  VOUT := 9.8
            ELSE IF VOUT < -9.8 THEN VOUT := -9.8;
        VOLT[J] := VOUT;
        RBDAC(1,J,VSCALE,VOLT);  { OUTPUT THE VOLTAGE }

        END;  { DO }

    END;  { IF MOVE }

END;  { DO }

PASS1 := FALSE;  { FLAG THAT FILTERS ARE INITIALIZED }

END;    { JSERVO }
```

```
(**E+*)

        /*******     FILE:   INIT35.PAS    *******/


                /*******************************/
                /*   PROCEDURE:  NORMALIZE     */
                /*******************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        30-MAR-82                                                */
/* FUNCTION:    MOVE HEXAPOD TO NORMALIZED POSITION                      */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              NORMALIZE;                                               */
/*                                                                       */
/*                                                                       */
/* PROCEDURES CALLED:   JSERVO, HALT, FOOTLINE                           */
/*                      TURNON, RBADC                                    */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*      REFERENCED:     XPA,    YPA,    ZPA                              */
/*                      MIDSTX, MIDSTY, MIDSTZ                           */
/*                                                                       */
/*      MODIFIED:       ZEROFORCE, FZERO                                 */
/*                                                                       */
/*************************************************************************/

PROCEDURE  NORMALIZE;


VAR     FOOT                    : INTEGER;  { FOOT INDEX        }
        FRSTCH                  : INTEGER;  { FIRST A/D CHANNEL }
        XCOORD, YCOORD, ZCOORD  : ARRAY6;   { FOOT COORDINATES  }


PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );
        EXTERNAL;

PROCEDURE RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
        EXTERNAL;

PROCEDURE HALT;  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;
```

```
BEGIN
    WRITELN;
    WRITELN( CHR(27), '@6 NORMALIZING ...');

    JSERVO( NOMOVE );  { FETCH ACTUAL FOOT POSITIONS }

    TURNON;  { TURN ON MOTOR POWER }

    FOR FOOT := 1 TO 6 DO { INITIALIZE COORDINATES }
        BEGIN
        XCOORD[FOOT] := XPA[FOOT];
        YCOORD[FOOT] := YPA[FOOT];
        ZCOORD[FOOT] := MIDSTZ;
        END;
    FOOTLINE( 3.0, XCOORD, YCOORD, ZCOORD );     { MOVE TO MIDSTANCE Z }


    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ]
            THEN ZCOORD[FOOT] := MIDSTZ - 5.0;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 UP }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ] THEN
            BEGIN
            XCOORD[FOOT] := MIDSTX[FOOT];
            YCOORD[FOOT] := MIDSTY[FOOT];
            END;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 OVER }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ] THEN
            BEGIN
            FRSTCH := ( FOOT - 1 ) * 3;     { COMPUTE FIRST A/D CHANNEL }
            RBADC( 3, FRSTCH, FSCALE, ZEROFORCE ); { READ FORCE OFFSET }
            FZERO[FOOT] := TRUE;          { FLAG THAT ZEROFORCE UPDATED }
            ZCOORD[FOOT] := MIDSTZ;
            END;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 DOWN }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 2, 3, 6 ]
            THEN ZCOORD[FOOT] := MIDSTZ - 5.0;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 2 UP }

    FOOTLINE( 5.0, MIDSTX, MIDSTY, ZCOORD );     { MOVE LEG SET 2 OVER }
```

```
    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 2, 3, 6 ] THEN
            BEGIN
            FRSTCH := ( FOOT - 1 ) * 3;     { COMPUTE FIRST A/D CHANNEL }
            RBADC( 3, FRSTCH, FSCALE, ZEROFORCE ); { READ FORCE OFFSET }
            FZERO[FOOT] := TRUE;           { FLAG THAT ZEROFORCE UPDATED }
            ZCOORD[FOOT] := MIDSTZ;
            END;
    FOOTLINE( 5.0, MIDSTX, MIDSTY, ZCOORD );     { MOVE LEG SET 2 DOWN }

    HALT;

    WRITELN;
    WRITELN( CHR(27), ' NORMALIZATION COMPLETE');
    WRITELN;

END;  /* NORMALIZE */
```

```
                /*******************************/
                /*   PROCEDURE:  INITIALIZE    */
                /*******************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82                                                */
/* FUNCION:     RAISE LEGS TO WALKING POSITION                           */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              INITIALIZE;                                              */
/*                                                                       */
/*                                                                       */
/* PROCEDURES CALLED:   FOOTLINE,  HALT,  TURNON                         */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*      REFERENCED:     BETA, RPHASE                                     */
/*                                                                       */
/*      MODIFIED:       PHASE, SUPPORT, ZPTERM                           */
/*                                                                       */
/*************************************************************************/

PROCEDURE  INITIALIZE;

VAR     FOOT                    : INTEGER;
        TPHASE                  : REAL;
        ZINIT                   : ARRAY6;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;


BEGIN
    WRITELN;
    WRITELN( CHR(27), '&6 INITIALIZING ...');

    SUPPORT := [ 1..6 ];   { START WITH ALL FEET IN SUPPORT PHASE }

    TURNON;  { TURN ON MOTOR POWER }
```

```
        /*** COMPUTE DESIRED INITIAL FOOT HEIGHTS ***/

        FOR FOOT := 1 TO 6 DO
            BEGIN
            ZPTERM[ FOOT ] := 0.0;  { INITIALIZE ATTITUDE CORRECTION TERM }

            IF RPHASE[ FOOT ] < BETA
                THEN
                    ZINIT[ FOOT ] := MIDSTZ
                ELSE
                    BEGIN
                    SUPPORT := SUPPORT - [FOOT]; { REMOVE FOOT FROM SUPPORT SET }
                    TPHASE := ( RPHASE[ FOOT ] - BETA ) / ( 1.0 - BETA );
                    ZINIT[ FOOT ] := MIDSTZ - FOOTLIFT * SIN( TPHASE * PI );
                    END;

                { END IF }

            END; { FOR FOOT }

        FOOTLINE( 5.0, MIDSTX, MIDSTY, ZINIT );    { RAISE LEGS IN TRANSFER PHASE }

        HALT;

        PHASE := 0.0;  { INITIALIZE KINEMATIC CYCLE PHASE }

        WRITELN;
        WRITELN( CHR(27), '$6 INITIALIZATION COMPLETE');
        WRITELN;

    END;  /* INITIALIZE */
```

```
                    /************************/
                    /*  PROCEDURE UPDOWN  */
                    /************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82                                                */
/* FUNCTION:    CHANGE THE BODY ELEVATION                                */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:  UPDOWN( HEIGHT );                */
/*              WHERE 'HEIGHT' IS THE DESIRED BODY ELEVATION             */
/*                                                                       */
/*                                                                       */
/* PROCEDURES CALLED:   FOOTLINE,  HALT,  TURNON                         */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*       REFERENCED:    MIDSTX, MIDSTY                                   */
/*                                                                       */
/*       MODIFIED:  NONE                                                 */
/*                                                                       */
/*************************************************************************/

PROCEDURE  UPDOWN( HEIGHT : REAL );


VAR     FOOT    : INTEGER;      /* FOOT INDEX           */
        ZCOORD  : ARRAY6;       /* FOOT Z COORDINATES   */


PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );
        EXTERNAL;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;

BEGIN
    TURNON;  { TURN ON MOTOR POWER }

    FOR FOOT := 1 TO 6 DO ZCOORD[FOOT] := HEIGHT;

    FOOTLINE( 3.0, MIDSTX, MIDSTY, ZCOORD );   { MOVE TO DESIRED ELEVATION }

    HALT;

END;   { UPDOWN }
```

```
(*$E+*)

        /*******     FILE:   LINE34.PAS    *******/

                /*******************************/
                /*   PROCEDURE:  FOOTLINE    */
                /*******************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82                                                */
/*                                                                       */
/* FUNCTION:    MOVE FEET FROM PRESENT POSITIONS TO                      */
/*              SPECIFIED POSITIONS IN STRAIGHT LINES                    */
/*              WITH ALL MOTIONS COMPLETED SIMULTANEOUSLY                */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              FOOTLINE( VELMAX, XCOORD, YCOORD, ZCOORD );              */
/*                                                                       */
/* PROCEDURES CALLED:   JSERVO                                           */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*      REFERENCED:     XPA,    YPA,    ZPA                              */
/*                                                                       */
/*      MODIFIED:       XPD,    YPD,    ZPD                              */
/*                      XRD,    YRD,    ZRD                              */
/*                      DT,     SPRING                                   */
/*                                                                       */
/*************************************************************************/

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );

CONST GAIN = 2.0;               /* POSITION ERROR TO RATE GAIN  */


VAR ERRORX, ERRORY, ERRORZ      /* POSITION ERROR FROM SETPOINT */
                : ARRAY6;

    PTIME, LTIME,               /* TEMPORARY STORAGE FOR TIME   */
    MAXERR                      /* MAXIMUM OF COORDINATE ERRORS */
                : REAL;

    FOOT                        /* FOOT INDEX                   */
                : INTEGER;


PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;
```

```
BEGIN
    SPRING := FALSE;    { DISABLE ACTIVE COMPLIANCE }
    JSERVO( NOMOVE );    { FETCH ACTUAL POSITIONS }

    FOR FOOT := 1 TO 6 DO { INITIALIZE DESIRED POSITIONS }
        BEGIN
        XPD[FOOT] := XPA[FOOT];
        YPD[FOOT] := YPA[FOOT];
        ZPD[FOOT] := ZPA[FOOT];
        END;

    LTIME    := TIME;
    DT       := 0.01;

    REPEAT
        MAXERR := 0.0;

        FOR FOOT := 1 TO 6 DO
            BEGIN

            ERRORX[FOOT] := XCOORD[FOOT] - XPA[FOOT];
            ERRORY[FOOT] := YCOORD[FOOT] - YPA[FOOT];
            ERRORZ[FOOT] := ZCOORD[FOOT] - ZPA[FOOT];

            IF ABS( ERRORX[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORX[FOOT] );
            IF ABS( ERRORY[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORY[FOOT] );
            IF ABS( ERRORZ[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORZ[FOOT] );

            END;  /* FOR FOOT */

        FOR FOOT := 1 TO 6 DO
            BEGIN
            IF ( GAIN * MAXERR ) < VELMAX { INCHES PER SECOND }
                THEN BEGIN
                     XRD[FOOT] := GAIN * ERRORX[FOOT];
                     YRD[FOOT] := GAIN * ERRORY[FOOT];
                     ZRD[FOOT] := GAIN * ERRORZ[FOOT];
                     END
                ELSE BEGIN
                     XRD[FOOT] := VELMAX * ERRORX[FOOT] / MAXERR;
                     YRD[FOOT] := VELMAX * ERRORY[FOOT] / MAXERR;
                     ZRD[FOOT] := VELMAX * ERRORZ[FOOT] / MAXERR;
                     END;

            XPD[FOOT] := XPD[FOOT] + XRD[FOOT] * DT;
            YPD[FOOT] := YPD[FOOT] + YRD[FOOT] * DT;
            ZPD[FOOT] := ZPD[FOOT] + ZRD[FOOT] * DT;

            END;  /* FOR FOOT */

        JSERVO( MOVE );
```

```
            PTIME    := TIME;
            DT       := (PTIME - LTIME) * 3600.0;
            LTIME    := PTIME;

        UNTIL MAXERR < 0.5;

        FOR FOOT := 1 TO 6 DO { CLEAN UP GLOBAL VARIABLES }
            BEGIN
            XRD[FOOT] := 0.0;
            YRD[FOOT] := 0.0;
            ZRD[FOOT] := 0.0;

            XPD[FOOT] := XCOORD[FOOT];
            YPD[FOOT] := YCOORD[FOOT];
            ZPD[FOOT] := ZCOORD[FOOT];
            END;

    END;  /* FOOTLINE */
```

```
                /****************************/
                /*    PROCEDURE:  HALT      */
                /****************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82                                                */
/*                                                                       */
/* FUNCTION:    STOPS ALL MOTION OF HEXAPOD                              */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              HALT;                                                    */
/*                                                                       */
/* PROCEDURES CALLED:   RBDAC                                            */
/*                                                                       */
/* GLOBAL VARIABLES                                                      */
/*      REFERENCED: NONE                                                 */
/*                                                                       */
/*      MODIFIED:       VELX,   VELY,   DPSI                             */
/*                      NVELX,  NVELY,  NDPSI                            */
/*                                                                       */
/*************************************************************************/

PROCEDURE HALT;

CONST VSCALE = 10.0;

VAR     JOINT   :INTEGER;       /* JOINT INDEX          */
        VOLT    :ARRAY18;       /* OUTPUT VOLTAGES      */

PROCEDURE  RBDAC( NOCHAN, FRSTCH: INTEGER;  SCALE: REAL;  VOLT: ARRAY18 );
        EXTERNAL;

PROCEDURE TURNOF;  EXTERNAL;

BEGIN
    FOR JOINT:= 0 TO 17 DO
        VOLT[JOINT] :=0.0;

    RBDAC( 18, 0, VSCALE, VOLT);   /* TURN OFF VOLTAGE TO ALL MOTORS */

    TURNOF;    /* TURN OFF MOTOR POWER */

    /*** INITIALIZE VELOCITY VECTORS ***/
    NVELX   := 0.0;   NVELY   := 0.0;   NDPSI   := 0.0;
    VELX    := 0.0;   VELY    := 0.0;   DPSI    := 0.0;

    WRITELN;
    WRITELN( CHR(27),'&6---HEXAPOD STOPPED---');

END;  /* HALT */
```

```
(*$E+*)

             /******  FILE:  LIBR34.PAS  ******/


                  /*************************/
                  /*   FUNCTION:  ATAN2    */
                  /*************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82.                                               */
/* FUNCTION:    IMPLEMENT THE FOUR - QUADRANT                            */
/*              INVERSE TANGENT FUNCTION                                 */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              ATAN2(Y, X);                                             */
/*              WHERE:  Y IS THE SIDE OPPOSED TO THE ANGLE               */
/*                      X IS THE SIDE ADJACENT TO THE ANGLE              */
/*                                                                       */
/*************************************************************************/

FUNCTION  ATAN2(Y, X : REAL) : REAL;


FUNCTION SIGN( X: REAL ): REAL; EXTERNAL;


BEGIN

IF ABS(X) > 0.00001
    THEN   { X IS NON-ZERO }
        IF X > 0.0
            THEN
                ATAN2 := ARCTAN(Y/X)
            ELSE
                ATAN2 := ARCTAN(Y/X) + PI * SIGN( Y )
        { END IF X }
    ELSE
        ATAN2 := PI / 2.0 * SIGN( Y );

END;   /* ATAN2 */
```

```
                    /************************/
                    /*   FUNCTION:  SIGN    */
                    /************************/

/*************************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                              */
/* DATE:        20-APR-82                                                */
/* FUNCTION:    IMPLEMENT THE SIGNUM FUNCTION                            */
/*                                                                       */
/* USER GUIDE:  THE CALLING FORMAT IS:                                   */
/*              SIGN( X );                                               */
/*                                                                       */
/*************************************************************************/

FUNCTION  SIGN( X : REAL ) : REAL;

BEGIN
    IF X >= 0.0
        THEN SIGN := 1.0
        ELSE SIGN := -1.0
END;
```

```
(*$E+*)

            /******  FILE:  DLNK34.PAS  ******/


                  /************************/
                  /*  PROCEDURE:  RBADC  */
                  /************************/

/*********************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                          */
/* DATE:        20-APR-82                                            */
/* FUNCTION:    READ IN INPUT VARIABLES: FORCE, RATE, &              */
/*              POSITION.                                            */
/*                                                                   */
/* USER GUIDE:  THE CALLING FORMAT IS:                               */
/*              RBADC( NOCHAN, FRSTCH, SCALE, INDATA );              */
/*            WHERE:                                                 */
/*              NOCHAN IS THE NUMBER OF CHANNELS TO BE READ          */
/*              FRSTCH IS THE FIRST CHANNEL TO BE READ:              */
/*                0 FOR READING FORCE VARIABLES                      */
/*                18 FOR READING JOINT RATES                         */
/*                36 FOR READING JOINT ANGLES                        */
/*              SCALE IS THE INPUT SCALING FACTOR                    */
/*              INDATA IS THE INPUT BUFFER                           */
/*                                                                   */
/*                                                                   */
/* PROCEDURES CALLED: NONE                                           */
/*                                                                   */
/* GLOBAL VARIABLES: NONE                                            */
/*                                                                   */
/*********************************************************************/

PROCEDURE RBADC(NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18);

VAR     I, J                    /* DATA POINTERS */
                : INTEGER;

        ADPORT  ORIGIN 166000B  /* LOCATION OF DATA LINK SHARED MEMORY */

                : ARRAY[ 0..63 ] OF INTEGER;


BEGIN  /* RBADC */

J := FRSTCH MOD 18;     /* OFFSET FROM START OF VARIABLE TYPE */

FOR  I := J TO (J + NOCHAN - 1) DO  { CONVERT & SCALE DATA }

    INDATA[I] := (777B - ADPORT[ FRSTCH-J + I ]) / 777B * SCALE;

END;   /* RBADC */
```

```
                    /************************/
                    /*  PROCEDURE:  RBDAC  */
                    /************************/

/*********************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                          */
/* DATE:        20-APR-82                                            */
/* FUNCTION:    SEND OUTPUT VOLTAGES TO DATA LINK                    */
/*                                                                   */
/* USER GUIDE:  THE CALLING FORMAT IS:                               */
/*              RBDAC( NOCHAN, FRSTCH, SCALE, OUTDATA )              */
 *            WHERE:                                                 */
/*              NOCHAN IS THE NUMBER OF OUTPUT CHANNELS              */
/*              FRSTCH IS THE FIRST CHANNEL TO BE OUTPUT             */
/*              SCALE IS THE OUTPUT SCALING FACTOR                   */
/*              OUTDATA IS THE OUTPUT BUFFER                         */
/*                                                                   */
/*                                                                   */
/* PROCEDURES CALLED: NONE                                           */
/*                                                                   */
/* GLOBAL VARIABLES: NONE                                            */
/*                                                                   */
/*********************************************************************/


PROCEDURE RBDAC(NOCHAN ,FRSTCH: INTEGER; SCALE: REAL; OUTDATA: ARRAY18);

CONST   READY = 200B;

VAR     I,                          /* DATA POINTER                 */
        TEMP                        /* TEMPORARY STORAGE FOR OUTPUT */
                : INTEGER;

        STATUS  ORIGIN 166244B  /* DATA LINK STATUS WORD            */
                : INTEGER;

        OUTPORT ORIGIN 166200B  /* DATA LINK OUTPUT PORTS           */
                : ARRAY[0..17] OF INTEGER;


BEGIN  /* RBDAC */

FOR I := FRSTCH TO (FRSTCH + NOCHAN - 1) DO
    BEGIN
    TEMP := 177B - ROUND(OUTDATA[I] * 177B / SCALE);
    WHILE (STATUS AND READY) = 0 DO { WAIT UNTIL DATA LINK READY };
    OUTPORT[I] := TEMP;
    END;   /* DO */

END;  /* RBDAC */
```

```
PROCEDURE CLOCKINIT;

VAR     STATUS  ORIGIN 170404B          : INTEGER;
        PRELOAD ORIGIN 170406B          : INTEGER;
        CLOCK   ORIGIN 170416B          : INTEGER;

BEGIN
PRELOAD := 0;  { USE FULL COUNTER INTERVAL }
STATUS := 413B;  { ENABLE COUNTER AT 100 Hz }
LASTCLOCK := CLOCK;
TOTALCLOCK := 0;
END;



FUNCTION DELTATIME : REAL;

VAR     CLOCK   ORIGIN 170416B          : INTEGER;
        NOWCLOCK                        : INTEGER;
        NUMTICKS                        : INTEGER;

BEGIN

NOWCLOCK := CLOCK;
NUMTICKS := ( NOWCLOCK - LASTCLOCK ) AND 377B;
TOTALCLOCK := TOTALCLOCK + NUMTICKS;
LASTCLOCK := NOWCLOCK;

DELTATIME := NUMTICKS / 100.0;

END;
```

```
(*$E+*)

                /******   FILE:  POWR34.PAS   ******/

/*****************************************************************************/
/*                                                                           */
/*      PROGRAMMER: DENNIS PUGH                                              */
/*      DATE: 20-APR-82                                                      */
/*                                                                           */
/*      GLOBAL VARIABLES: NONE                                               */
/*                                                                           */
/*  THIS FILE CONTAINS SUBROUTINES WHICH UTILIZE THE SELF-DIAGNOSTIC         */
/*  CAPABILITIES AND THE REMOTE POWER CONTROL CAPABILITIES OF THE            */
/*  DIGITAL DATA LINK. THE EXTERNALLY USEFUL ROUTINES ARE 'TESTON',          */
/*  'TURNOF', AND 'TURNON'. NONE OF THESE REQUIRE ARGUMENTS.                 */
/*                                                                           */
/*****************************************************************************/




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

PROCEDURE PWAIT;    /* A ROUTINE WHICH WAITS UNTIL ALL FEEDBACK CHANNELS */
                    /* HAVE BEEN UPDATED AFTER A CHANGE IN STATUS        */
VAR NOW:REAL;

BEGIN
    NOW:=TIME;
    WHILE ((TIME-NOW)*3600.) < 0.004 (* SECONDS *) DO (* NOTHING *) ;
END;  /* PWAIT */

/*****************************************************************************/
```

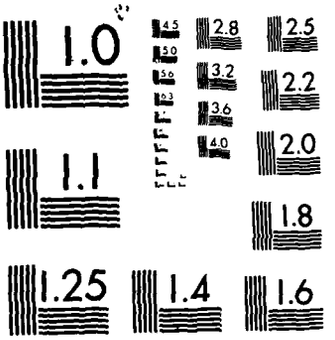
```
/*****************************************************************************/
/*                                                                           */
/* TURNOF:                                                                   */
/*      (TURN OFF) IS A ROUTINE WHICH TURNS OFF THE POWER TO THE             */
/*      SERVO MOTORS. IT SHOULD BE USED BEFORE EXITING THE CONTROL           */
/*      PROGRAM, AND ADDITIONALLY, IT SHOULD BE CALLED WHENEVER THE          */
/*      HEXAPOD IS NOT ACTUALLY SERVOING. THIS WILL INCREASE SAFETY          */
/*      AND REDUCE THE RISK OF DAMAGE TO THE HEXAPOD. THE EFFECTS            */
/*      OF CALLING TURNOF CAN BE REVERSED BY CALLING TURNON.                 */
/*                                                                           */
/*****************************************************************************/

PROCEDURE TURNOF;

(*$C    COMSTAT=^O166244        ;ADDRESS OF COMMAND & FORWARD STATUS
       POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD

       .MACRO SEND Y,Z,?T
T:     TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
       BPL     T
       MOV     Y,Z             ;SEND DATA
       .ENDM
*)

BEGIN

(*$C    SEND    #0,COMSTAT      ;COMMAND NORMAL MODE
       SEND    #2,POWER        ;TURN OFF MOTOR POWER
*)

END;   /* PROCEDURE TURNOF */

/*****************************************************************************/
```

```
/*************************************************************************/

PROCEDURE ASK;   /* A PROCEDURE WHICH GIVES THE OPTION TO ABORT */

(*$C   .MCALL  EXIT$S  *)

VAR LETTER:CHAR;

BEGIN
    WRITELN('DO YOU WISH TO ABORT?  (Y/N)');
    READLN(LETTER);
    IF LETTER = 'Y' THEN
        BEGIN
        WRITELN;
        WRITELN(' *** PROGRAM ABORTED ***');
        TURNOF;
        (*$C EXIT$S *)  /* EXIT THE PROGRAM */
        END;
END;  /* PROCEDURE ASK */

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




/*************************************************************************/
/*                                                                       */
/*  TURNON:                                                              */
/*      (TURN ON) IS A ROUTINE WHICH TURNS ON THE HEXAPOD MOTOR &        */
/*      ELECTRONICS POWER. IT ASSUMES THAT 'TESTON' HAS BEEN CALLED      */
/*      SUCCESSFULLY. IT SHOULD BE CALLED AT THE START OF EVERY          */
/*      REAL TIME SECTION (WHEN THE HEXAPOD IS ACTIVELY SERVOING).       */
/*                                                                       */
/*************************************************************************/

PROCEDURE TURNON;

(*$C    COMSTAT=^O166244        ;ADDRESS OF COMMAND & STATUS WORDS
        POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD
        PSTAT=^O166176          ;ADDRESS OF POWER STATUS WORD

        .MACRO SEND Y,Z,?T
T:      TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
        BPL     T
        MOV     Y,Z             ;SEND DATA
        .ENDM
*)
```

```
BEGIN

        (*$C    BIT     #4,PSTAT                ;TEST KILL CIRCUIT STATUS
                BEQ     KIL                     ;BRANCH IF INACTIVE
        *)

        WRITELN;
        WRITELN('THE KILL CIRCUIT IS ACTIVE, PLEASE DEACTIVATE.');

        (*$C
        WAITK:  BIT     #4,PSTAT        ;TEST KILL CIRCUIT
                BNE     WAITK           ;LOOP UNTIL DEACTIVATED
        *)

        WRITELN(' THANK YOU');

        (*$C
        KIL:    SEND    #3,POWER        ;TURN ON ELECTRONICS & MOTOR POWER
        *)

        PWAIT;
        PWAIT;

        (*$C    MOV     @#PSTAT,R1      ;CHECK FOR MP & EP ON
                BIC     #^O177774,R1    ;MASK OFF MP & EP STATUS BITS
                CMP     #3,R1           ;CHECK IF BOTH BITS ON
                BEQ     DONE            ;BRANCH IF CORRECT
        *)

        WRITELN;
        WRITELN(' POWER CONTROLLER MALFUNCTION');
        ASK;
        WRITELN;
        WRITE('PLEASE SWITCH TO MANUAL MODE,');
        WRITELN(' THEN TURN ON YELLOW & GREEN LIGHTS');

        (*$C
        WAITP:  MOV     @#PSTAT,R1      ;CHECK FOR MP & EP ON
                BIC     #^O177774,R1    ;MASK OFF MP & EP STATUS BITS
                CMP     #3,R1           ;CHECK IF BOTH BITS ON
                BNE     WAITP           ;LOOP UNTIL POWER ON
        *)

        WRITELN;
        WRITELN(' THANK YOU');

        (*$C    DONE:
        *)

END;  /* TURNON */

/*****************************************************************************/
```

AD-A125 466 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. FEB 83 3/3

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL



```
/*************************************************************************/
/*                                                                       */
/*  TESTON:                                                              */
/*      (TEST & TURN ON) IS AN INTERACTIVE PROCEDURE WHICH SHOULD BE     */
/*      CALLED AT THE START OF EVERY REAL-TIME HEXAPOD CONTROL PROGRAM.  */
/*      IT AUTOMATICALLY TESTS DATA LINK OPERATION AND THEN TURNS ON     */
/*      THE POWER SYSTEMS IN THE PROPER SEQUENCE. ANY ABNORMAL           */
/*      CONDITIONS ARE REPORTED TO THE OPERATOR.                         */
/*                                                                       */
/*************************************************************************/

PROCEDURE TESTON;

VAR     I,
        N,
        DATA,
        LOC:
                INTEGER;

(*$C    ADBASE=^O166000         ;BASE ADDRESS OR A/D'S
        DABASE=^O166200         ;BASE ADDRESS OF D/A'S
        DABUF=^O166236          ;D/A CHANNEL USED FOR FORWARD TESTS
        COMSTAT=^O166244        ;ADDRESS OF FORWARD STATUS WORD
                                ;  AND MODE COMMAND WORD
        PSTAT=^O166176          ;ADDRESS OF POWER STATUS WORD
        POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD

        .MACRO SEND Y,Z,?T
T:      TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
        BPL     T
        MOV     Y,Z             ;SEND DATA
        .ENDM
*)

BEGIN

        WRITELN;
        WRITE('PLEASE TURN ON DATA LINK POWER (RED LIGHT),');
        WRITELN(' THEN ENTER A CARRIAGE RETURN.');
        READLN;


        (*******  FEEDBACK TEST ROUTINE  *******)

        WRITELN(' PERFORMING DATA LINK FEEDBACK TEST');

        (*$C    SEND    #0,POWER        ;TURN OFF POWER SYSTEMS
                SEND    #1,COMSTAT      ;COMMAND TEST MODE #1
        *)
```

```
PWAIT;
PWAIT;
N:=0;
FOR I:=1 TO (MAXINT DIV 64) DO
        BEGIN
        (*$C    MOV     #^O77,R1        ;INITIALIZE ADDRESS COUNTER
        LOOP:   ASL     R1              ;CONVERT TO WORD ADDRESS
                MOV     ADBASE(R1),R2   ;FETCH DATA
                ASR     R1              ;RESTORE COUNTER
                MOV     R1,R3           ;GENERATE EXPECTED DATA IN R3
                SWAB    R3              ;  BY DUPLICATING BITS 0-3
                ASR     R3              ;  IN BITS 6-9
                ASR     R3
                ADD     R1,R3
                BIC     #^O176000,R3    ;DATA NOW GENERATED
                CMP     R2,R3           ;CHECK IF DATA IS CORRECT
                BEQ     CONT
                MOV     R1,LOC(SP)      ;SAVE ADDRESS OF BAD DATA
        *)

                N:=N+1;
                IF N<20 THEN
                    BEGIN
                    WRITE(' ERROR DETECTED ON LOOP',I,', ADDRESS',LOC);
                    WRITELN;
                    END;

        (*$C
        CONT:   DEC     R1              ;DECREMENT ADDRESS COUNTER
                BPL     LOOP            ;LOOP UNTIL R1<0
        *)

        END;  /* FOR I */

IF N <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN(N,' TRANSMISSION ERRORS DETECTED IN',MAXINT,' TESTS.');
        ASK;
        END;


(*******  FEEDFORWARD TEST ROUTINE  *******)

WRITELN;
WRITELN(' PERFORMING DATA LINK FEEDFORWARD TEST');

(*$C     SEND    #2,COMSTAT      ;COMMAND TEST MODE #2
*)
```

```
N:=0;
PWAIT;
FOR DATA:=0 TO 127 DO
        BEGIN
        (*$C    SEND    DATA(SP),DABUF  ;SEND DATA
        *)

        PWAIT;

        (*$C    MOV     @#ADBASE,R1     ;READ BACK DATA
                CMPB    R1,DATA(SP)     ;COMPARE DATA SENT & RECIEVED
                BEQ     GOOD            ;BRANCH IF DATA GOOD
                INC     N(SP)           ;INCREMENT ERROR COUNTER
        GOOD:   COM     DATA(SP)        ;FLIP BITS FOR NEXT OUTPUT
                SEND    DATA(SP),DABUF  ;SEND DATA
        *)

        PWAIT;

        (*$C    MOV     @#ADBASE,R1     ;READ BACK DATA
                CMPB    R1,DATA(SP)     ;COMPARE DATA SENT & RECIEVED
                BEQ     GOOD2           ;BRANCH IF DATA GOOD
                INC     N(SP)           ;INCREMENT ERROR COUNTER
        GOOD2:  COM     DATA(SP)        ;RESTORE DATA
        *)

        END;  /* FOR DATA */

IF N <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN(N,' TRANSMISSION ERRORS DETECTED IN 256 TESTS.');
        ASK;
        END;


(*******  POWER CONTROLLER MODE TEST  *******)

(*$C     SEND    #0,COMSTAT      ;COMMAND NORMAL MODE
*)

PWAIT;
PWAIT;

(*$C     TSTB    PSTAT   ;TEST IF POWER CONTROLLER IN COMPUTER MODE
        BMI     COMP    ;BRANCH IF IT IS
*)

WRITELN;
WRITELN('PLEASE PLACE POWER CONTROLLER IN COMPUTER MODE.');
```

```
        (*$C
        WAITC:  TSTB    PSTAT           ;TEST FOR COMPUTER MODE
                BPL     WAITC           ;LOOP UNTIL MODE CHANGED
        *)

        WRITELN(' THANK YOU');

        (*$C    COMP:
        *)


        (******* ZERO DAC ROUTINE  *******)

        FOR I:=0 TO 17 DO
                BEGIN
                (*$C    MOV     I(SP),R1        ;MOVE I TO R1
                        ASL     R1              ;CONVERT TO WORD ADDRESS
                        SEND    #^O177,DABASE(R1) ;SEND ZERO VOLTS TO JOINT
                *)
                END;

        TURNON;

        WRITELN;
        WRITELN(' INITIALIZATION COMPLETE');
        WRITELN;

END;   /* TESTON */

/*****************************************************************************/
```

```
;               /*******  FILE:  KYCK34.MAC  ******/
;
; ***************************************************************************
; *                                                                         *
; *  PROGRAMMERS:  CHARLES KLEIN, DENNIS PUGH                               *
; *  DATE:  20-APR-82                                                       *
; *                                                                         *
; *  GLOBAL VARIABLES: NONE                                                 *
; *                                                                         *
; *  PURPOSE:  ASSEMBLER LEVEL KEYBOARD INPUT FOR SPECIAL-PURPOSE           *
; *            USE WITH HIGH LEVEL LANGUAGES.                               *
; *                                                                         *
; *  SUBROUTINES CONTAINED:                                                 *
; *                                                                         *
; *     KEYINT: INITIALIZES TERMINAL CHARACTERISTICS AND PLACES AN          *
; *             INPUT REQUEST ON THE SYSTEM QUEUE. KEYINT SHOULD BE         *
; *             CALLED ONLY ONCE. NO ARGUMENTS ARE REQUIRED.                *
; *                                                                         *
; *     KEYCK:  RETURNS THE CHARACTER TYPED ON THE KEYBOARD IN ITS          *
; *             ONE ARGUMENT (VARIABLE TYPE CHAR). IF NO CHARACTER HAS      *
; *             BEEN INPUT SINCE THE LAST CALL, A NULL CHARACTER IS         *
; *             RETURNED (ASCII 0).  SAMPLE CALL: KEYCK(CHARACTER);        *
; *                                                                         *
; *     KYWAIT: IS TO BE CALLED WHEN IT IS DESIRED TO SUSPEND PROGRAM       *
; *             EXECUTION UNTIL A CHARACTER IS INPUT, THUS FREEING THE      *
; *             SYSTEM FOR OTHER TASKS. EXECUTION IS RESUMED WHEN AN        *
; *             INPUT IS RECIEVED. KEYCK CAN THEN BE USED TO FETCH THE      *
; *             CHARACTER. NO ARGUMENTS ARE REQUIRED.                       *
; *                                                                         *
; *     IOKILL: IS USED TO CANCEL THE INPUT REQUEST WHEN A HIGH-LEVEL       *
; *             READ (READLN) IS TO BE PERFORMED. NO ARGUMENTS ARE          *
; *             REQUIRED.                                                   *
; *                                                                         *
; *     INQUE:  IS CALLED TO REASSERT THE INPUT REQUEST AFTER AN IOKILL     *
; *             AND HIGH-LEVEL READ. NO ARGUMENTS ARE REQUIRED.             *
; *                                                                         *
; *                                                                         *
; *         *** ALL SUBROUTINES ARE IN FORTRAN FORMAT. ***                  *
; *                                                                         *
; ***************************************************************************
;
;
        .LIST   TTM
        .MCALL  QIO$S,WTSE$S,ALUN$S
;
; LOCAL SYMBOL DEFINITION
        LUN2    = 2
        EFN1    = 11
        EFN2    = 12
;
```

```
;
; LOCAL DATA BLOCKS:
;
        .PSECT  DATA,D,RW
MSG:    .BLKB   1               ;CHARACTER BUFFER
        .EVEN
TCHAR:  .BYTE   TC.FDX,1        ;TERMINAL CHARACTERISTICS
IOST:   .BLKW   2               ;I/O STATUS RETURN LOCATION
;
        .PSECT
;
KEYINT::
        ALUN$S  #LUN2,#'TI                              ;ASSIGN LUN TO TERMINAL
        QIO$S   #IO.ATT,#LUN2                           ;ATTACH TERMINAL
        QIO$S   #SF.SMC,#LUN2,,,,,<#TCHAR,#2>           ;SET TO FULL DUPLEX
        QIO$S   #IO.RNE,#LUN2,#EFN2,,#IOST,,<#MSG,#1>   ;ASSERT INPUT REQUEST
        BCS     ERROR                                   ;SIGNAL IF ERROR
        RTS     PC
;
;
KEYCK:: TST     (R5)+                                   ;POINT R5 TO ARGUMENT
        CLRB    @0(R5)                                  ;PUT NULL IN ARGUMENT
        CMPB    #IS.SUC,IOST                            ;TEST FOR SUCCESSFUL READ
        BNE     RETURN                                  ;RETURN IF NOT
        MOVB    MSG,@0(R5)                              ;PUT CHAR. IN ARGUMENT
INQUE:: QIO$S   #IO.RNE,#LUN2,#EFN2,,#IOST,,<#MSG,#1>   ;REASSERT INPUT REQUEST
        BCS     ERROR                                   ;SIGNAL IF ERROR
RETURN: RTS     PC
ERROR:  MOV #2,R0                                       ;SET ERROR FLAG
        IOT
;
;
KYWAIT::
        WTSE$S  #EFN2           ;WAIT FOR KEYBOARD INPUT WITH LITTLE OVERHEAD
        RTS     PC
;
;
IOKILL::
        QIO$S   IO.KIL,#LUN2,#EFN1      ;CANCEL I/O REQUESTS
        WTSE$S  #EFN1                   ;WAIT FOR COMPLETION
        RTS     PC
;
        .END
```

```
;          FILE: CMPL34.CMD
;
; FUNCTION: INDIRECT COMMAND FILE TO
;           COMPILE & ASSEMBLE ALL FILES
;           OF ROBOT 3.4
;
PAS ROBT34=GBLF34,ROBT34
MAC ROBT34=ROBT34
PAS PLAN34=GBLF34,PLAN34
MAC PLAN34=PLAN34
PAS FOOT34=GBLF34,FOOT34
MAC FOOT34=FOOT34
PAS SERV34=GBLF34,SERV34
MAC SERV34=SERV34
PAS INIT34=GBLF34,INIT34
MAC INIT34=INIT34
PAS LINE34=GBLF34,LINE34
MAC LINE34=LINE34
PAS LIBR34=GBLF34,LIBR34
MAC LIBR34=LIBR34
PAS DLNK34=GBLF34,DLNK34
MAC DLNK34=DLNK34
PAS POWR34=POWR34
MAC POWR34=POWR34
MAC KYCK34=KYCK34
TKB @ROBT34
```

```
;       FILE: ROBT34.CMD
;
; FUNCTION: PROVIDES LINKAGE INFORMATION
;           FOR THE TASK BUILDER
;
ROBT34/FP,ROBT34/CR/-SP=ROBT34
PLAN34
FOOT34
LIBR34
SERV34
DLNK34
INIT34
LINE34
KYCK34
POWR34
[1,1]PASLIB/LB
/
COMMON=IOPAGE:RW
extsct=$$heap:5000
UNITS=7
//
```

REFERENCES

1 Bekker, M.G., Introduction to Terrain-Vehicle Systems. Ann Arbor, MI: Univ. Michigan, 1969.

2 Mosher, R.S., "Exploring the Potential of a Quadruped," presented at Inter. Automotive Engineering Conf., Detroit, MI, SAE Paper No. 690191, Jan., 1969.

3 McGhee, R.B., "Control of Legged Locomotion Systems," in Proc. 1977 Joint Autom. Contr. Conf., San Francisco, CA, pp. 205-215, June, 1977.

4 Klein, C.A., and Briggs, R.L., "Use of Active Compliance in the Control of Legged Vehicles," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-10, No. 7, July, 1980.

5 Buckett, J.R., Design of an On-board Electronic Joint Control System for a Hexapod Vehicle, M.S. thesis, Ohio State University, Columbus, OH, March, 1977.

6 Briggs, R.L., Real-Time Digital Control of an Electrically Powered Vehicle Leg Using Vector Force Feedback, M.S. thesis, Ohio State University, Columbus, OH, December, 1977.

7 Briggs, R.L., A Real-Time Digital System for Control of a Hexapod Vehicle Utilizing Force Feedback, Ph.D. dissertation, Ohio State University, Columbus, OH, June, 1979.

8 Sun, S.S., A Theoretical Study of Gaits for Legged Locomotion Systems, Ph.D. dissertation, Ohio State University, Columbus, OH, March, 1974.

9 Chao, C.S., Real-Time Multiprocessor Control of a Hexapod Vehicle, Ph.D. dissertation, Ohio State University, Columbus, OH, Aug., 1979.

10 Orin, D.E., "Supervisory Control of a Multilegged Robot," Robotics Research, Vol. 1, No. 1, Spring, 1982.

11 Paul, R.P., Robot Manipulators: Mathematics, Programming, and Control, M.I.T. Press, Cambridge, Massachusetts, 1981.

12 Maney, J.J., A Comparative Study of Real-Time Computer Control Techniques for a Multi-Jointed Linkage Mechanism, M.S. thesis, Ohio State University, Columbus, OH, June, 1980.

13 Vohnout, V.J., Mechanical Design of an Energy Efficient Robotic Leg for Use on a Multi-Legged Walking Vehicle, M.S. thesis, Ohio State University, Columbus, OH, June, 1982.

14 Whitney, D.E., "Force Feedback Control of Manipulator Fine Motions," in Proc. of Seventeenth Joint Autom. Contr. Conf., Purdue University, pp. 694-699, July, 1976.

15 Klein, C.A., and Wahawisan, W., "Use of a Multiprocessor for Control of a Robotic System," The International Journal of Robotics Research, Issue 2, Summer, 1982.