19

A TRIDENT SCHOLAR PROJECT REPORT

NO. 216

"ROBOTIC CONTROL USING MUSCULAR AND NEURAL ELECTRICAL SIGNALS"

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

U.S.N.A. -- Trident Scholar Project Report; No. 216 (1994)

"ROBOTIC CONTROL USING MUSCULAR AND NEURAL ELECTRICAL SIGNALS"

by

Midshipman William M. Gotten, Jr., Class of 1994
U.S. Naval Academy
Annapolis, Maryland

Professor Kenneth A. Knowles
Weapons and Systems Engineering Department

Accepted for Trident Scholar Committee

Francis D Correll
Chair

19 May 1994
Date

REPORT DOCUMENTATION PAGE

Form Approved
OMB no. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
19 May 1994

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Robotic control using muscular and neural electrical signals

5. FUNDING NUMBERS

6. AUTHOR(S)
William M. Gotten, Jr.

7. PERFORMING ORGANIZATIONS NAME(S) AND ADDRESS(ES)
U.S. Naval Academy, Annapolis, MD

8. PERFORMING ORGANIZATION REPORT NUMBER USNA Trident Scholar project report; no. 216 (1994)

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Accepted by the U.S. Trident Scholar Committee

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words) The human body is capable of producing measurable electrical potentials ranging from the nanovolt to millivolt range. The actions needed to produce these potentials can be as simple as blinking an eye or flexing a muscle. The voltages can even be created within and by the brain, although this process is not well understood at the present time. This design project explores the possibility and practicality of harnessing these potentials into signals that can reliably control a robotic arm.

14. SUBJECT TERMS electromyography, muscles, prosthetics, robotics

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Abstract:

The human body is capable of producing measurable electrical potentials ranging from the nanovolt to millivolt range. The actions needed to produce these potentials can be as simple as blinking an eye or flexing a muscle. The voltages can even be created within and by the brain, although this process is not well understood at the present time. This design project explores the possibility and practicality of harnessing these potentials into signals that can reliably control a robotic arm.

Keywords: Electromyography
Muscles
Prosthetics
Robotics

Preface:

There is a particular reason that I chose to work on a project centered around the human body as opposed to, for instance, levitating Tiddly Winks using varying magnetic fields and superglue. The body is the ultimate machine. It is amazing how millions of individual cells work together to take in information, analyze it, and then react to it. The brain alone can do differential calculus faster than the fastest computer; the heart is more reliable than almost any pump ever designed; the algorithm necessary to control one finger would fill books, yet the body can control ten of them independently and effortlessly.

The television series *The Six Million Dollar Man*, which I believe was based in part on a novel called *Cyborg*, caught my attention at an early age. But despite offers from my grandparents to pay my way through Princeton and then through medical school in order to keep me out of the Navy, I realized I could never stand the sound and smell of ripping open bones and guts as part of my job description (unless I was making that happen with laser-guided 500-pound bombs). For this reason, the possibility of combining my study in Control Systems Engineering with my long-held interest in the human body--without having to draw blood--has so much appeal for me. The specific idea of using the muscle and brain signals arose from the fact that working with control input signals has been the major focus of every Control Systems and Electrical Engineering course I have taken at the Naval Academy. These considerations were the primary impetus for designing and carrying out this project.

On a different note--while I am still in that part of the paper where I can write in the first person--I would like to take a little space to thank a number of people for their support in helping me complete this project:

Professor William B. Bennett, U. S. Naval Academy, for his help with the Electrical Engineering aspects of the project.

Professor William I. Clement, U. S. Naval Academy, for his help with some of the Computer Engineering aspects of the project

Dr. Mark Cunningham, Baptist Memorial Hospital, Memphis, TN., for introducing me to the idea of electromyography

Professor Robert A. Demoyer, U. S. Naval Academy, for aiding me in my efforts to download data from the digital oscilloscope in my laboratory

Professor Carl E. Wick, U. S. Naval Academy, for troubleshooting the communications problems I was having with the robot

LT Ziemke, Bancroft Hall Medical Clinic, U. S. Naval Academy, for furthering my understanding of electromyography

Mr. and Mrs. William M. Gotten, Sr. (who never understood the mechanics of the research project, but thought it was great stuff, anyway)

Professor Kenneth A. Knowles (who actually believed I could do it, and then made sure I did)

Miss Jennifer Howard (who knew what this project meant to me)

MIDN 1/C Erik Smith (with whom I could--and did--commiserate. Often.)

Mr. Sam Hawkins (who went into grieving whenever I needed parts, but always made sure I got them)

Captain B. F. "Hawkeye" Pierce, USA (who existed only for a half-hour each night, but brought perspective back each time)

and

Murphy (the robot--who operated just often enough to make me realize that this research project would, eventually, somehow . . . work)

Table of Contents:

Topic: Page:

Appendices (cont'd)

Topic: Page:

Objectives:

- Determine whether the human body can produce effective control signals from the brain and/or muscle groups

- Determine which microscopic body electric signals can be utilized for effective control and how electronics can best be designed to process them

- Implement a robot control system design that can be reliably and predictably controlled by the processed signals

- Make the entire system easy to use

- Determine additional practical applications for the system

Problem Statement:

It is well-known in the medical community that the human body is capable of producing measurable voltage potentials. The familiar electrocardiogram, or EKG, is easily the most familiar device used in this rcspect. However, the heart is not the only organ capable of producing voltages large enough to be picked up by equipment commonly available to the technical community. Individual muscles, entire muscle groups, and the brain are also frequently monitored by instruments designed specifically for such purposes.

This project was initially entitled "Non-Prehensile Telerobotic Control Using Muscular and Neural Means." This foreboding title was chosen in order to express concisely what is, in general, a simple idea. The term "non-prehensile" reflects a self-imposed criteria that the hands and fingers play no role in the signal-generating process. The assumption is that the hands are not available, for one reason or another, to provide any input. "Telerobotic control" means that the control signals for a robot (or any other "controlled" device, for that matter) do not originate from its usual primary controller. In other words, the signals that control it are first created outside the actual controller and then eventually transmitted to it. "Muscular and neural means" are, of course, the manner in which those signals are created in the first place. The ultimate goal, then, is for a person to be able to control various motions of a robot arm by harnessing the measurable voltages created naturally within the body. In theory, for example, if the controller "flexes" the left calf (thereby raising its voltage potential), the robot could respond by extending the arm along its "elbow" axis.

The potential applications of such a device are many. The most obvious employment is in the area of prosthetics. To be used in this manner, muscle signals could be rerouted into the body to stimulate either paralyzed muscles or artificial limbs. In a similar fashion, some industrial robots could be programmed to accept these signals. In fact, this usage would probably require the least amount of modification to the original design. A more aggressive idea would be to implement this design into the control system of an airplane or car. Here, muscle signals could be used to control a steering mechanism, or simply to control various switches in a cockpit or on a dashboard.

Background:

Scientists and medical researchers have long recognized the existence of electrical signals produced naturally by the human body. Once the technology became available to reliably monitor such signals, researchers began looking for ways to use them diagnostically.

Electromyography (EMG) focuses specifically on the monitoring of the signals produced by muscles. Correct analysis of these EMG signals indicate "when the muscles are electrically active" (Grieve, p. 109). Most often, physicians use EMG eith:r to monitor muscle activity or to assess some forms of muscle damage. EMG can also be used in conjunction with research on paralyzed muscles. There are numerous examples of this type of research; Murray (1987) describes one which is an electric walking machine for paraplegics. This particular design uses surface electrodes on the chest for sensors, sends the signal through a stimulator, and then back to the paralyzed legs. Boggs (1992) describes another one for a child's artificial hand that is controlled by inputs from the wrist muscles. The implementation of EMG signals as controllers in these fashions approaches very closely the focus of this project.

Electroencephalography (EEG) focuses specifically on the monitoring of brain waves. This is a topic that, despite many years of the knowledge of the existence of brain waves, has found no completely adequate analysis process. Callaway (1975) has reliably documented some potential difficulties of using EEG:

> Gross relations between the EEG and brain function are quite convincing. However, such findings are controversial. The very value

> of the EEG in neurology lies in the fact that wavelike activities often signal "not operating," with various pathological conditions (such as epilepsy, brain tumors, and toxic states) producing more or less characteristic not-operating signals. The brain does emit job-specific signals when it does something, and such specific signals may or may not be embedded in larger not-operating signals arising from other areas. Generally, for a process to show up in the raw EEG it must, like sleep or epilepsy, involve a large portion of the brain all at once. Unfortunately, right from the beginning it seemed as if such repetitive waves in the EEG told more about what the mind was *not* doing than about what the mind was doing. Berger [the father of electroencephalography] himself observed that 10-cycle alpha waves from over the visual cortex appear when one ceases to use the visual apparatus. The slow waves of sleep disappear when we dream.

Certainly, Callaway's assessment of the possibilities of using brain waves as control signals are very dim indeed. The implications are that by concentrating on modifying brain waves, a user might actually *eliminate* them from the control picture. This result depends, of course, on the ability to *find* the brain waves in the first place.

Method of Research:

To see if electromyography and electroencephalography could be implemented into an effective control design required dividing the project into several distinct phases. The results of each phase determined, to a great degree, the ambitions of the following stages. The first phase of the research was to configure equipment to detect electromyographic signals due to muscle movements. The sensors used in this phase were 10 mm, gold-plated surface electrodes that are commercially available through medical instrument companies. One such electrode is shown in Figure (1). Also necessary during the initial phase of the research was a special conductive gel which is also sold commercially. Combined with the gel, the surface electrodes were able to register a very small signal on a frequency analyzer. One such signal is provided on Appendix (E-2). The addition of supporting analog amplifiers and filters allowed the muscle signals to be picked up much more efficiently.

The second phase of the research was to explore the characteristics of the voltages produced by the muscles. The primary goal of the experimentation with the muscle groups was to find which groups could be used for independent, predictable and dependable control signal patterns despite the presence of noise. A secondary goal was to determine the difficulty of providing these particular input signals. In support of these goals, eight different muscle groups were chosen to be the focus of research. These original groups were the biceps, brachioradialis (along the top of the forearm), vastus medialis (commonly known as the quadriceps), semitendinosus (also known as the hamstring), tibialis anterior (along the upper half of the shin), the gastrocnemius (commonly known as the calf), peroneus longus (a series of muscles just above and outside of the ankle), and the soleus (on top of the foot where the foot joins the lower leg). The exact locations of each of these muscle groups is shown in Figure (2). Unfortunately, four of these groups--the forearm, hamstring, ankle, and shin--were eventually shown not to be effective signal generators for reasons that will be described in detail later. However, the fact that only eight areas were chosen in the first place does not mean that there are not other muscle groups quite capable of producing effective control signals. Further research can be devoted to isolating other areas that might work quite well.

The next phase of the project centered around the robotics mechanism, shown in Figure (3). The robot chosen for this research, a SCORBOT-ER V Plus, is a sophisticated robot arm that has five axes of rotation (shoulder, elbow, wrist pitch, wrist roll, and base rotation) and a gripper that can open and close. First, the robot had to be configured to work with the same hardware and software that supported the muscle signals. This goal was met after the appropriate handshaking signals were found (they are shown in the program in Appendix (AB)). Second, a way of controlling each of the five axes and the gripper had to be developed. The most convenient way to complete this task would be to have one sensor controlling each direction of motion. Since the robot could move two ways along each axis, a minimum of twelve sensors would be needed to implement the design in this way. Unfortunately, results from an earlier phase of research had already suggested that only four muscle groups out of the original eight sampled would provide adequate signals.

Accordingly, only eight areas on the body could be used as signal generators (i.e., one group on each side). One solution to this problem would have been to research other muscle groups to determine their effectiveness. Another would have been simply not to use the axes which seemingly could not be controlled. Rather than accepting the setback of researching more muscle areas, and further, not accepting the loss of two axes of rotation, different *combinations* of sensors were used to control the remaining motions. For example, activating the left biceps could control the robotic arm's shoulder axis in one direction. The right biceps could move it in the other. But both biceps activated together also provided a perfectly viable signal, and thus the apparent problem was solved.

The fo. th stage was to design and construct a physical harness that could allow a person to control robot using muscle signals alone. Also necessary at this time was the design of computer software appropriate for complementing the task. The design and extent of the programming was achieved according to the results achieved earlier in the research.

The fifth stage of the project was to actually hook up a user to the robot and find out the effectiveness of his control over it. Unfortunately, because of time restraints, the only user during the research and testing was the designer himself. Frequent, though often slight, improvements were still readily made, and the entire integrated system is currently ready for further research.

The sixth and final stage of the research was dedicated to trying to harness brain waves as another form of control. This initial idea was an ambitious one, but the potential for success was not realized during the course of this research. For reasons which will be described in detail later, it was effectively impossible to isolate signals characteristic of brain waves. After several tries that came well short of success, the efforts at employing brain waves as control signals were abandoned in favor of the more predictable muscle signals.

Theory of Operation:

The chain of events that allows a robot to be controlled by simply flexing a muscle is outlined in Figure (4). The process which allows any of the project to work at all is the generation of electrical energy in muscles. The strict mechanics of how the signal is processed by the human body from the brain down to each nerve ending is really not important to the workings of the project. It is adequate that this chain of events simply happens. Voltages produced in the body, just like voltages produced by Duracell batteries or available from wall sockets, have no meaning unless they are produced *relative* to a voltage level somewhere else. Surprisingly, finding places on the body to use as signal generators (i.e., muscles) was fairly easy, but finding a place with which to establish a reference (i.e., the "common" or "ground") took a little more effort.

The choices of the eight particular muscle groups that were examined during the course of this research were as much a matter of intuition and logic as anything else. Those muscles which were generally large and/or very close to the skin surface were the most likely candidates. There was also some further thought given to how some muscles might interfere with each other. The biceps and triceps are such a pair of interfering muscle groups. Since the nature of the biceps is to retract the arm, and the nature of the triceps is to extend it, it made little logical sense to test the independence of both groups from each other. A similar problem was eventually discovered between the sensors on the feet and those on the calves. Fortunately, this was a problem that could be overcome with some practice on the part of the user.

Candidates for the common reference voltage were numerous. Areas of the body that tended not to have muscles that could provide interfering signals themselves were the prime candidates. The area of skin just below the knee was the first choice while electrode gel was still being used. Once the necessity for this paste was overcome later in the research, further investigation was conducted to define additional ground sites. The ear lobe, the back of the neck, and the forehead were the best candidates discovered in these attempts until a fortuitous event occurred. While working on establishing yet another choice for a common, the grounding electrode was held in the designer's mouth so that he could use both hands to work. Oddly enough, muscle signal patterns improved dramatically, and noisy interference in the air almost disappeared! As a result, the mouth became the instant winner for the common location for the remainder of the project. There is no doubt that keeping an electrode in the mouth could be an inconvenience. However, no other part of the body was determined capable of providing a common with the noise rejection characteristics of the mouth without some sort of surface preparation (gel, moistening the skin, etc.).

Active Electrodes

Initially, it was thought that the 10 mm surface electrodes would be incapable of picking up signals without the help of electrode gel. It was this gel that increased the conductivity of the skin enough that muscle signals could be picked up on an oscilloscope. Unfortunately, the gel was also rather smelly and very messy, and was not conducive to the completion of a user-friendly robotics project, as this one was intended to be.

Another idea presented by Nishimura, Tomita, and Horiuchi (1992) showed that the conductive gel could be eliminated without compromising a significant amount of the electromyographic signals. They decided to duplicate the high input impedance of human skin by feeding each electrode into an operational amplifier configured as a voltage follower. This configuration has been termed an "active electrode." With its high input impedance, this circuit has reduced the effects of the high surface contact resistance, which required the use of conductive gel in the first place. Additionally, only a negligible amount of signal is lost in the process because of the very low output impedance that is characteristic of most modern op-amps. The op-amps themselves are configured as simple non-inverting voltage followers, the schematics of which are shown in Figure (5). There is one noticeable drawback with this strategy, however. The gel had the quality of isolating the patches of skin on which it was used. The active electrodes have no such quality, and are more subject to interference from other muscle groups. As a result, eliminating the gel also increased the necessity of locating muscles that are independent of other muscle groups.

Amplification

The control signals generated within the body are very small. If they were otherwise, people would have a great deal of trouble getting along without continuously shocking each other. At a muscle source, it is estimated that the voltage potentials created by movement are in the 60-90 millivolt range (Boggs, p. 109). These raw muscle signals, as detected by the active electrodes on the skin surface, have an amplitude range on the order of one millivolt down to only a few microvolts.

Research has shown that the signal magnitudes are more commonly in the microvolt region.

As a result, it was necessary to amplify the muscle signals to a level that the rest of the project could actually work with. This was done in two stages, both of which use analog variable inverting amplifiers such as the one shown in Figure (6). The first stage amplifies the actual raw muscle signal approximately one million times. The second amplification stage, with a gain of about 10, is at the end of the electronics chain. The reason for this second amplification is to provide normalized voltage levels (i.e., between 0.1 V and 10 V), independent of which muscles were being used. These amplifiers each provide a 180 degree phase shift in the signal, as can be seen in the amplifier derivation in Appendix (W). Because the software works primarily with ratios, all of the 180 degree phase changes cancel themselves out.

60- and 180-Hz Filters

The fact that muscle signals are so small is not the only barrier to their being read accurately. If this were the case, then amplification alone would be sufficient. But the air is full of noise that creates interference even greater than that created by conflicting muscle groups. Most of it is at a frequency of 60 Hz (in the United States), and is caused by various nearby electrical equipment. This equipment is normally powered by standard wall plugs, which operate at the 60 Hz frequency and are the actual source of the problem. Unfortunately, this noise completely masks out the signal that this project is trying to harness. One particular example of this problem can be easily seen in Appendix (D). Here, no muscle signal has been sent, and no filtering has occurred. The result is a noisy 60 Hz sine wave with an amplitude of about 0.4 volts peak-to-peak. This makes reading tiny muscle impulses effectively impossible.

Looking at the frequency response of an unamplified, unfiltered muscle signal such as the one in Appendix (E-2), one can easily see that this particular signal stays within the bandwidth defined by about 10 Hz to about 250 Hz. Therefore, the 60 Hz problem cannot be avoided by simply listening to different frequencies, because the noise is embedded well within the frequencies of interest. The solution chosen to combat this problem was an analog band-stop filter, such as one of the ones suggested by Berlin (1978). The design of the particular filter used is shown in Figure (7). This is an adjustable-frequency band-stop filter which carries an attenuation of approximately 30 dB at 60 Hz, a result that can be seen graphically in Appendix (B-1). This design has proven itself to be a workhorse of a filter, and is easily adjustable with only one variable potentiometer. This design does not eliminate 60 Hz noise completely, but it does so enough to make the project functional. An attempt was made to cascade two of these filters in order to eliminate the remaining noise almost completely. There was not enough success to warrant placing two such filters in each of the electronic circuit cards necessary for each electrode.

Fourier analysis of the noise signals in the air would predict that there might be more than simply a 60 Hz component. This prediction turns out to be not only correct, but also influential to the operation of the project. Another look at the frequency response in Appendix (E-2) does, in fact, show spikes at each odd multiple of 60 Hz (i.e., 180 Hz, 300 Hz, and 420 Hz). Fortunately, everything above 200 Hz gets filtered later on in the electronics, but the 180 Hz component is, again, right in the middle of the important frequencies. At first it was thought that the 180 Hz component would not be much of a problem, but this belief turned out to be false. In fact, the 180 Hz signal became the dominant noise factor once the 60 Hz filter was in place. For example, a careful look at the hamstrings noise plot in Appendix (I-1) reveals that the variations occurred at 180 Hz.

Another band-stop filter was needed, but the subordinate nature of the 180 Hz signal did not warrant the complexity of a modified version of the 60 Hz filter in Figure (7). Instead, a simpler version was documented by the Staff of Research and Education Association (1982). This circuit can be found in Figure (8), while its frequency response is in Appendix (B-2). Now this circuit, although simpler, actually proved to be considerably less reliable than the 60 Hz filter, most likely because the circuit does not allow for much variation within the individual parts. It turned out that two separate variable potentiometers were needed to make it work correctly, which gives it a considerable operational disadvantage when compared to the 60 Hz filter. The mathematical proof behind how this circuit works can be found in Appendix (X).

High- and Low-Pass Butterworth Filters

Frequency responses of the eight muscle groups tested, found in Appendix (E), show that the muscle signals remain typically in the low frequencies. This fact made it advantageous to filter out the rest of the frequency spectrum, which contains a substantial amount of noise. There are actually any number of filters that could have been ir .lemented in this project design to accomplish this goal. The Butterworth analog filter series in particular was chosen because of its frequency characteristics. Sedra and Smith (1991) provide a normalized frequency response curve for the different orders of Butterworth filters in Chapter 11 of their textbook. These filters provide an almost horizontal frequency response in the pass band, and a consistent attenuation in the reject band. The third order system was chosen originally because its attenuation approaches a practical 60 dB/decade, which has since proven to be enough for the purposes of this design. Berlin (1978) made it very easy to choose the correct components, as well. The pass and reject band frequencies were chosen experimentally using data very similar to the graphs shown in Appendix (E). Finally, both the high- and low-pass Butterworths are so similar in design (merely trade some resistors and capacitors) that this series was an immediate first choice. This design has since proved to be extremely reliable. The final schematic can be found in Figures (9) and (10), and the mathematical derivation in Appendix (Y). The frequency response of each of the two filters can be found in Appendices (B-3) and (B-4).

AC to DC Converter

All of the muscle signals have, up to this point, retained their naturally sinusoidal nature. But now that the amplification and filtering processes have brought the muscle voltages to reasonable levels of clarity, thought must be given to transferring the data to its ultimate destination.

The software that implements the muscle signals was designed for direct current voltages (i.e. DC, or non-sinusoidal voltages). Therefore, the mostly-sinusoidal inputs (AC) need to be rectified into a signal that is approximately DC. One way to do this would have been to program the 80386 computer to do the job itself. Unfortunately, this process, which has to be done either at or very close to real-time, would slow down all of the other jobs that the computer had to do already. Therefore, another electronic circuit was put in to do that specific task in analog.

The AC/DC converter used in this design is based on one found in Markus' circuit handbook (1974). Its principal operating components are two diodes and a 10 μF capacitor, which effectively (1) rectify the incoming signals and (2) take the time-average of the result. The nature of the averaging process means that there will always be a delay in the response, but that delay is very slight. This delay is on the order 0.25 seconds (see Appendix (C)), which is acceptably small for the needs of this project.

The original AC/DC converter was designed for 100 kHz [sic], weak AC signals that were to be converted to DC, although ripple error at frequencies as low as 20 Hz are reported to be less than 1% (Markus, p. 478). This very small error did not necessitate replacing component parts to lower the design frequency. There were, however, some slight modifications to this design. These changes included the use of TL082 operational amplifiers rather than the specified LM101A, the use of less-precise resistors, and the elimination of a couple of offset capacitors for which the TL082 does not allow. Experimentation showed that this AC/DC converter would malfunction if given too great an input voltage. This malfunction would normally manifest itself with a saturated output at the operational amplifier rail voltage (15 V).

Therefore, additional care had to be taken to insure that the signal going into it was not amplified excessively. A copy of the circuit can be found in Figure (11).

Analog-to-Digital Converter

The signal is now at least mostly DC, but it is still an analog signal. The computer, the robot, and its controllers, however, are designed for digital signals. Therefore, the signal must be translated from analog values to digital ones. Data Translation's 2801A Analog-to-Digital Converter is the means by which the analog electrical signals from the body reach the 80386 computer. Programs written in QuickBASIC drive the board and read the data from it (see Appendix (AB) for specifics on how this is done). A DT707A Break-out board is simply a screw-terminal board which connects directly to the output of the amplifier that follows the AC/DC converter. The DT707A board did, however, have to be modified slightly with the addition of some resistors. The reason for this modification is that the project design calls for independent grounding attachments rather than the single common lead designed into the board.

IBM 80386

The computer used in this project was a Unisys IBM-compatible 80386 (25 MHz) machine. Almost any similar machine would be adequate, however, given that it can handle the same programs and communications software. The Unisys had to be configured for the project as follows:

General Purpose Interface Board (GPIB):

This board, installed in one of the expansion slots, allowed the computer to talk to the HP54501A Oscilloscope and the HP3562A Signal Analyzer. Only in this way only could generated data be transferred from these devices to the computer (see Appendices (Z) and (AA) for details) and later plotted. Creating muscle data plots by hand would have been impossible given the amount of data generated.

RS-232 Communications:

The computer had to be capable of handling RS-232 communications. The Unisys did not have to be specially modified to meet this requirement, as the circuitry is built-in. The communications criteria were necessary to control the SCORBOT.

DT2801A Analog-to-Digital Converter:

The computer had the Data Translations DT2801A A/D board connected to another of the expansion slots. This board was necessary to translate the analog signals from the circuitry to digital numbers that the computer could digest.

Software:

Only two commercial programs were really necessary for completion of this project. The first was Microsoft's QuickBASIC compiler. This program was necessary because its extensive library software could communicate well with both the GPIB and DT2801A. It could also talk to the robot and read the A/D converter in the same program. Also necessary was some sort of spreadsheet program. Quattro-Pro for Windows was used because it could import ASCII data from the QuickBASIC data-generation routines. These were actually configured for use with MATLAB 4.0, but were additionally compatible with Quattro. The compatibility was especially important because Quattro-Pro was the program that plotted and printed results from the oscilloscope and signal analyzer.

Digital Analysis (A Side Note):

One of the major early decisions in this project was whether to use analog or digital techniques to measure and filter the muscle signals. Analog techniques were chosen principally because of the designer's familiarity with the field. But hindsight shows that with a moderately fast computer (i.e., capable of sampling at 2 kHz or better) these signals could have been filtered and amplified with greater accuracy using digital methods than the analog equivalent. This is another appropriate topic for future research in this field; digital methods would certainly alleviate a number of the problems encountered with the great deal of analog circuitry and component parts that were required. The analog circuitry would still be necessary to a degree to prevent aliasing, but very likely the digital filtering would be preferable.

Robot Controller Program:

As mentioned above, Microsoft's QuickBASIC was the language of choice for the main controller program. This controller program, listed in Appendix (AB), has 4 different functions: (1) Home the Robot, (2) Reset Muscle Threshold Values, (3) Activate the System (Robot), and (4) Activate the System (Graphics). Choice (1) simply returns the SCORBOT to its home position, a function normally chosen at the beginning and/or end of a robotics "session." Choice (2) provides two functions. The first is to set the noise levels for the system. This subroutine first allows adjustment of all the amplifiers and filters to provide an optimum signal. Second, it sets the threshold level for muscle activity. This function operates by recording the level at which the user wants the program to activate the robot. Choice (3) is the operational part of the program in terms of physical robot control. This subroutine shows the user what percentage of his threshold value he is currently sending to the robot. It also activates the robot accordingly if muscle activity exceeds the threshold setting. Choice (4) was designed in case the robot was either inoperative or unavailable. It simply controls a small graphics cursor using the same control inputs that the robot would use if it were, in fact, connected. The graphics cursor has functions such as move right, move left, change size, change color, and plot. This part of the program also shows a variation on how muscle input can be used.

Implementation:

Considerable effort was put into deciding how to integrate the user into the system. Because there were eight sensors, each with its own four-foot wire that had to be connected before each use, Velcro straps were cut to size and fitted with one sensor each via electrical tape. Doing this made strapping on the electrodes fairly easy and adjustable. Unfortunately, the Velcro stuck to everything, which meant that care had to be taken when storing or removing the straps. Given that there were now eight sizeable Velcro straps, it seemed reasonable to integrate all of them into a suit of some sort. A standard Navy flight suit was adequate for the purpose, although it failed to reduce the problems inherent with the Velcro. A view of the legs of this instrumented suit is shown in Figure (12).

Because each sensor required its own circuit board, one of which is shown in Figure (13), a small box was designed to house them. Furthermore, a 6-pin socket was modified to be a plug for six of the sensors (the other two electrodes and the ground were already conveniently connected using sockets from an old EMG meter). The box was also fitted with holes through which the wires for the power supply, the A/D converter, and the nine sensor wires could be threaded.

Interference:

The human muscular system is incredibly complex. "Simple" actions such as walking need hundreds of muscles acting in concert to result in a smooth action. As a result, consciously activating one particular muscle may also unconsciously activate one or more other muscle groups. Activation of a muscle group may also send interfering signals to active electrodes elsewhere on the body. For this reason, only four of the original eight sensors could be effectively incorporated into the flight suit design created for this project.

A large percentage of the graphs included in the appendices are dedicated to the effects of muscle interference. Each investigation into the interference characteristics of a particular muscle required three sets of data: (1) a "noise" level (i.e., not activated) for reference, (2) a conscious activation of one without the conscious activation of the other, and (3) vice versa. These results were often quite interesting, and are summarized on the next page.

Interference Characteristics of Sampled Muscle Groups

("On" means "activated," "Off" means "not activated")
("High" means "high interference")

"On" →→ ↓ "Off" ↓	Biceps	Forearm	Hamstring	Quadriceps	Ankle	Shin	Foot	Calf
Biceps	N/A	Low	Low	Low	Low	Low	Low	Low
Forearm	High	N/A	Low	Low	Low	Low	Low	Low
Hamstring	Low	Low	N/A	High	Low	Low	Low	Low
Quadriceps	Low	Low	High	N/A	Low	Low	Low	Low
Ankle	Low	Low	High	Medium	N/A	Low	Low	High
Shin	Low	Low	High	Medium	Low	N/A	Low	Medium
Foot	Low	Low	Medium	Medium	Low	High	N/A	Medium
Calf	Low	Low	High	Medium	Low	Low	Low	N/A

One surprising result from this investigation showed that the muscles commonly known as the hamstrings tended to interfere with just about every other muscle in the leg. Another interesting result was the indication that simply raising the foot off the ground (as if to "tap") actually involves the entire lower leg.

Not all of the muscle groups implemented in the research design were completely independent of one another. For example, the foot and the calf signals interfered with each other. The quadriceps area was capable of affecting the entire leg. This was where training significantly affected the results. Knowing that some interference existed, different muscles could be moved in distinct ways as not to interfere with the other groups. Although occasional problems still occurred, this solution was adequate.

These specific data which supports the table on the previous page can be found in Appendices (G) through (V).

The Problem with Neural Signals:

An advertisement in *The Computer Applications Journal* suggested that appropriate circuitry dedicated for the measurement of brain waves could be ordered from Circuit Cellar, Ink. This device was known as the Hemispheric Activation Level Detector, or simply HAL-4. Unfortunately, the HAL-4 apparently failed to be able to pick out anything that would suggest that brain waves could be used as control signals. The results, in fact, appeared to be more noise than anything else. The equipment was tried under several different circumstances without any improvements in the results; some of these circumstances were daydreaming, concentrating on different emotions, different placement of the electrodes, and rapid blinking. None of these circumstances could be differentiated from the others based on the results. As the completion date for this research drew near, emphasis was placed on finalizing a working design based on the predictable muscle groups. As a result, the electroencephalogram research had to be postponed indefinitely.

Observations:

The results of this research showed that it is certainly possible to use electrical signals produced naturally by the body as control signals. These results are evidenced by the fact that each of the axes of the SCORBOT were predictably controllable over short periods of time using only muscle electrical signals as inputs. In this respect, the objectives of this project were met. In the efforts to produce a working design project, though, certain aspects of research had to be foregone because of time considerations. As a result, many of the results are much more qualitative than quantitative.

It should be noted that some of the data indicated by the graphs in the appendices of this document do not accurately reflect what the muscle signals were actually doing over a given length of time. For example, the HP3562A provided a frequency analysis of the muscle signals that was about 5 seconds delayed. But monitoring the results for quite some time showed no particular pattern in any of the eight muscle signals sampled. The only conclusion that could be drawn from the frequency analysis appeared to be that the muscle signals limit themselves to about 10 Hz on the low end and 300 Hz on the high end of the frequency spectrum. On occasion, it would appear that one particular muscle response, such as that of the forearm in Appendix (E-6), was out of the ordinary (the forearm response has a wider range of frequencies than most of the other muscle groups). Monitoring that and other signals over periods of time showed that other muscles sometimes varied in their frequency responses to a similar degree.

Time responses provided little more information. With the frequencies being more-or-less random, only amplitude and phase seemed worth noting for the purpose of mathematical analysis. Since amplitude was adjustable for each sensor, it really meant little more than mere existence of a signal. If each muscle could be monitored using the same sensor under the same conditions, then more could be done with the results. Phase appeared to matter little, as well; the interference graphs (Appendices (G) - (V)) are slightly misleading in the sense that the two data plots on each were taken sequentially rather than simultaneously. Again, monitoring the oscilloscope provided little else that seemed significant.

This is not to say that a rigid mathematical analysis of the muscle signals is not in order. In fact, there are many additional ways that these signals ought to be examined. Perhaps a single band pass filter at a frequency of 110 Hz cascaded with an amplifier would provide better information. Another idea would be to isolate all signals below 60 Hz in the hopes of eliminating that problem altogether. In any case, different techniques can be applied to achieve similar results as those found in this research. Perhaps better ways to harness the muscle signals can be found without too much difficulty.

What should be noted in addition to the above are conditions in which qualitative measurements could be made. Muscle signals were much more difficult to find when it was cold outside. Conversely, they were much easier to recognize when a thin layer of sweat existed on the skin. Prepping the skin with rubbing alcohol appeared to have little if any affect on the results. Finally, on some days there was apparently more noise in the air than on others (as indicated by the frequency analyzer). This, of course, affected the signal accordingly, although there was no apparent reason for the change in the noise levels.

Given that it is possible to use muscle signals as control signals, further research in this field is certainly appropriate. The feasibility of harnessing these control signals is limited at this point, and will be so until a more regimented analysis of the signals can be undertaken. With any luck, a successful quantitative analysis of the seemingly random signals is actually possible. If it is, then the possibility of exploring new technology for the human body will take another leap.

-------- ILLUSTRATIONS --------

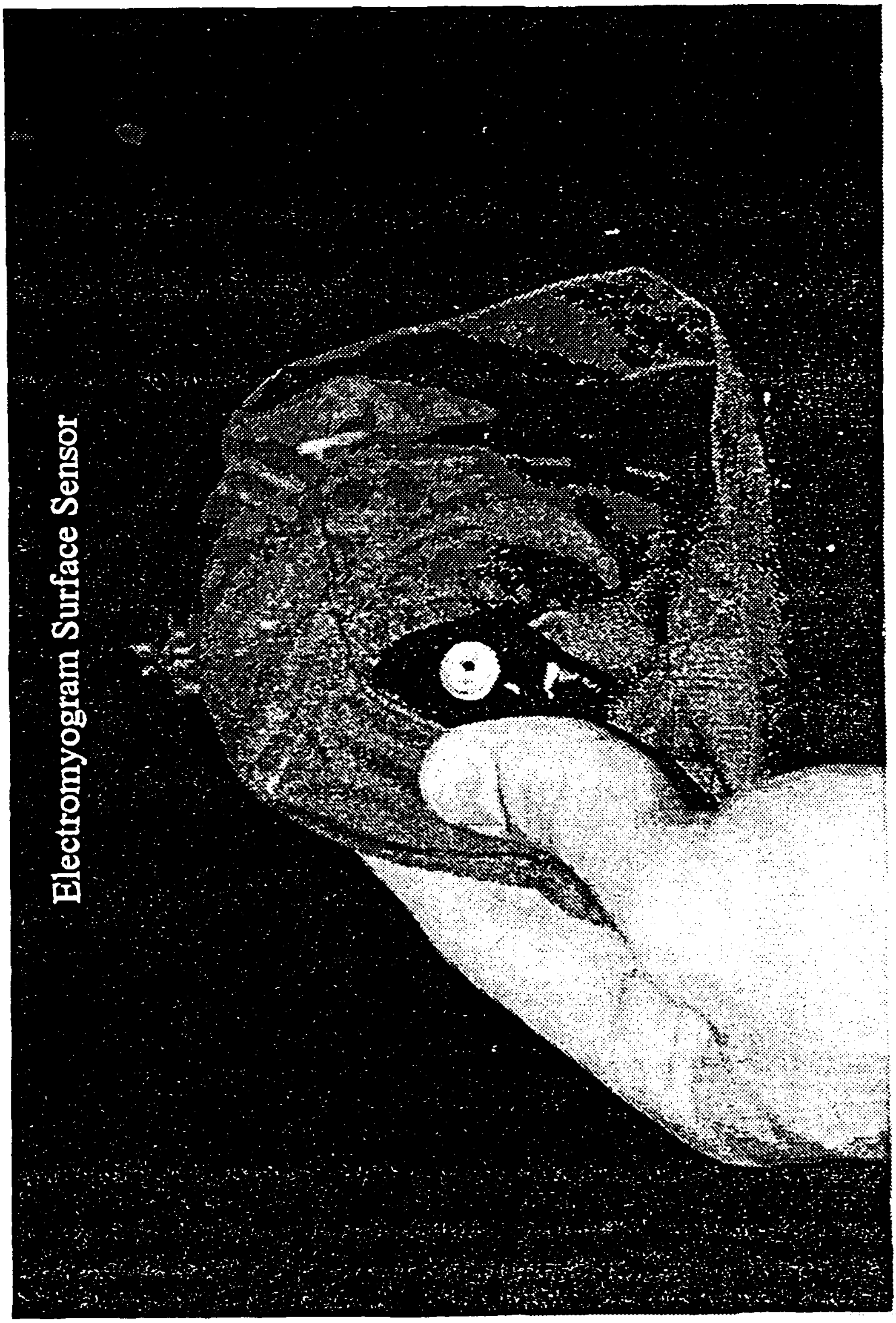


Figure (1)

Sensor Locations

Tried and implemented

Tried but not used

"Biceps" = Biceps

"Forearm" = Brachioradialis

"Quadriceps" = Vastus Medialis

"Hamstring" = Semitendinosus

"Shin" = Tibialis Anterior

"Calf" = Gastrocnemius

"Ankle" = Peroneus Longus

"Foot" = Soleus

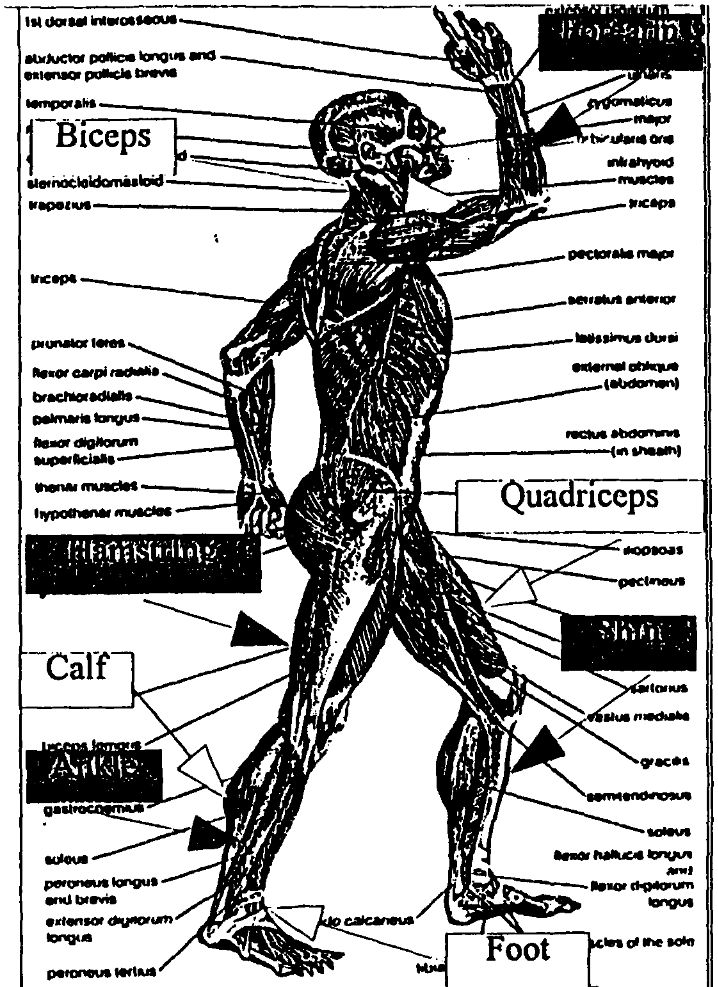


Figure 13: Lateral aspect of the human muscular system.

Reprinted by permission of Faber and Faber Ltd from Anatomy of the Human Body by R D Lockhart, G.F. Hamilton, and F W. Fyfe (second edition) 1959

Figure (2)

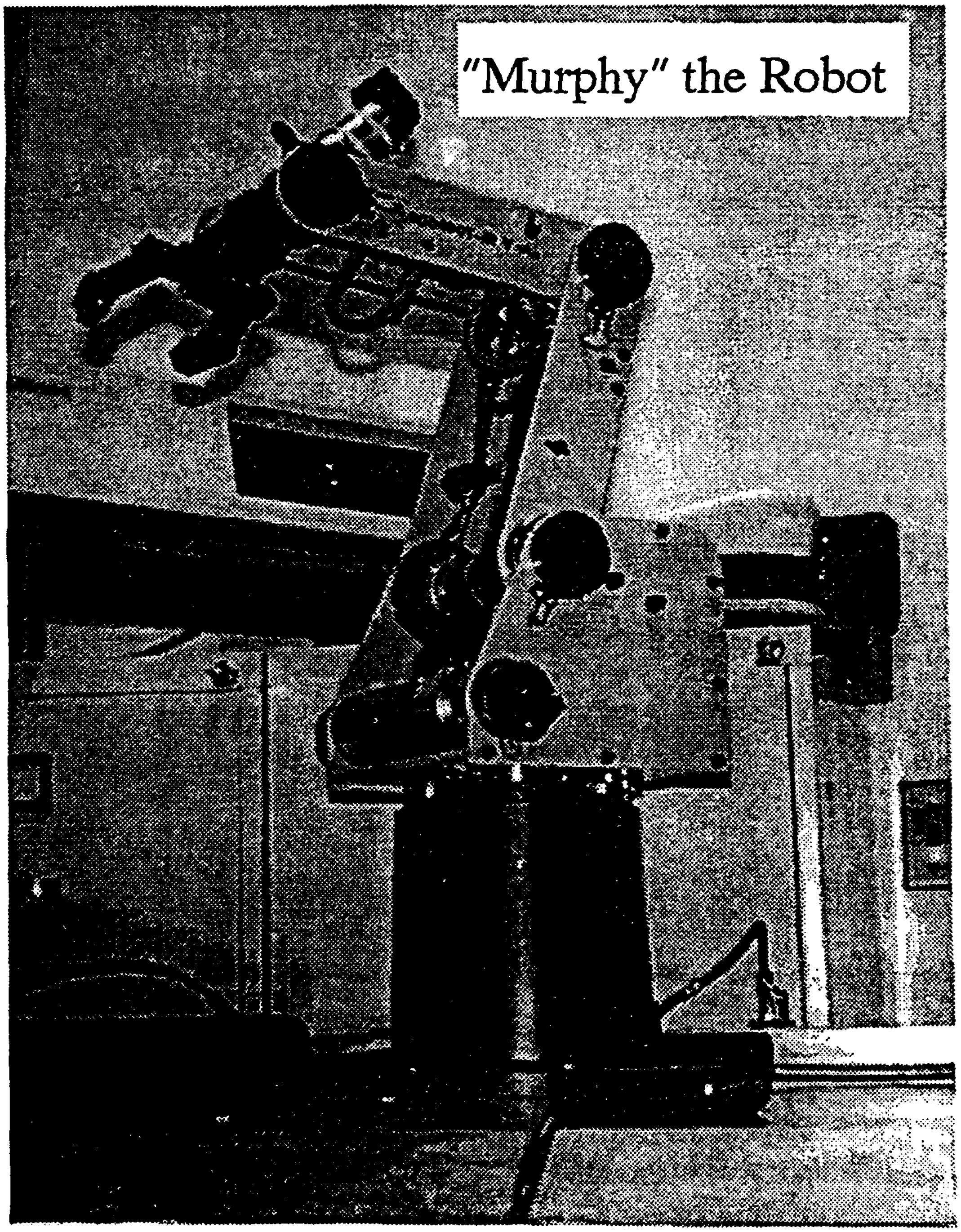


Figure (3)

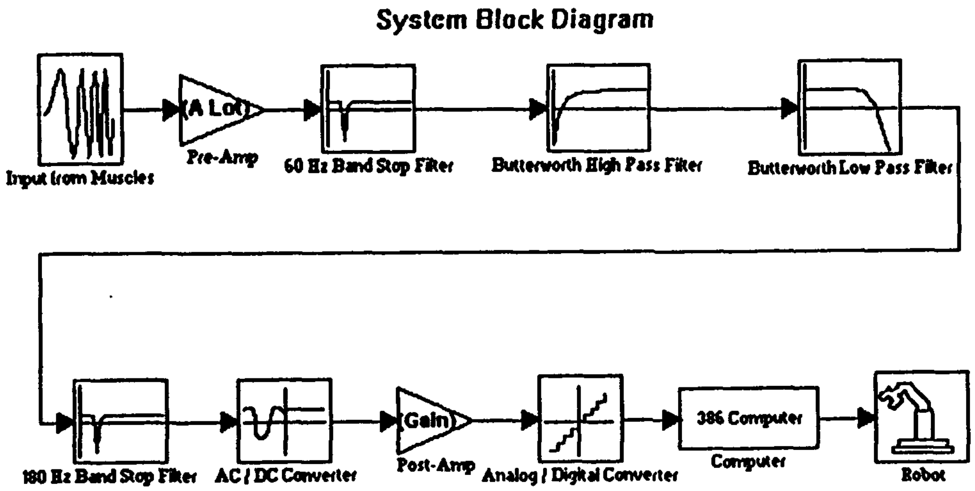
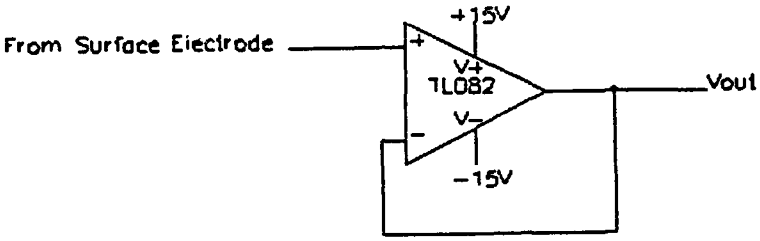

System Block Diagram
(A Lot)
Input from Muscles
Pre-Amp
60 Hz Band Stop Filter
Butterworth High Pass Filter
Butterworth Low Pass Filter
(Gain)
386 Computer
180 Hz Band Stop Filter
AC / DC Converter
Post-Amp
Analog / Digital Converter
Computer
Robot

Figure (4)



Active Electrode

Figure (5)

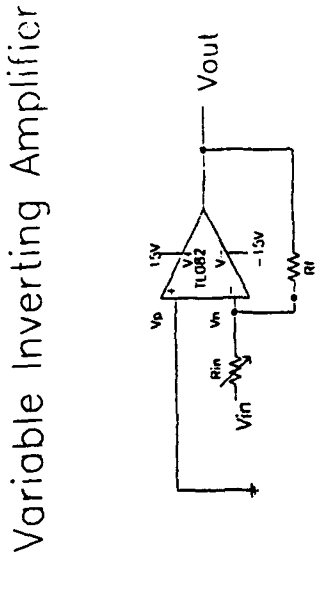

Variable Inverting Amplifier
Vout
15V
V+
TL082
V-
-15V
Rf
Vp
Vn
+
-
Rin
Vin

Figure (6)

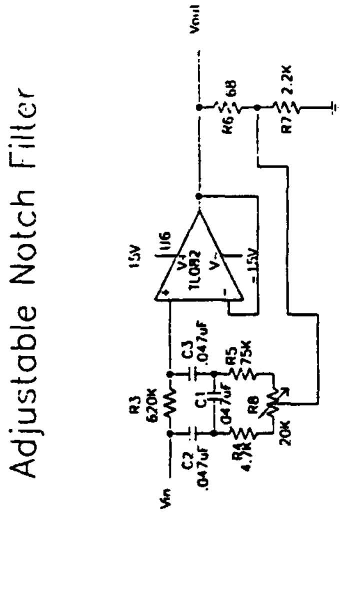

Adjustable Notch Filter
Vin
R3
620K
C2
.047uF
C1
.047uF
C3
.047uF
R4
4.7K
R5
75K
R8
20K
15V
U6
V+
TL082
V-
-15V
+
-
Vout
R6
68
R7
2.2K

Figure (7)

180 Hz Notch Filter

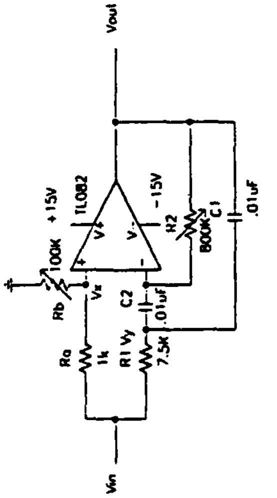


Figure (8)

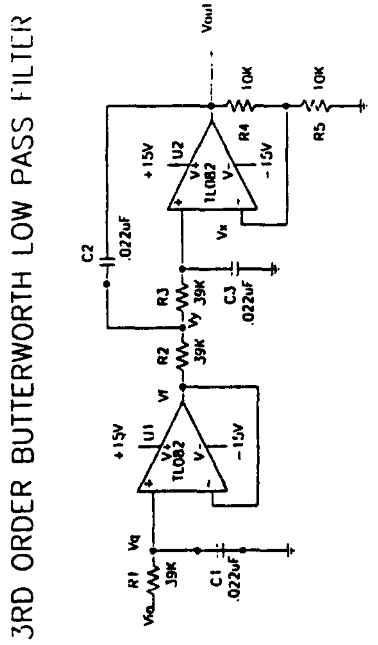

3RD ORDER BUTTERWORTH LOW PASS FILTER
Via
R1
39K
Vq
C1
.022uF
+15V
U1
V+
TL082
V-
-15V
Vf
R2
39K
Vy
R3
39K
C3
.022uF
C2
.022uF
Vx
+15V
U2
V+
TL082
V-
-15V
R4
10K
R5
10K
Vout

Figure (9)

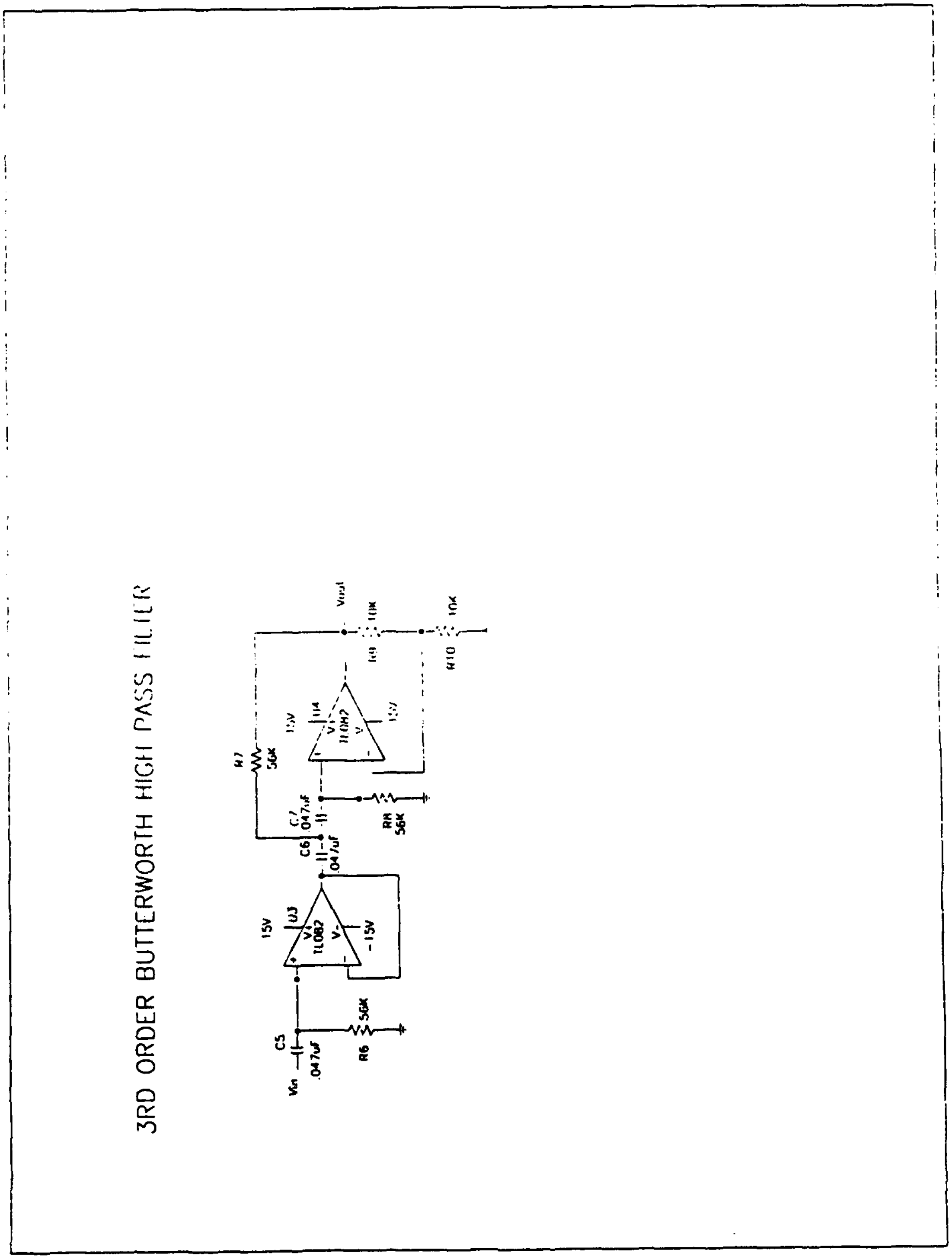

3RD ORDER BUTTERWORTH HIGH PASS FILTER
Vin
C5
.047uF
R6
56K
15V
U3
V+
TL082
V-
-15V
C6
.047uF
C7
.047uF
R7
56K
R8
56K
15V
U4
V+
TL082
V-
15V
Vout
R9
10K
R10
10K

Figure (10)

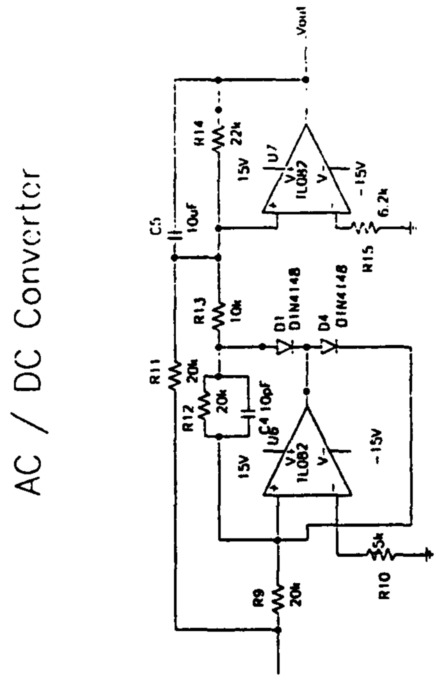

AC / DC Converter
Vin
R9
20k
R11
20k
R12
20k
C4
10pF
U6
V+
V-
1L082
15V
-15V
R10
5k
R13
10k
D1
D1N4148
D4
D1N4148
C5
10uF
R14
22k
U7
V+
V-
1L082
15V
-15V
R15
6.2k
Vout

Figure (11)

Prototype Sensor Configuration (Legs only)

Figure (12)

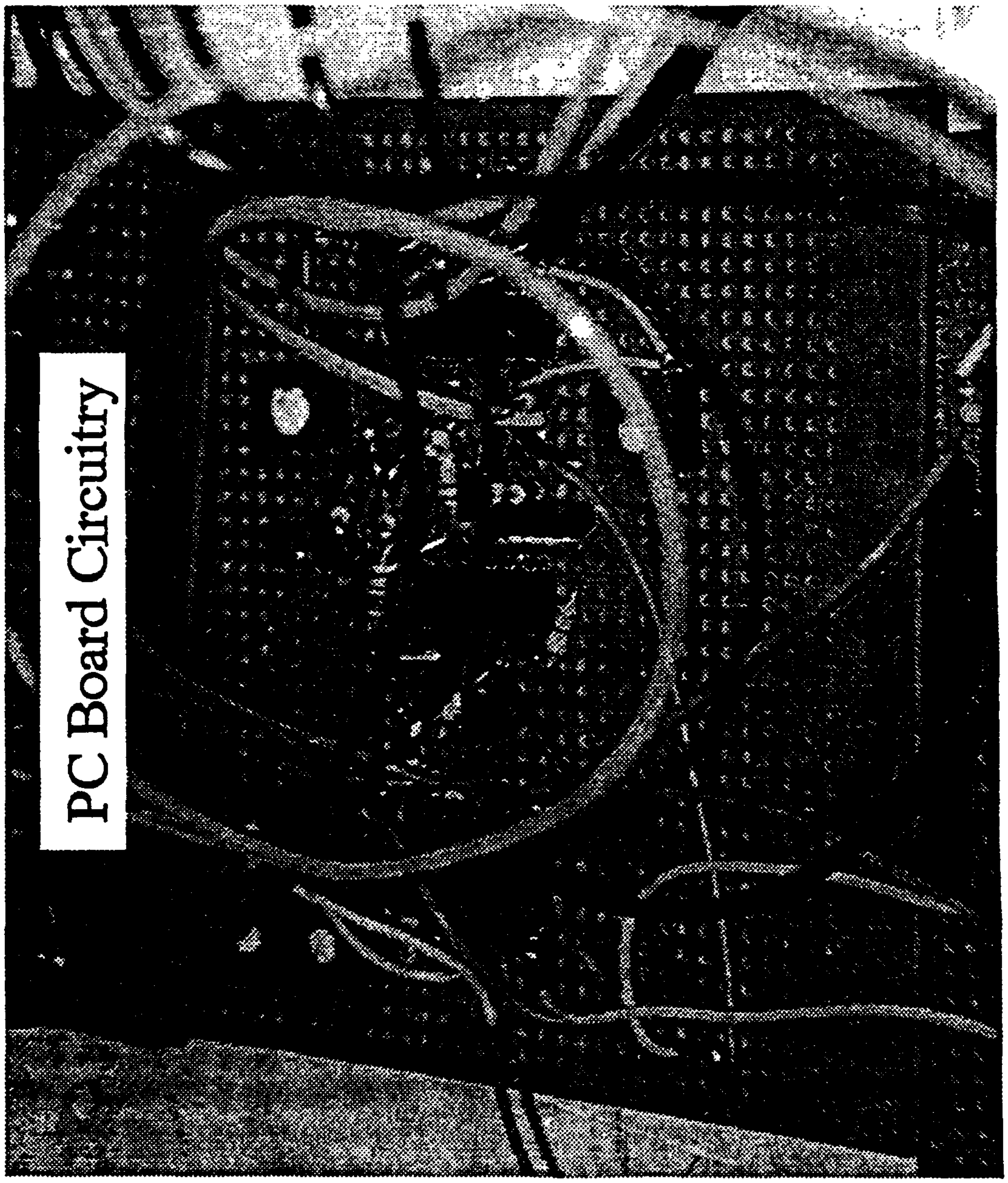


Figure (13)

References:

Berlin, Howard M. *Design of Active Filters, with Experiments.* Indianapolis: H. W. Sams, 1978.

Bennett, Professor William B. United States Naval Academy, Annapolis, MD.

Boggs, Robert N. "Tiny Motor Powers Child's Prosthesis." *Design News* 04 May 1992: 109-110.

Callaway, Enoch, M.D. *Brain Electrical Potentials and Individual Psychological Differences.* New York: Grune & Stratton, 1975.

Circuit Cellar Ink. *The Computer Applications Journal.* 38 (1993): 69.

Clement, Professor William I. United States Naval Academy, Annapolis, MD.

Cunningham, Dr. Mark, M.D., Baptist Memorial Hospital, Memphis, TN. Personal Interview. 28 December 1992.

Data Translation, Inc. *User Manual for DT2801 Series Single Board Analog and Digital I/O Systems for the IBM Personal Computer.* 1987.

Demoyer, Professor Robert A. United States Naval Academy, Annapolis, MD.

Dobkin, R. C. "Precision AC / DC Converters." *Guidebook of Electronic Circuits.* Eds. John Markus, Tyler G. Hicks, and Patricia A. Allen. New York: McGraw-Hill, Inc., 1974: 478.

Grass Instrument Company Recording Accessories and Spare Parts for EEG and Polygraphs Catalog. January, 1993.

Grieve, D.W., et al. *Techniques for the Analysis of Human Movement.* Princeton: Princeton Book Company, 1976.

Hewlett-Packard Company. *HP3562A Dynamic Signal Analyzer Operating Manual.*

Hewlett-Packard Company. *HP3562A Dynamic Signal Analyzer Programming Manual.*

Hewlett-Packard Company. *HP54501A Digitizing Oscilloscope Programming Reference.* 1988.

Knowles, Professor Kenneth A. United States Naval Academy, Annapolis, MD.

Lam, Harry Y-F. *Analog and Digital Filters: Design and Realization.* Englewood Cliffs, N.J.: Prentice-Hall, 1979.

The MathWorks, Inc. *The Student Edition of MATLAB for MS-DOS Personal Computers.* Englewood Cliffs, N.J.: Prentice-Hall, 1992.

Microsoft Corporation. *Microsoft QuickBASIC: Programming in BASIC.* 1988.

Murray, Charles J. "EMG Signal Retrieval Controls Muscle Stimulation." *Design News* 21 December 1987. pp. 52-53.

National Instruments Corporation. *NI-488.2 Software Reference Manual for MS-DOS.* 1992.

The New Encyclopedia Britannica. "Muscles." 1993 ed.

Nishimura, S., Y. Tomita, and T. Horiuchi. "Clinical Application of an Active Electrode Using an Op-Amp." *IEEE Transactions on Biomedical Engineering.* 39 (1992): 1096-1099.

Nunez, Paul L. *Electric Fields of the Brain: The Neurophysics of EEG.* New York: Oxford University Press, 1981.

Sedra, Adel S., and Kenneth C. Smith. *Microelectronic Circuits.* Philadelphia: Saunders College Publishing, 1991.

Staff of Research and Education Association. *The Electronics Problem Solver.* New York: Research and Education Association. 1982.

Texas Instruments. *Linear Circuits Data Book: Op Amps, Comparators, Timers, Regulators, A/D Peripherals.* 1984.

Wick, Professor Carl E. United States Naval Academy, Annapolis, MD.

Ziemke, LT, USN, M.D. Bancroft Hall Medical Clinic, United States Naval Academy, Annapolis, MD.

-------- APPENDICES --------

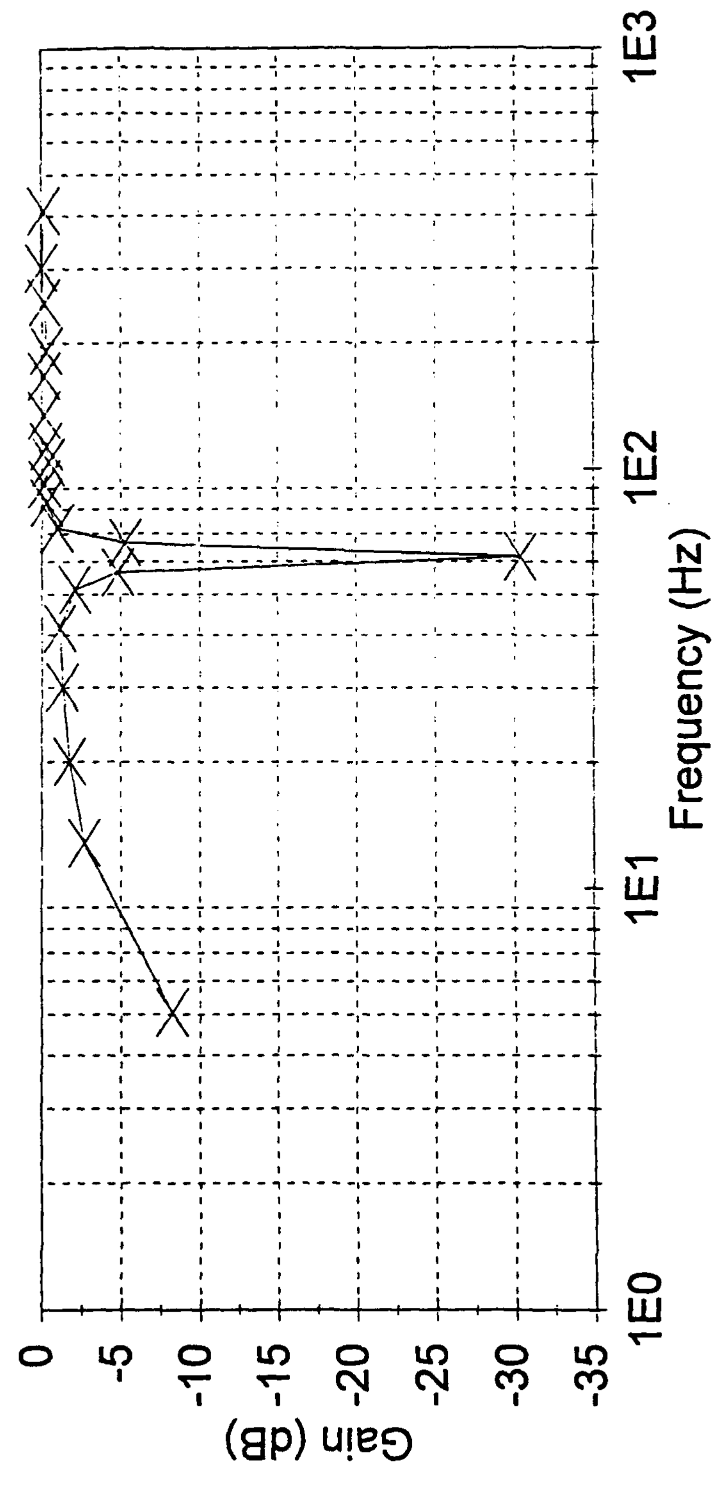
Equipment List:

The following is the complete equipment list necessary for carrying out the research for this project.

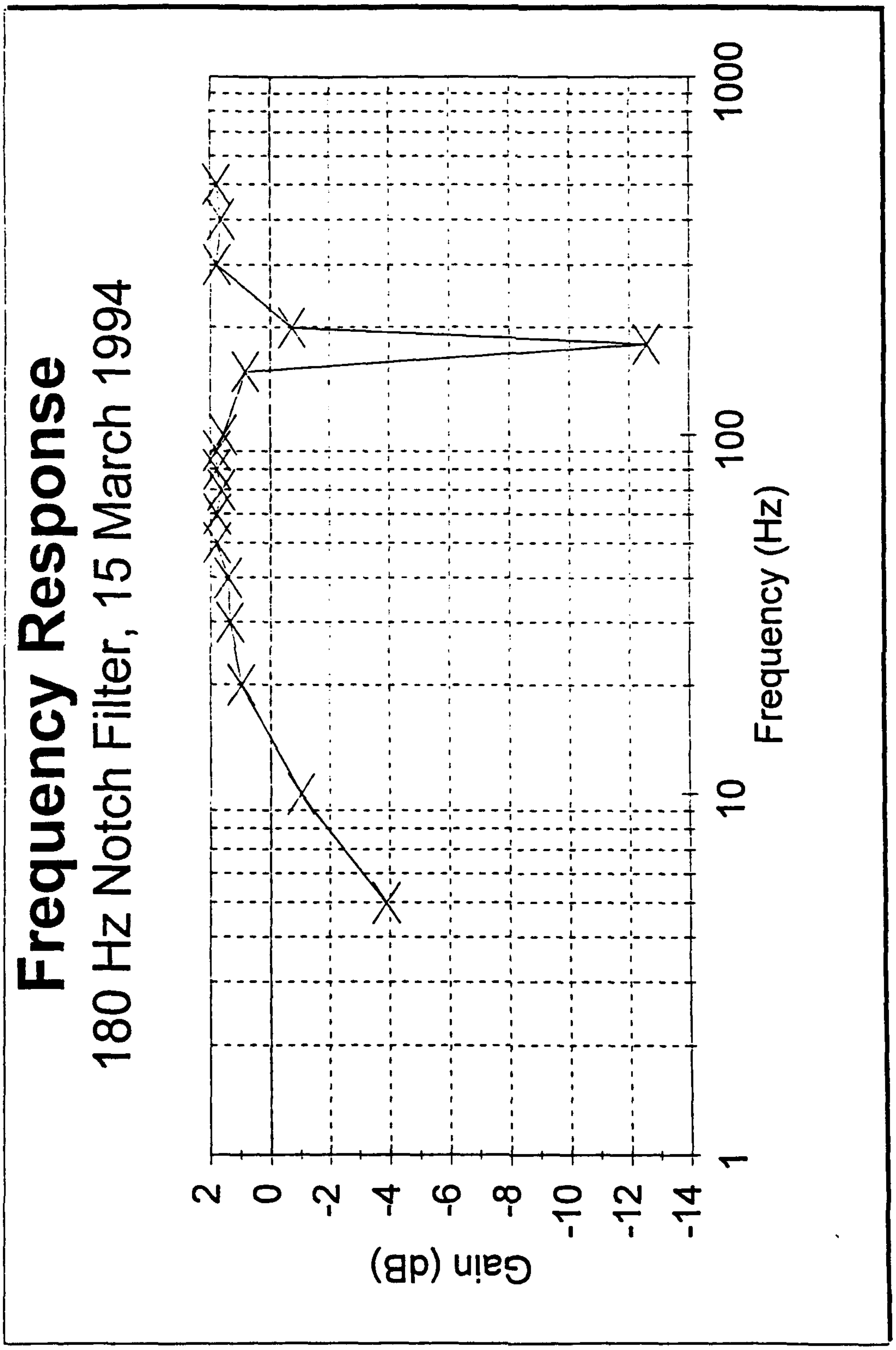
1) Hewlett-Packard 54501A Digitizing Oscilloscope, manuals, and interface cables
2) Hewlett-Packard 3562A Frequency Analyzer, manuals, and interface cables
3) National Instruments NI-488.2 General Purpose Interface Bus, software, and interface cables
4) Gould 1602 Oscilloscope
5) Frequency Generator
6) Data Translations DT2801 Analog to Digital Converter, software, and manuals
7) Data Translations DT707A Differential Input Breakout Board
8) Unisys IBM-compatible computer (386, 25 MHz) with Microsoft Windows v3.1
9) Microsoft QuickBASIC 4.0 and manuals
10) MATLAB 4.0 and manuals
11) PSpice circuit design program
12) WordPerfect 5.2 for Microsoft Windows v3.1
13) Quattro-Pro for Microsoft Windows v3.1
14) Breadboard with DC voltage sources
15) Global Specialties 1302 Power Supply
16) Gold-plated surface electrodes (10 mm and 5 mm, each commercially available for about $6.00)
17) Electrode Conducting Gel
18) Robot arm (e.g. SCORBOT ER V Plus), manuals, cables, controller, and accessories
19) Standard electrical components (resistors, capacitors, solder, wire, etc.)
20) Circuit Cellar Hemispheric Activation Level Detector (HAL-4) and manual
21) Hewlett-Packard Laser-Jet IV Printer

Frequency Response

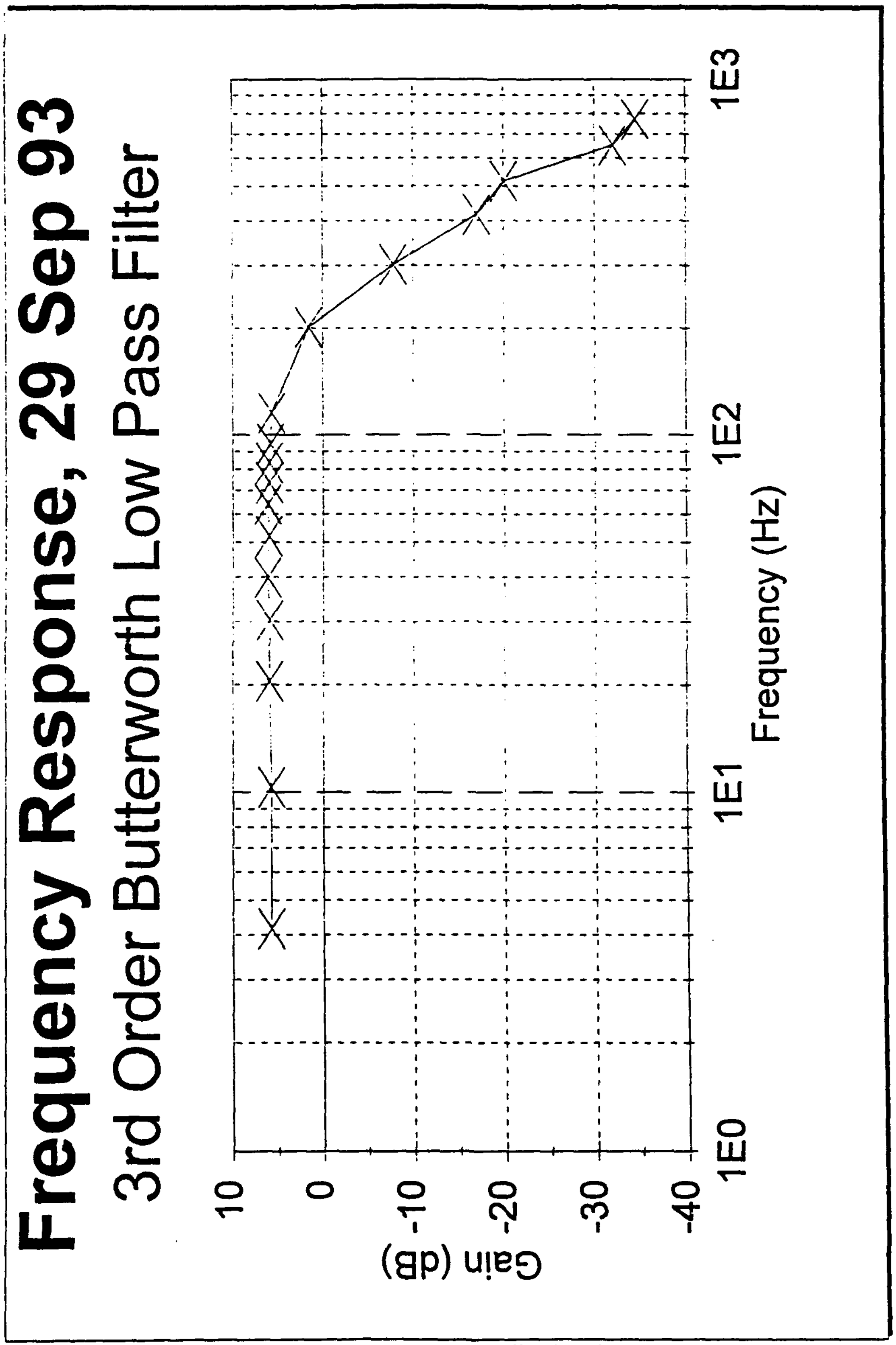
60 Hz Notch Filter, 29 September 1993



Appendix (B-I)


Frequency Response
180 Hz Notch Filter, 15 March 1994
Gain (dB)
2
0
-2
-4
-6
-8
-10
-12
-14
Frequency (Hz)
1
10
100
1000

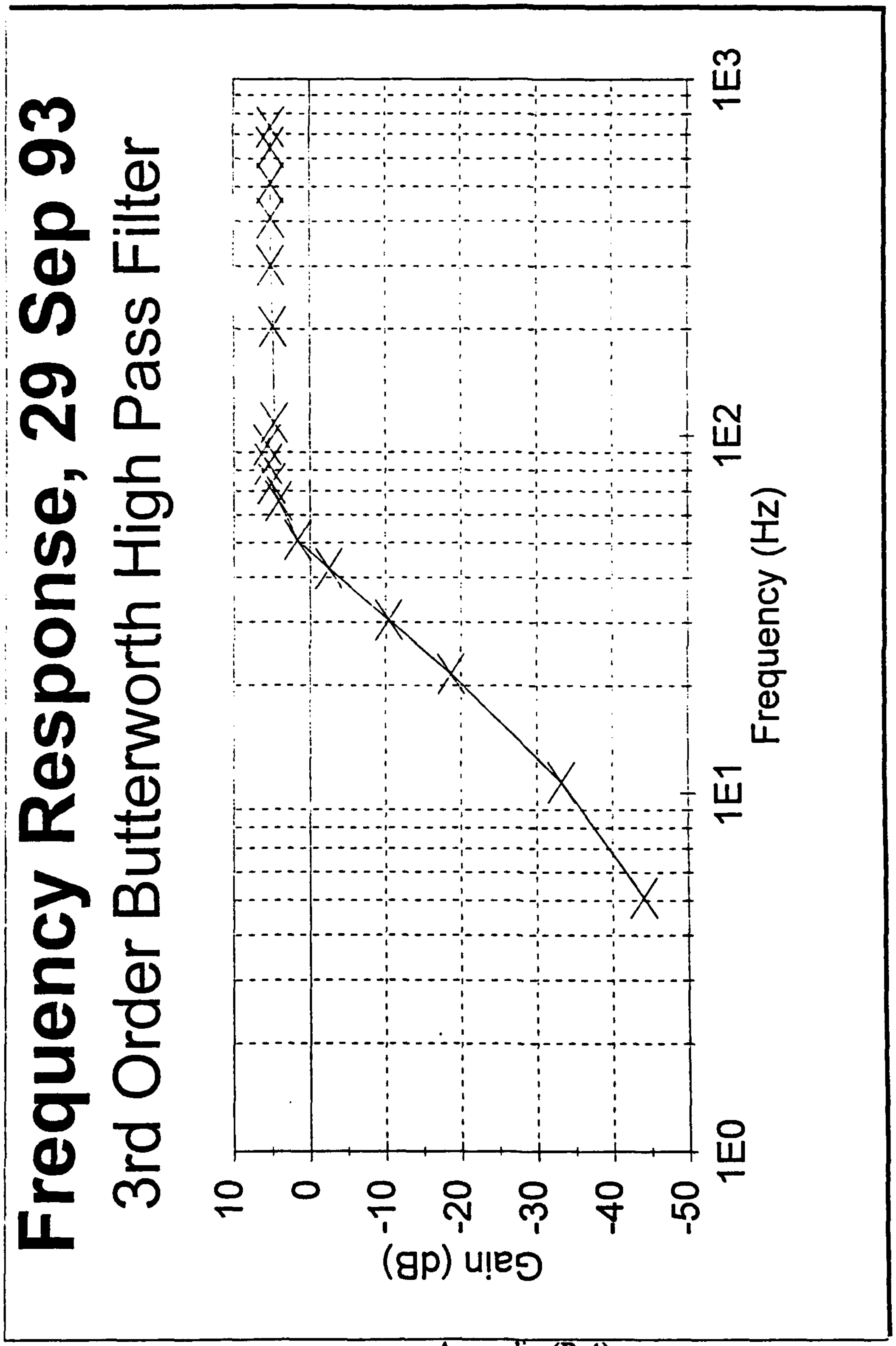
Appendix (B-2)



Frequency Response, 29 Sep 93

3rd Order Butterworth Low Pass Filter

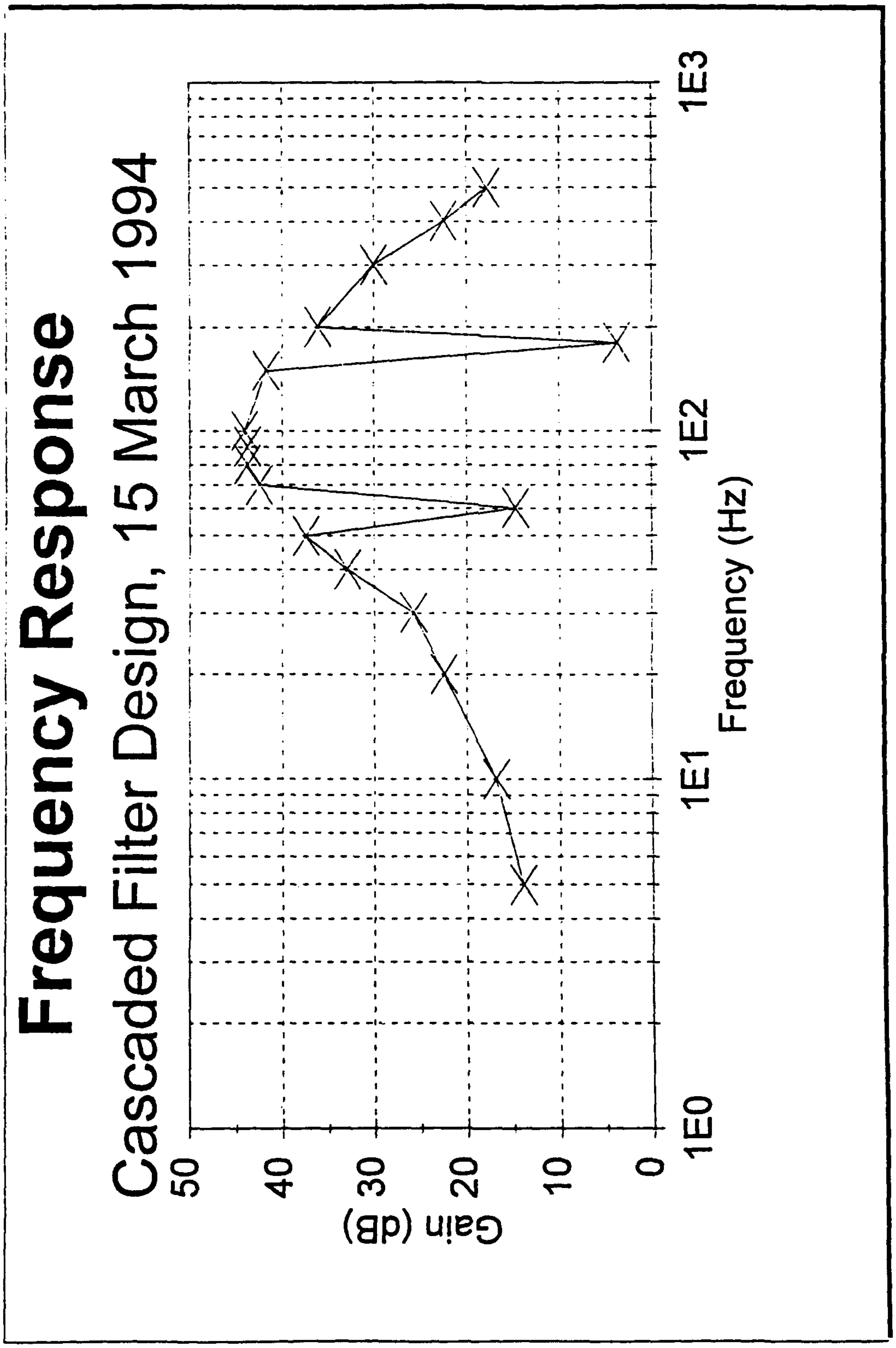
Appendix (B-3)



Frequency Response, 29 Sep 93

3rd Order Butterworth High Pass Filter

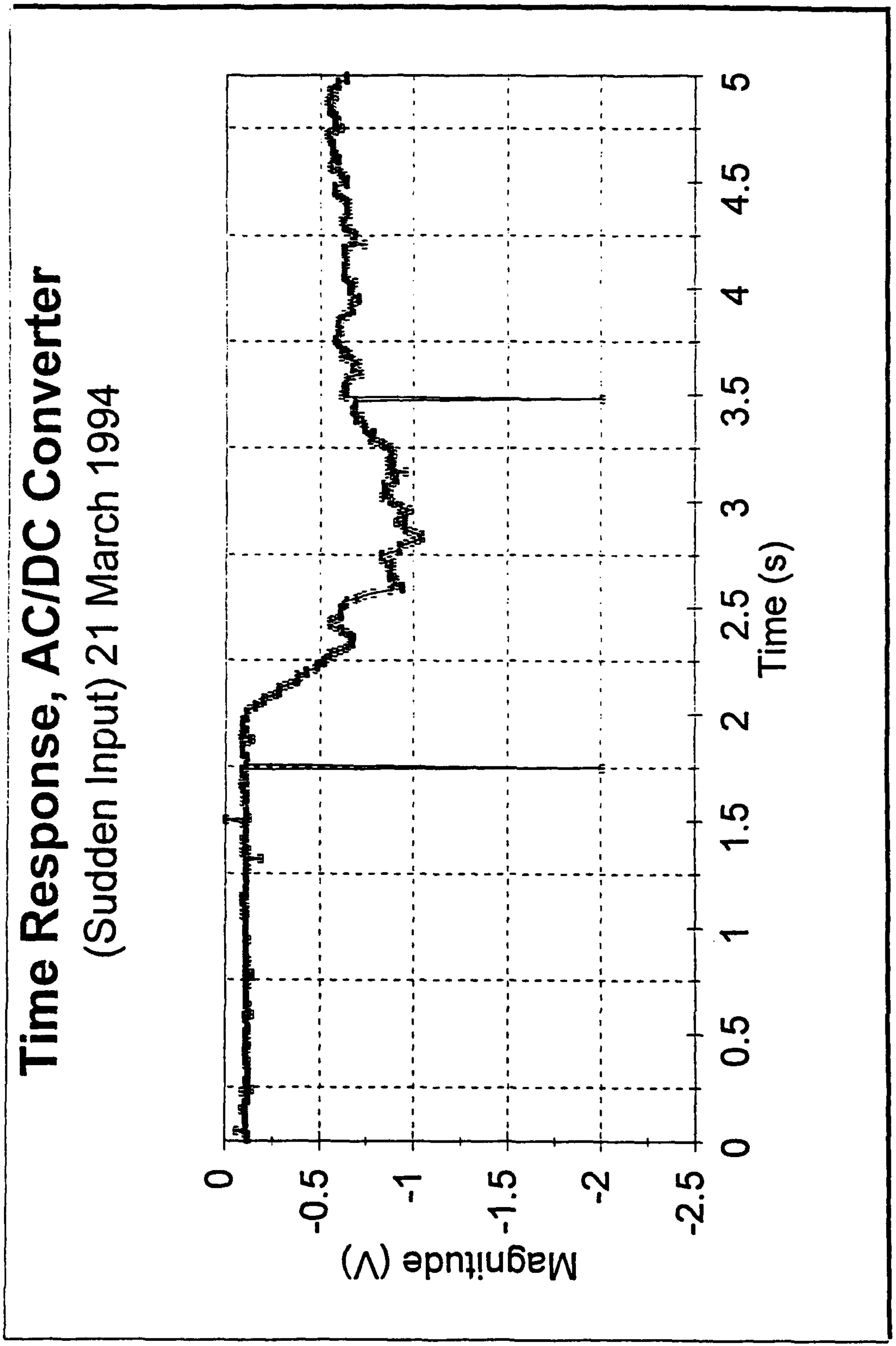
Appendix (B-4)



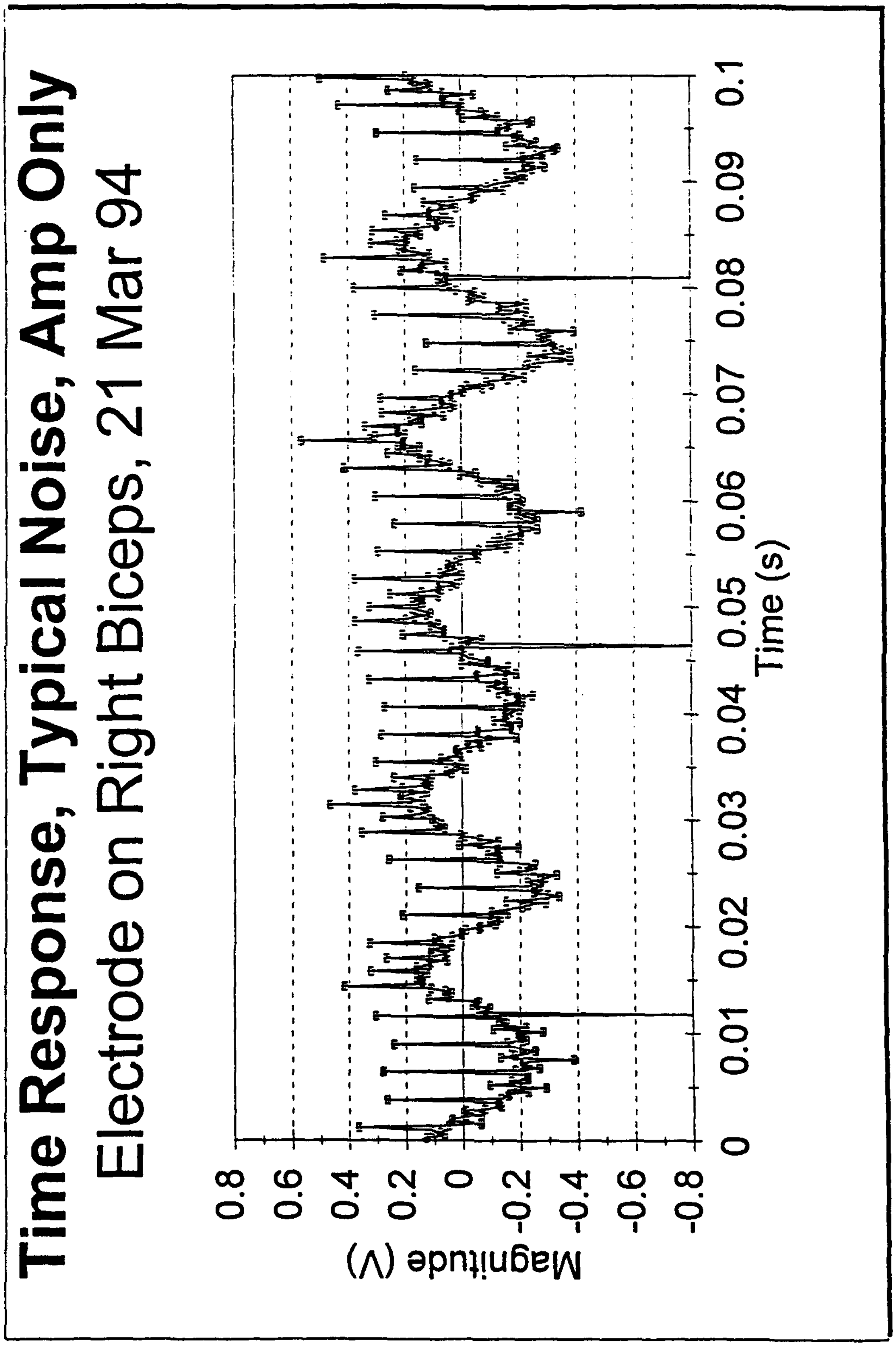
Frequency Response

Cascaded Filter Design, 15 March 1994

Appendix (B-5)


Time Response, AC/DC Converter
(Sudden Input) 21 March 1994
Magnitude (V)
0
-0.5
-1
-1.5
-2
-2.5
Time (s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

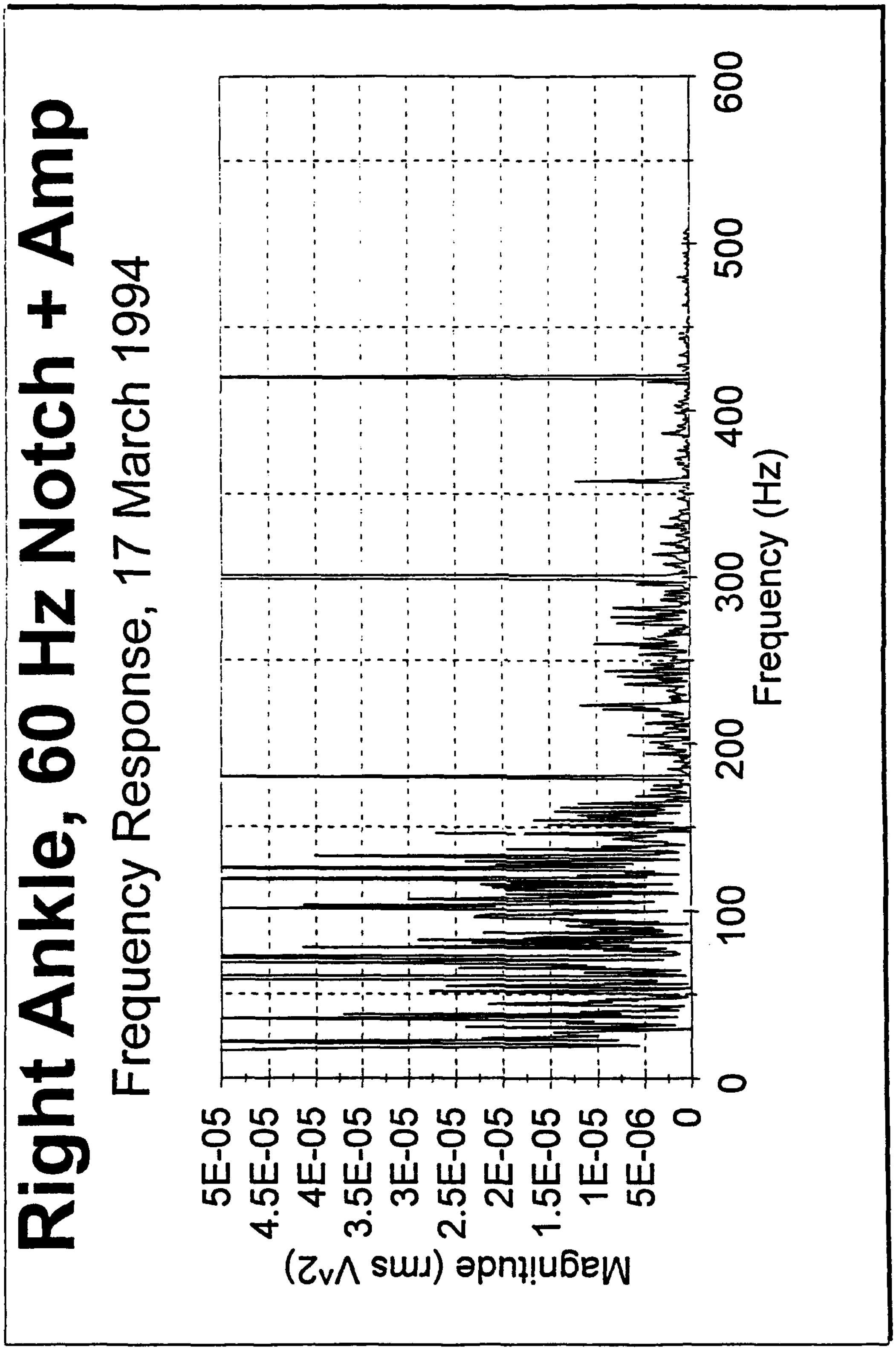
Appendix (C)



Time Response, Typical Noise, Amp Only

Electrode on Right Biceps, 21 Mar 94

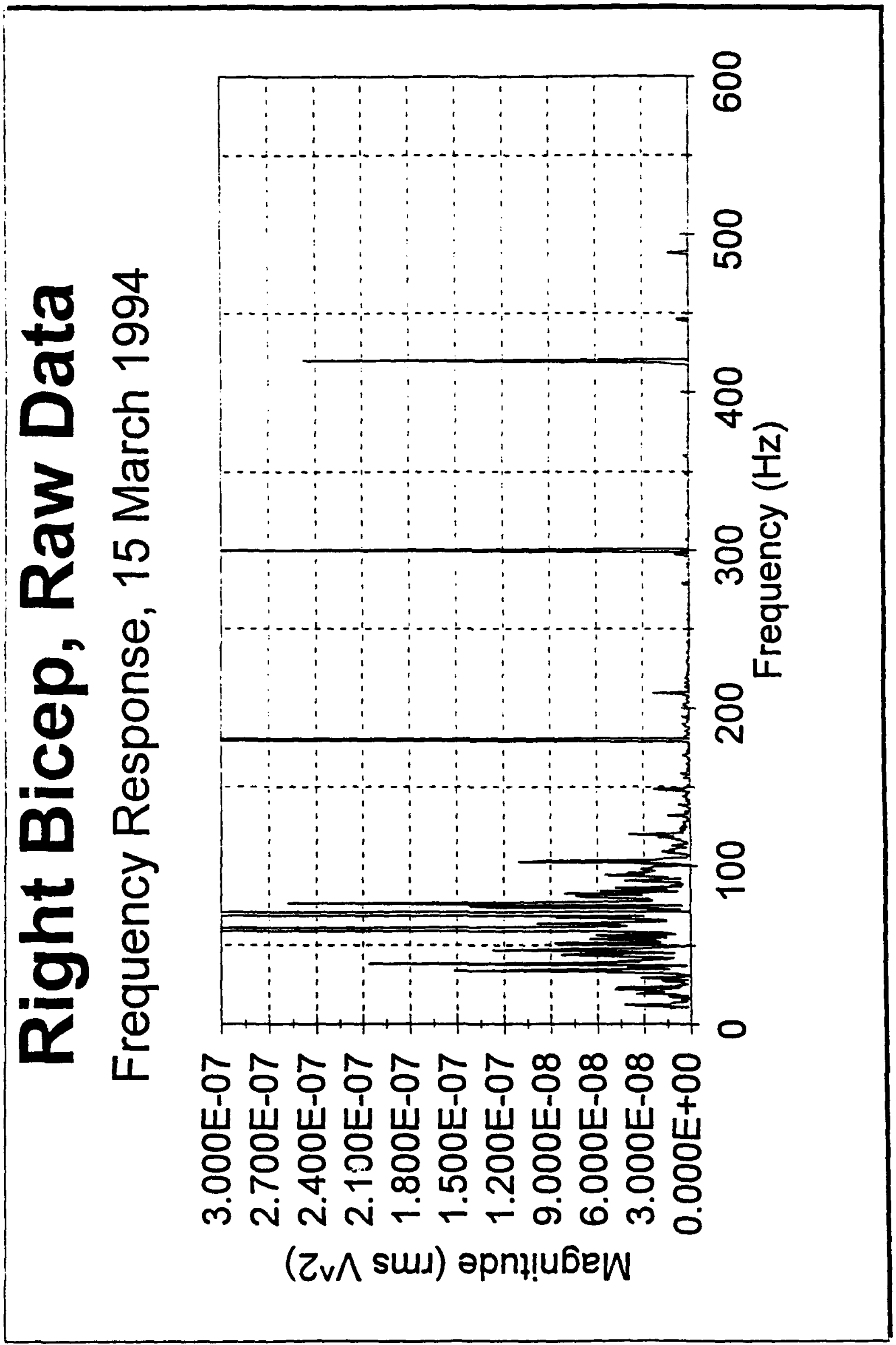
Appendix (D)



Right Ankle, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

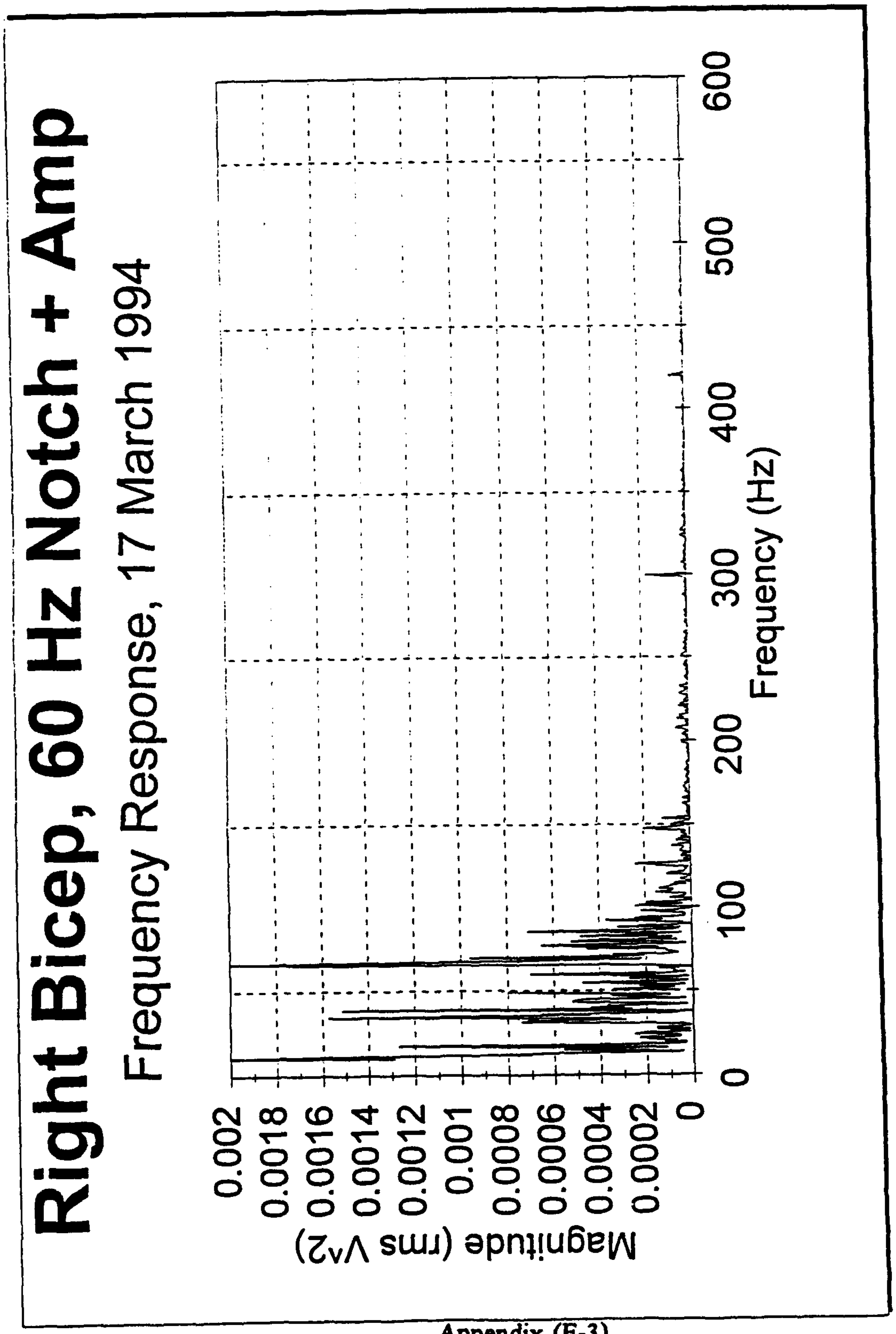
Appendix (E-1)



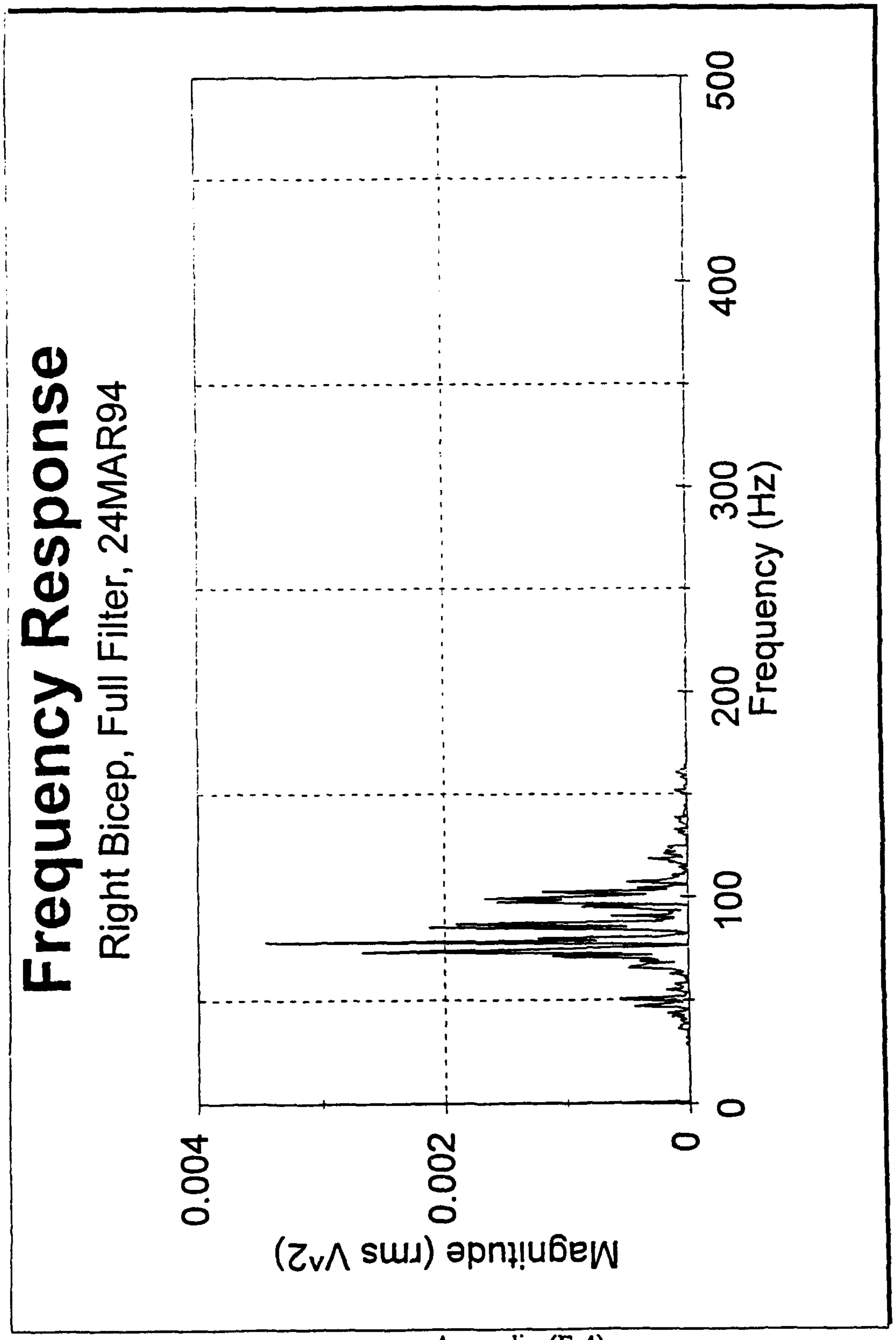
Right Bicep, Raw Data

Frequency Response, 15 March 1994

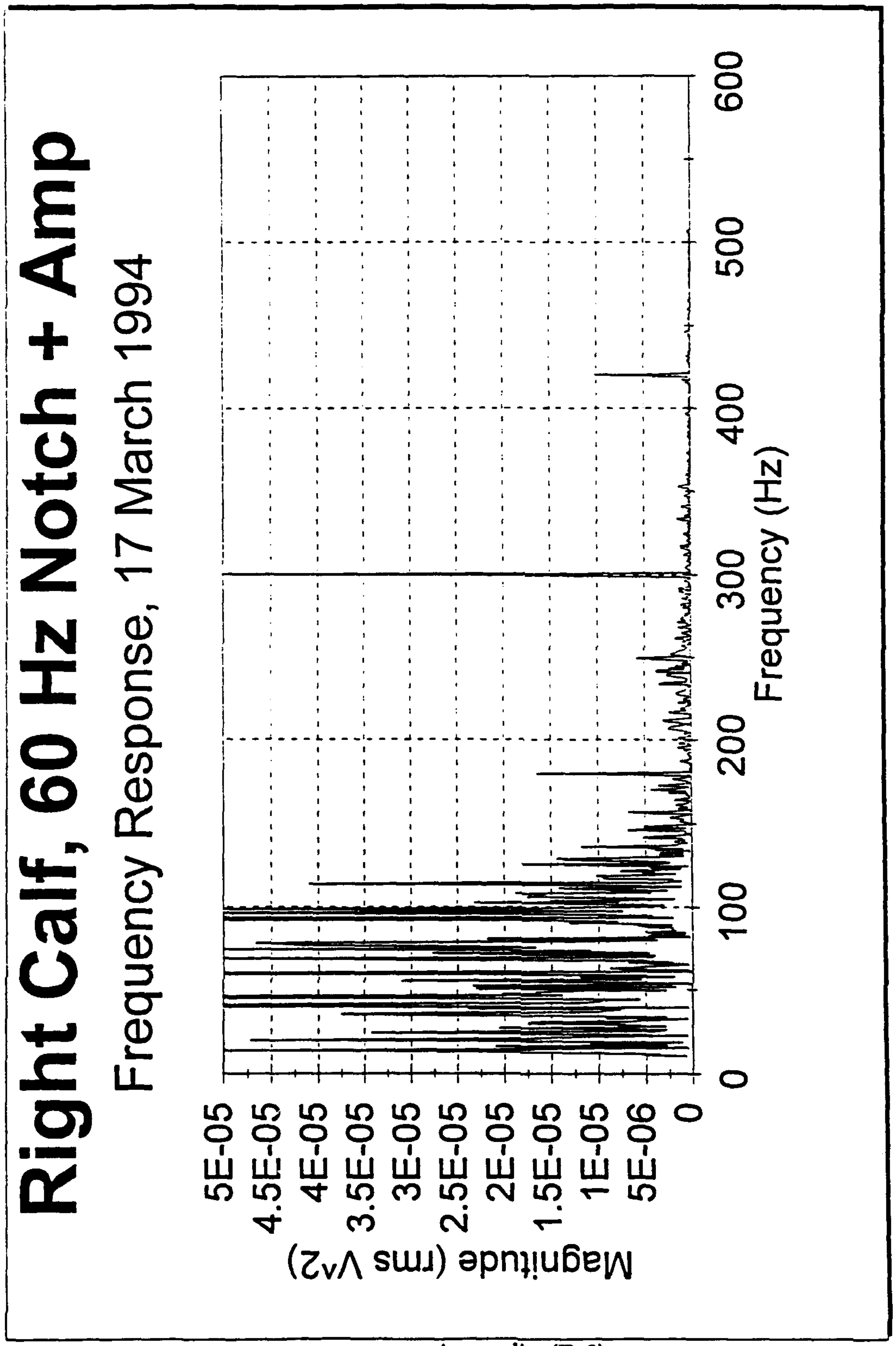
Appendix (E-2)


Right Bicep, 60 Hz Notch + Amp
Frequency Response, 17 March 1994
Magnitude (rms V^2)
0.002
0.0018
0.0016
0.0014
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
0
100
200
300
400
500
600
Frequency (Hz)

Appendix (E-3)


Frequency Response
Right Bicep, Full Filter, 24MAR94
Magnitude (rms V^2)
0.004
0.002
0
Frequency (Hz)
0
100
200
300
400
500

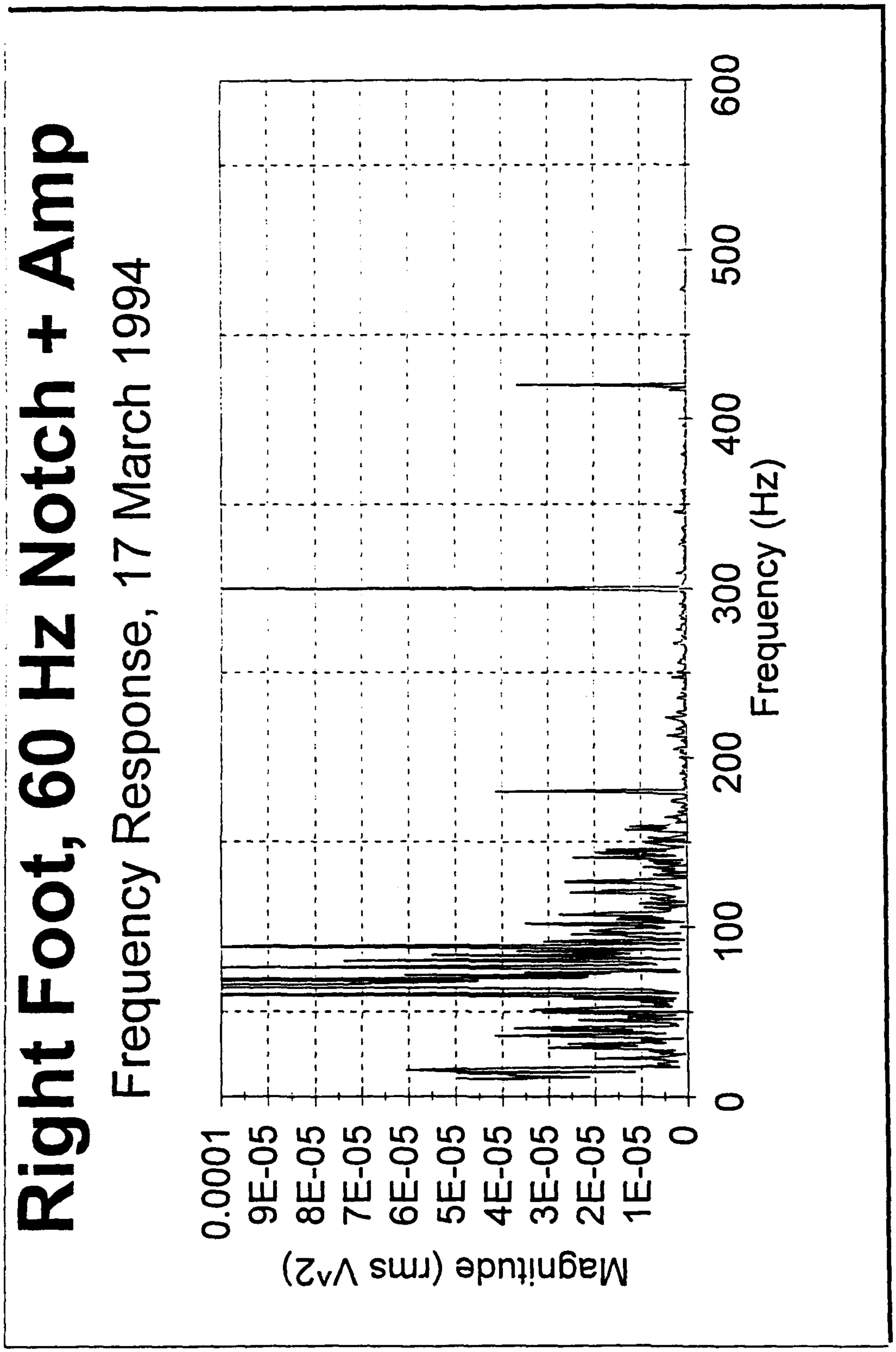
Appendix (E-4)



Right Calf, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

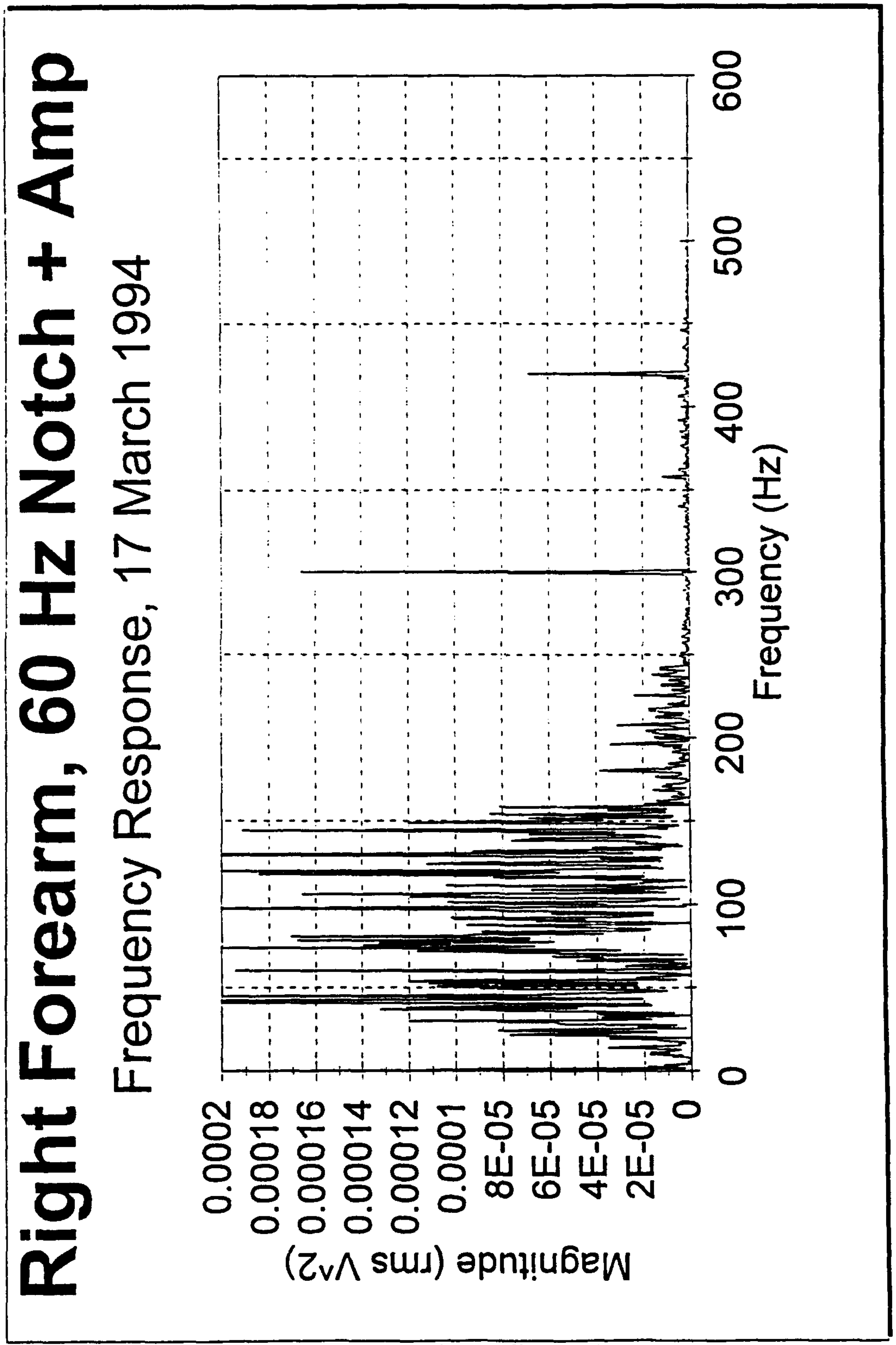
Appendix (E-5)



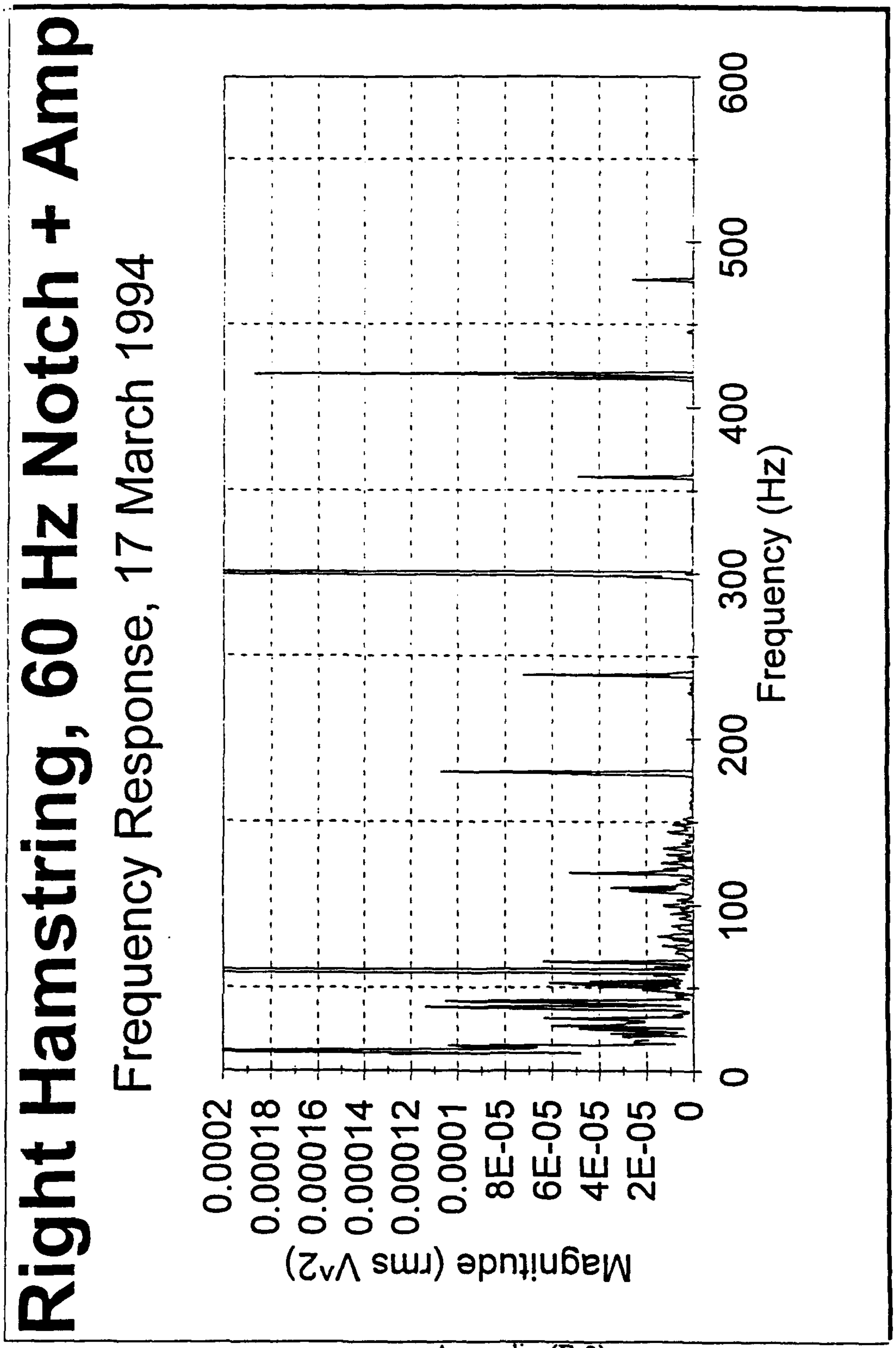
Right Foot, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

Appendix (E-6)


Right Forearm, 60 Hz Notch + Amp
Frequency Response, 17 March 1994
Magnitude (rms V^2)
0.0002
0.00018
0.00016
0.00014
0.00012
0.0001
8E-05
6E-05
4E-05
2E-05
0
Frequency (Hz)
0
100
200
300
400
500
600

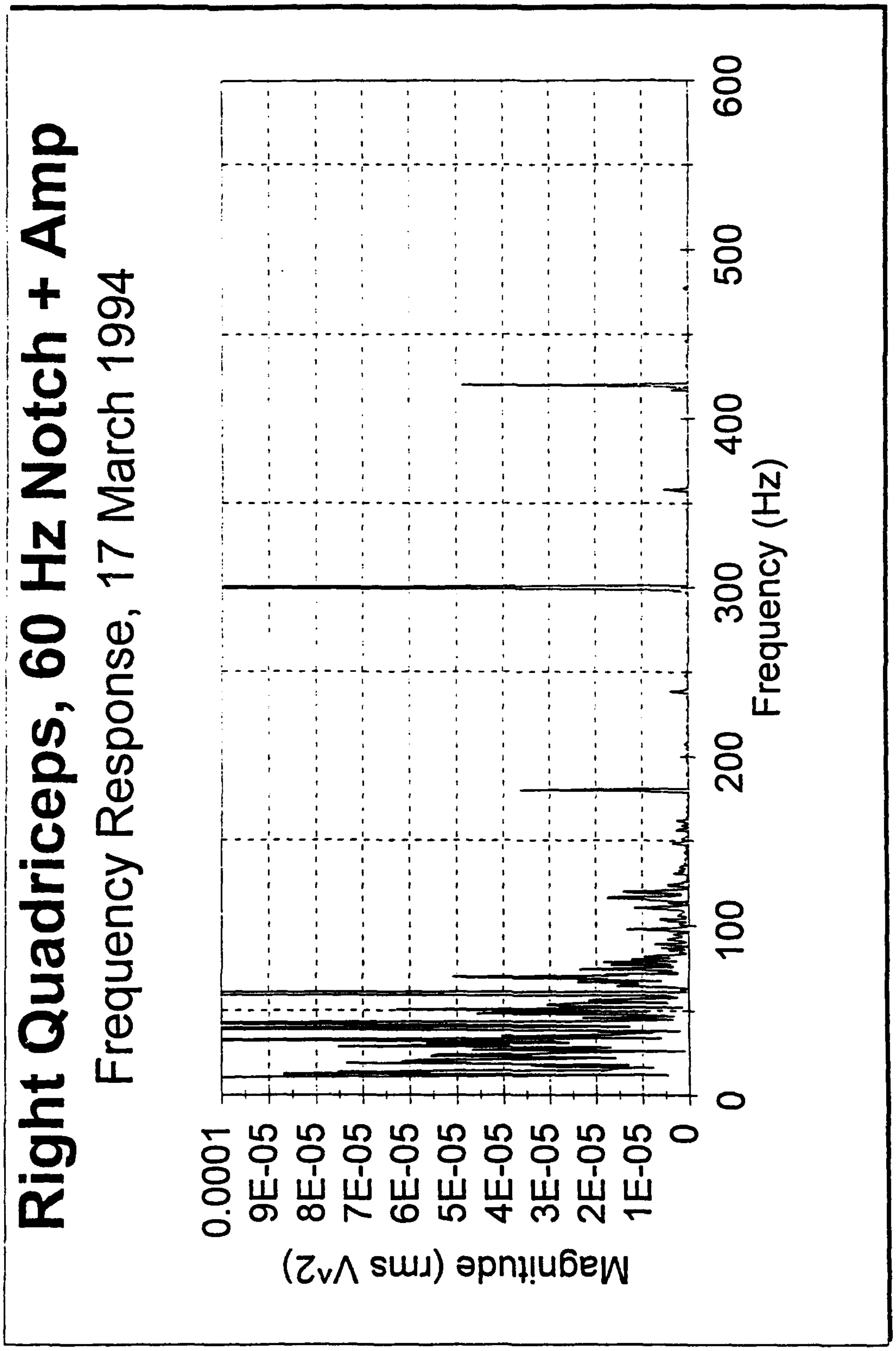
Appendix (E-7)



Right Hamstring, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

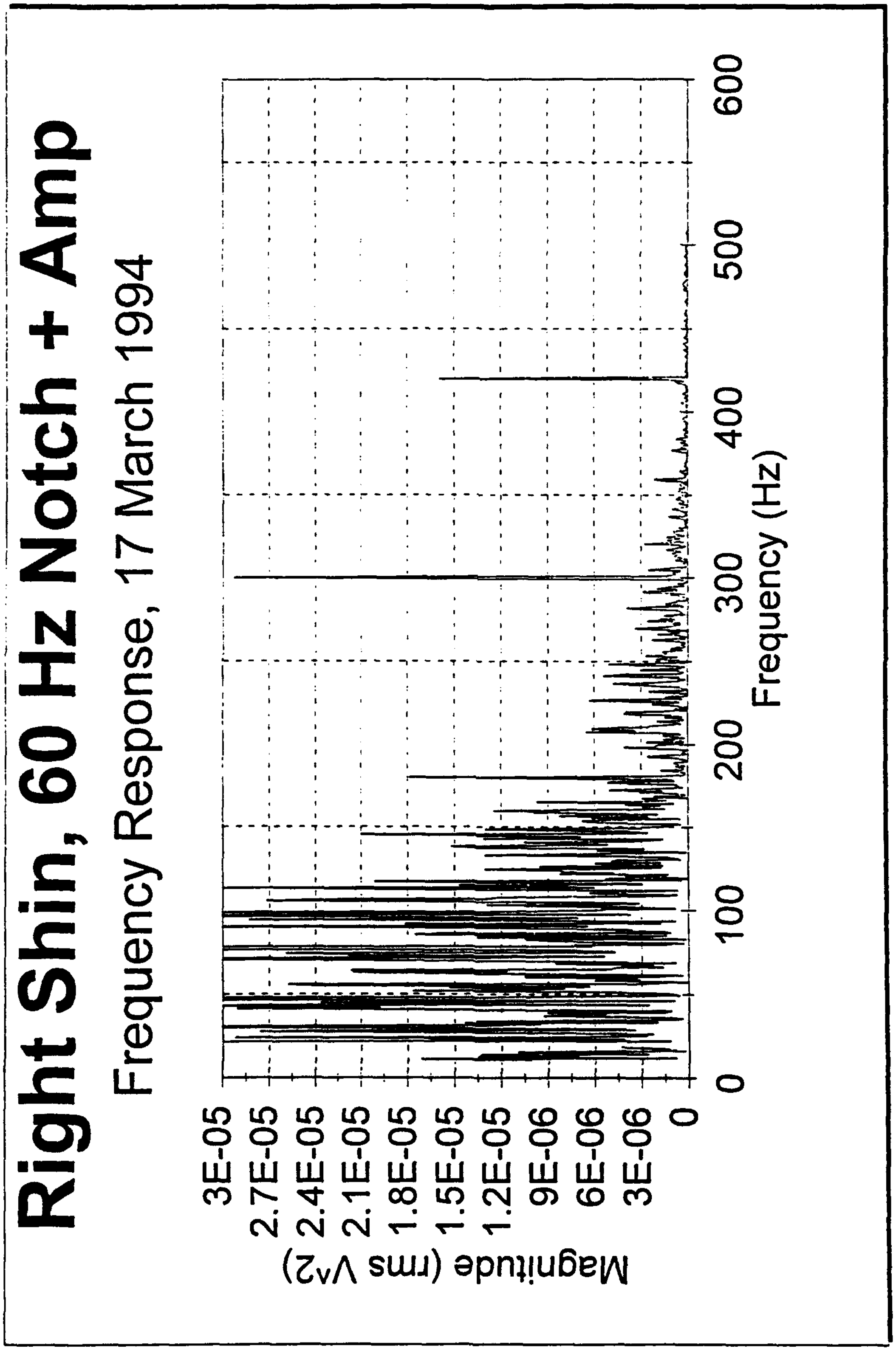
Appendix (E-8)



Right Quadriceps, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

Appendix (E-9)



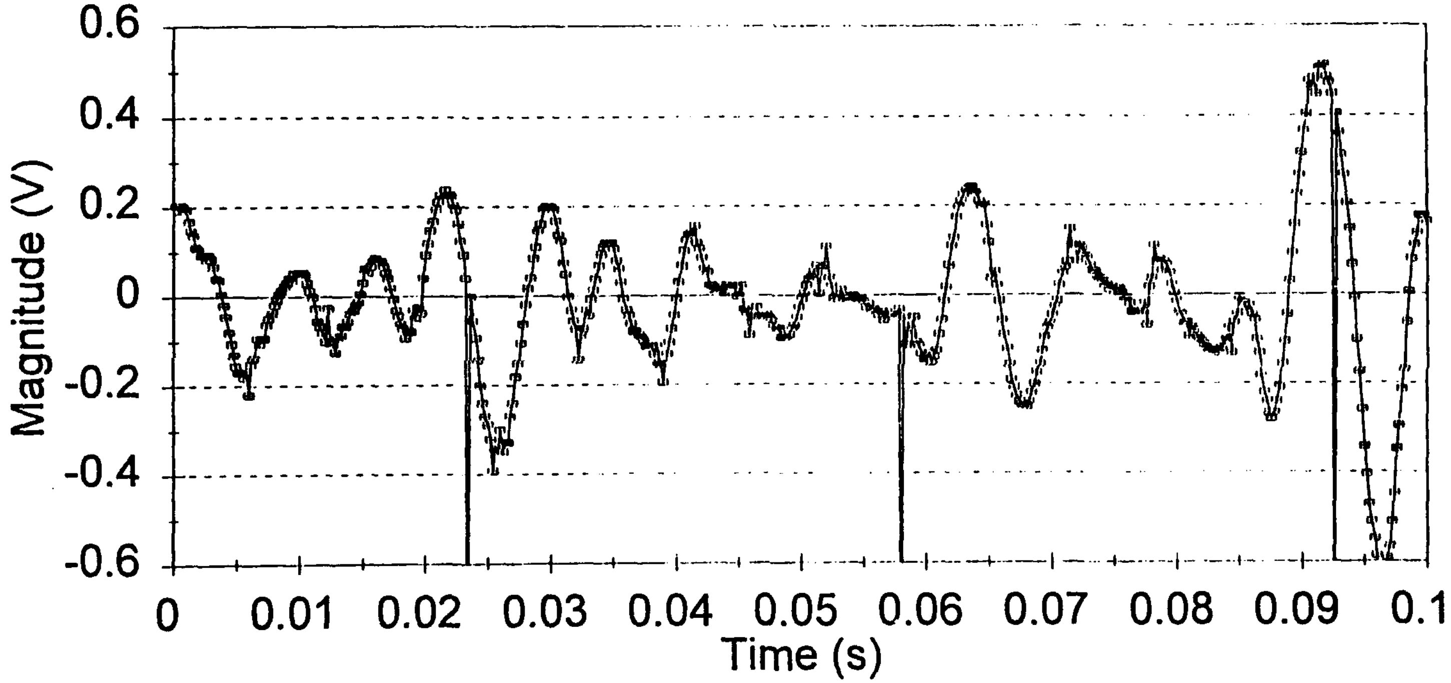
Right Shin, 60 Hz Notch + Amp

Frequency Response, 17 March 1994

Appendix (E-10)

Time Response, Right Ankle

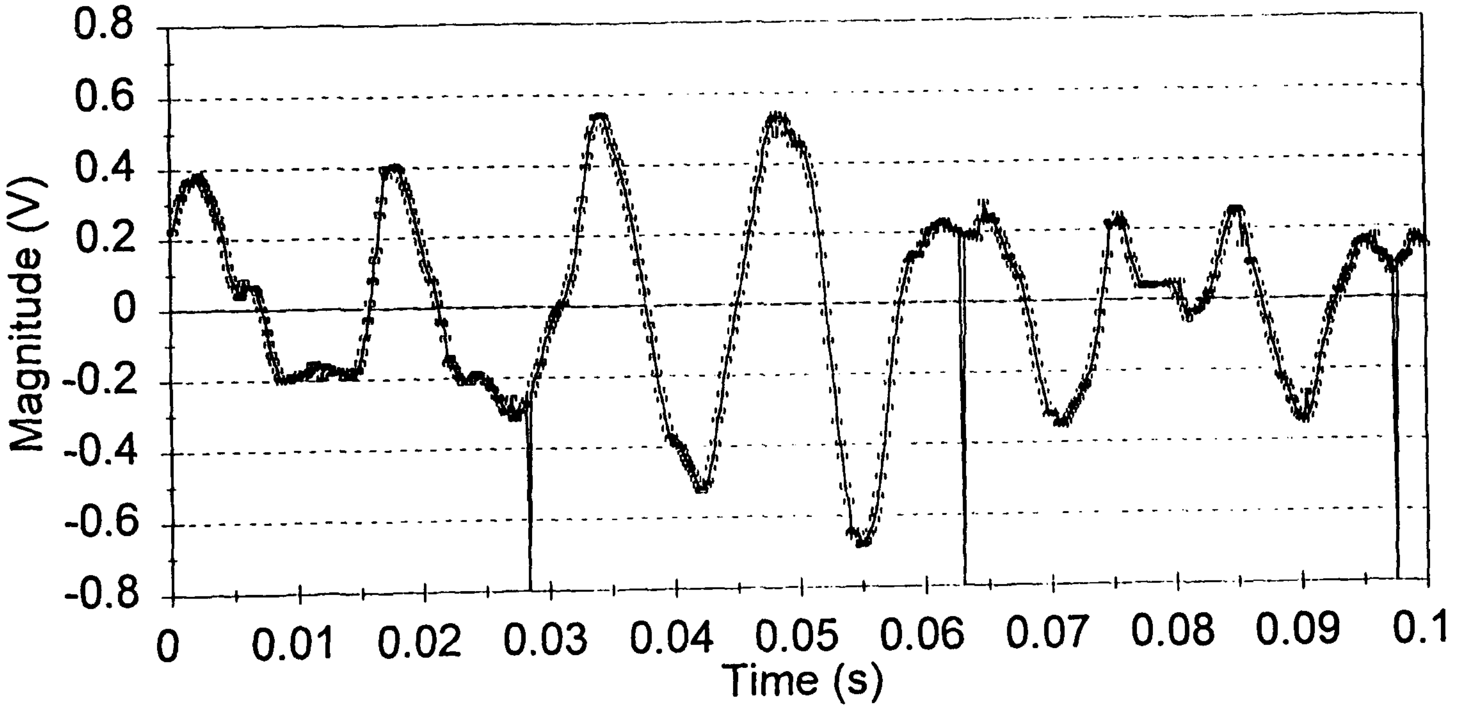
21 March 1994



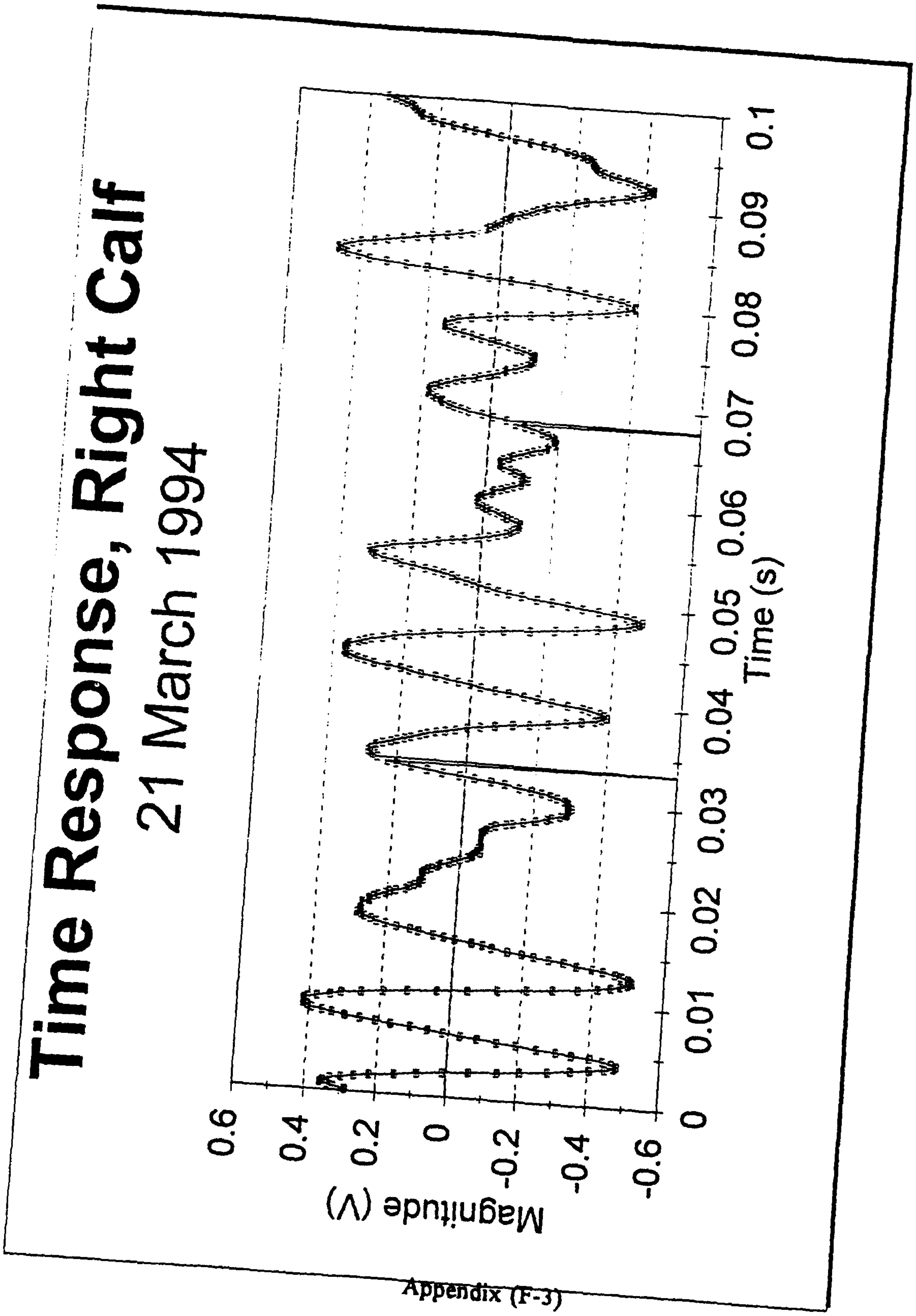
Appendix (F-1)

Time Response, Right Biceps

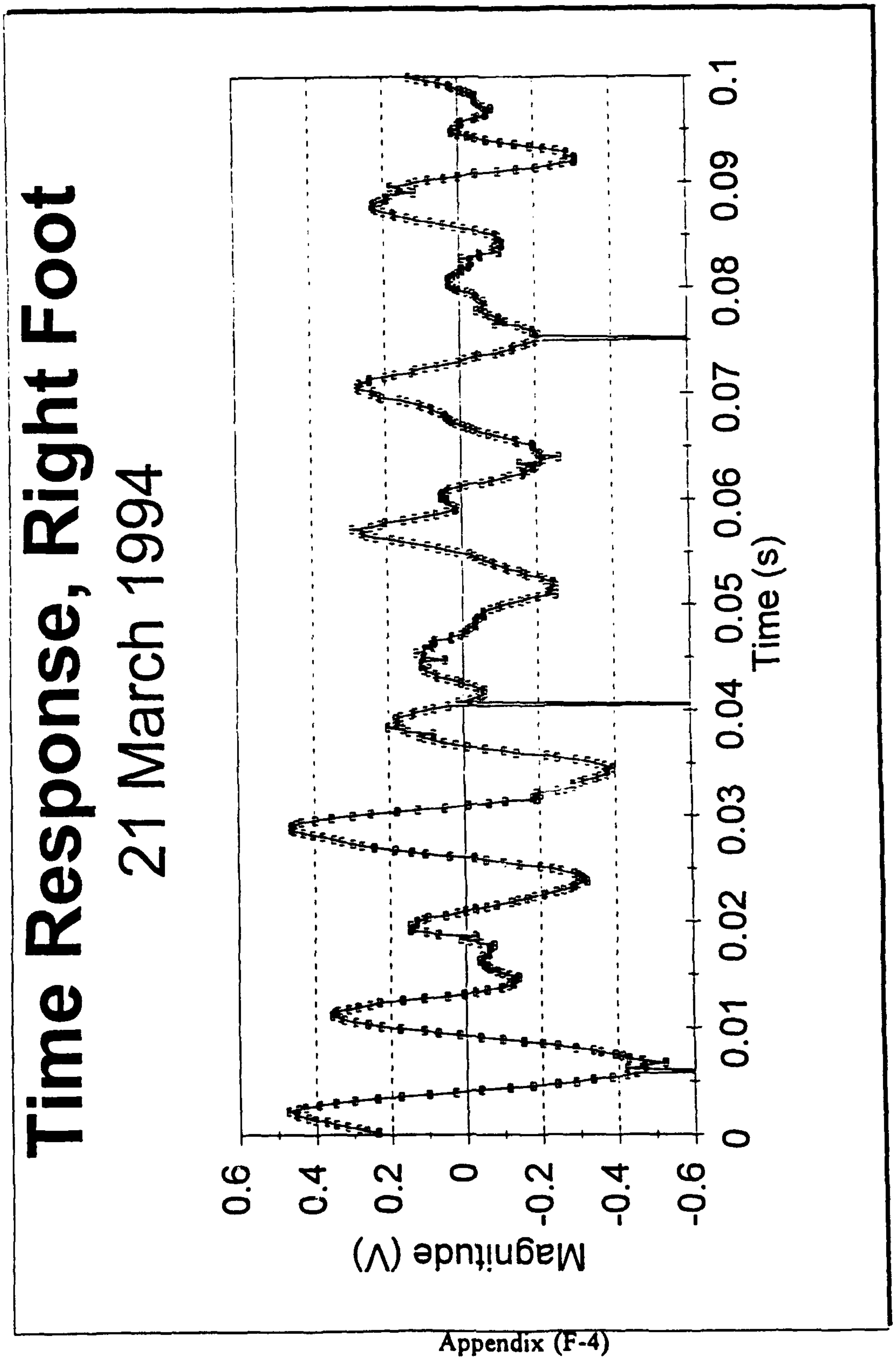
21 March 1994



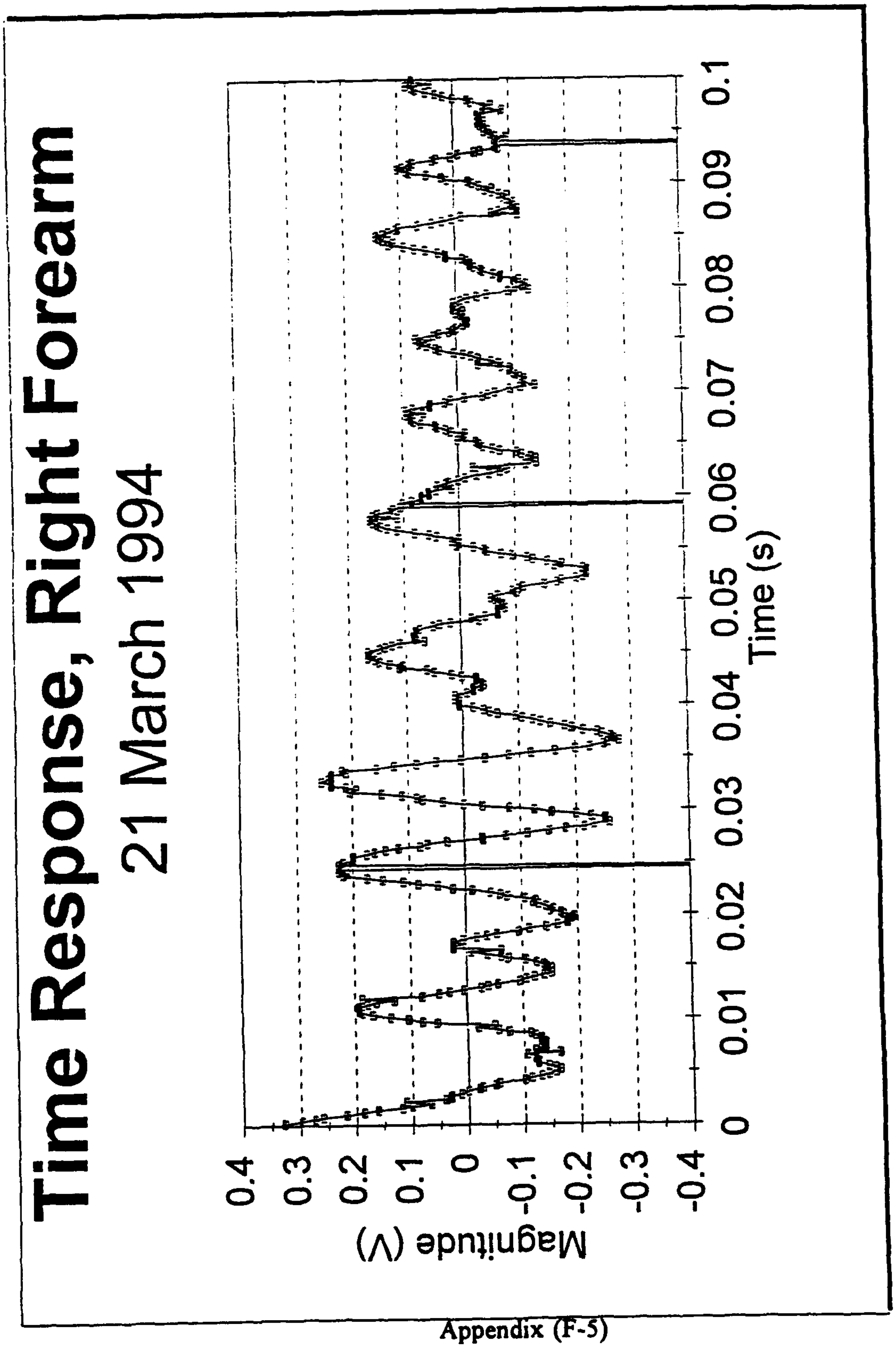
Appendix (F-2)


Time Response, Right Calf
21 March 1994

Appendix (F-3)


Time Response, Right Foot
21 March 1994
Magnitude (V)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Time (s)

Appendix (F-4)



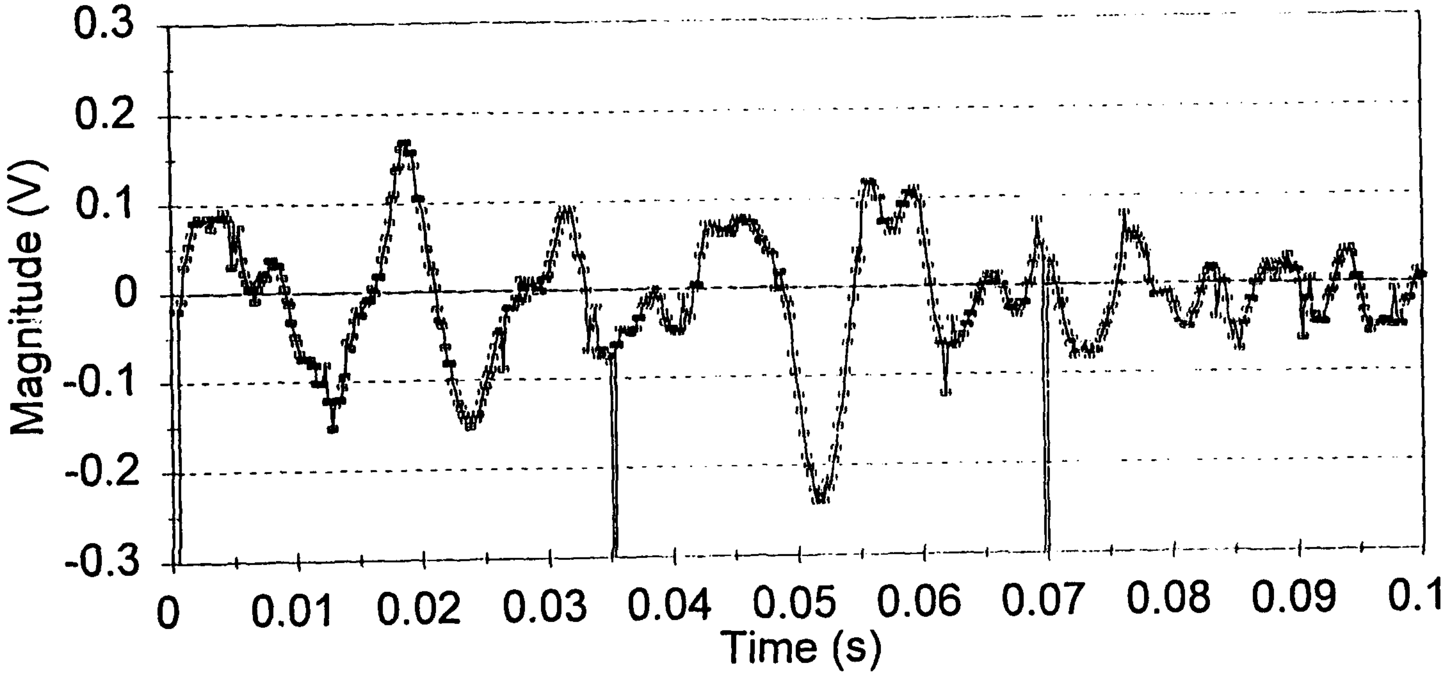
Time Response, Right Forearm

21 March 1994

Appendix (F-5)

Time Response, Right Hamstring

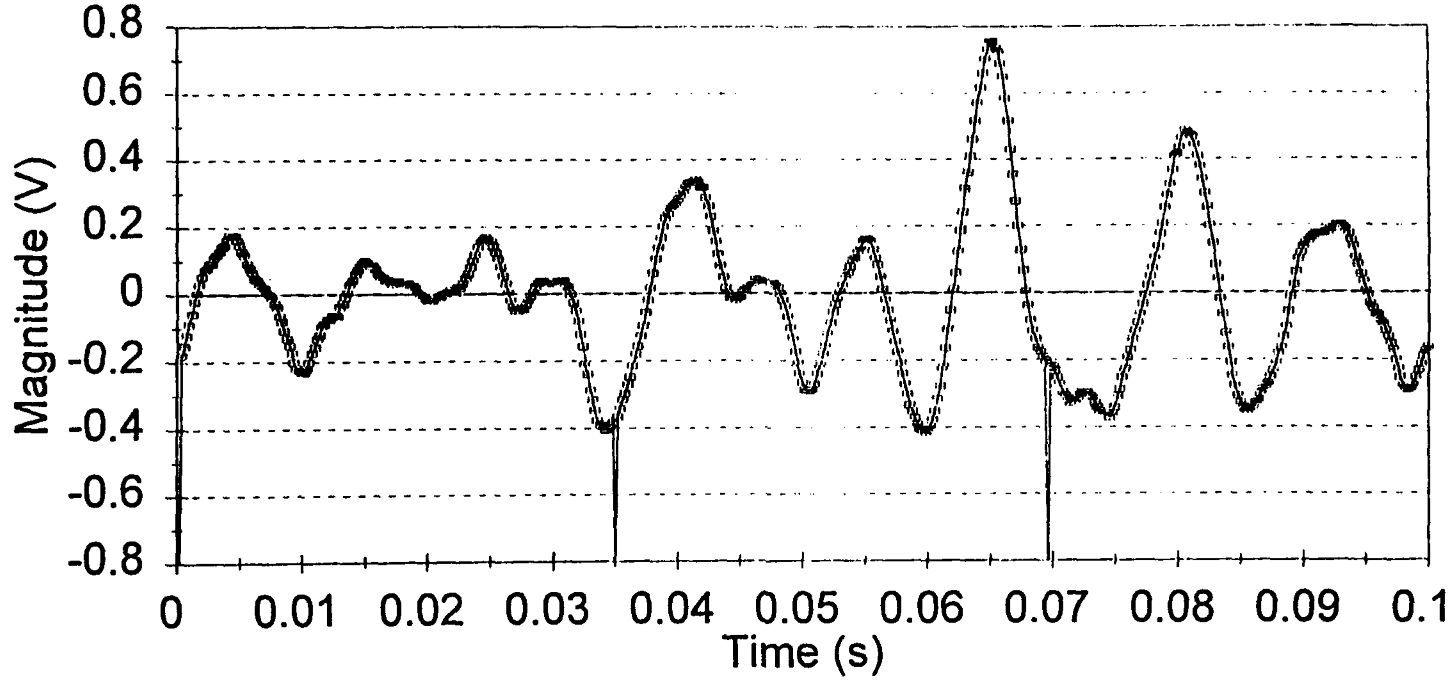
21 March 1994



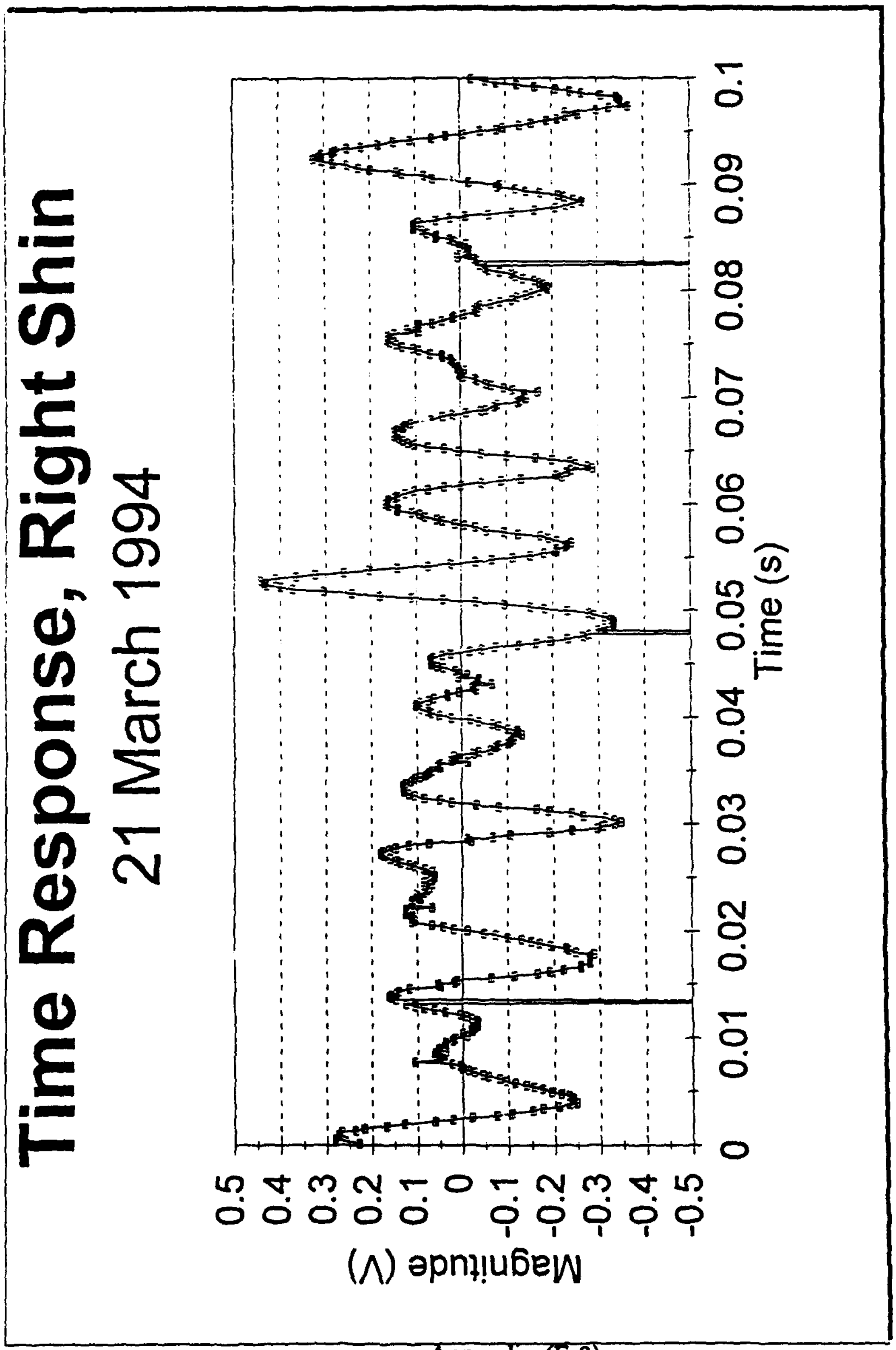
Appendix (F-6)

Time Response, Right Quadriceps

21 March 1994

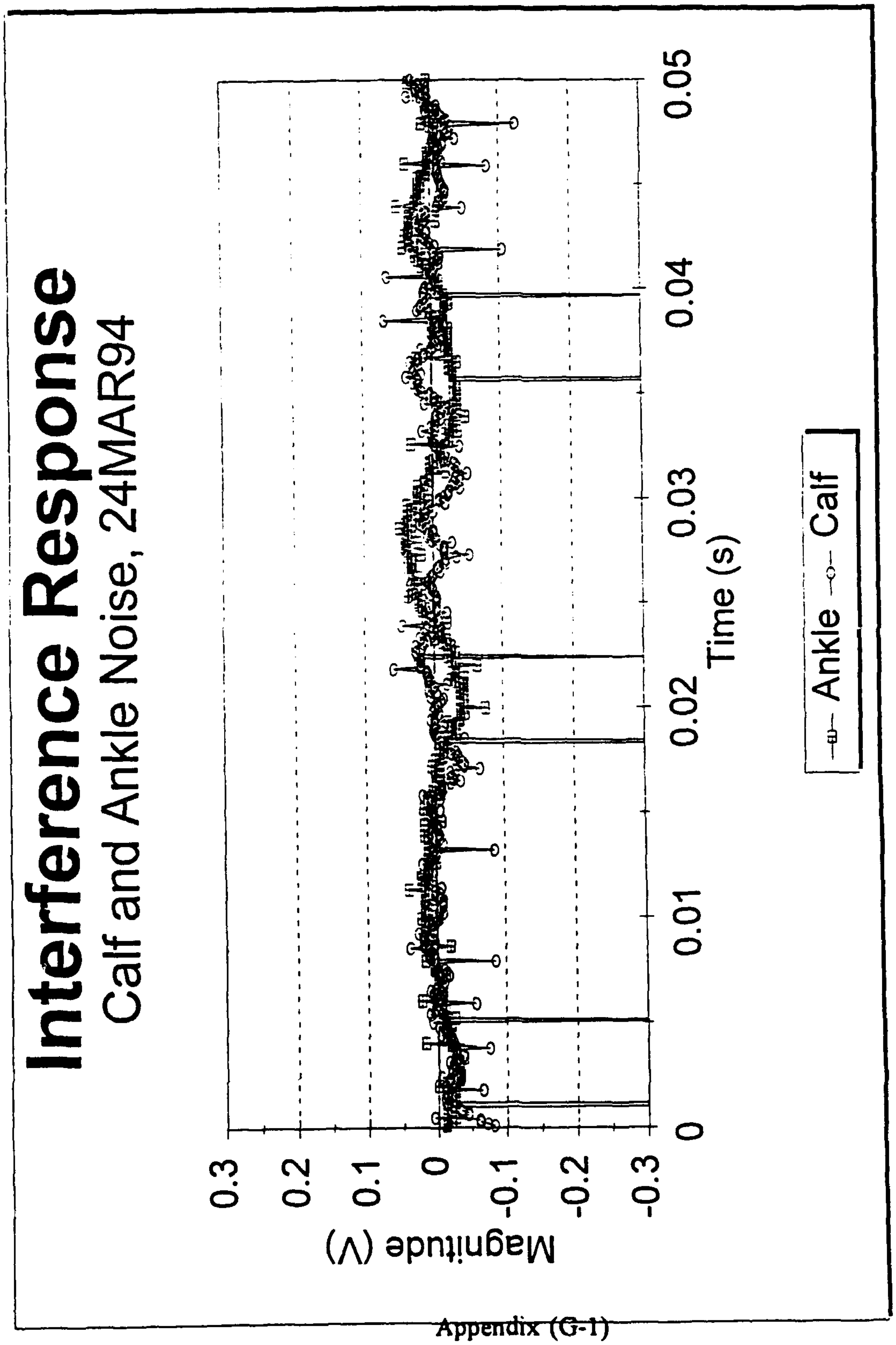


Appendix (F-7)



Time Response, Right Shin
21 March 1994

Appendix (F-8)



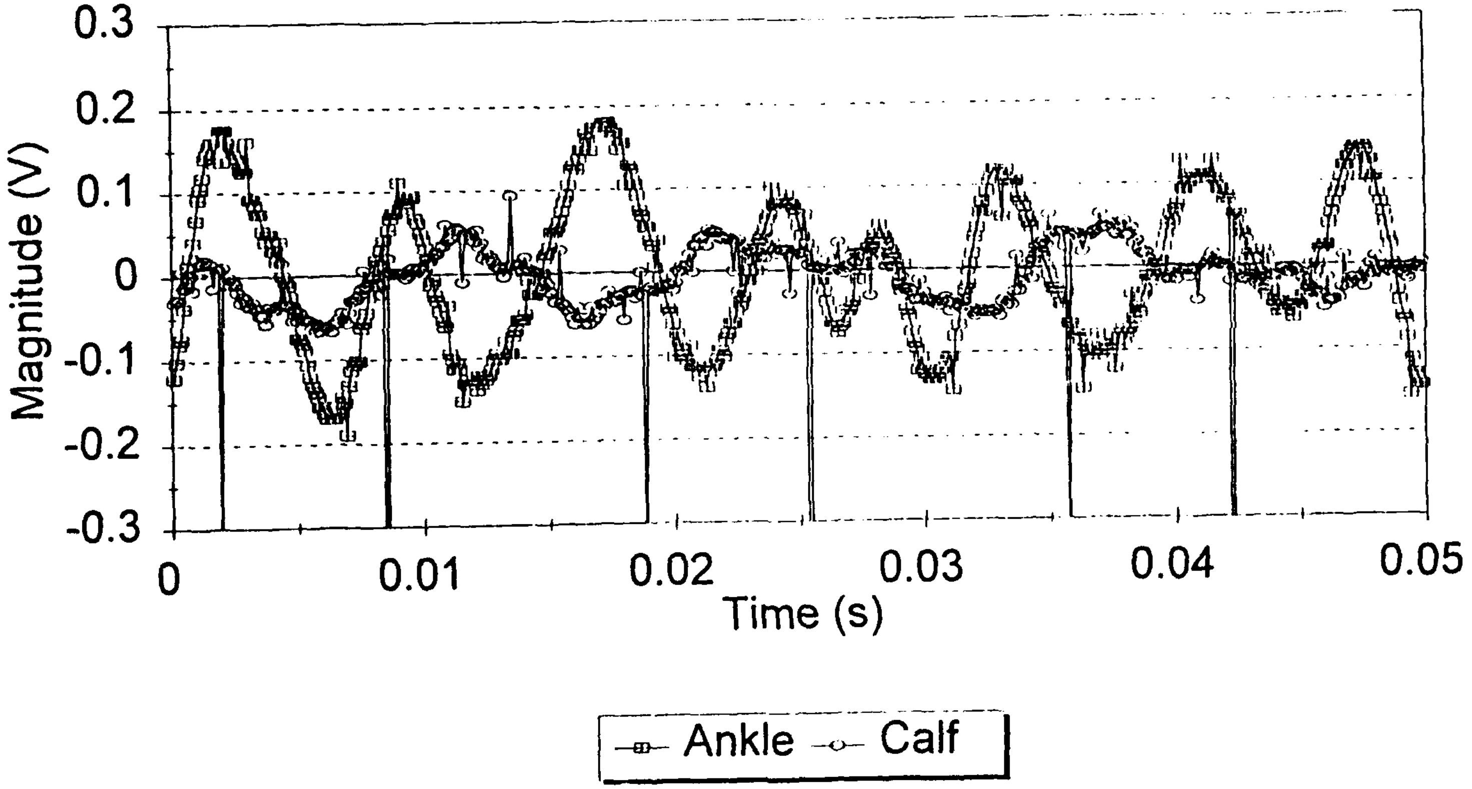
Interference Response

Calf and Ankle Noise, 24MAR94

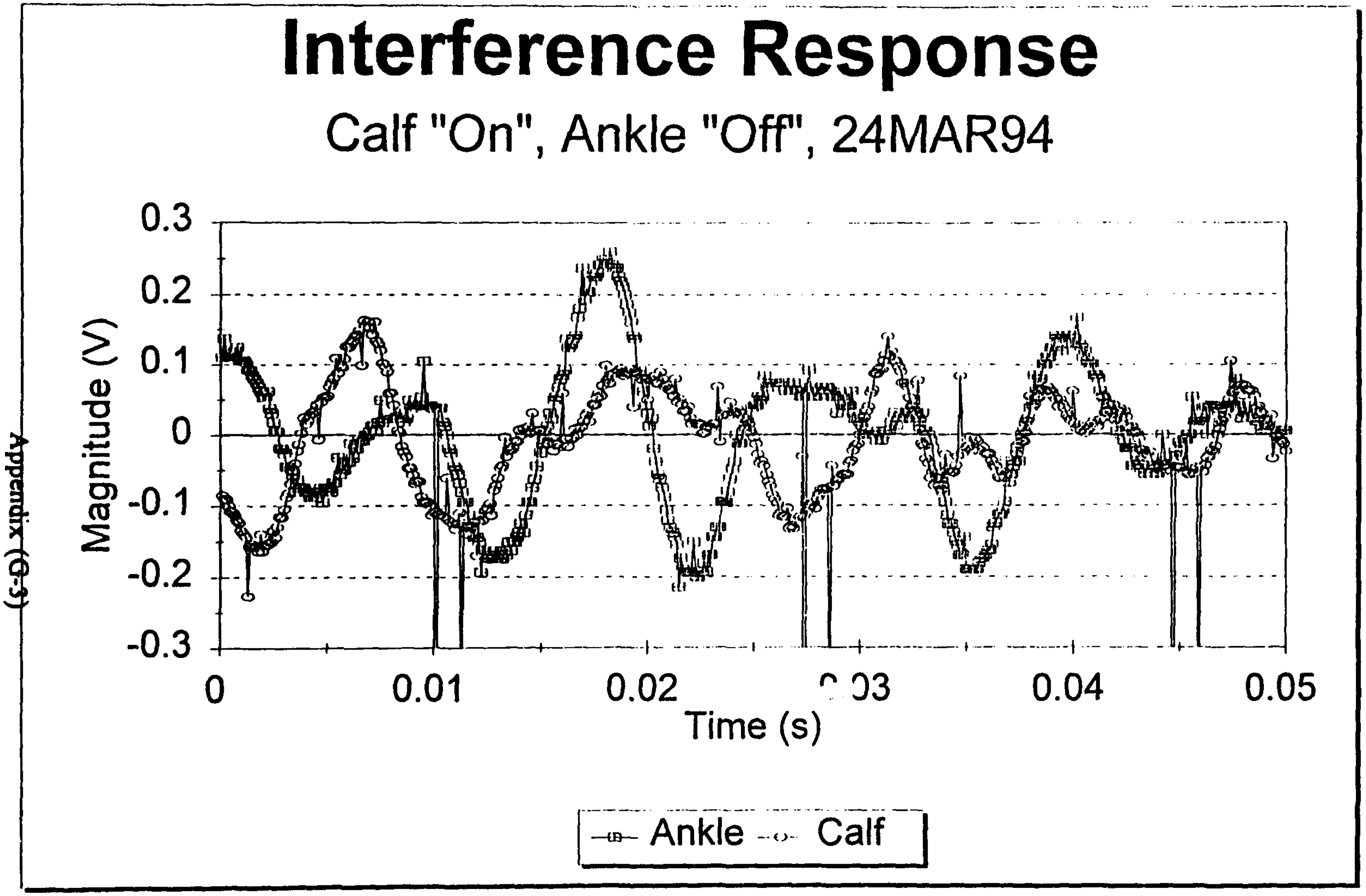
Appendix (G-1)

Interference Response

Ankle "On", Calf "Off", 24MAR94



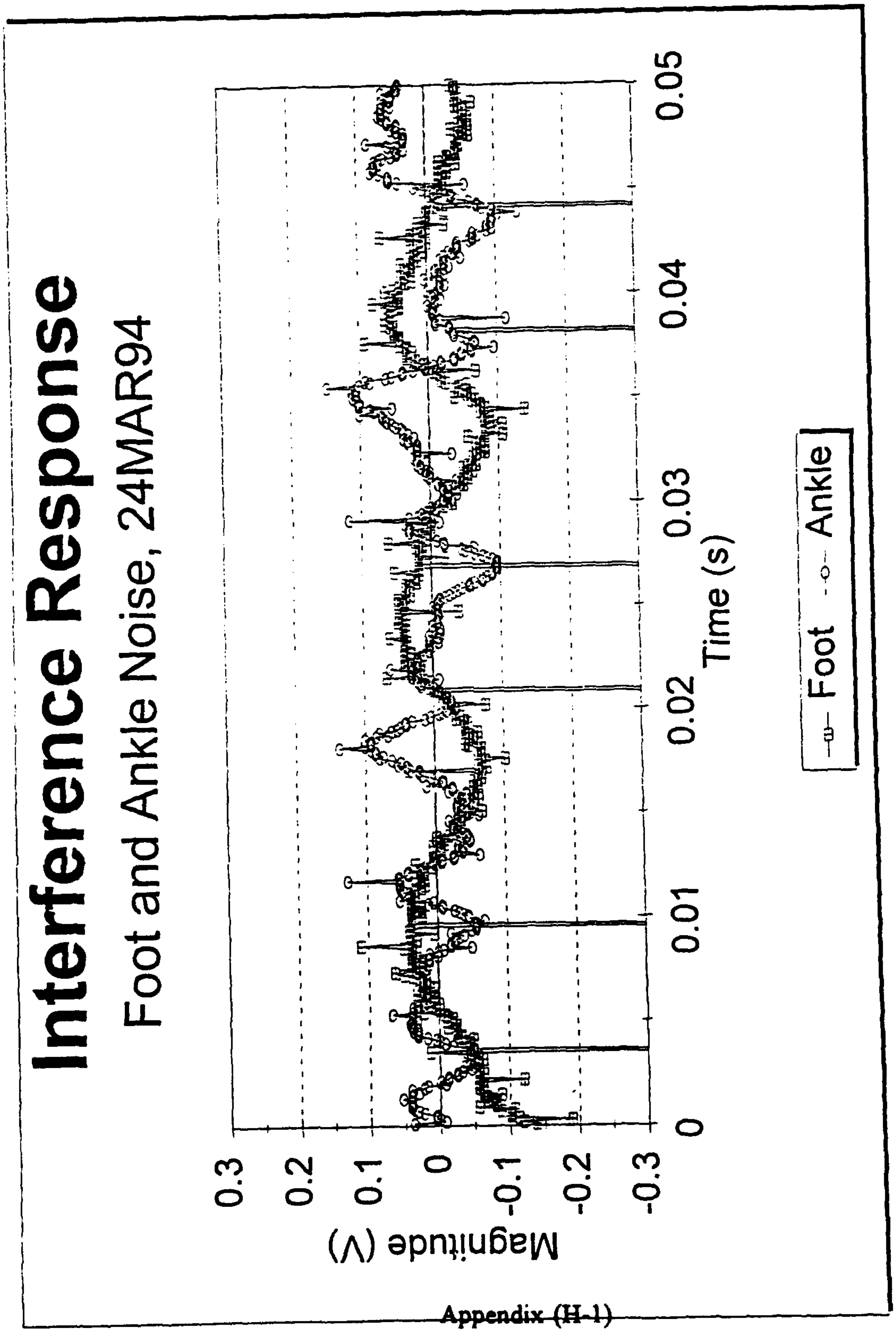
Appendix (G-2)



Interference Response

Calf "On", Ankle "Off", 24MAR94

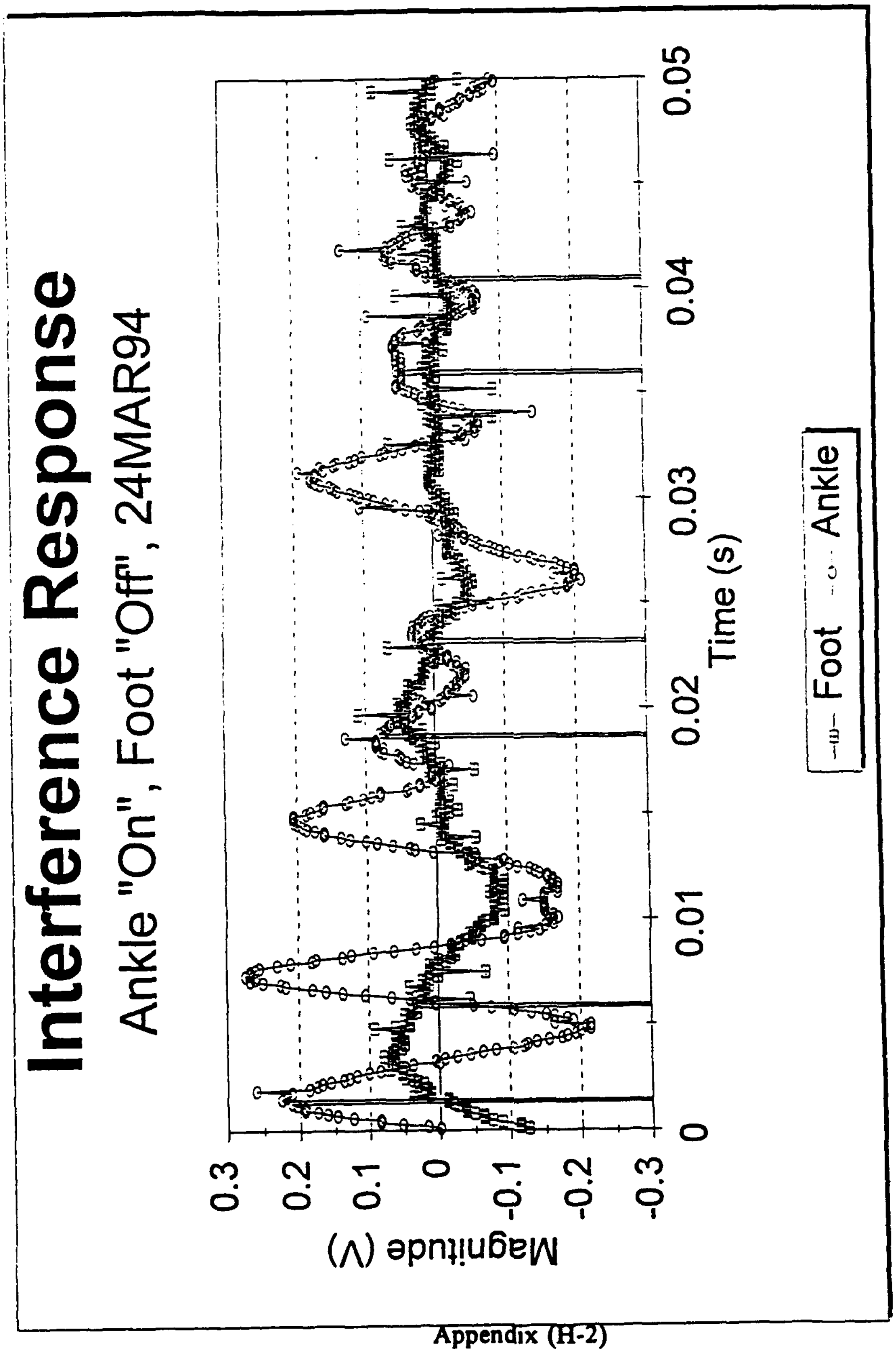
Appendix (G-3)



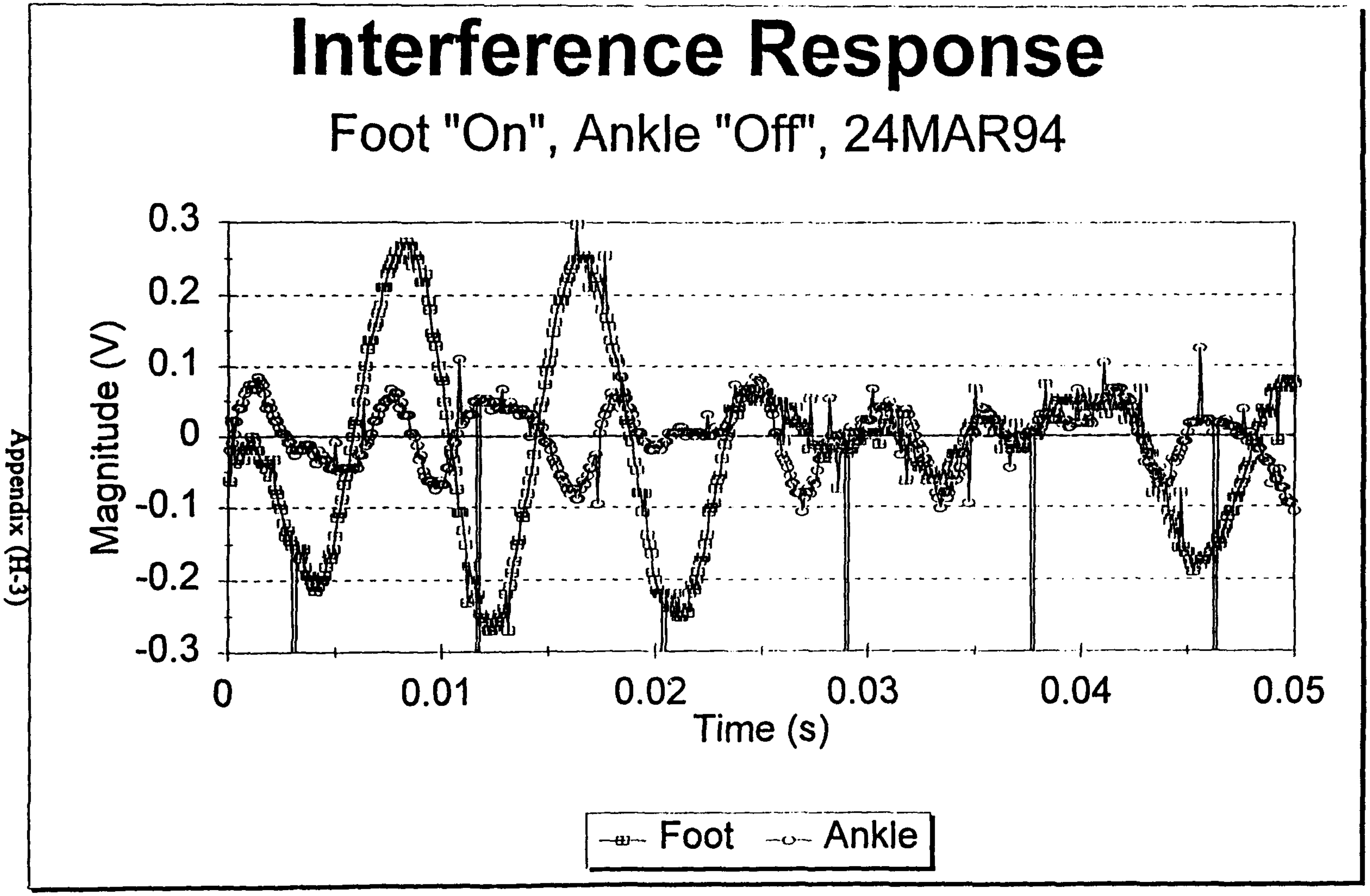
Interference Response

Foot and Ankle Noise, 24MAR94

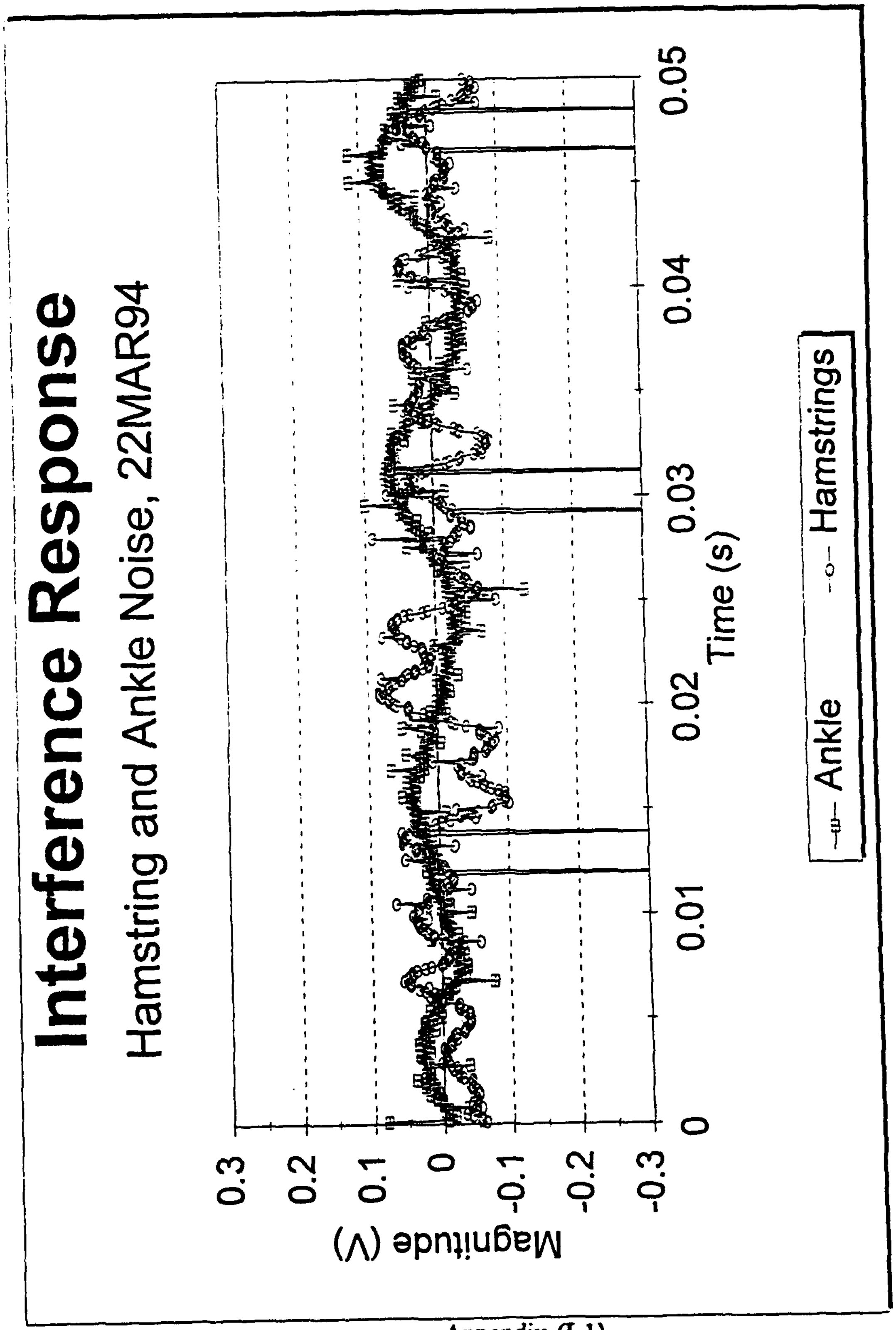
Appendix (H-1)


Interference Response
Ankle "On", Foot "Off", 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Foot
Ankle

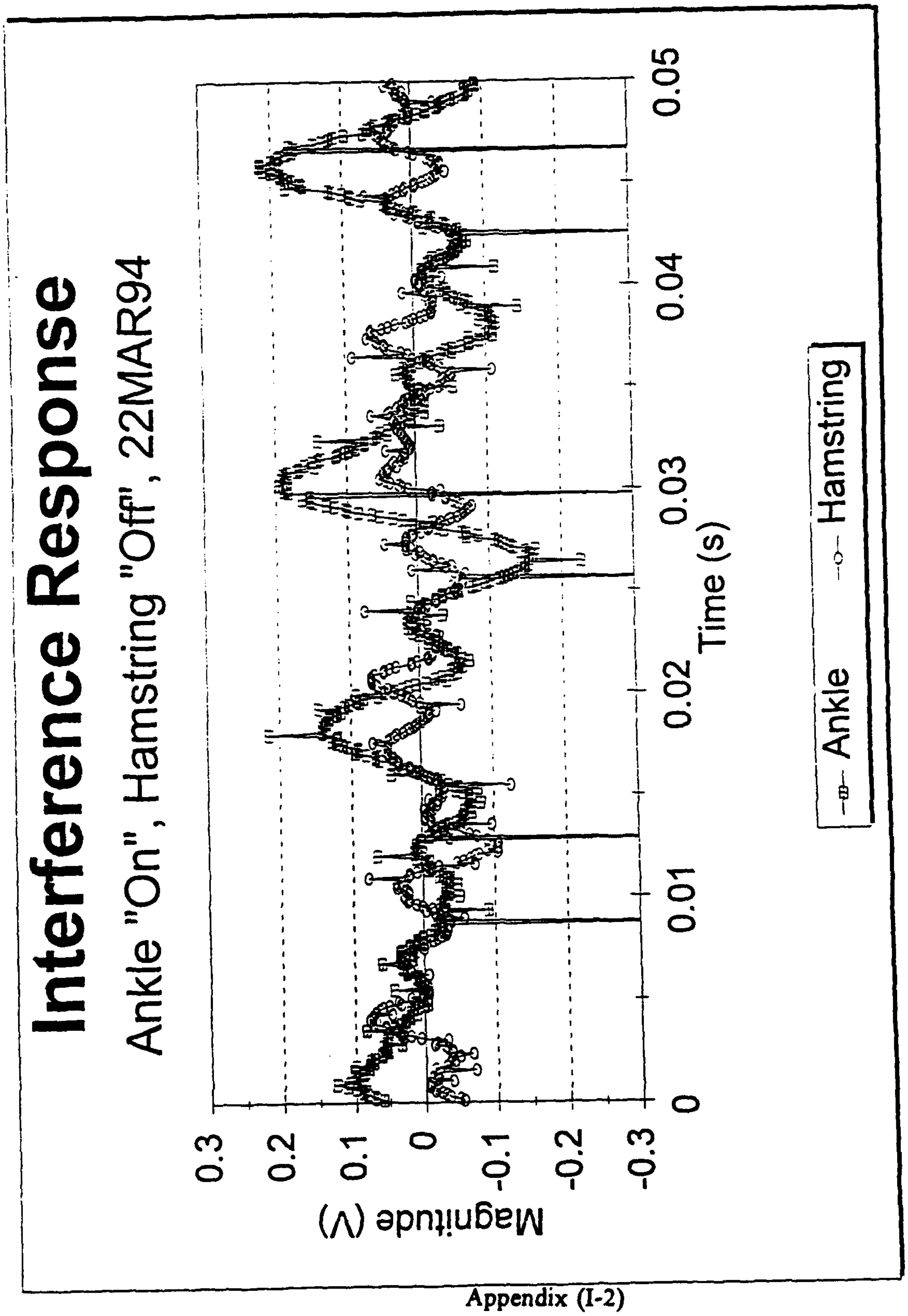
Appendix (H-2)


Interference Response
Foot "On", Ankle "Off", 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Ankle

Appendix (H-3)


Interference Response
Hamstring and Ankle Noise, 22MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Ankle
Hamstrings

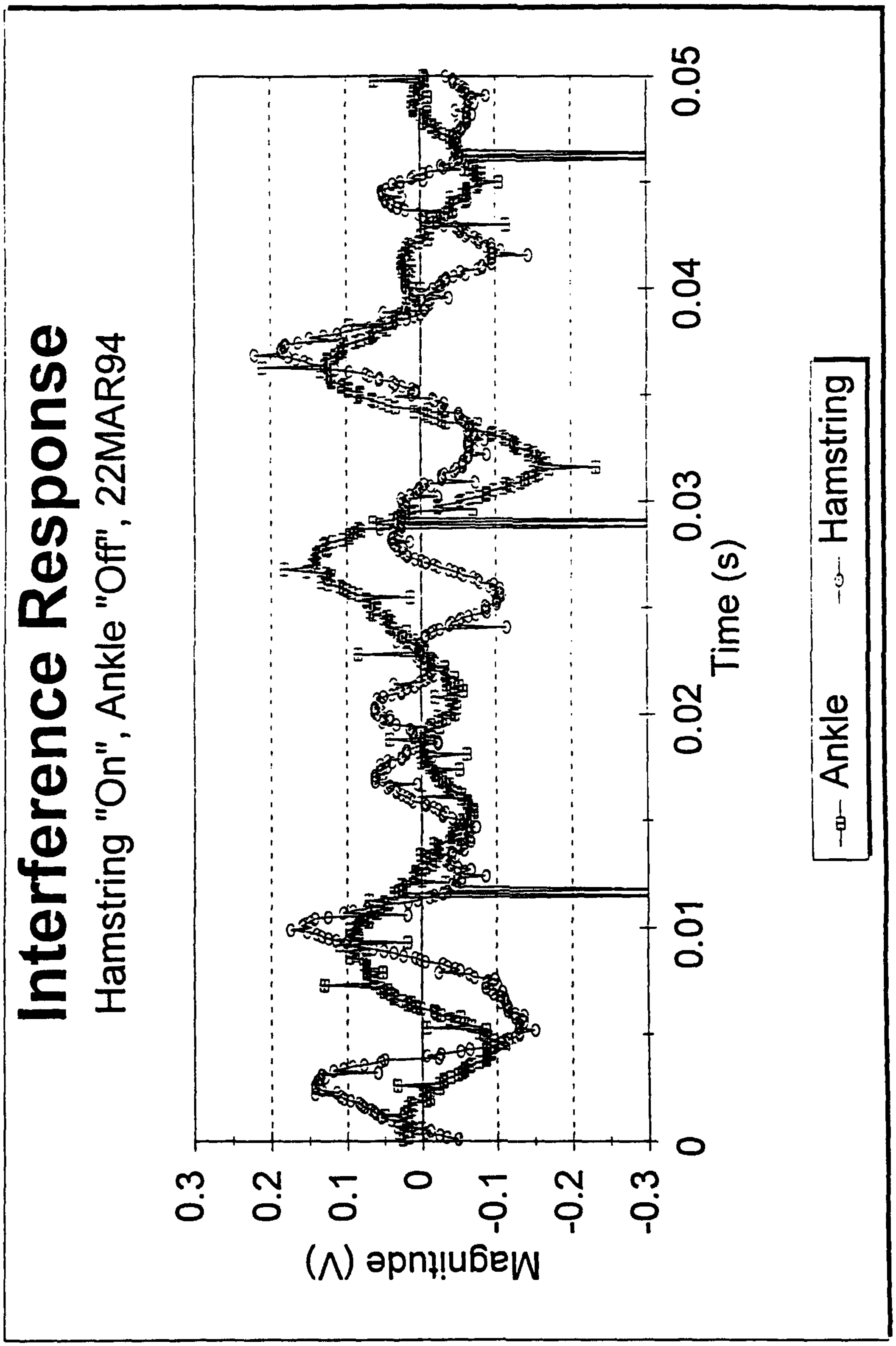
Appendix (I-1)



Interference Response

Ankle "On", Hamstring "Off", 22MAR94

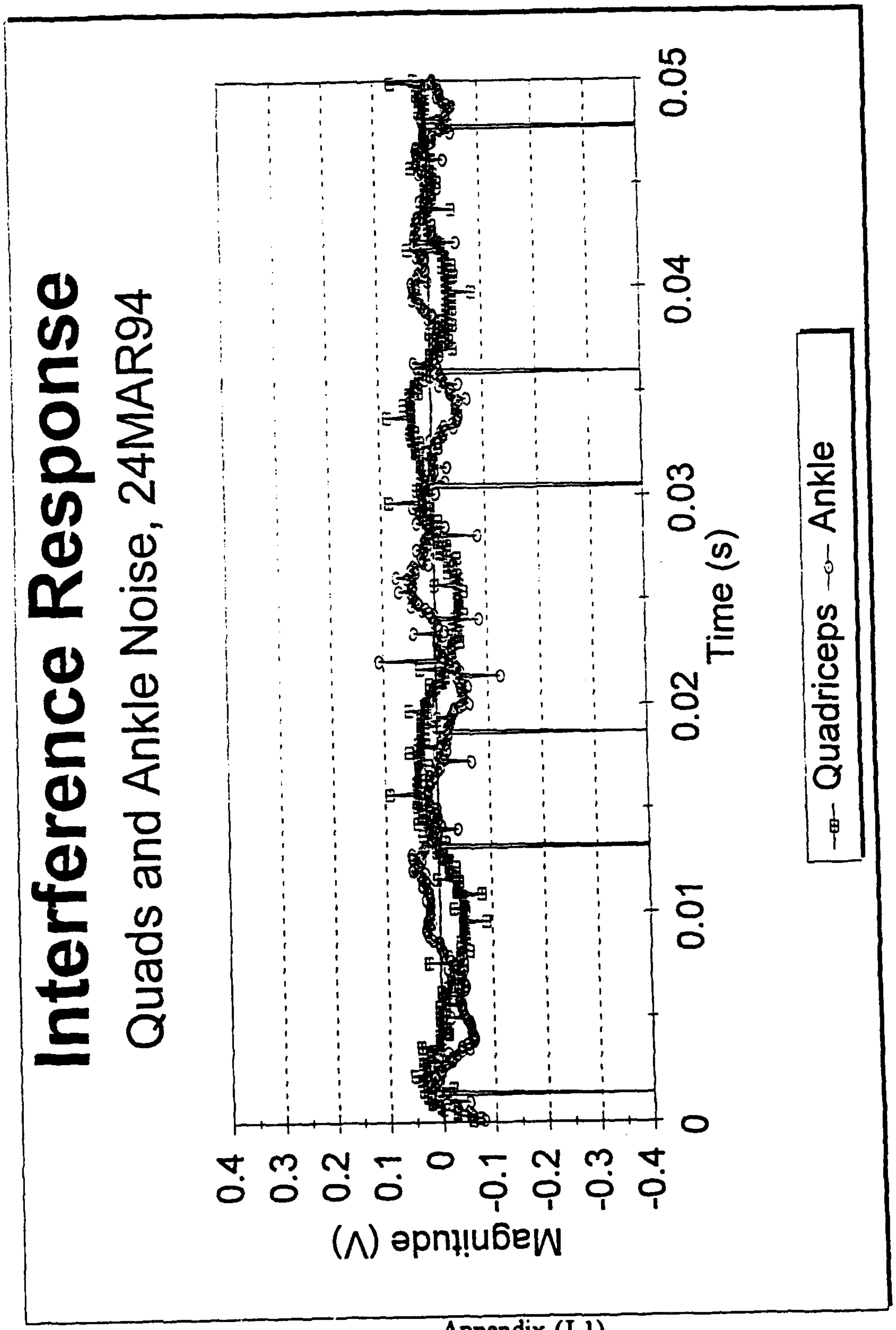
Appendix (I-2)



Interference Response

Hamstring "On", Ankle "Off", 22MAR94

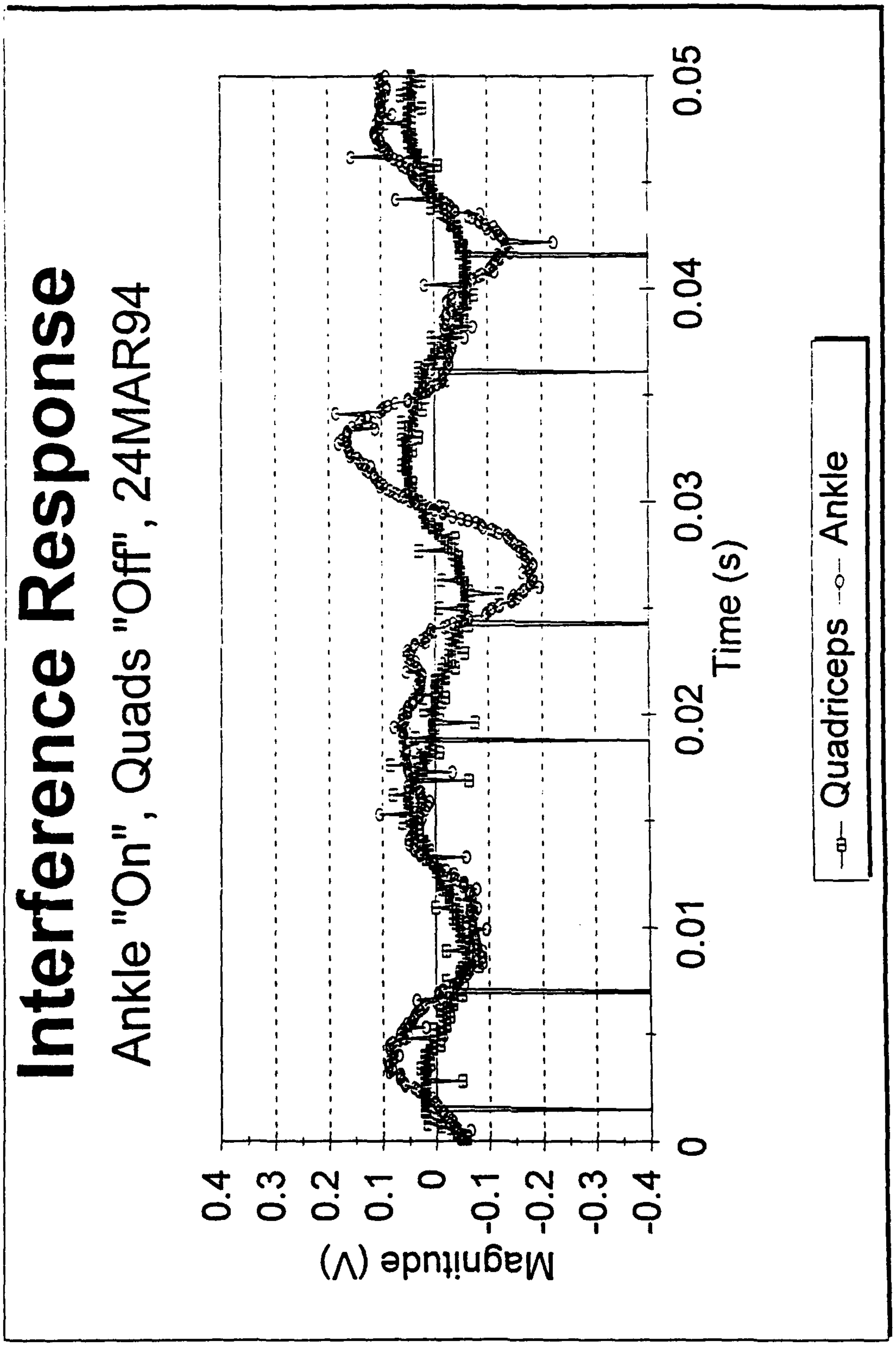
Appendix (I-3)



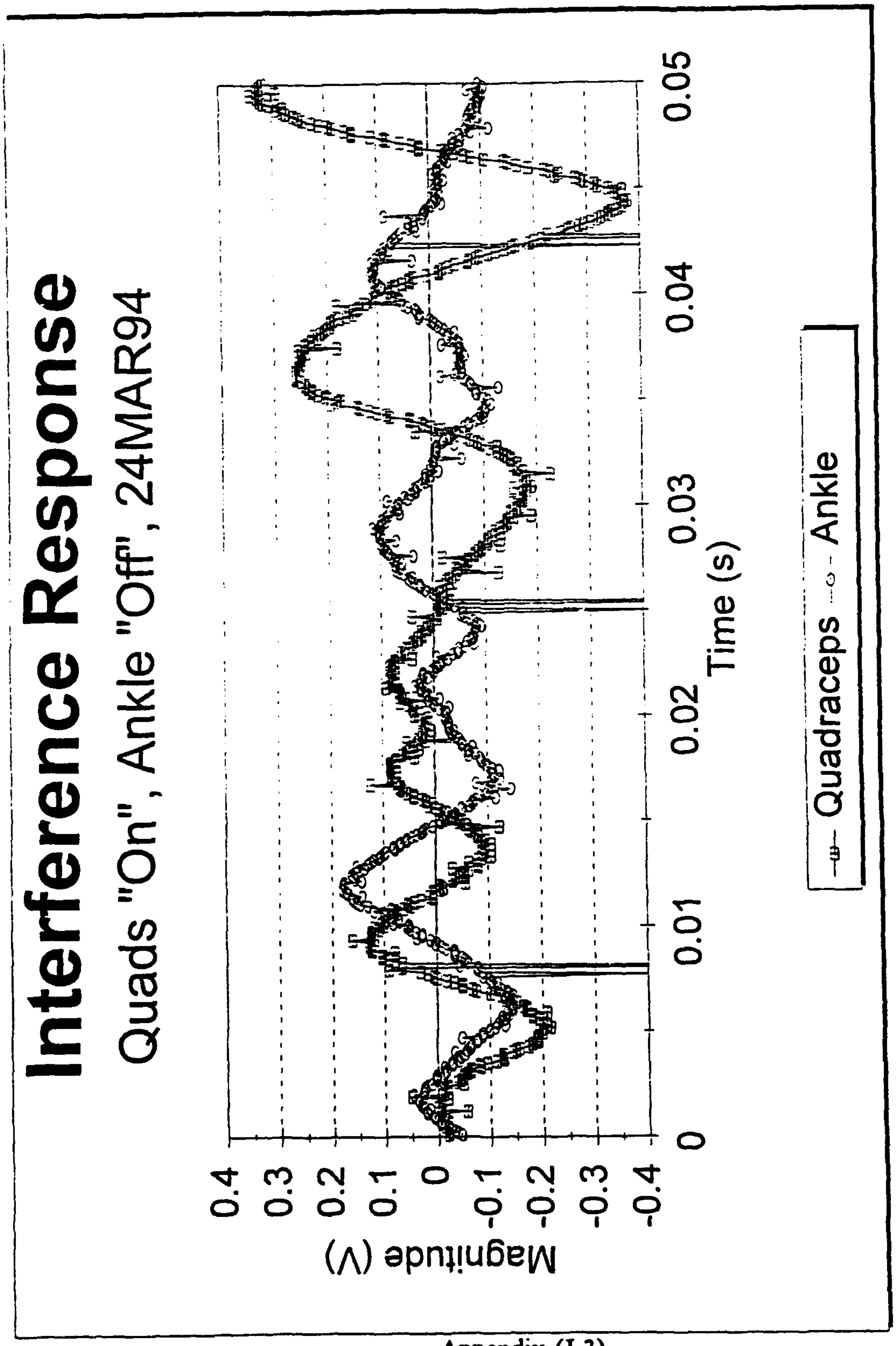
Interference Response

Quads and Ankle Noise, 24MAR94

Appendix (J-1)


Interference Response
Ankle "On", Quads "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Quadriceps
Ankle

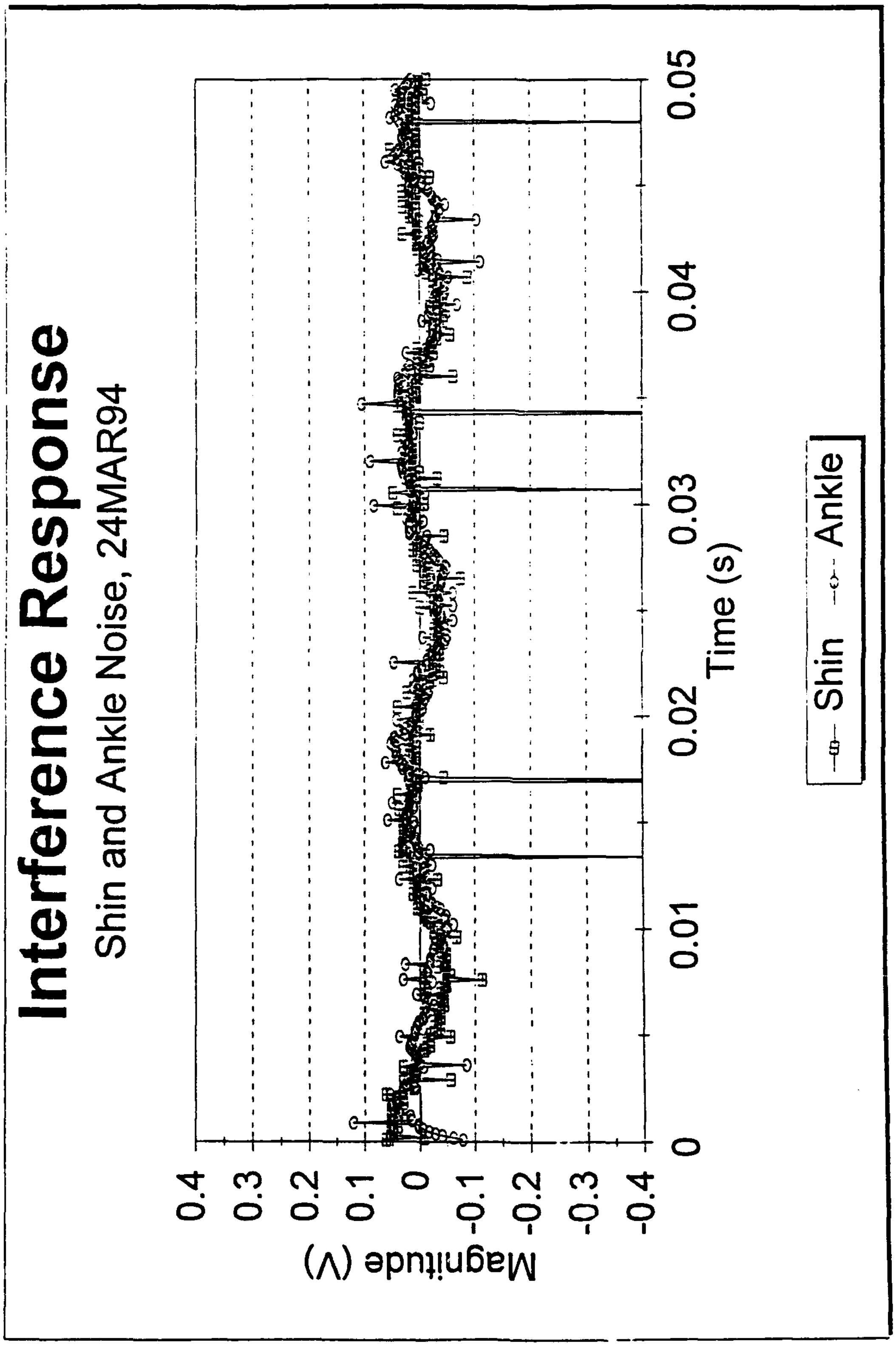
Appendix (J-2)



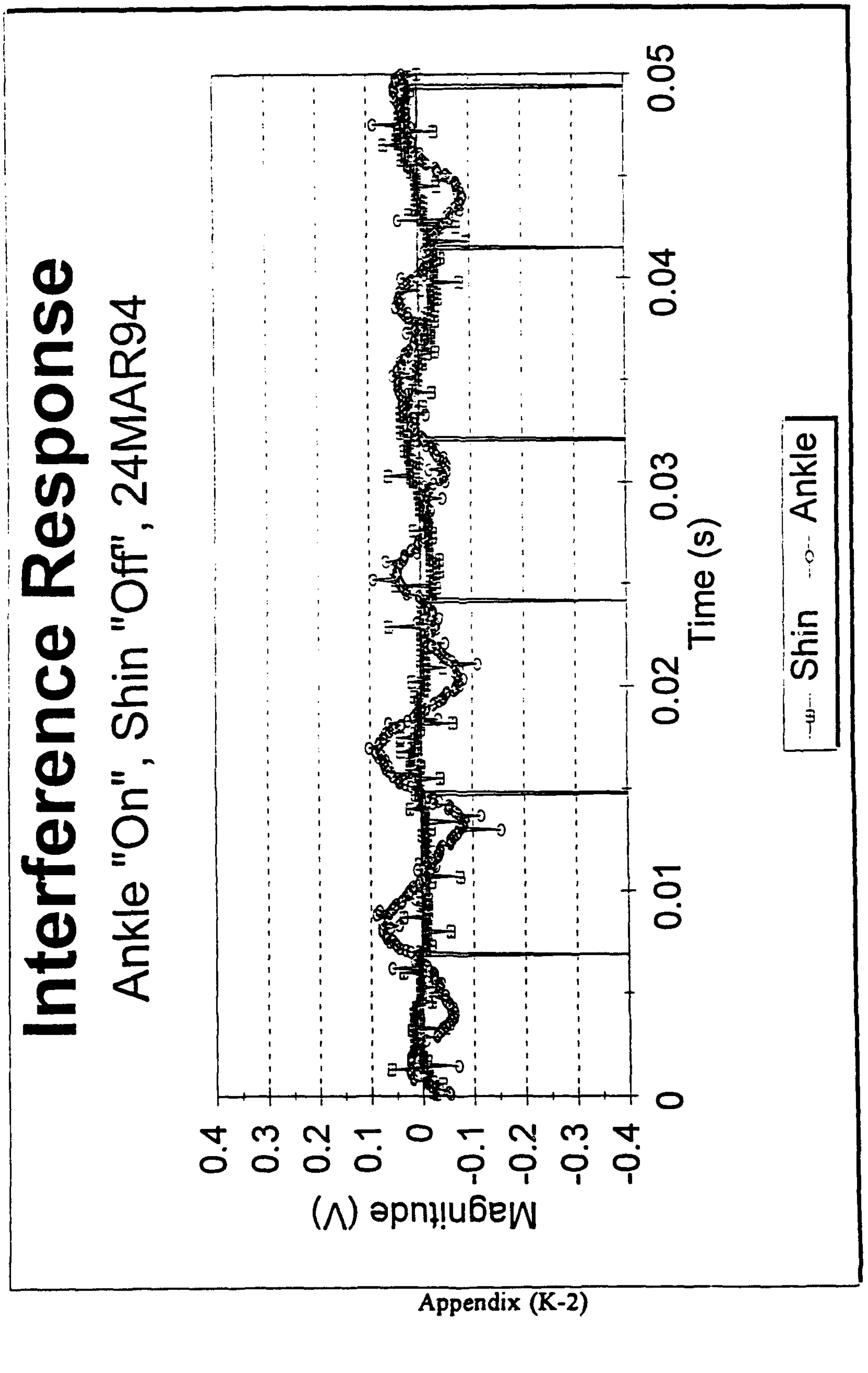
Interference Response

Quads "On", Ankle "Off", 24MAR94

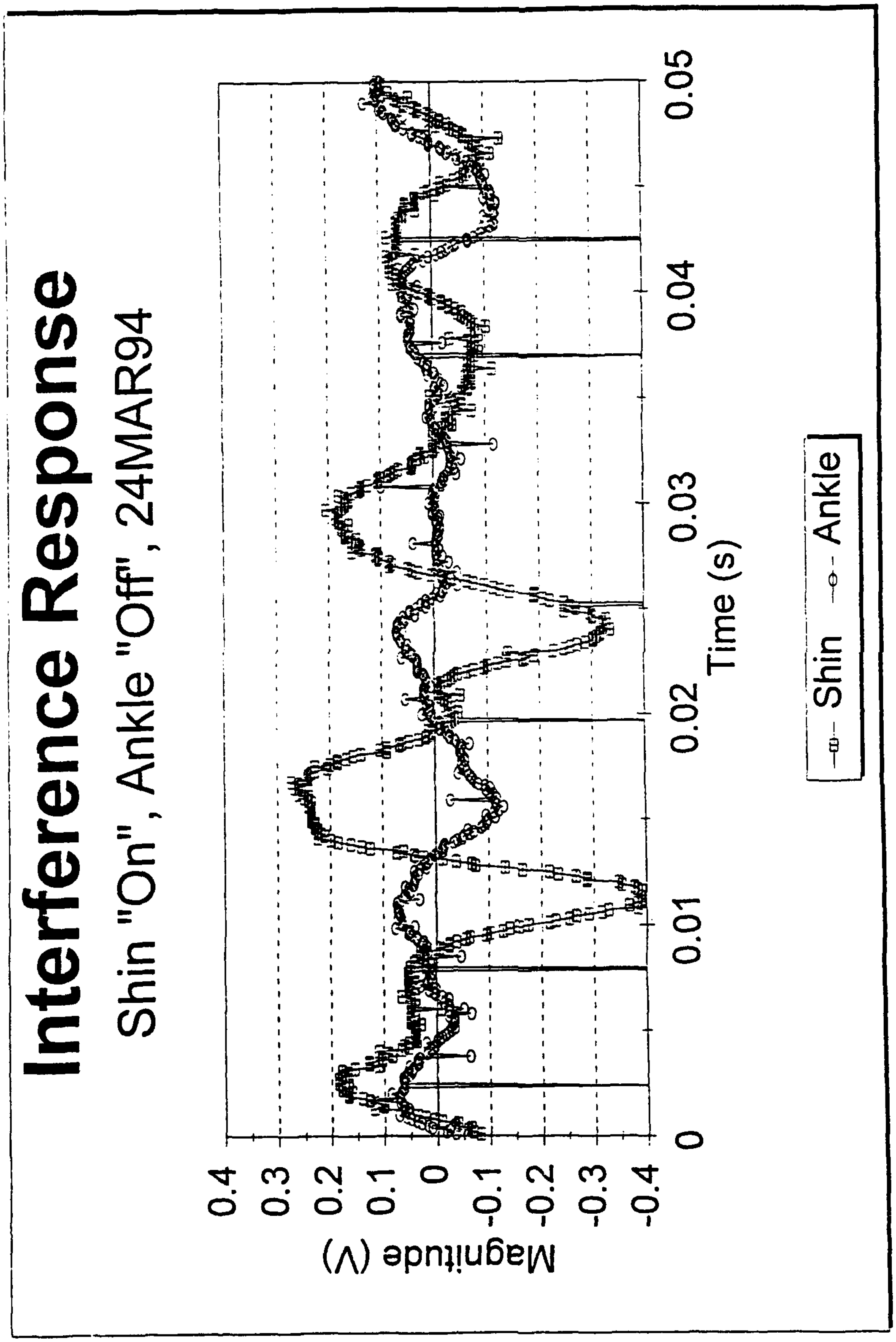
Appendix (J-3)


Interference Response
Shin and Ankle Noise, 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Ankle

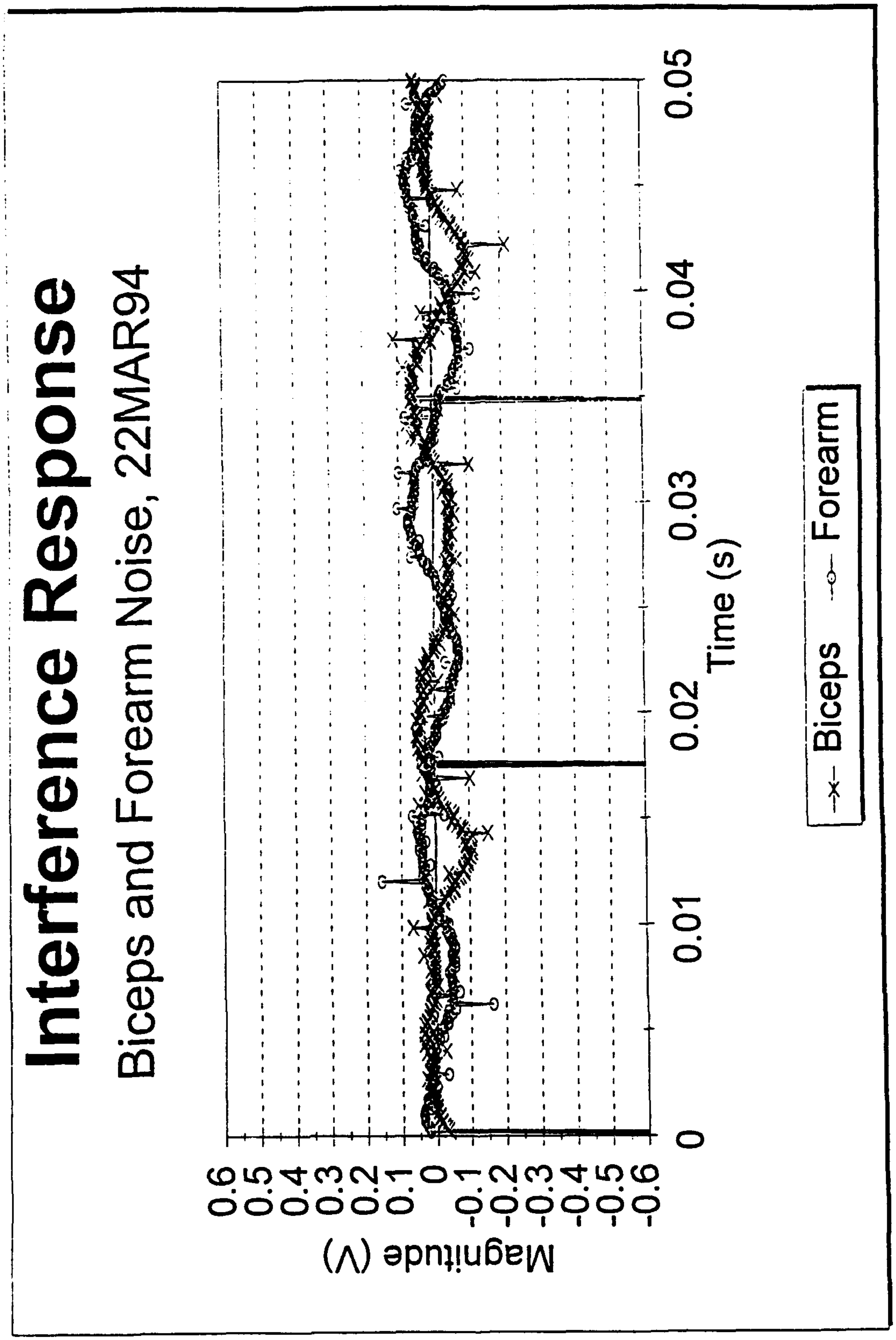
Appendix (K-1)


Interference Response
Ankle "On", Shin "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Ankle

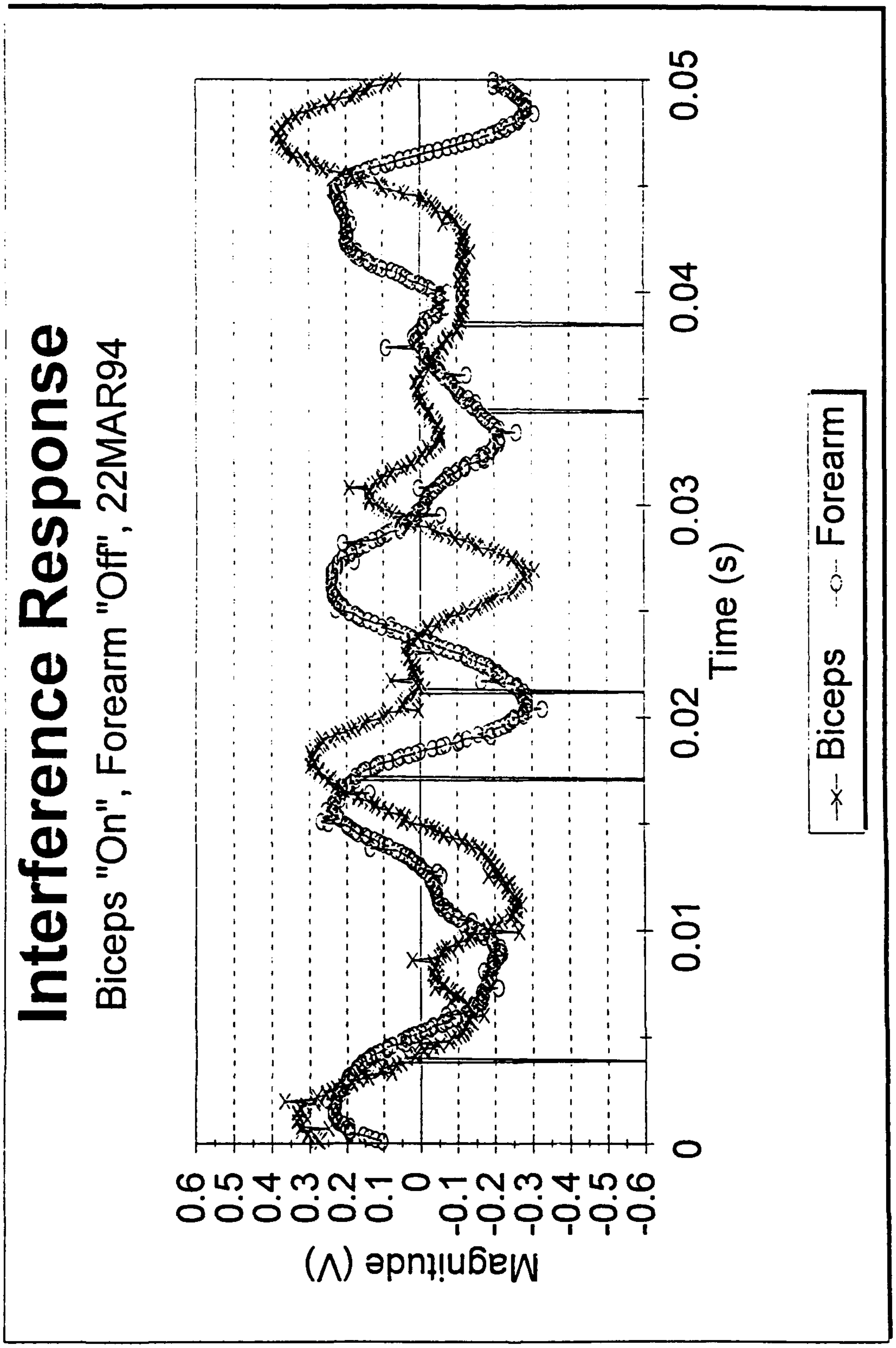
Appendix (K-2)


Interference Response
Shin "On", Ankle "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Shin
Ankle

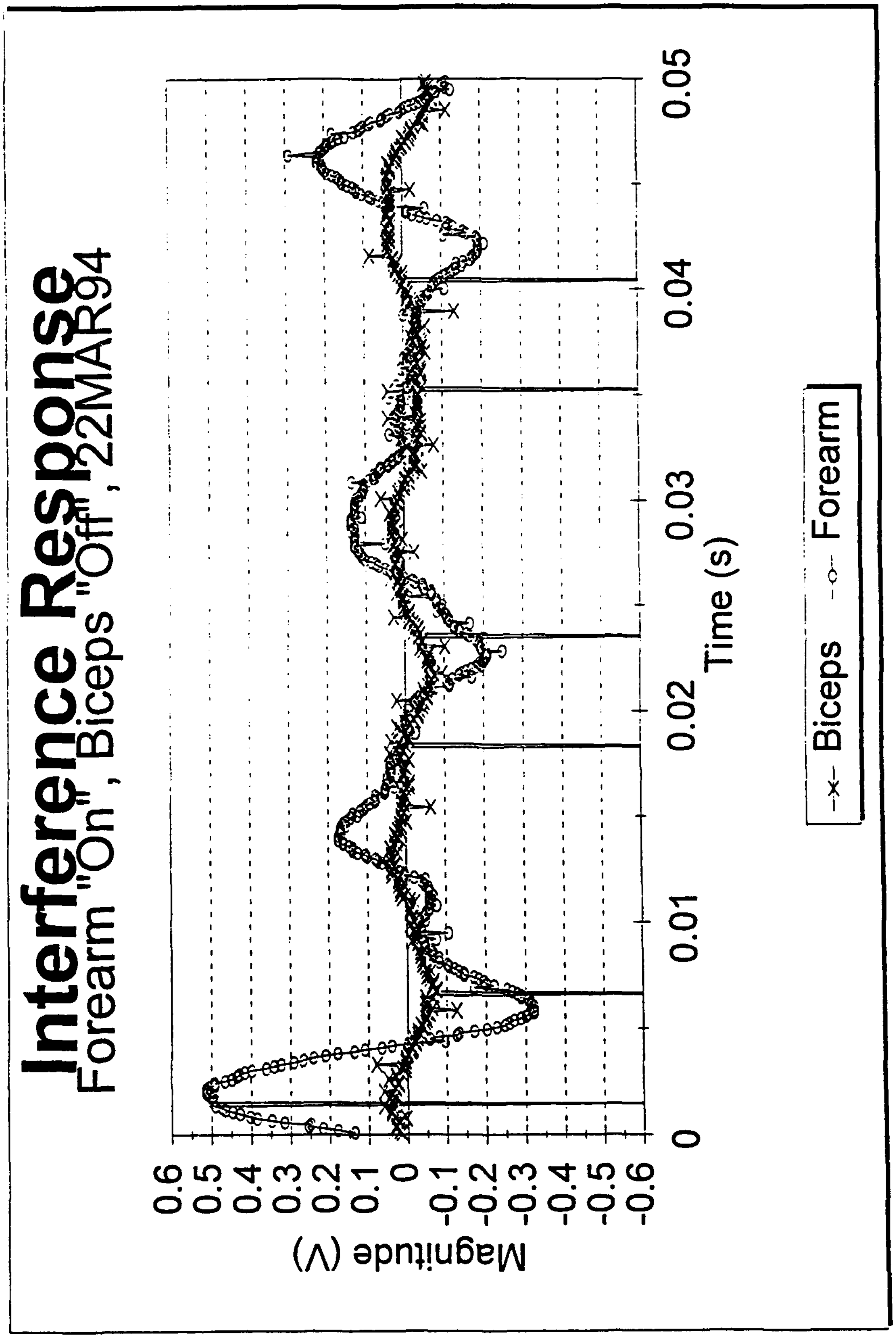
Appendix (K-3)


Interference Response
Biceps and Forearm Noise, 22MAR94
Magnitude (V)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Biceps
Forearm

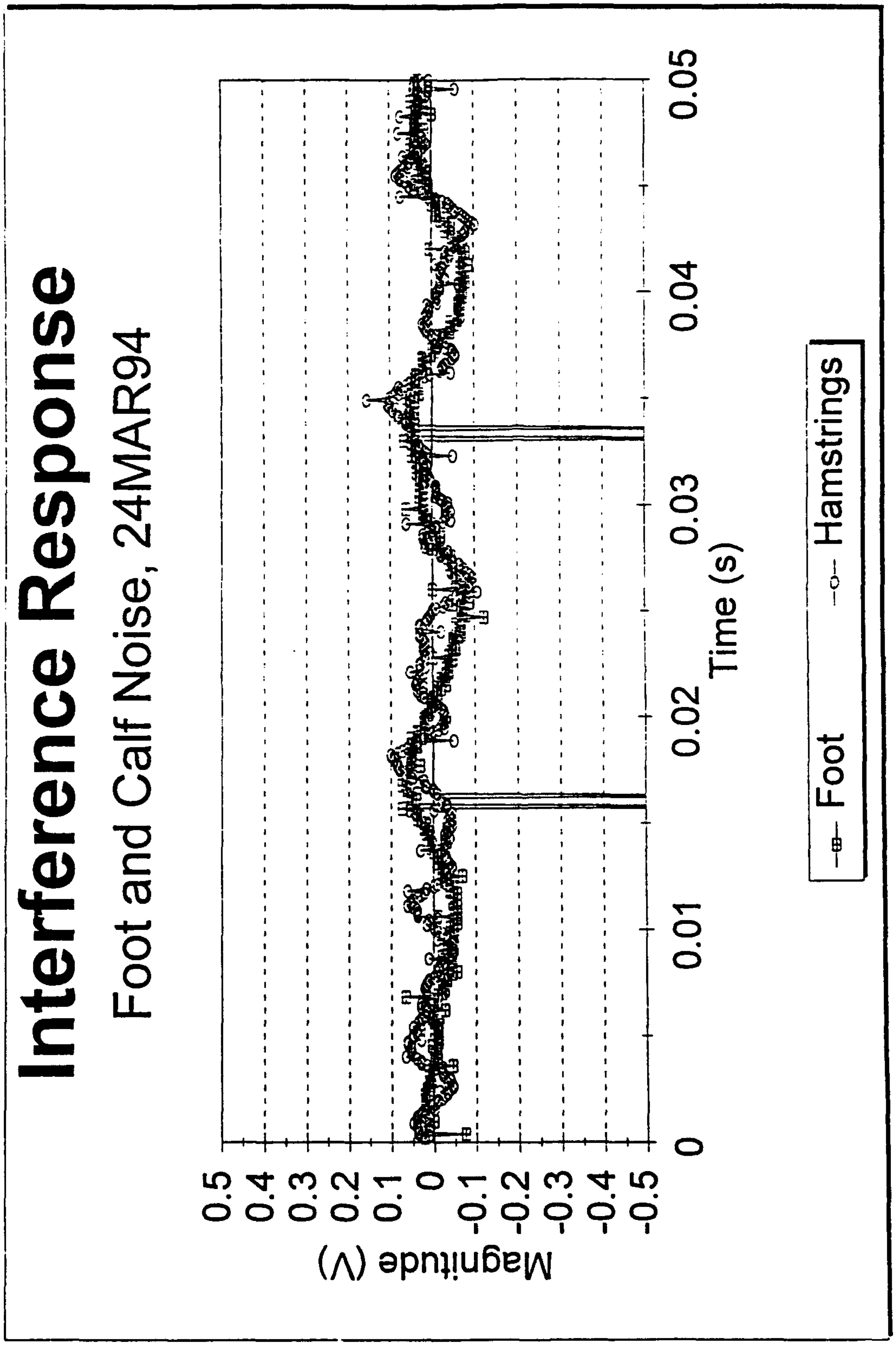
Appendix (L-1)


Interference Response
Biceps "On", Forearm "Off", 22MAR94
Magnitude (V)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Biceps
Forearm

Appendix (L-2)


Interference Response
Forearm "On", Biceps "Off", 22MAR94
Magnitude (V)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Biceps
Forearm

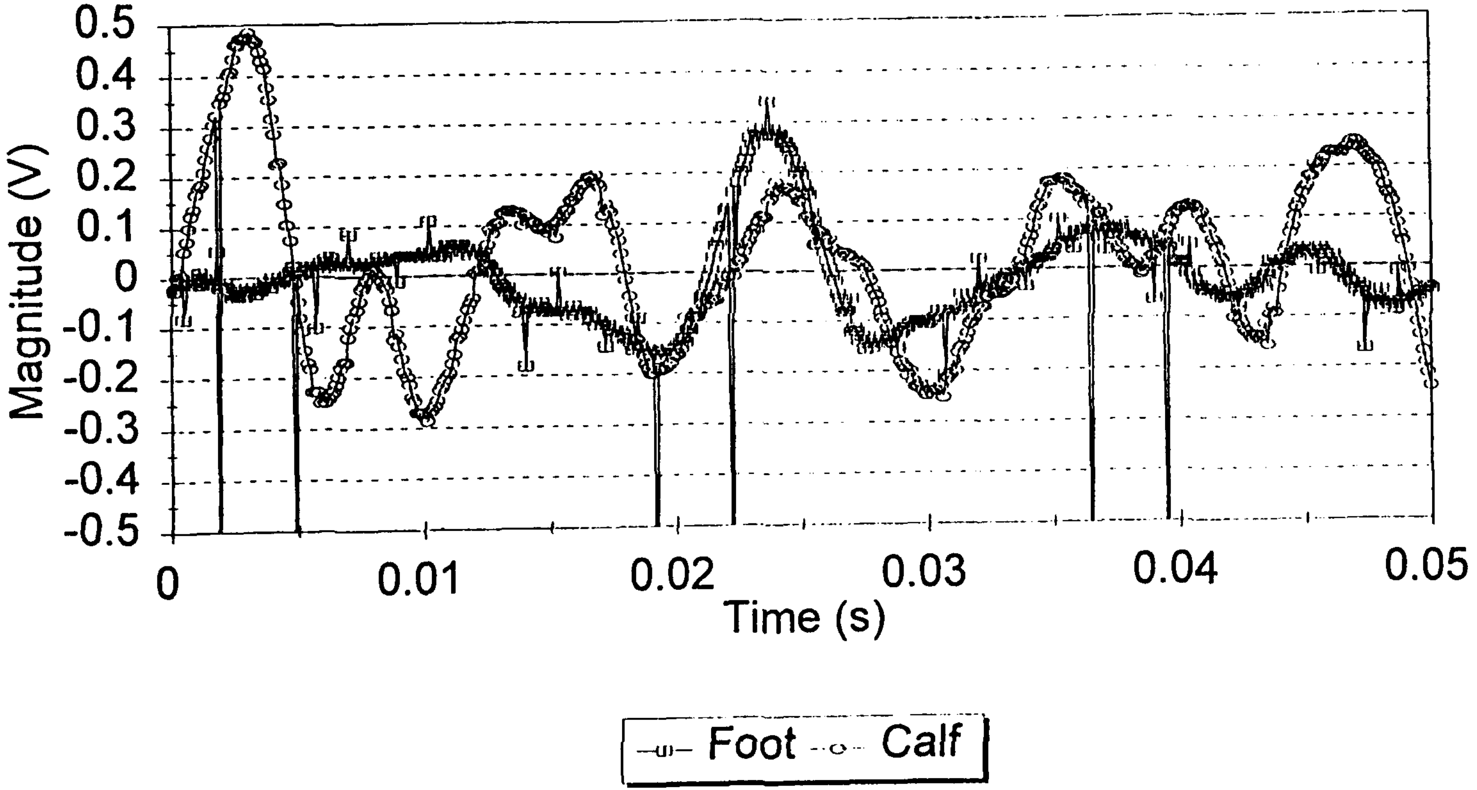
Appendix (L-3)


Interference Response
Foot and Calf Noise, 24MAR94
Magnitude (V)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Hamstrings

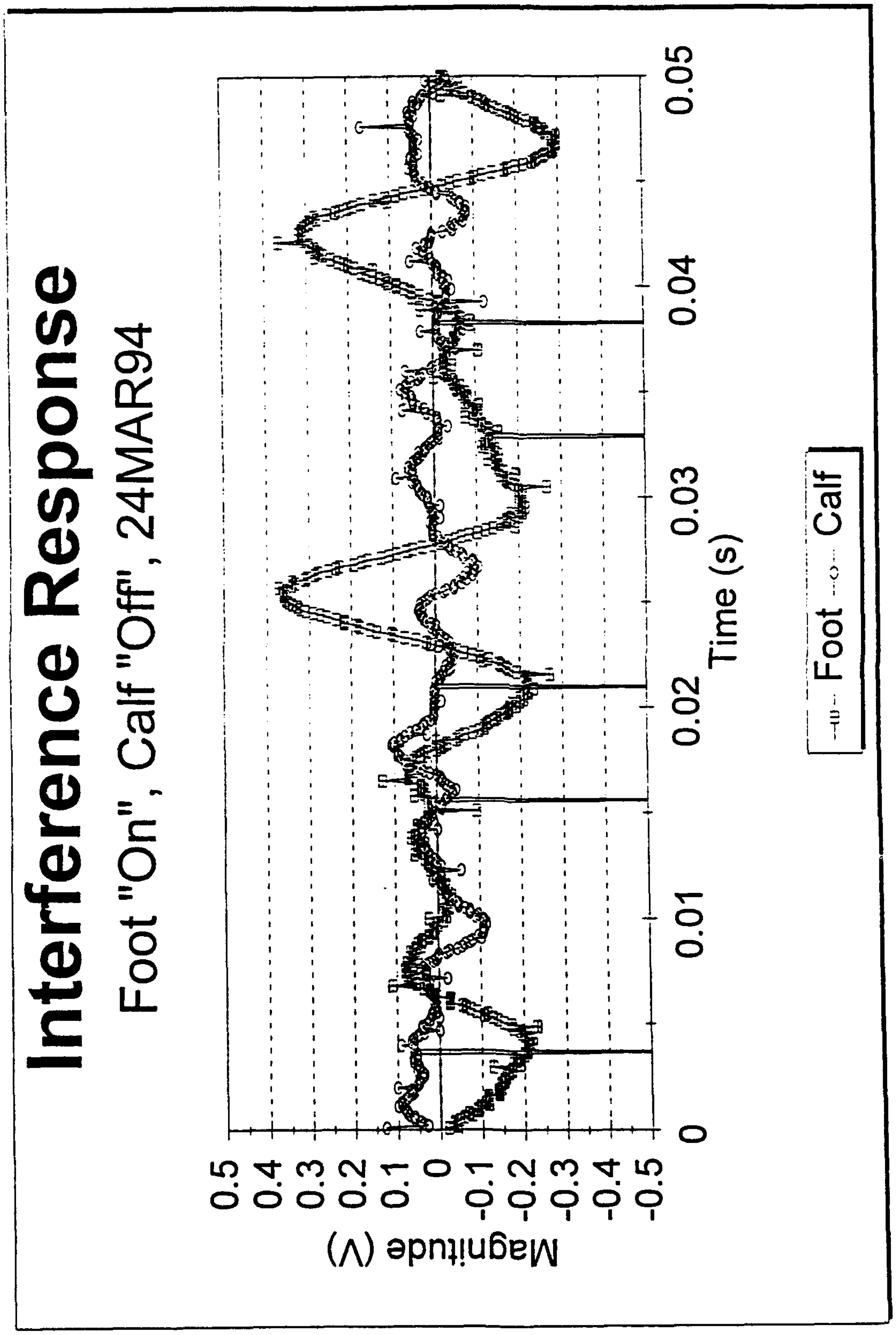
Appendix (M-1)

Interference Response

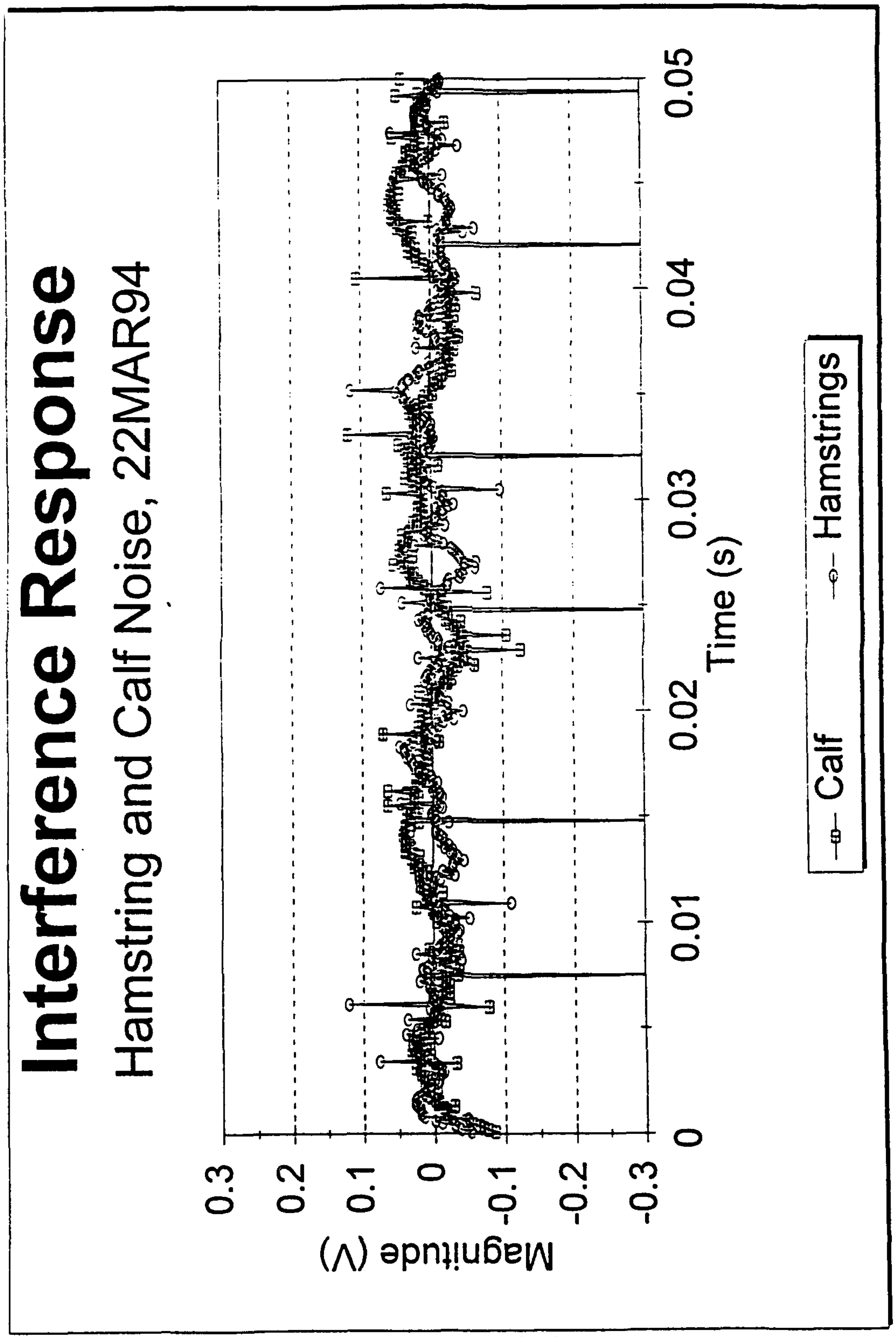
Calf "On", Foot "Off", 24MAR94



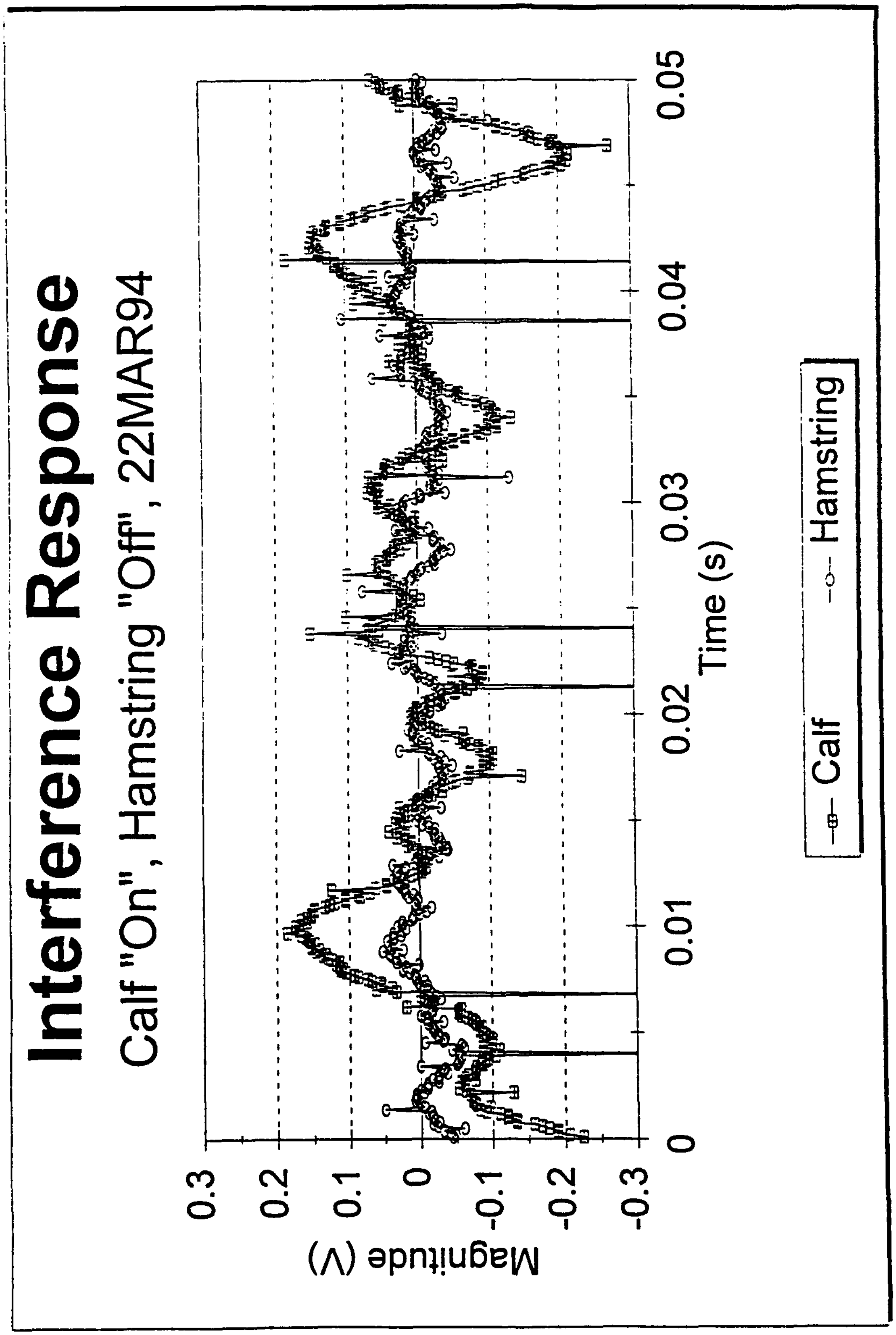
Appendix (M-2)


Interference Response
Foot "On", Calf "Off", 24MAR94
Magnitude (V)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Calf

Appendix (M-3)


Interference Response
Hamstring and Calf Noise, 22MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Calf
Hamstrings

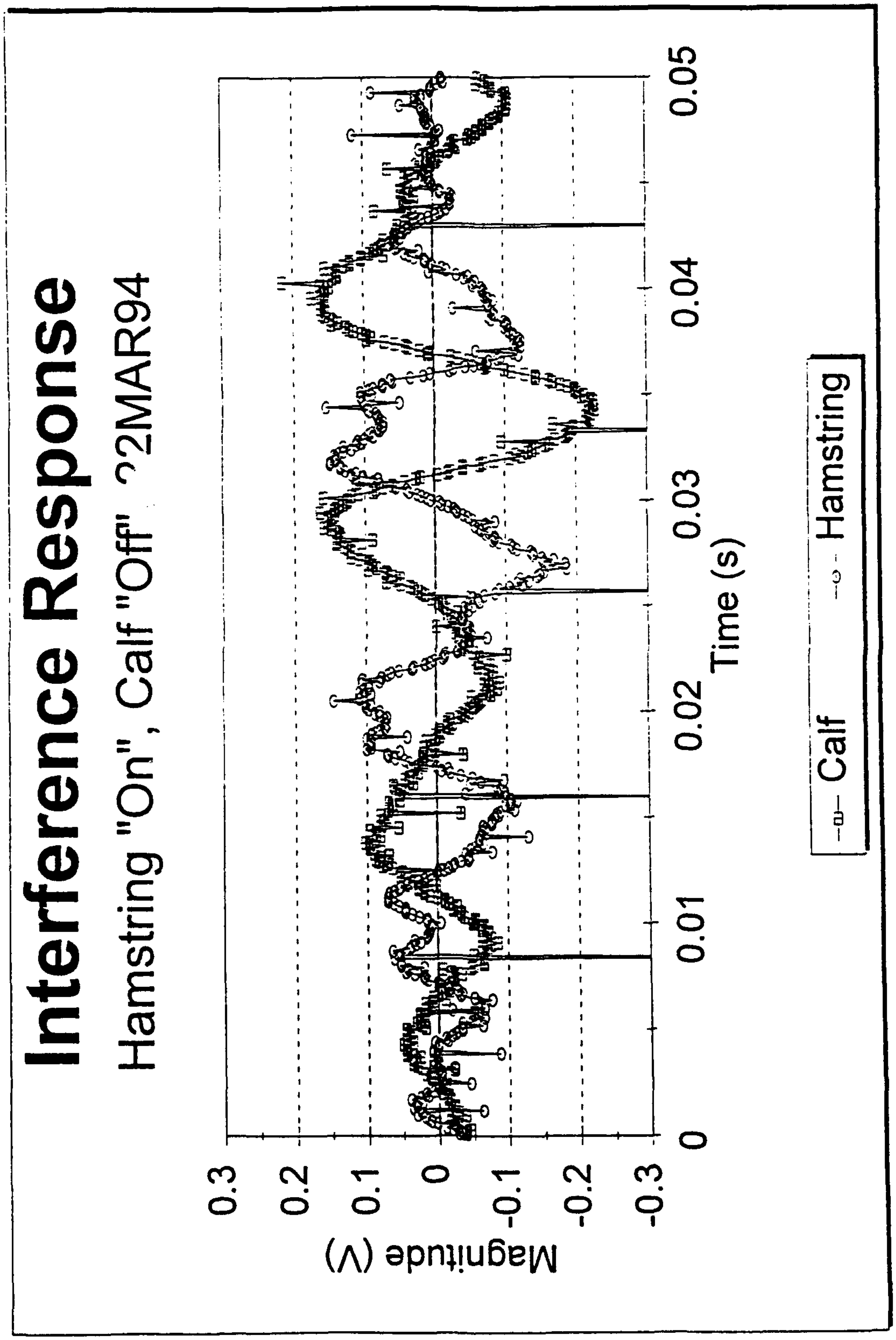
Appendix (N-1)



Interference Response

Calf "On", Hamstring "Off", 22MAR94

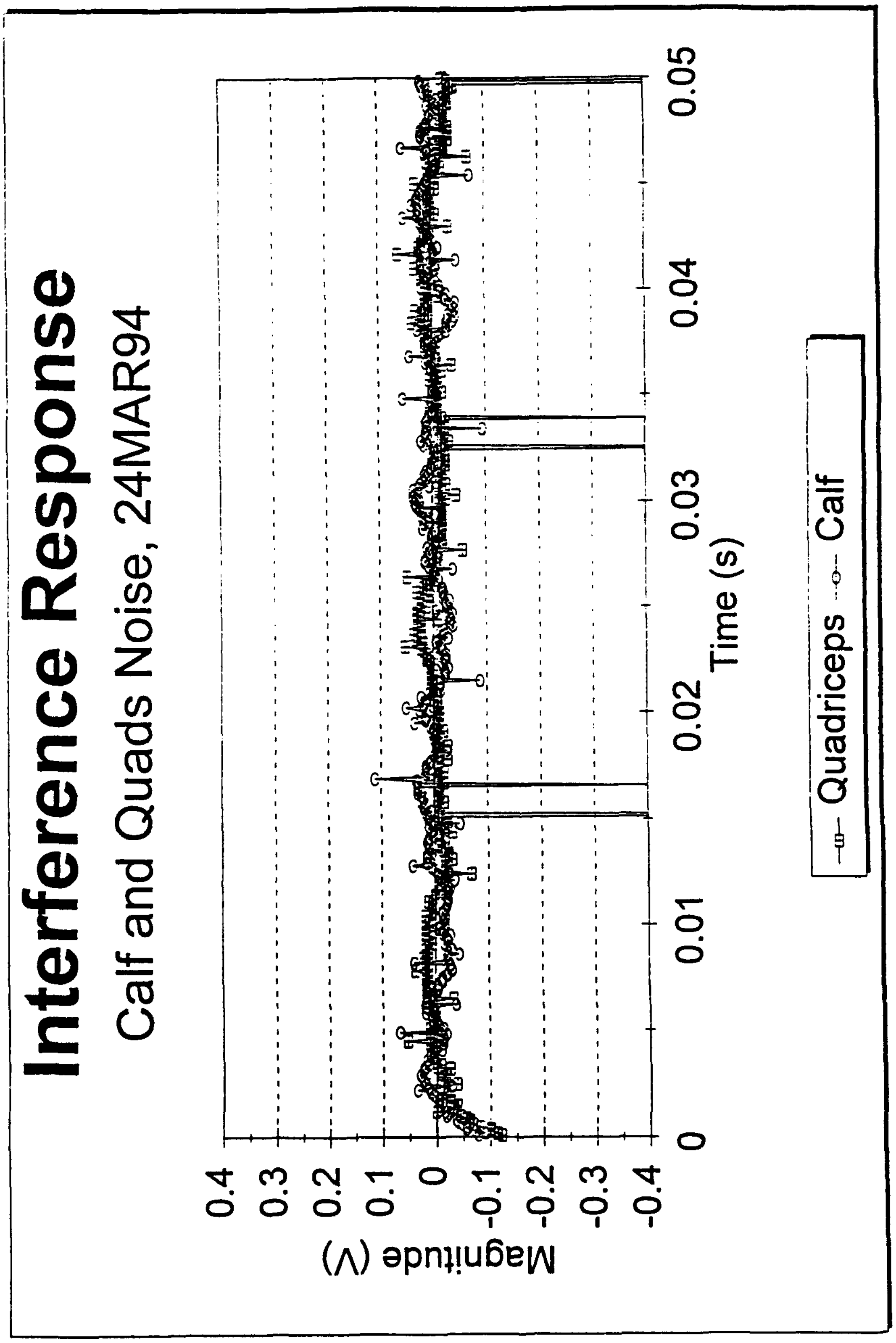
Appendix (N-2)



Interference Response

Hamstring "On", Calf "Off", 22MAR94

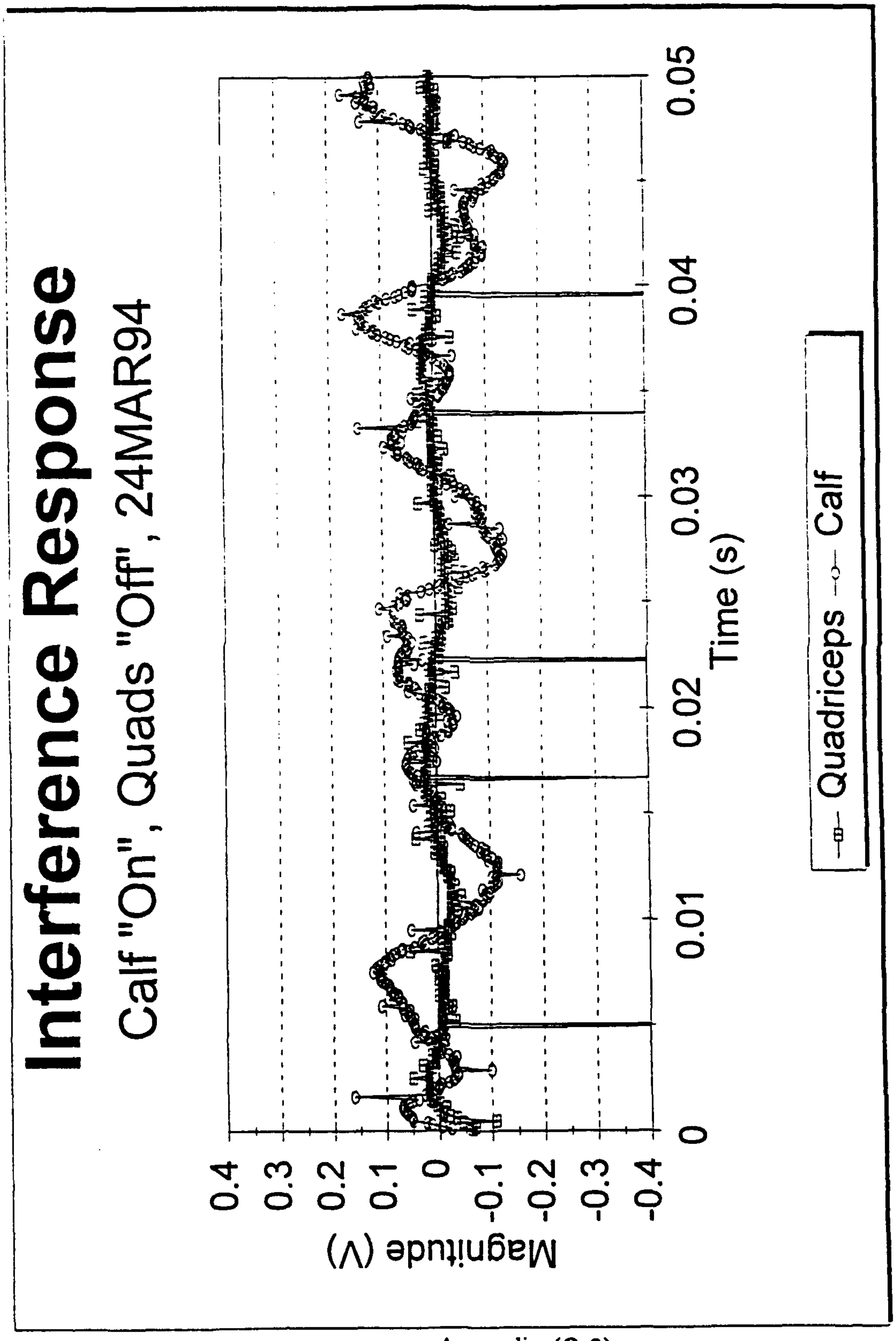
Appendix (N-3)



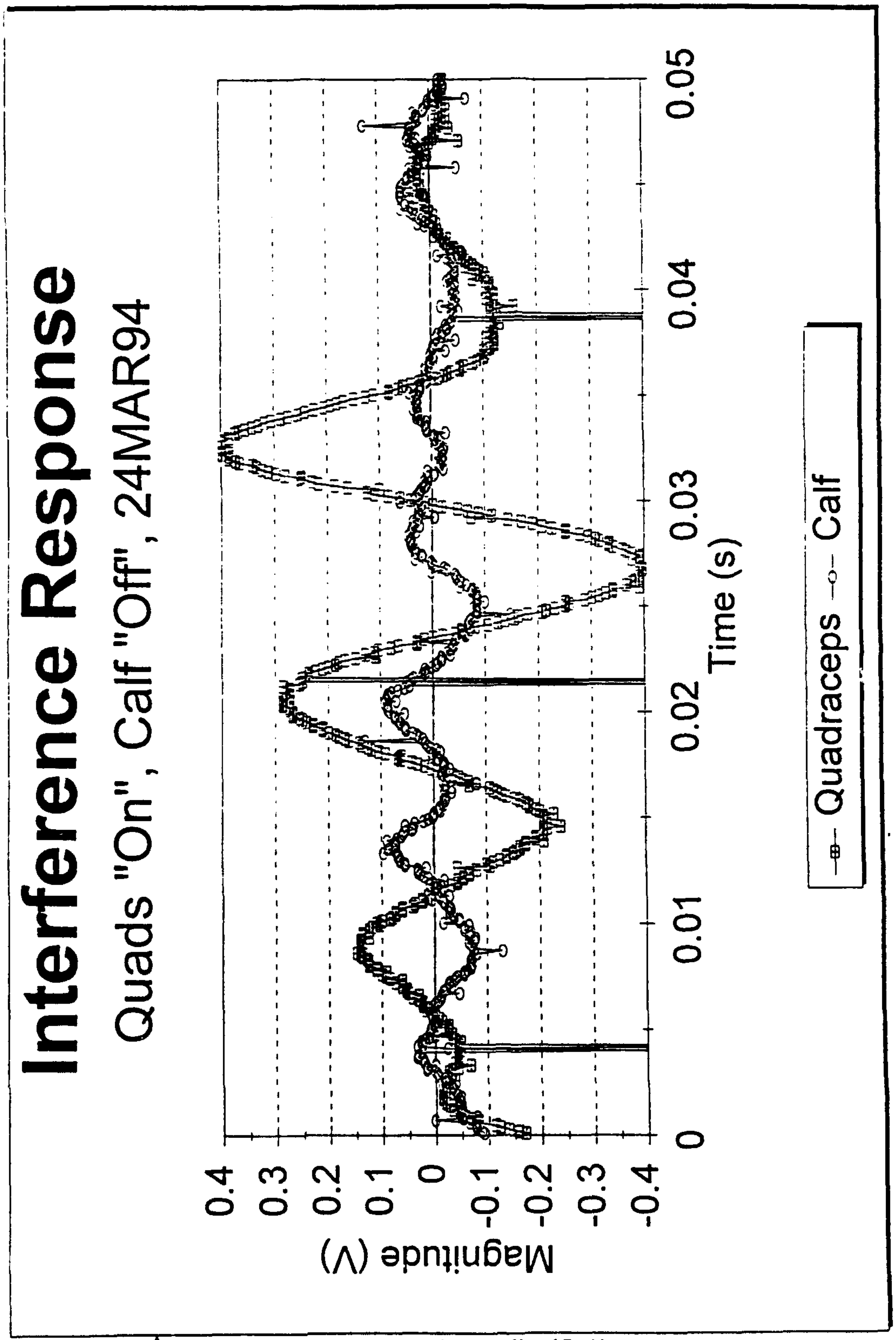
Interference Response

Calf and Quads Noise, 24MAR94

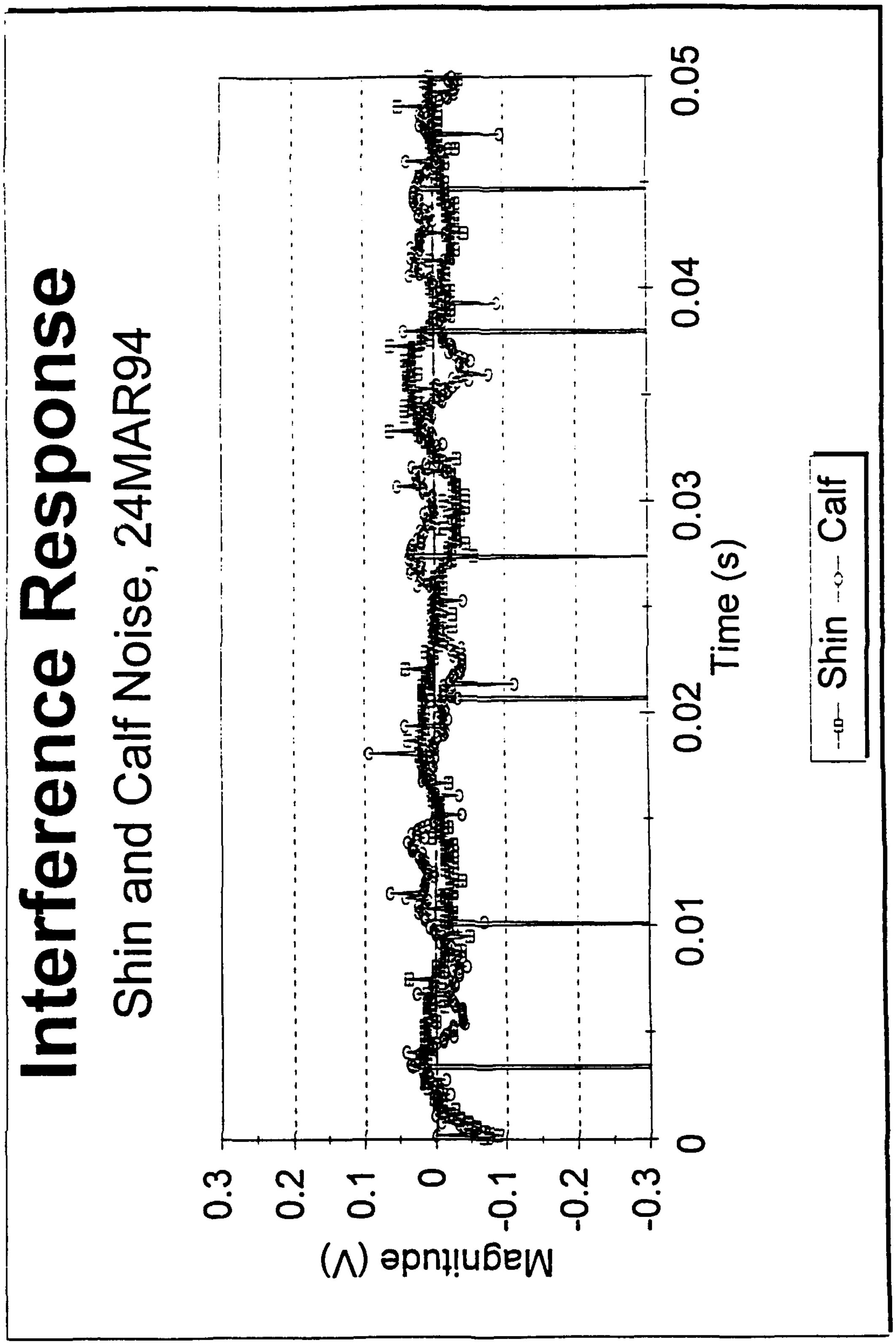
Appendix (O-1)


Interference Response
Calf "On", Quads "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Quadriceps
Calf

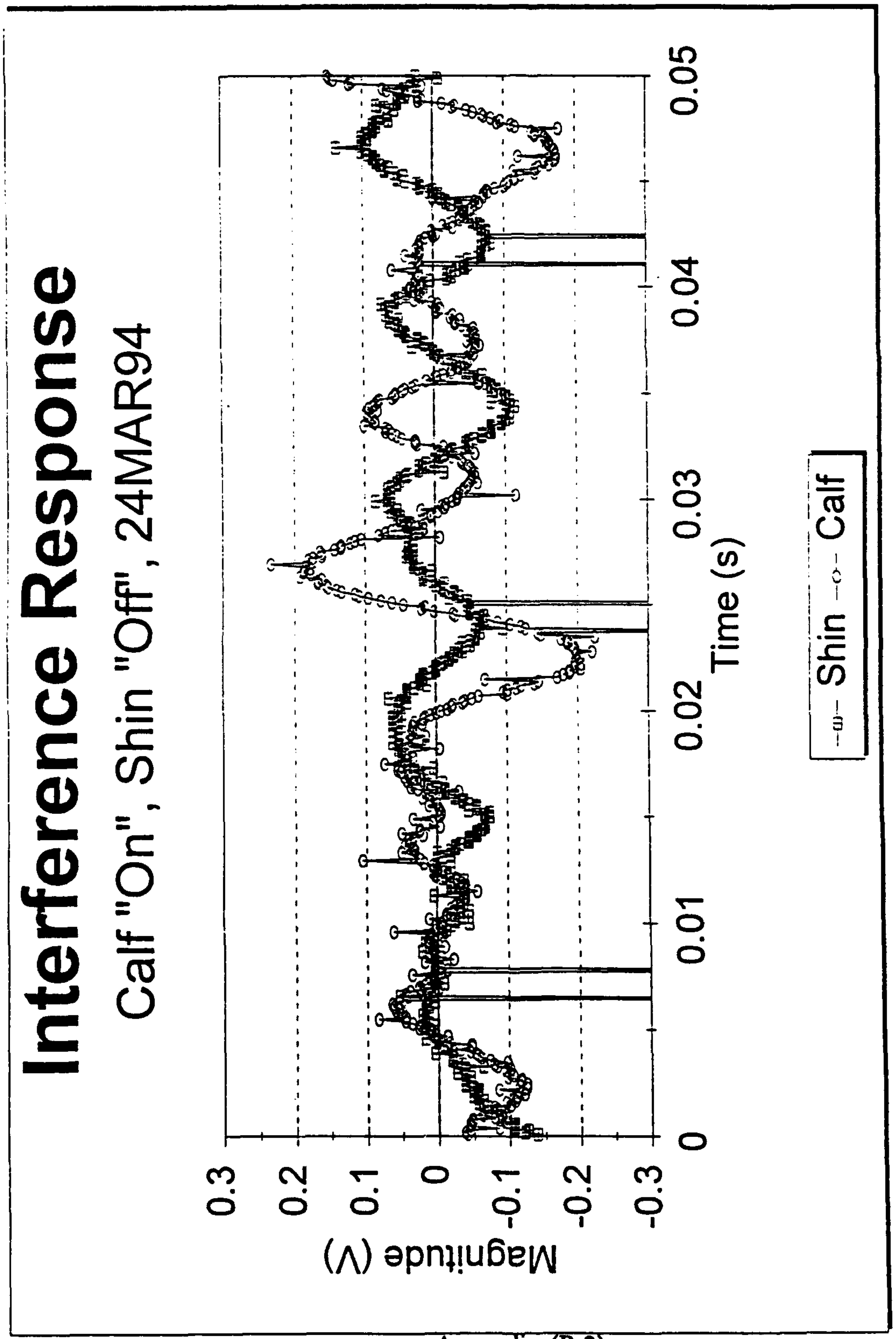
Appendix (O-2)


Interference Response
Quads "On", Calf "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Quadraceps
Calf

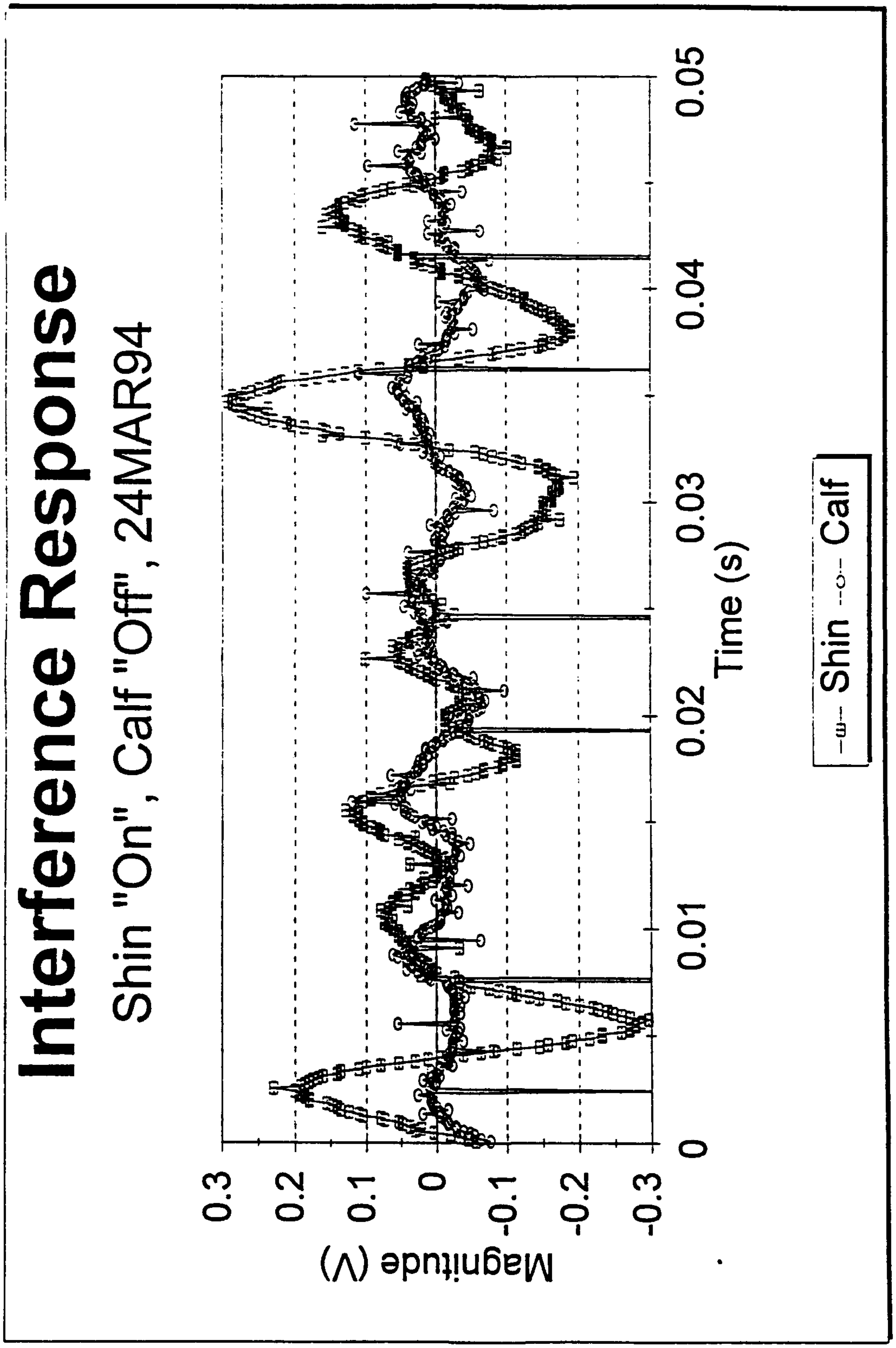
Appendix (O-3)


Interference Response
Shin and Calf Noise, 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Shin
Calf

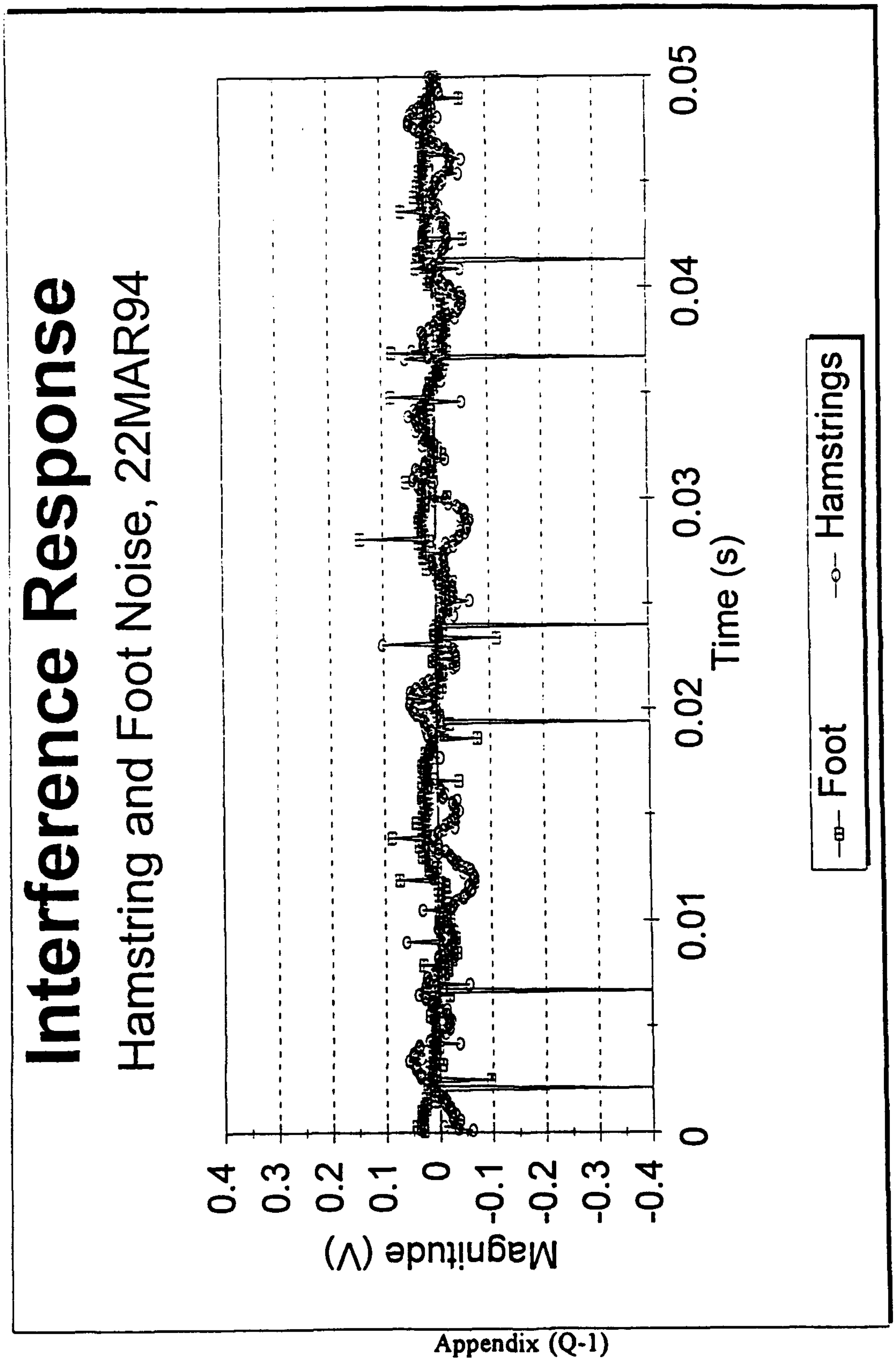
Appendix (P-1)


Interference Response
Calf "On", Shin "Off", 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Calf

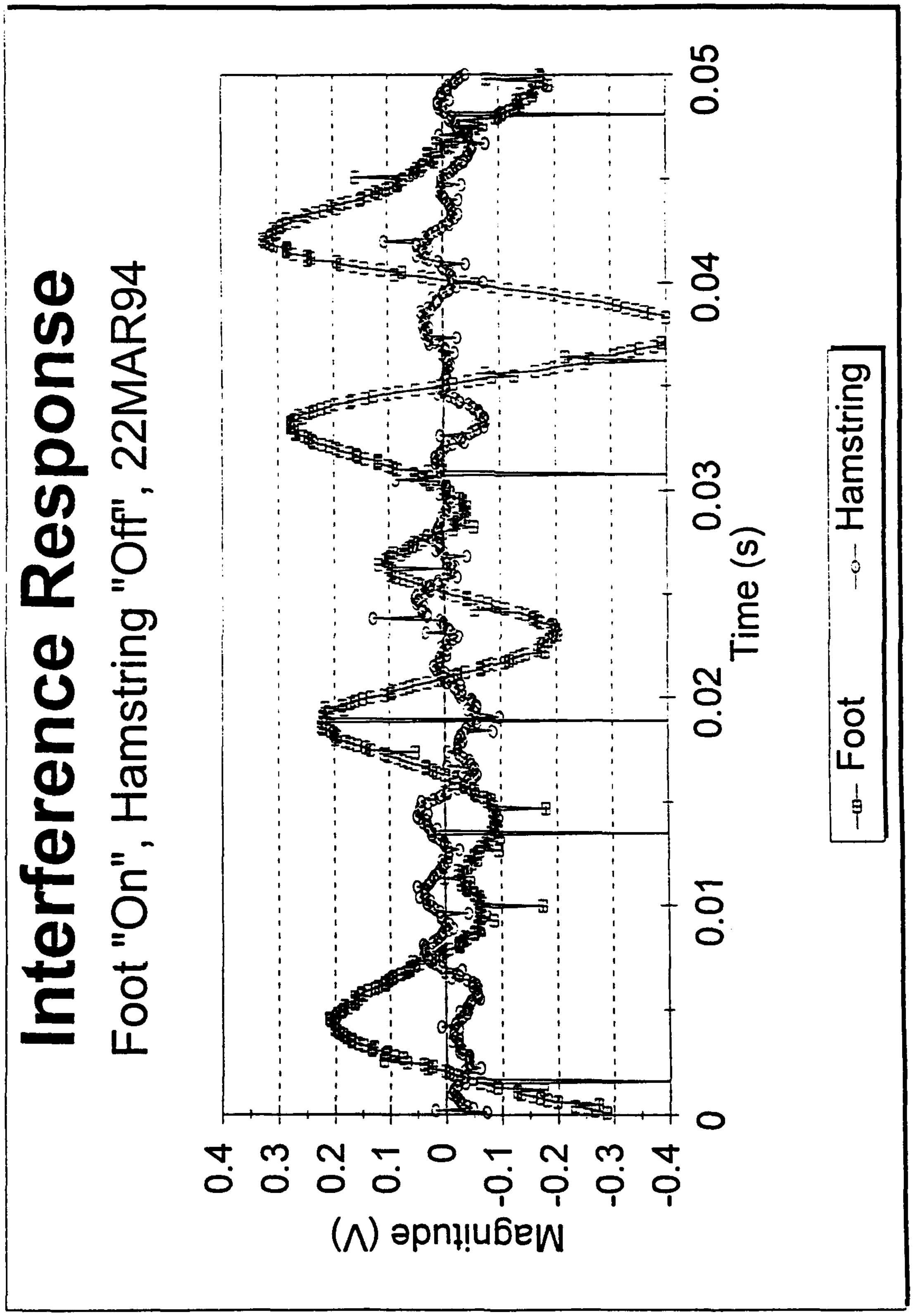
Appendix (P-2)


Interference Response
Shin "On", Calf "Off", 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Shin
Calf

Appendix (P-3)


Interference Response
Hamstring and Foot Noise, 22MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Hamstrings

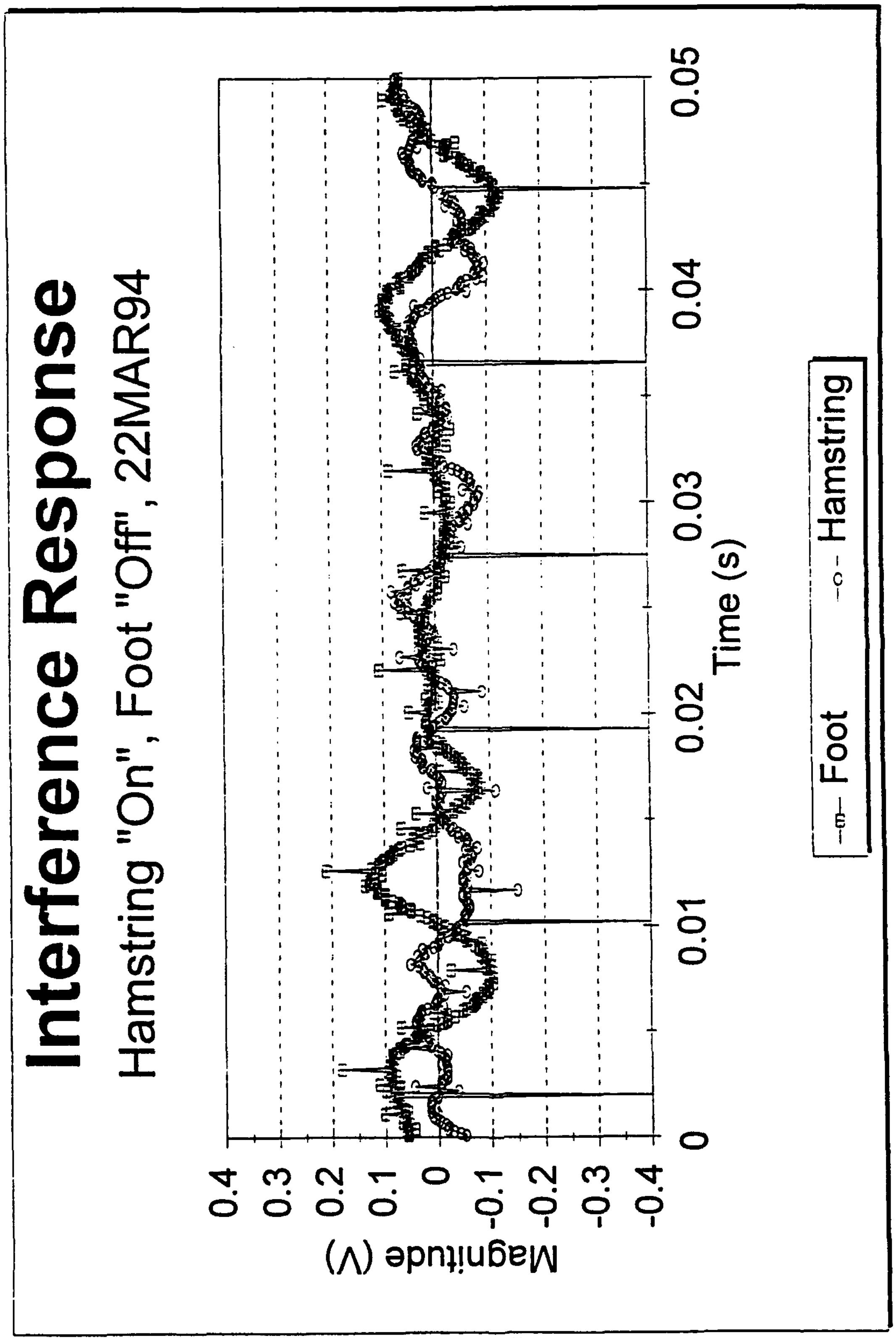
Appendix (Q-1)



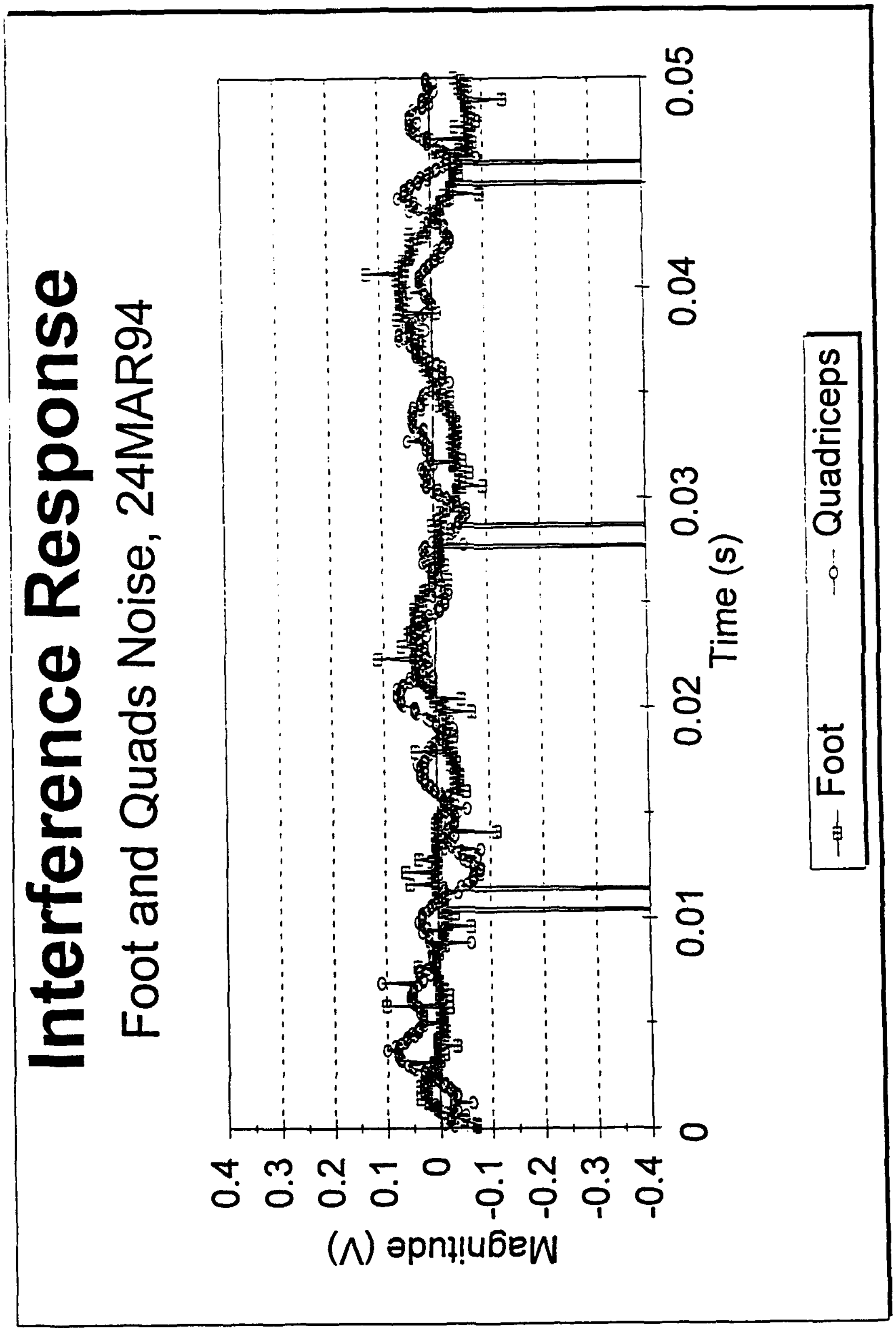
Interference Response

Foot "On", Hamstring "Off", 22MAR94

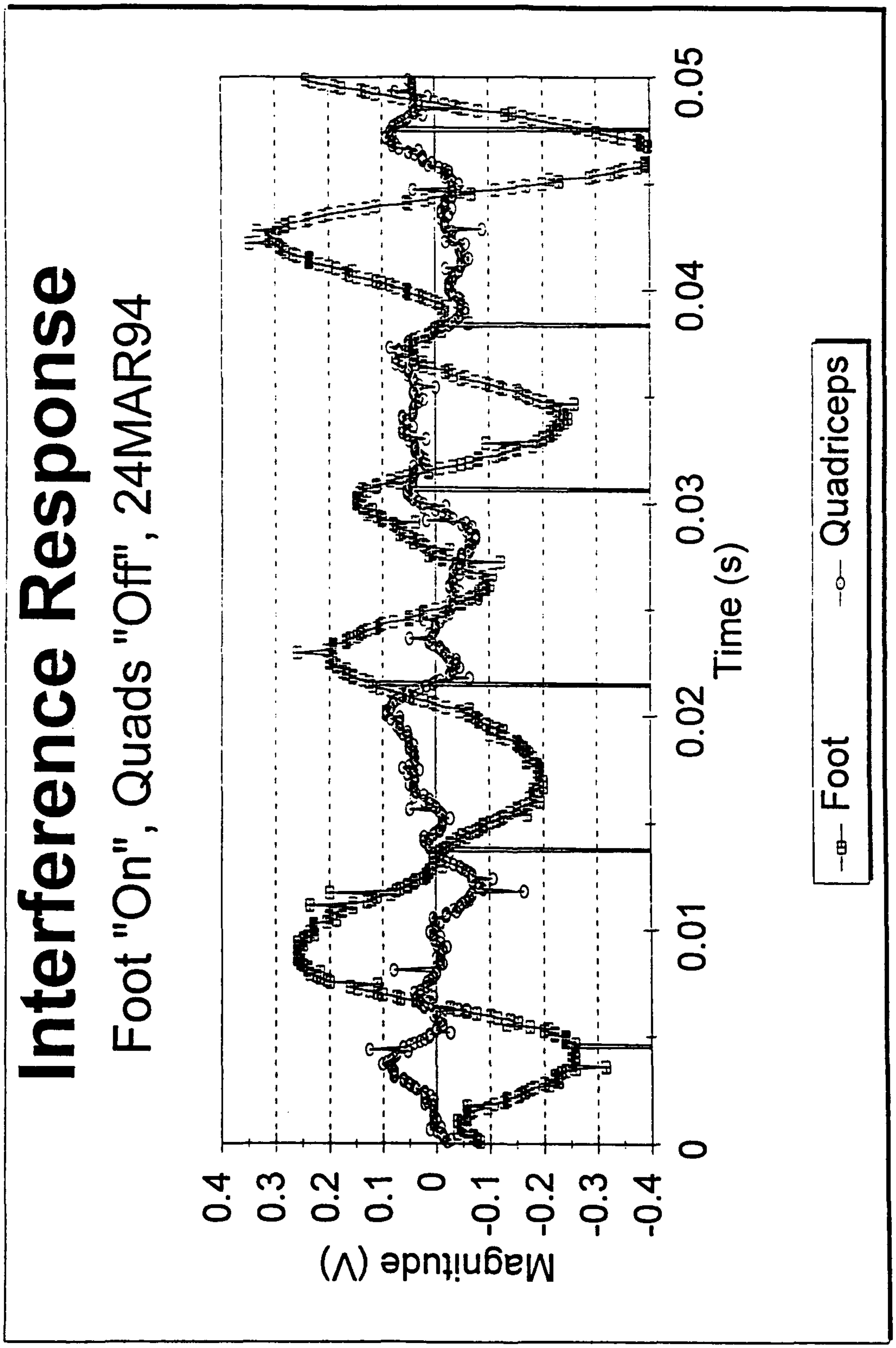
Appendix (Q-2)


Interference Response
Hamstring "On", Foot "Off", 22MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Hamstring

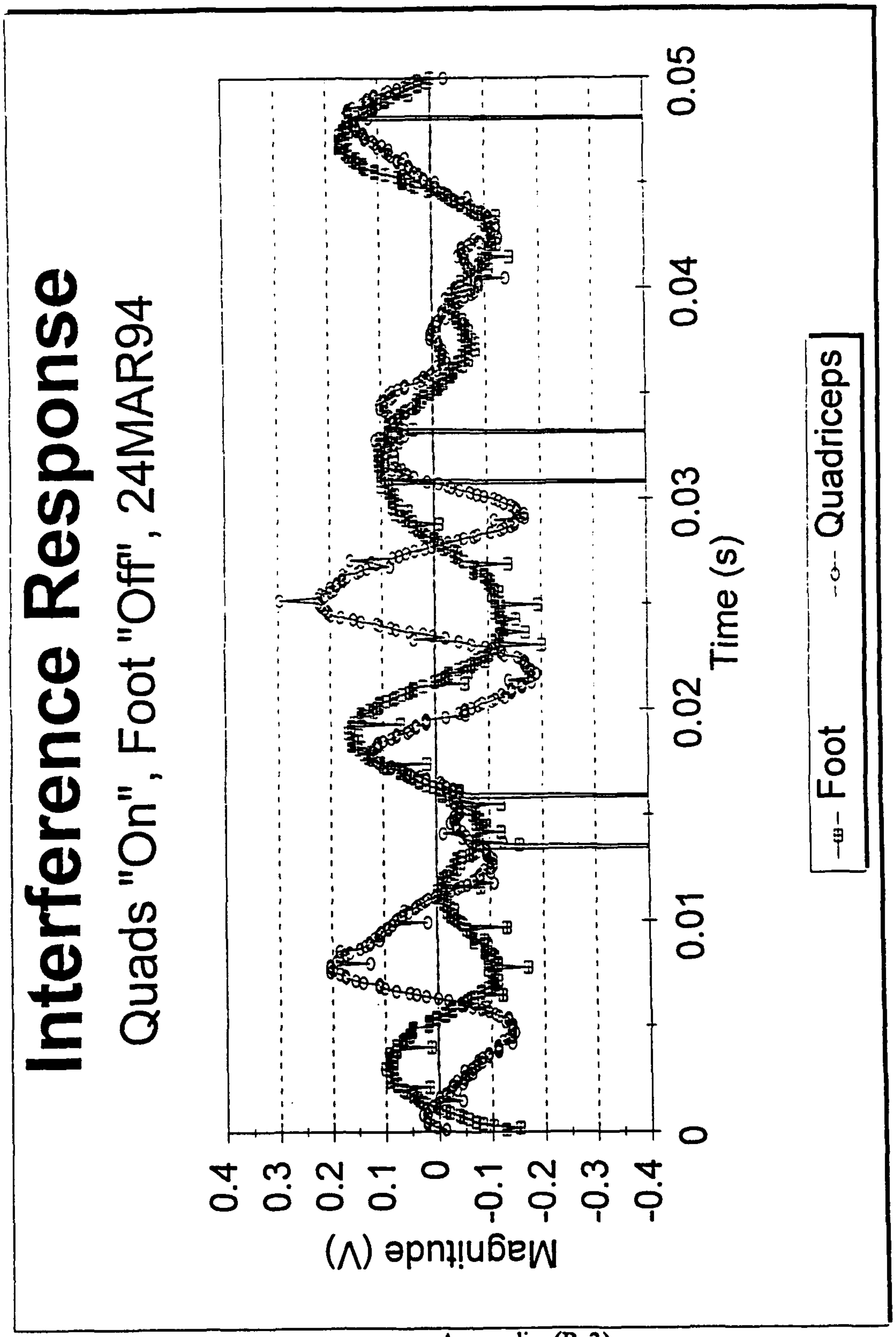
Appendix (Q-3)


Interference Response
Foot and Quads Noise, 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Foot
Quadriceps

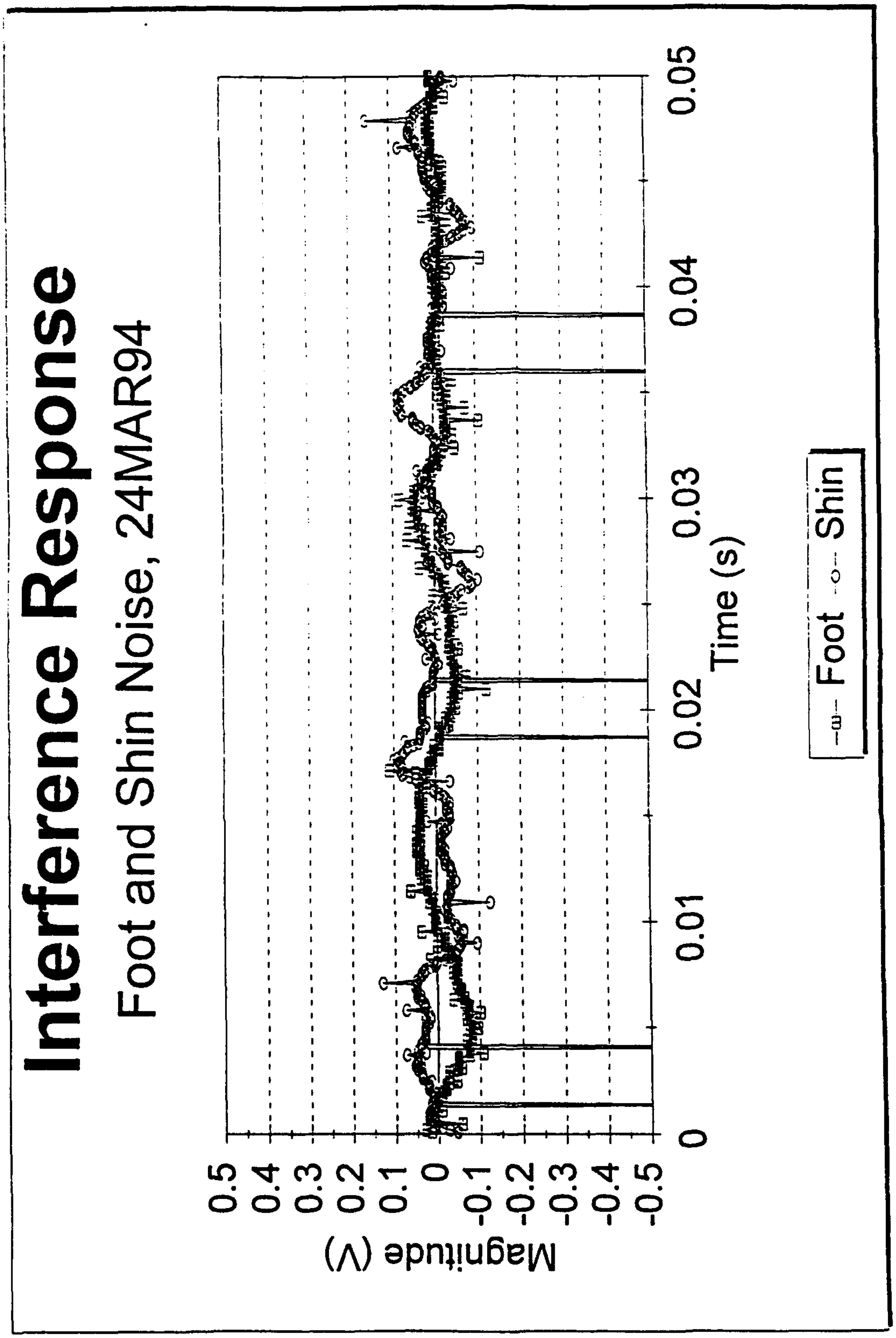
Appendix (R-1)


Interference Response
Foot "On", Quads "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Quadriceps

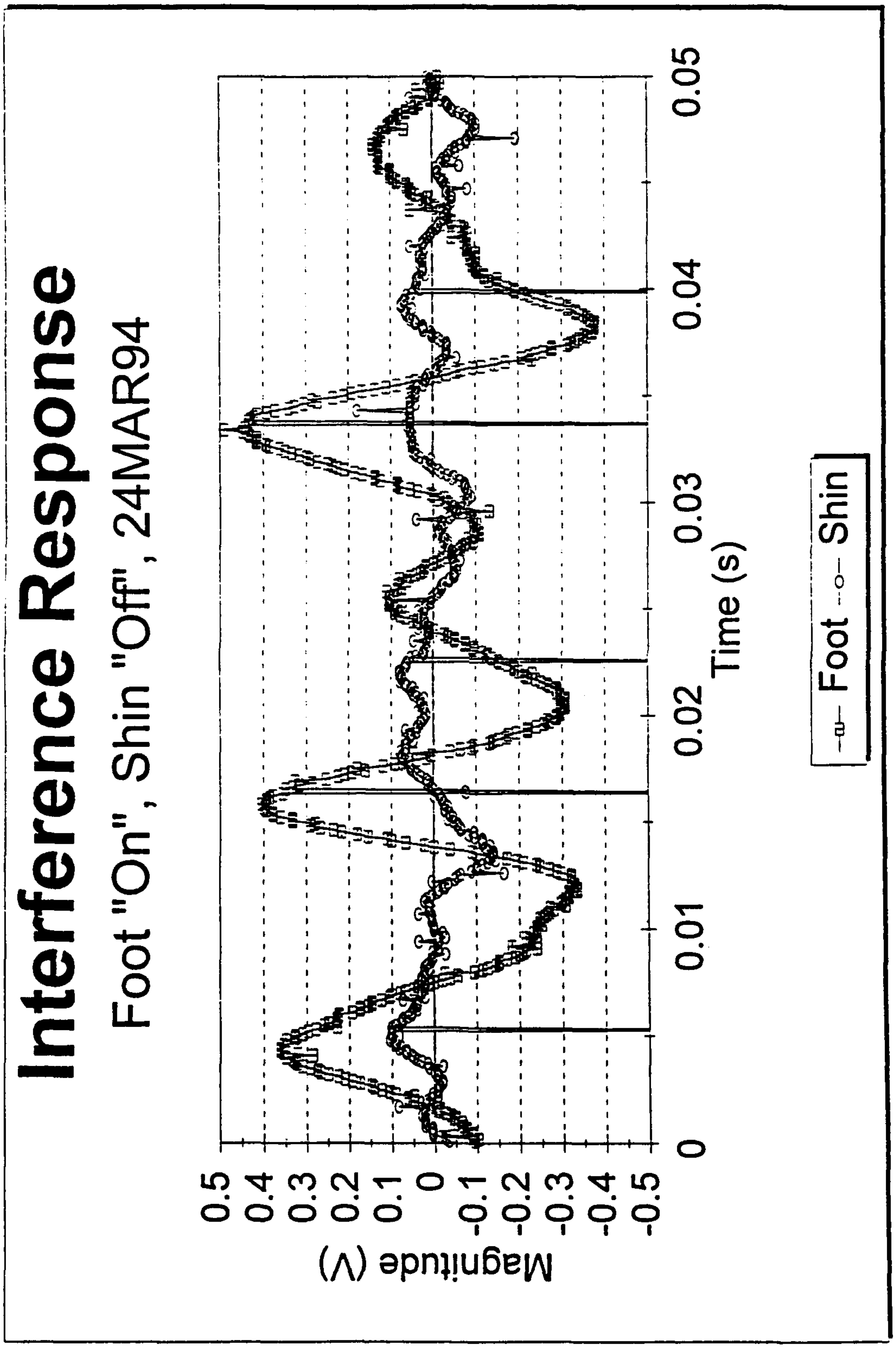
Appendix (R-2)


Interference Response
Quads "On", Foot "Off", 24MAR94
Magnitude (V)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Quadriceps

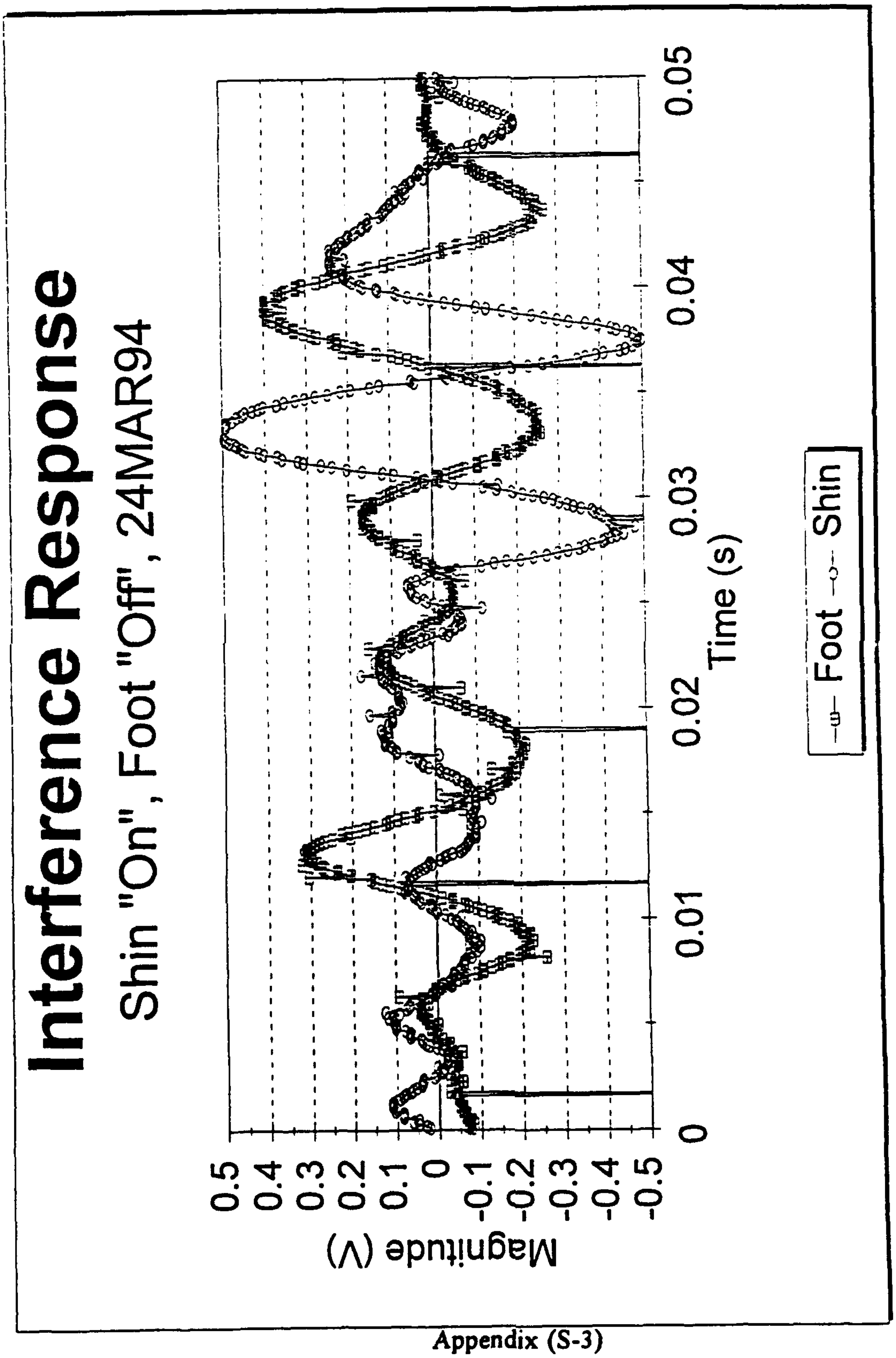
Appendix (R-3)


Interference Response
Foot and Shin Noise, 24MAR94
Magnitude (V)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Foot
Shin

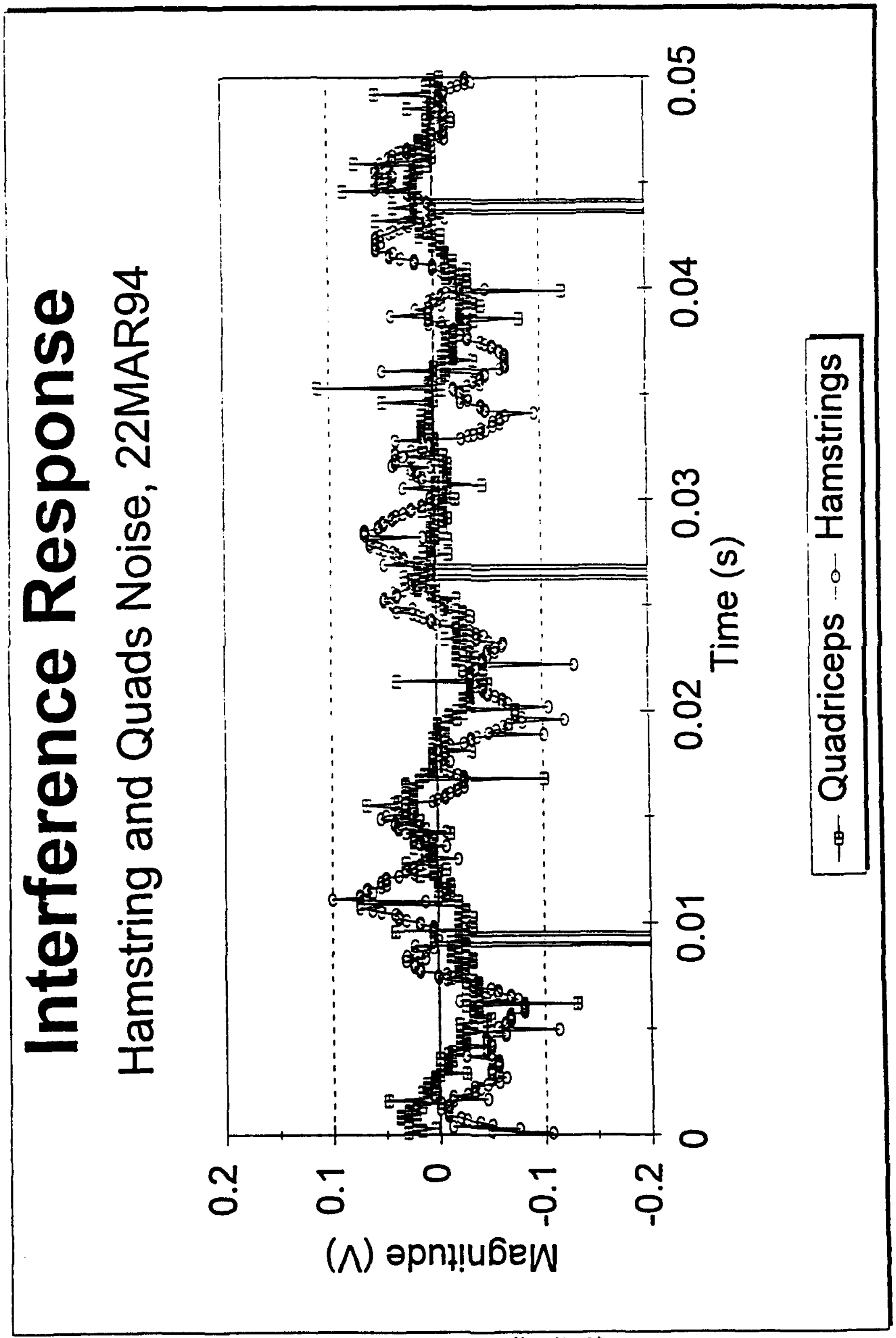
Appendix (S-1)


Interference Response
Foot "On", Shin "Off", 24MAR94
Magnitude (V)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Shin

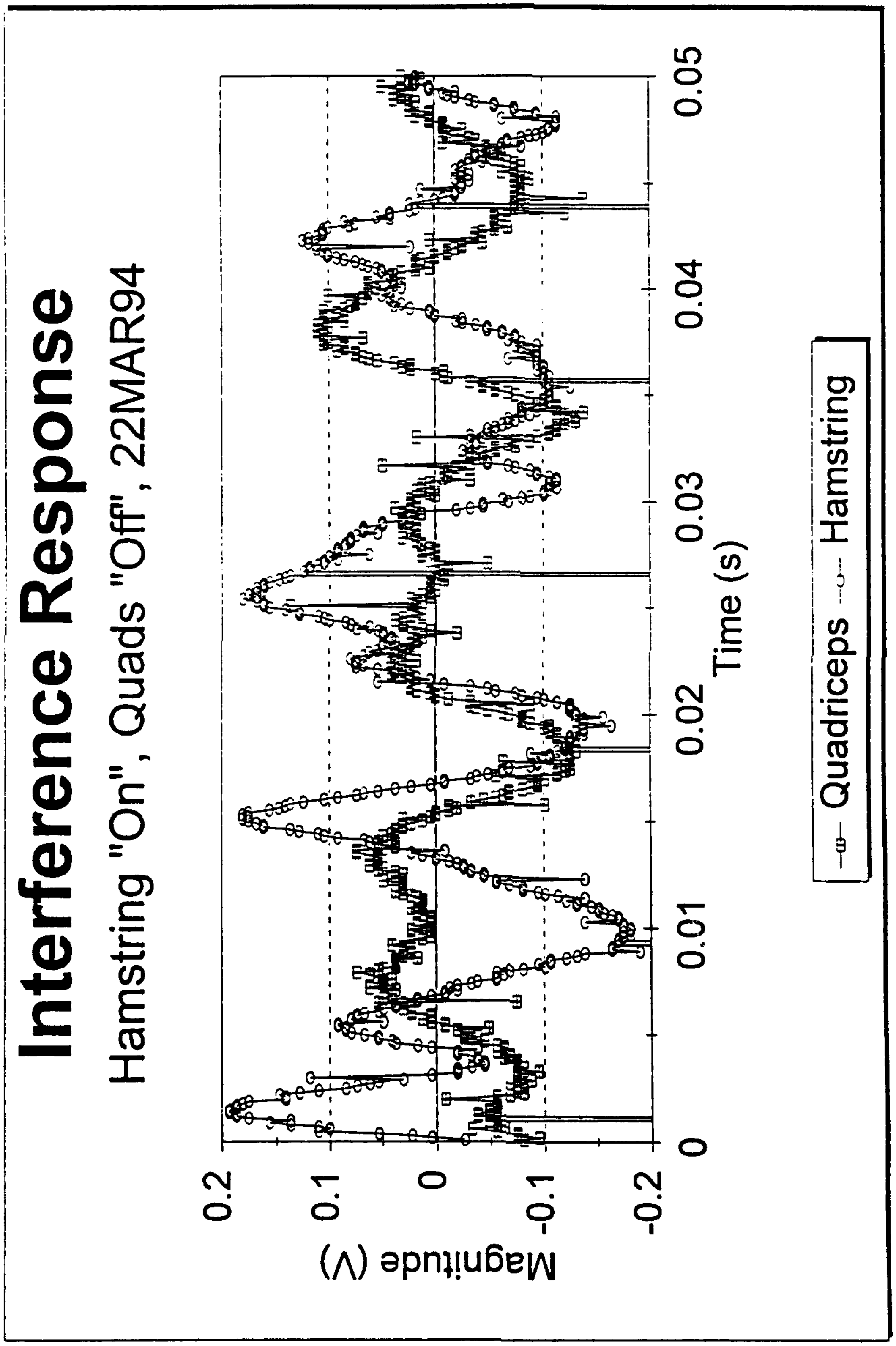
Appendix (S-2)


Interference Response
Shin "On", Foot "Off", 24MAR94
Magnitude (V)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Foot
Shin

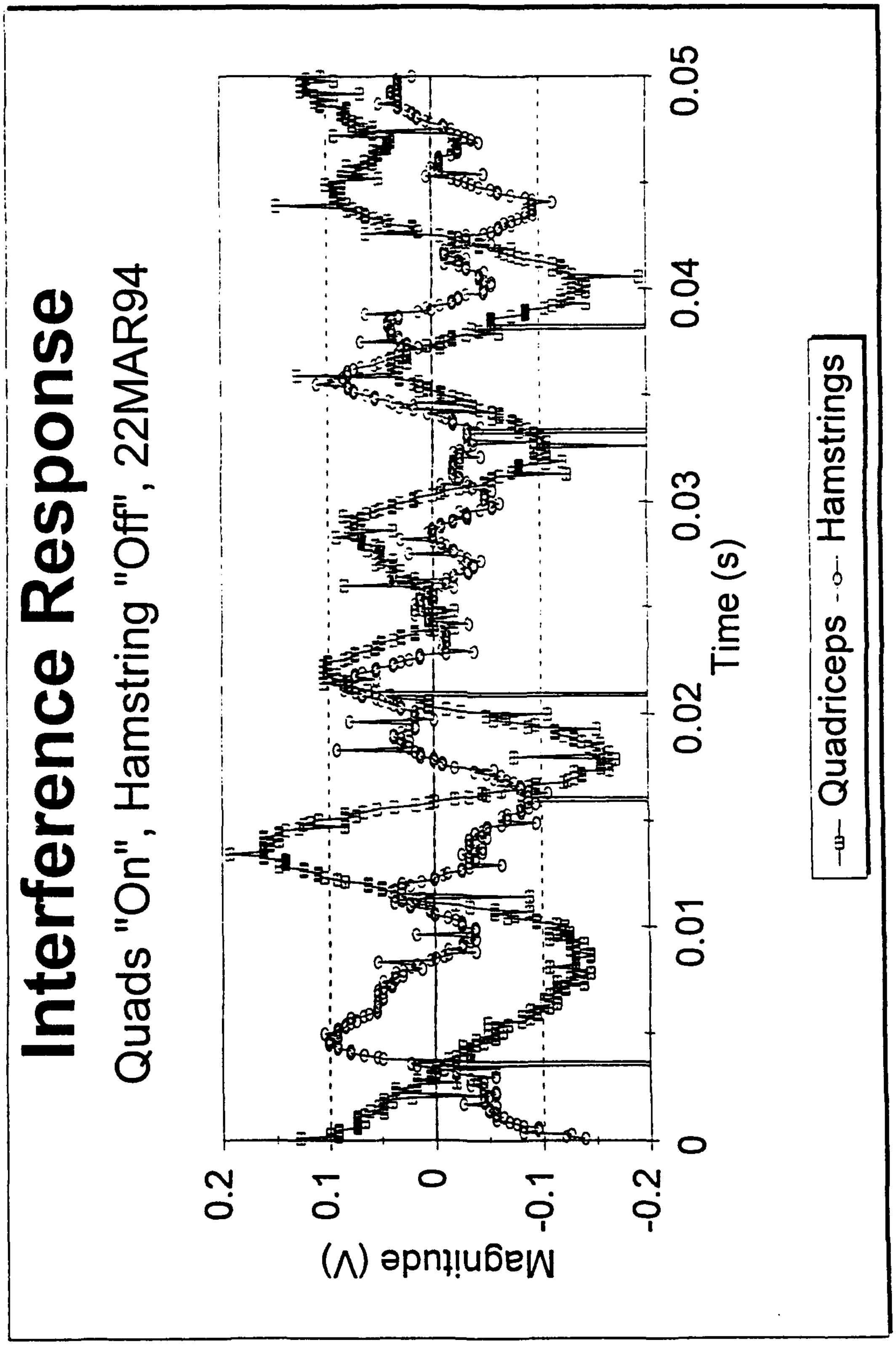
Appendix (S-3)


Interference Response
Hamstring and Quads Noise, 22MAR94
Magnitude (V)
0.2
0.1
0
-0.1
-0.2
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Quadriceps
Hamstrings

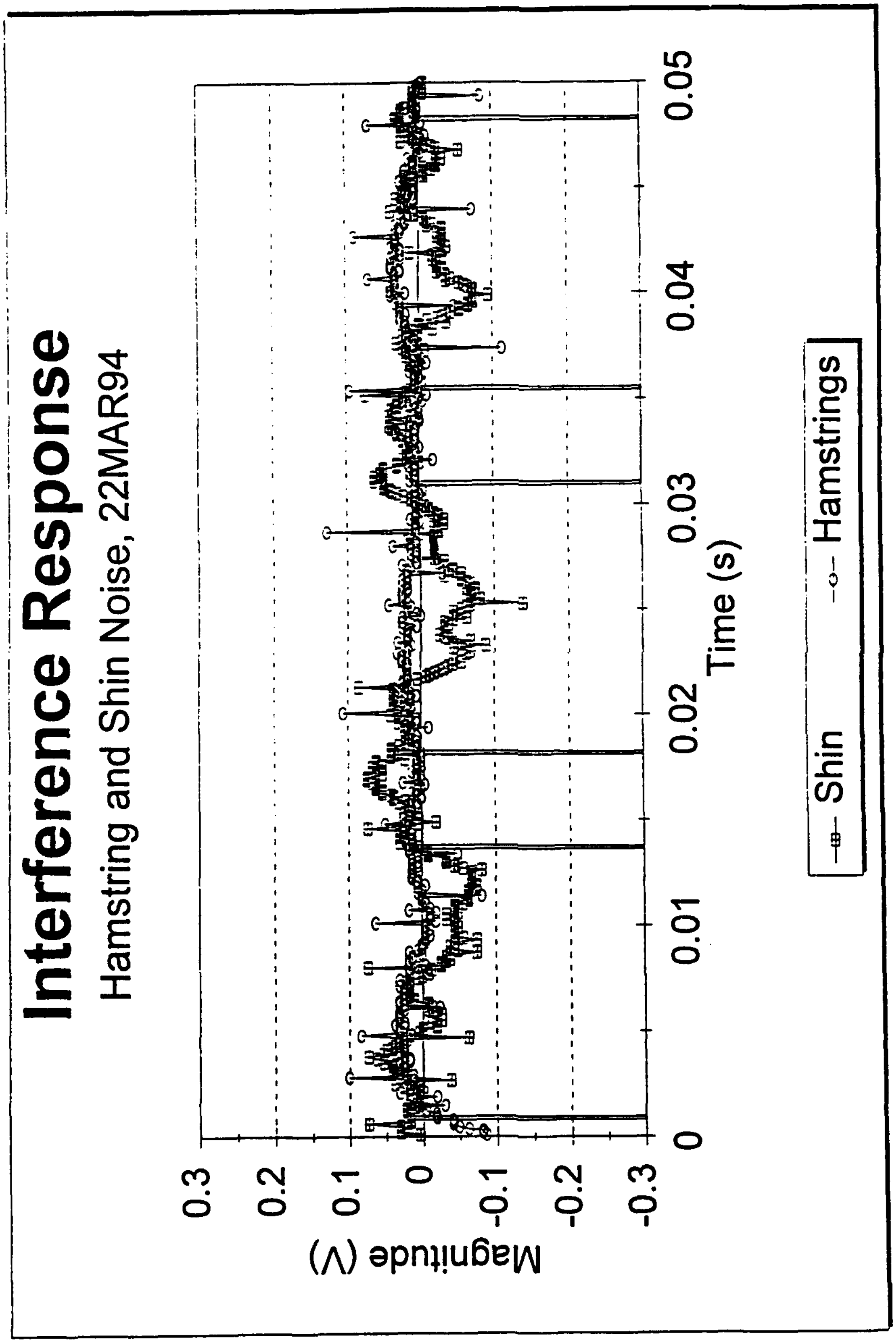
Appendix (T-1)


Interference Response
Hamstring "On", Quads "Off", 22MAR94
Magnitude (V)
0.2
0.1
0
-0.1
-0.2
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Quadriceps
Hamstring

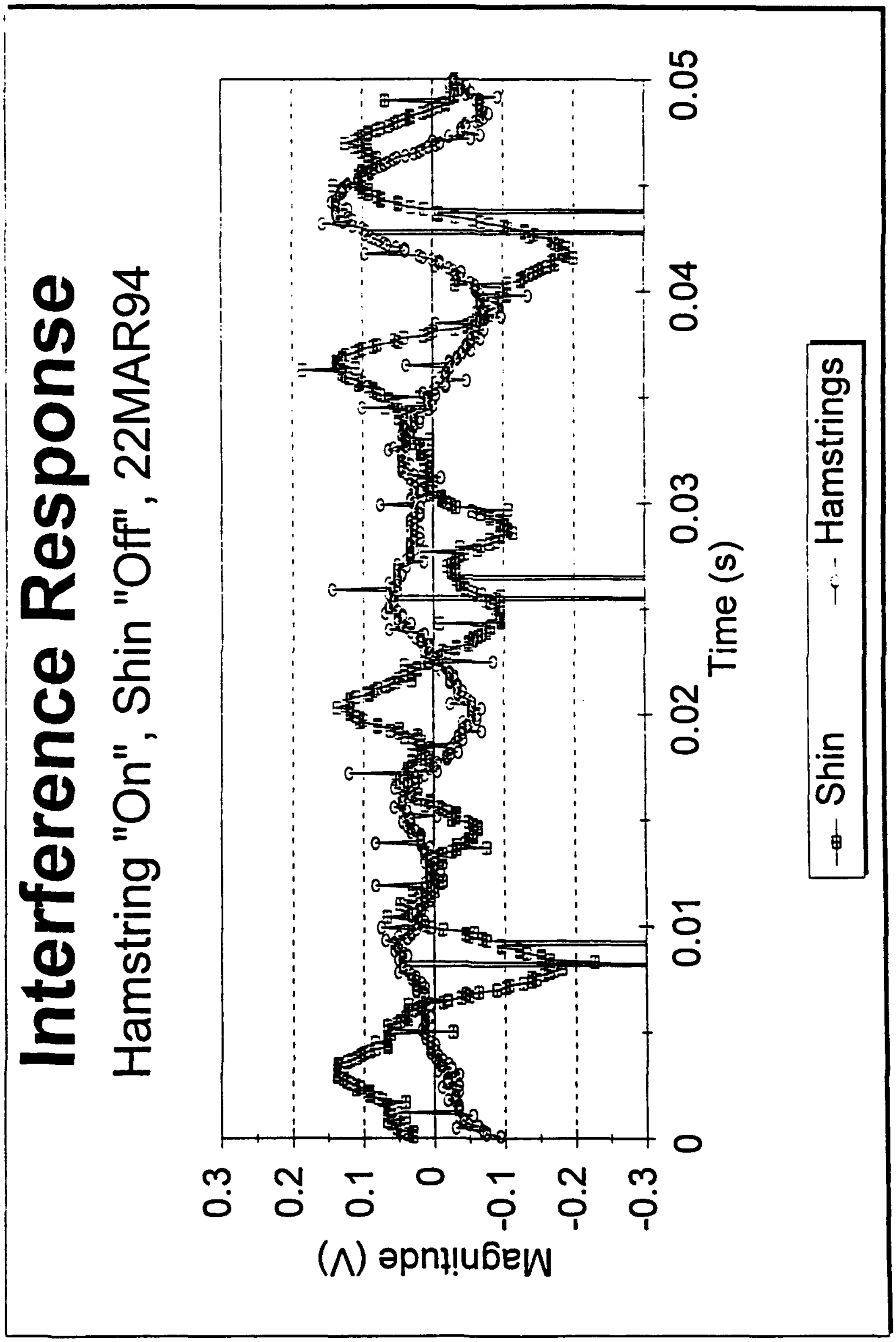
Appendix (T-2)


Interference Response
Quads "On", Hamstring "Off", 22MAR94
Magnitude (V)
0.2
0.1
0
-0.1
-0.2
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Quadriceps
Hamstrings

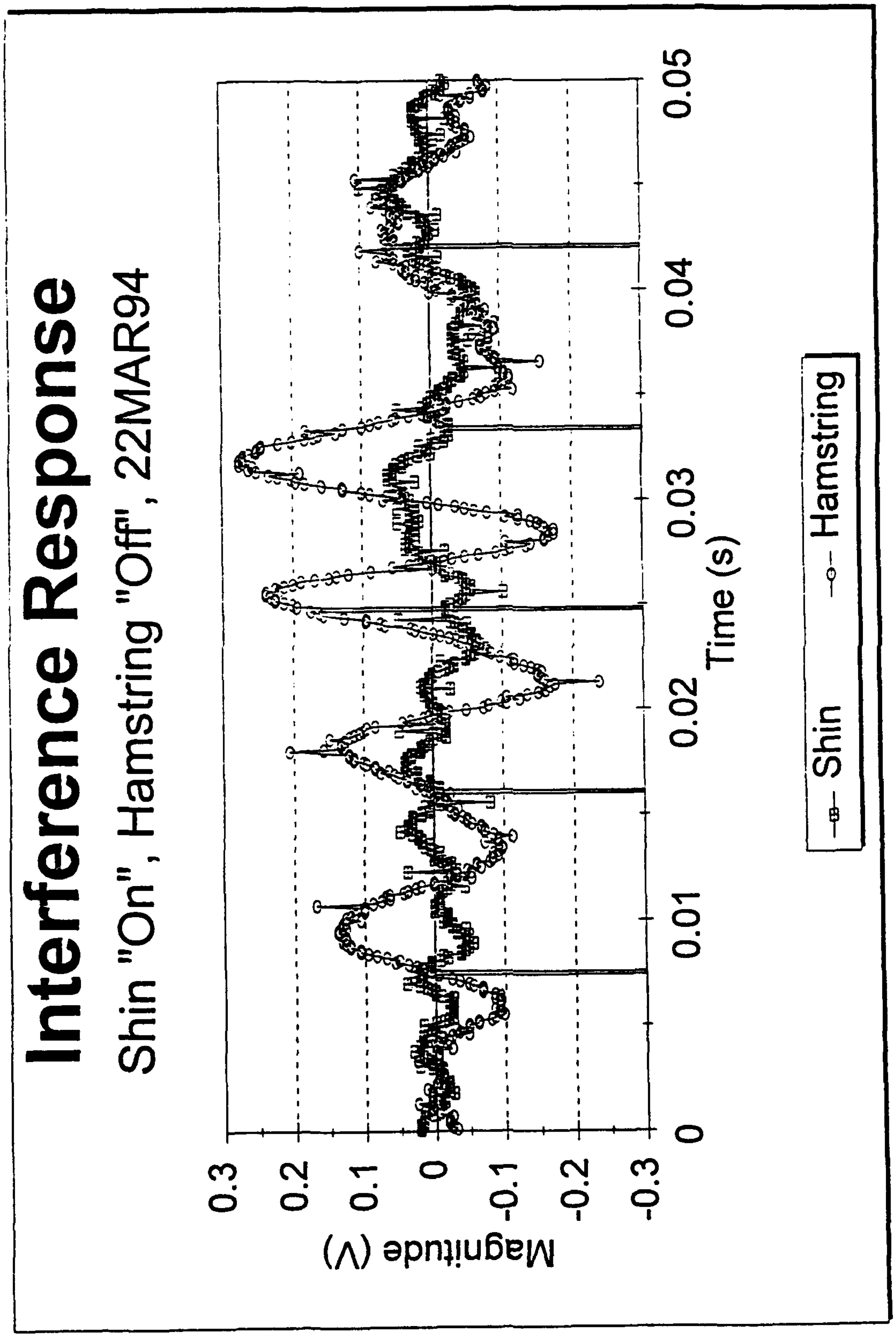
Appendix (T-3)


Interference Response
Hamstring and Shin Noise, 22MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Hamstrings

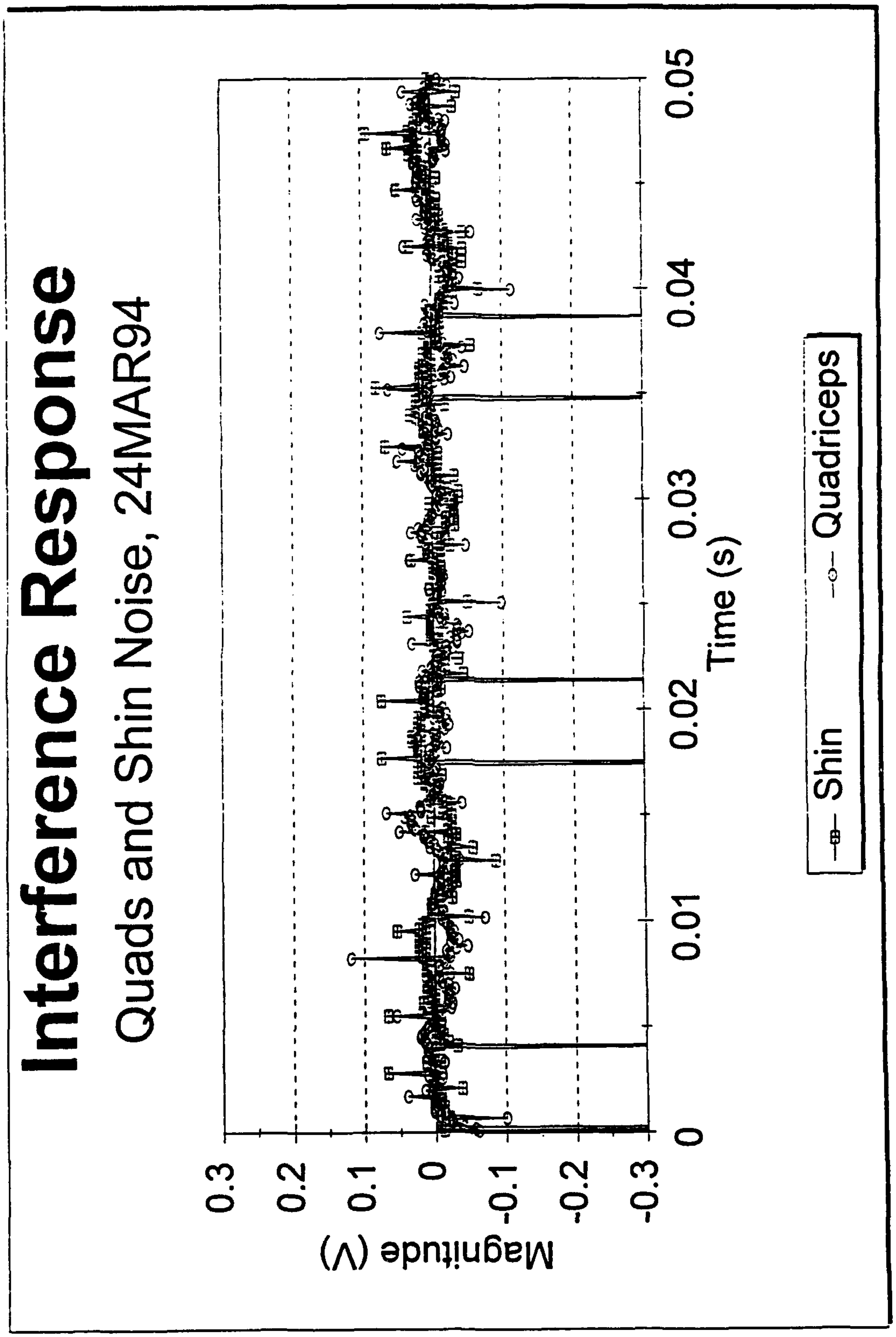
Appendix (U-1)


Interference Response
Hamstring "On", Shin "Off", 22MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Hamstrings

Appendix (U-2)


Interference Response
Shin "On", Hamstring "Off", 22MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
Time (s)
0
0.01
0.02
0.03
0.04
0.05
Shin
Hamstring

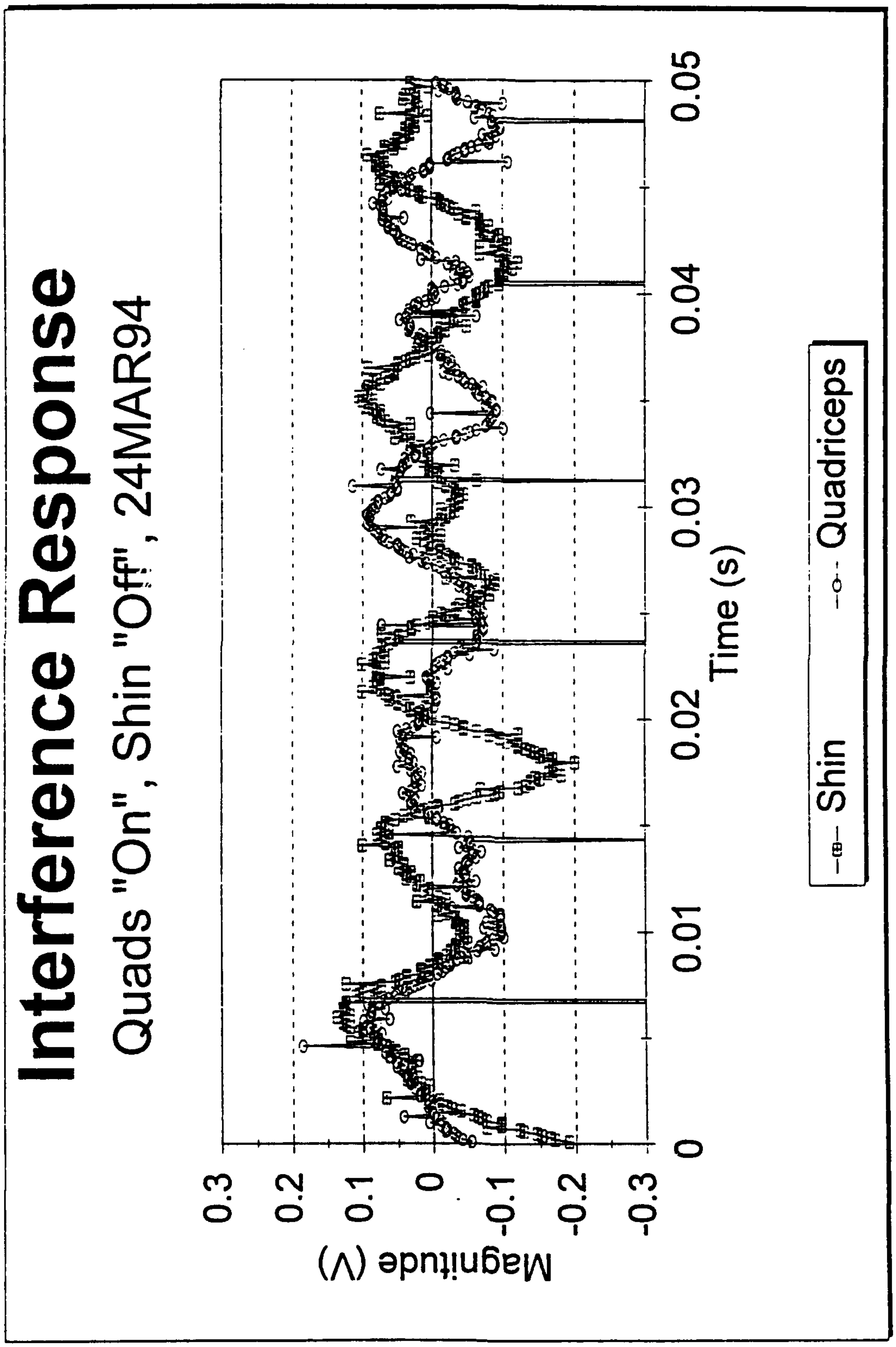
Appendix (U-3)



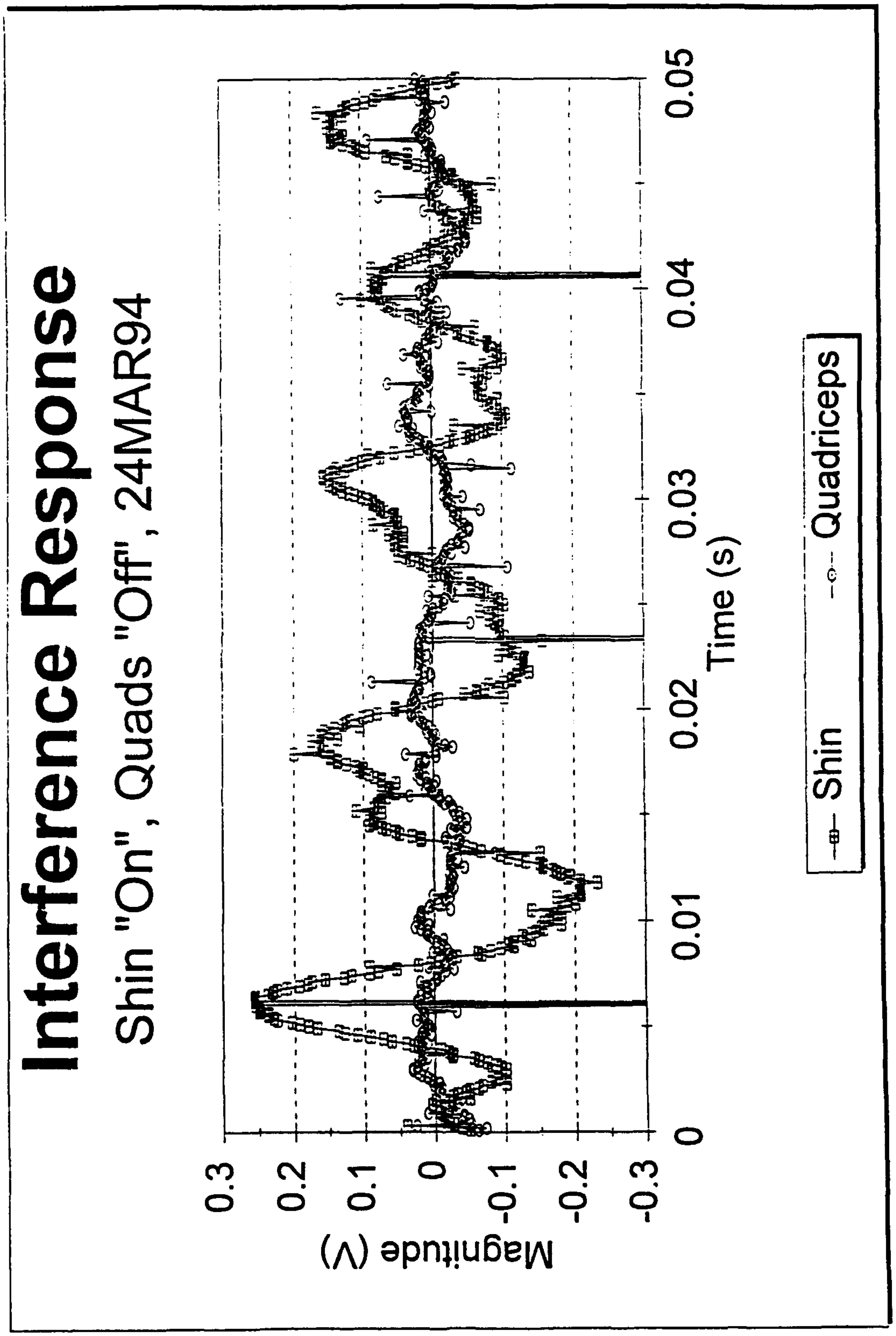
Interference Response

Quads and Shin Noise, 24MAR94

Appendix (V-1)


Interference Response
Quads "On", Shin "Off", 24MAR94
Magnitude (V)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.01
0.02
0.03
0.04
0.05
Time (s)
Shin
Quadriceps

Appendix (V-2)



Interference Response

Shin "On", Quads "Off", 24MAR94

Appendix (V-3)

Derivation for Pre- and Post-Amplifiers:

There are two devices within the circuit design that are dedicated to amplification. One amplifies the signal which comes from the surface electrodes (on the skin), and the other amplifies the DC signal coming from the AC / DC converter. They are based on the design of a simple inverting amplifier, and are the same with the exception of the output gain. The math refers to the amplifier design in Figure (6), and goes as follows (ideal Op-Amp assumed):

$$\frac{V_{in}-V_n}{R_{\epsilon}}+\frac{V_{out}-V_n}{R_f}=0$$

But $V_n = V_p = 0$;
Therefore,

$$\frac{V_{out}}{V_{in}}=-\frac{R_f}{R_{in}}$$

Thus, by varying either R_{in} or R_f (R_{in} in the case of these amplifiers), one can get a very simple amplifier with an easily adjustable gain. The fact that the amplifiers invert is moot; any computer program that uses these numbers can be made to compensate for the phase shift of the signal.

Proof for 180 Hz Notch Filter:

Figure (8) shows the circuit that was used as a filter at 180 Hz. The use of only one Op-Amp should not lead one to believe that this is, in fact, a simple circuit. The algebra in the mathematics is hardly simple.

This filter needs only three equations, however, to begin its solution. They are:

$$V_{in}\left(\frac{R_b}{R_b+R_a}\right)=V_x$$

$$\frac{V_{out}-V_x}{R_2}=(V_y-V_x)\,sC_1$$

$$(V_{out}-V_y)\,sC_2+\frac{V_{in}-V_y}{R_1}=(V_y-V_x)\,sC_1$$

Some hefty manipulation of the above into a transfer function yields this result:

$$\frac{V_{out}}{V_{in}}=\frac{s^2(C_1C_2R_1R_BR_2)+s(C_2R_1R_B+C_1R_1R_B-C_1R_AR_2)+R_B}{(R_A+R_B)\,[s^2(C_1C_2R_1R_2)+s(C_1R_1+C_2R_1)+1]}$$

According to *The Electronics Problem Solver* (1982) #10-100, the following numbers should be appropriate:

$C_1=C_2=0.01\mu F$ $R_1=8.8K\Omega$ $R_2\approx 884K\Omega$ $R_A=1K\Omega$ $R_B=50K\Omega$

However, at the time of assembly, it was not noticed that the capacitors had values closer to 0.014μF. This led to the circuit not working properly. Some trial and error finally resulted in success using these values:

$C_1\approx C_2\approx 0.014\mu F$ $R_1=7.5K\Omega$ $R_2\approx 543K\Omega$ $R_A=1K\Omega$ $R_B\approx 11K\Omega$

Derivation for Third Order Butterworth Filter:

A rigid derivation of the entire Butterworth Filter Series is well beyond the scope of this project. The result of all the derivations and such, in brief, is a standard filter type whose component resistors and capacitors have been tabulated in order that one can design filters by cookbook. Lam (1979) goes into detail about how the Butterworth Filter Series actually works; below, however, is the math behind this particular Third Order Butterworth Low-Pass Filter. Electrical components refer to the diagram in Figure (9).

The first stage is relatively benign. The use of ideal operational amplifier characteristics, which are commonly known among engineers[1], but are also described in Chapter 2 of Sedra and Smith's textbook *Microelectronic Circuits* (1991), provides the following:

$$V_f = V_q$$

$$\frac{V_{in}-V_q}{R_1} = \frac{V_x-0}{\frac{1}{sC_1}}$$

Rearranging yields the transfer function

$$\frac{V_f}{V_{in}} = \frac{1}{sC_1R_1+1}$$

[1] e.g. infinite input resistance, infinite open-loop gain, negligible output resistance

The second stage is much more complicated. It is, however, governed by only a few simple equations.

$$V_x = \frac{V_{out} R_5}{R_4 + R_5}$$

$$\frac{V_f - V_y}{R_2} + \frac{V_{out} - V_y}{\frac{1}{sC_2}} = \frac{V_y - V_x}{R_3}$$

$$\frac{V_y - V_x}{R_3} = \frac{V_x}{\frac{1}{sC_3}}$$

These equations are simplified by the criteria that $R_1 = R_2 = R_3$, $C_1 = C_2 = C_3$, and $R_4 = R_5$. The first two sets of equalities are characteristic of the Butterworth, and the third provides a gain of 2 courtesy of the design implementation by Berlin. Doing some substituting and rearranging results in this rather nasty equation:

$$\frac{V_{in}}{1+sCR} + V_{out} sCR + \frac{V_{out}}{2} = \frac{V_{out}}{2}(1+sCR)(2+sCR)$$

The simplified version of this brings out this equation:

$$\frac{V_{out}}{V_{in}} = \frac{\frac{2}{R^2C^2}}{s^2 + s^2\left(\frac{3}{RC}\right) + s\left(\frac{3}{R^2C^2}\right) + \frac{1}{R^3C^3}}$$

Now, substituting numbers for R and C (R = KΩ, C=0.022μF) results in this final equation.

$$\frac{V_{out}}{V_{in}} = \frac{3.16x10^9}{s^3 + 1165.5s^2 + 1358392s + 1.58x10^9}$$

So ends the mathematic derivation of the transfer function of this filter. The transfer function for the High-Pass Filter, which is not included, may be derived by using the same basic equations, but substituting C_1, C_2, and C_3 for R_1, R_2, R_3, and vice versa.

HP 3562A Signal Analyzer Screen Dump Program:

```
PROGRAM "HP3562A.BAS"
'This program dumps a data trace from the HP3562A Dynamic Signal Analyzer
'and puts the values into a MATLAB-compatible file.
'Written by William M. Gotten, USNA '94 for his Trident Project
'21 September 1993

REM $INCLUDE: 'c:\qb45\qbdecl.bas'     'This is the command library for the
                                       '  National Electronics GPIB Card

board% = 0                 'The board number (default=0)
PAD% = 11                  'The primary address location (default=11)
SAD% = 0                   'The secondary address location (default=0)
TMO% = 15            'The Time-Out timer (default = 15, about 20ms)
EOT% = 1                   'END message enabled -- don't know what this does
EOS% = 0                   'EOS mode disabled -- don't know what this does
devname$ = "gpib0"   'This is what the card likes to be called
DIM y(3000)                'This array stores data numbers
DIM x(3000)                'This array stores x-values
c$ = SPACE$(1)             'The size of one byte of information
CLS                        'Clear the screen
CALL IBFIND(devname$, UD%)  'What is the numeral location of the device
CALL IBDEV(board%, PAD%, SAD%, TMO%, EOT%, EOS%, UD%)
                                               'Initialize the device
CALL DevClear(0, 11)            'Clear the device buffer
CALL IBWRT(UD%, "dcas")         'Do a data trace dump in ASCII
FOR i = 1 TO 2                  'Read but ignore the first two data bytes
     CALL IBRD(UD%, c$)
NEXT

a$ = " "                        'Initialize the variables (clear them)
c$ = " "
WHILE ASC(c$) <> 13             'While not a predetermined byte
     CALL IBRD(UD%, c$)         'Read one byte at a time
     a$ = a$ + c$          'Update a$
WEND                            'Loop
y(0) = VAL(a$)                  'This is the final value of a$
CALL IBRD(UD%, c$)              'Read and ignore the next data byte

c$ = " "
FOR i = 1 TO 50                 'This is the Coordinate Transform Header
```

```
        a$ = c$
        WHILE ASC(c$) <> 13
                CALL IBRD(UD%, c$)
                a$ = a$ + c$
        WEND
        CALL IBRD(UD%, c$)
        y(i) = VAL(a$)
NEXT

c$ = " "
FOR i = 51 TO 116          'This is the data header
        a$ = c$
        WHILE ASC(c$) <> 13
                CALL IBRD(UD%, c$)
                a$ = a$ + c$
        WEND
        CALL IBRD(UD%, c$)
        y(i) = VAL(a$)
NEXT
c$ = " "
FOR i = 117 TO y(0)                'Here are the data numbers
        a$ = c$
        WHILE ASC(c$) <> 13
                CALL IBRD(UD%, c$)
                a$ = a$ + c$
        WEND
        CALL IBRD(UD%, c$)
        y(i) = VAL(a$)
NEXT
FOR i = 117 TO y(0)                'This manufactures the x axis
        x(i) = y(49) + (i - 117) * y(106)  'y(49) is x(0) and y(106) is dx
NEXT
INPUT "Enter filename for the trace data -->", yfilename$
OPEN "a:\" + yfilename$ + ".mat" FOR OUTPUT AS #1      'Configure for
                                                         '  MATLAB
FOR i = 117 TO y(0) - 1
        PRINT #1, LTRIM$(STR$(x(i))),
        PRINT #1, LTRIM$(STR$(y(i)))
NEXT i                             'This writes it to disk for MATLAB
CLOSE #1
CALL DevClear(0, 11)               'Clear the device buffer
CALL IBWRT(UD%, "lcl")  'Return the device to local mode
```

HP 54501A Oscilloscope Screen Dump Program:

```
PROGRAM "HP54501A.BAS"

'This program dumps a data trace from the HP54501 Digitizing Oscilloscope
'and puts the values into a MATLAB-compatible file.

'Written by William M. Gotten, USNA '94 for his Trident Project
'19 October 1993

REM $INCLUDE: 'c:\qb45\qbdecl.bas'     'This is the command library for the
                                       '  National Electronics GPIB Card

board% = 0                  'The board number (default=0)
PAD% = 7                    'The primary address location (default=7)
SAD% = 0                    'The secondary address location (default=0)
TMO% = 15           'The Time-Out timer (default = 15, about 20ms)
EOT% = 1                    'END message enabled -- don't know what this does
EOS% = 0                    'EOS mode disabled -- don't know what this does
devname$ = "gpib0"  'This is what the card likes to be called
DIM y1(600)                 'These arrays stores data numbers
DIM y2(600)
DIM y3(600)
DIM y4(600)
DIM x(600)                  'This array stores x-values
CLS                         'Clear the screen

CALL IBFIND(devname$, ud%)     'What is the numeral location of the device
CALL IBDEV(board%, PAD%, SAD%, TMO%, EOT%, EOS%, ud%)
                                         'Initialize the device
CALL DevClear(0, 7)            'Clear the device buffer
CALL ibwrt(ud%, "waveform:format ascii")      'Sets the style of the output

PRINT "Measure channel 1 (y/n)?"  'Determines which channels to
chan1$ = INPUT$(1)                '  take a look at
PRINT "Measure channel 2 (y/n)?"
chan2$ = INPUT$(1)
PRINT "Measure channel 3 (y/n)?"
chan3$ = INPUT$(1)
PRINT "Measure channel 4 (y/n)?"
chan4$ = INPUT$(1)
```

```
PRINT "Press any key to start . . ."
start$ = INPUT$(1)

a$ = " "
c$ = "x"
CALL ibwrt(ud%, "waveform:xreference?")         'Data point associated with
WHILE ASC(c$) <> 10                              'x-origin data value.  This should be
        CALL ibrd(ud%, c$)              '  zero.
        a$ = a$ + c$
WEND
xref = VAL(a$)

a$ = " "                                 'Time difference between consecutive data
c$ = "x"                                 '  points
CALL ibwrt(ud%, "waveform:xincrement?")
WHILE ASC(c$) <> 10
        CALL ibrd(ud%, c$)
        a$ = a$ + c$
WEND
xinc = VAL(a$)

a$ = " "
c$ = "x"
CALL ibwrt(ud%, "waveform:xorigin?")    'Time of first data point with
WHILE ASC(c$) <> 10                      '  respect to the trigger point
        CALL ibrd(ud%, c$)
        a$ = a$ + c$
WEND
xorig = VAL(a$)

IF chan1$ = "y" THEN
        a$ = " "                         'Initialize the variables
        c$ = "x"
        CALL ibwrt(ud%, "waveform:source channel1")     'Look at Channel 1
        CALL ibwrt(ud%, "Digitize Channel1")            'What it says
        CALL ibwrt(ud%, "waveform:yreference?")         'Data point where
y-origin
        WHILE ASC(c$) <> 10                             '  occurs
                CALL ibrd(ud%, c$)       'Read the data -- end of line
                a$ = a$ + c$             '  indicator is a space
        WEND
        yref = VAL(a$)
        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yincrement?")         'Voltage difference
between
        WHILE ASC(c$) <> 10                              '  consecutive data
points
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yinc = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yorigin?")   'Voltage value at center of
        WHILE ASC(c$) <> 10                    '  screen
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yorig = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:points?")    'How many points are there?
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        numpoints = VAL(a$)

        a$ = " "
        c$ = " "
        CALL ibwrt(ud%, "waveform:data?")      'Queries for data
        FOR counter = 1 TO numpoints
                a$ = " "
                c$ = " "
                WHILE ASC(c$) <> 44 AND ASC(c$) <> 10
                        CALL ibrd(ud%, c$)
                        a$ = a$ + c$
                WEND
                ytemp = VAL(a$)
                y1(counter) = ((ytemp - yref) * yinc) + yorig
        NEXT                        'Algorithm for converting to voltage
END IF

IF chan2$ = "y" THEN
        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:source channel2")
        CALL ibwrt(ud%, "Digitize Channel2")
        CALL ibwrt(ud%, "waveform:yreference?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yref = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yincrement?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yinc = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yorigin?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yorig = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:points?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        numpoints = VAL(a$)
        a$ = " "
        c$ = " "
        CALL ibwrt(ud%, "waveform:data?")
        FOR counter = 1 TO numpoints
                a$ = " "
                c$ = " "
                WHILE ASC(c$) <> 44 AND ASC(c$) <> 10
                        CALL ibrd(ud%, c$)
                        a$ = a$ + c$
                WEND
                ytemp = VAL(a$)
                y2(counter) = ((ytemp - yref) * yinc) + yorig
        NEXT
END IF

IF chan3$ = "y" THEN
        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:source channel3")
        CALL ibwrt(ud%, "Digitize Channel3")
        CALL ibwrt(ud%, "waveform:yreference?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yref = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yincrement?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yinc = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yorigin?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
```

```
    yorig = VAL(a$)

    a$ = " "
    c$ = "x"
    CALL ibwrt(ud%, "waveform:points?")
    WHILE ASC(c$) <> 10
        CALL ibrd(ud%, c$)
        a$ = a$ + c$
    WEND
    numpoints = VAL(a$)
    a$ = " "
    c$ = " "
    CALL ibwrt(ud%, "waveform:data?")
    FOR counter = 1 TO numpoints
        a$ = " "
        c$ = " "
        WHILE ASC(c$) <> 44 AND ASC(c$) <> 10
            CALL ibrd(ud%, c$)
            a$ = a$ + c$
        WEND
        ytemp = VAL(a$)
        y3(counter) = ((ytemp - yref) * yinc) + yorig
    NEXT
END IF

IF chan4$ = "y" THEN
    a$ = " "
    c$ = "x"
    CALL ibwrt(ud%, "waveform:source channel4")
    CALL ibwrt(ud%, "Digitize Channel4")
    CALL ibwrt(ud%, "waveform:yreference?")
    WHILE ASC(c$) <> 10
        CALL ibrd(ud%, c$)
        a$ = a$ + c$
    WEND
    yref = VAL(a$)

    a$ = " "
    c$ = "x"
    CALL ibwrt(ud%, "waveform:yincrement?")
    WHILE ASC(c$) <> 10
        CALL ibrd(ud%, c$)
```

```
                a$ = a$ + c$
        WEND
        yinc = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:yorigin?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        yorig = VAL(a$)

        a$ = " "
        c$ = "x"
        CALL ibwrt(ud%, "waveform:points?")
        WHILE ASC(c$) <> 10
                CALL ibrd(ud%, c$)
                a$ = a$ + c$
        WEND
        numpoints = VAL(a$)
        a$ = " "
        c$ = " "
        CALL ibwrt(ud%, "waveform:data?")
        FOR counter = 1 TO numpoints
                a$ = " "
                c$ = " "
                WHILE ASC(c$) <> 44 AND ASC(c$) <> 10
                        CALL ibrd(ud%, c$)
                        a$ = a$ + c$
                WEND
                ytemp = VAL(a$)
                y4(counter) = ((ytemp - yref) * yinc) + yorig
        NEXT
END IF

FOR counter = 1 TO numpoints                'Algorithm for determining time values
        x(counter) = (counter - xref) * xinc + xorig
NEXT

INPUT "Enter filename for the trace data -->", filename$
filename$ = filename$ + ".mat"
```

```
OPEN "a:\" + filename$ FOR OUTPUT AS #1     'Configure for MATLAB
OPEN "d:\matlab\scopeout.mat" FOR OUTPUT AS #2
FOR i = 1 TO numpoints - 1
        PRINT #1, LTRIM$(STR$(x(i))); " ",
        PRINT #2, LTRIM$(STR$(x(i))); " ",
        IF chan1$ = "y" THEN
                PRINT #1, LTRIM$(STR$(y1(i))),
                PRINT #2, LTRIM$(STR$(y1(i))),
        END IF
        IF chan2$ = "y" THEN
                PRINT #1, LTRIM$(STR$(y2(i))),
                PRINT #2, LTRIM$(STR$(y2(i))),
        END IF
        IF chan3$ = "y" THEN
                PRINT #1, LTRIM$(STR$(y3(i))),
                PRINT #2, LTRIM$(STR$(y3(i))),
        END IF
        IF chan4$ = "y" THEN
                PRINT #1, LTRIM$(STR$(y4(i))),
                PRINT #2, LTRIM$(STR$(y4(i))),
        END IF
        PRINT #1, CHR$(13)
        PRINT #2, CHR$(13)
NEXT i                                  'This writes it to disk for MATLAB
CLOSE #1
CLOSE #2

CALL DevClear(0, 7)                     'Clear the device buffer
CALL ibwrt(ud%, "run")                  'Starts the scope up again
CALL ibwrt(ud%, "system:key 39")        'Returns the scope to local control
```

Robot Controller Program:

```
'Program "Control"
'This program is the main controller program for the Trident project.
'It takes care of any initialization that must occur, handles the A/D
'converter board, and also talks to the robot.

'Program designed and written by William M. Gotten, Jr. '94 for his
'Trident Research Project.

DECLARE SUB Control.Graphics ()
DECLARE SUB Control.the.Robot ()
DECLARE SUB Home.The.Robot ()
DECLARE SUB Reset.Muscle.Values ()
DECLARE SUB A2D.Readings (Voltage())
DECLARE SUB Menu (Menu.Choice)
CLEAR
COM(2) ON
ON COM(2) GOSUB Speak                'If the robot has something to say

DIM SHARED Noise(0 TO 7)             'The noise reference array
DIM SHARED Inputs$(0 TO 7)           'The names of the sources
DIM SHARED Threshold(0 TO 7)  'The activation levels array
DIM SHARED Source(0 TO 7)            'The current levels array
DIM SHARED Percent.of.noise(0 TO 7)  'Percentage of threshold to noise
CONST FALSE = 0, TRUE = NOT FALSE
Menu.Choice = -1
Inputs$(0) = "(1) Left Bicep"
Inputs$(1) = "(2) Right Bicep"
Inputs$(2) = "(3) Left Foot"
Inputs$(3) = "(4) Right Foot"
Inputs$(4) = "(5) Left Calf"
Inputs$(5) = "(6) Right Calf"
Inputs$(6) = "(7) Left Quad"
Inputs$(7) = "(8) Right Quad"
cursrow = 20                  'Default for a subscreen
col1 = 1                             'Default for a subscreen

SCREEN (0)                           'Set the text screen
DO UNTIL Menu.Choice = 0
      CALL Menu(Menu.Choice)         'The Menu subroutine
```

```
    IF Menu.Choice = 0 THEN  'Exit the Program
        END
    END IF

    IF Menu.Choice = 1 THEN  'Return the robot to its
        Home.The.Robot               '  Home Position
    END IF

    IF Menu.Choice = 2 THEN  'Reset the noise and threshold
        Reset.Muscle.Values          '  values
    END IF

    IF Menu.Choice = 3 THEN  'Control a robot using muscle
        Control.the.Robot            '  inputs
    END IF

    IF Menu.Choice = 4 THEN          'Control a graphics cursor using
        Control.Graphics             '  muscle inputs.  This can be useful for
    END IF                           '  testing the program if the robot is
LOOP                                 '  unavailable.
END

Speak:                               'This subroutine prints to the screen what
                                     '  the robot sends back to the RS-232.
row = CSRLIN                         'Grab the current cursor position
col = POS(0)
VIEW PRINT 20 TO 24                  'Write to a subscreen at the bottom
LOCATE cursrow, col1                 'The last known position on the
                                     '  subscreen
char$ = INPUT$(1, #1)                'Grab a character from the RS-232 buffer
PRINT char$;                         'Print it
cursrow = CSRLIN                     'Where am I?
col1 = POS(0)
VIEW PRINT 1 TO 19                   'Back to the upper screen
LOCATE row, col                      'Put the cursor back there
RETURN

SUB A2D.Readings (Voltage())

'Modified version of a program in the DT2801 manual called MANEP06.
'
'Modified slightly to read and return all 8 bipolar channels
```

```
'by William M. Gotten, Jr. '94 for his Trident Project.  14 January 1994.
'Comments are from MANEP06.

10                      ' Read A/D Immediate command example
20 '
30                      ' Define constants
40 '
60 Base.Address = &H2EC
70 Command.Register = Base.Address + 1
80 Status.Register = Base.Address + 1
90 Data.Register = Base.Address
100 Command.Wait = &H4
110 Write.Wait = &H2
120 Read.Wait = &H5
130 '
140 Cclear = &H1
150 Cadin = &HC
160 Cstop = &HF
170 DIM Gain(0 TO 3)
175 Base.Factor# = 4096
180 Base.Channels = 8
190 Gain(0) = 1
191 Gain(1) = 2
192 Gain(2) = 4
193 Gain(3) = 8
200 '
210                     'Stop and clear the DT2801 series board
220 '
230 OUT Command.Register, Cstop
240 Temp = INP(Data.Register)
245 WAIT Status.Register, Write.Wait, Write.Wait
250 WAIT Status.Register, Command.Wait
260 OUT Command.Register, Cclear
1000 '
1010                    'Set the desired A/C channel and gain code
1015 '
1040 ADgain = 0
1070 '
1080 FOR ADchannel = 0 TO 7
1100 '
1120                    'Wait until the DT2801 series board
1123                    '  DATA IN FULL flag is clear and READY
```

```
1125            '  flag is set, then write the READ A/D
1130            '  IMMEDIATE command byte to the Command Register.
1140 '
1145 WAIT Status.Register, Write.Wait, Write.Wait
1150 WAIT Status.Register, Command.Wait
1160 OUT Command.Register, Cadin
1170 '
1180            'Wait until the DT2801 series board DATA IN
1185            '  FULL flag is clear, then write the A/D gain
1190            '  byte to the Data In Register
1200 '
1210 WAIT Status.Register, Write.Wait, Write.Wait
1220 OUT Data.Register, ADgain
1230 '
1240            '  Wait until the DT2801 serices board DATA IN
1245            '  FULL flag is clear, then write the A/D channel
1250            '  byte to the Data In Register
1260 '
1270 WAIT Status.Register, Write.Wait, Write.Wait
1280 OUT Data.Register, ADchannel
1290 '
1300            '  Read two bytes of A/D data from the Data
1305            '  Out Register, waiting for a set DATA OUT
1310            '  READY (or READY) flag before each read,
1320            '  and combine the two bytes into one word.
1330 '
1340 WAIT Status.Register, Read.Wait
1350 Low = INP(Data.Register)
1360 WAIT Status.Register, Read.Wait
1370 High = INP(Data.Register)
1380 Data.value# = High * 256 + Low
1390 '
1400            'Wait until the DT2801 series board
1403            '  DATA IN FULL flag is clear and READY flag
1405            '  is set, indicating command completion, then
1410            '  check the Status Register ERROR flag.
1420 '
1425 WAIT Status.Register, Write.Wait, Write.Wait
1430 WAIT Status.Register, Command.Wait
1440 Status = INP(Status.Register)
1450 IF (Status AND &H80) THEN GOTO 2030
1460 '
```

```
1470                    'Calculate and print the A/D reading in volts
1480 '
1500 Factor# = (10 / Base.Factor#) / Gain(ADgain)
1510 Uni.volts# = Data.value# * Factor#
1520 Bi.volts# = Uni.volts# * 2 - (10 / Gain(ADgain))
1530 '
1560 Voltage(ADchannel) = Bi.volts#
1570 '
1630 NEXT
2000 EXIT SUB
2010                'Error.
2020 '
2030 PRINT
2040 PRINT "Error!!"
2050 PRINT
2080 END SUB

SUB Control.Graphics

'This subroutine is the main program for the graphics portion of
'this program. The graphics cursor (a circle) in this case is controlled by
'inputs from the body. This subroutine can be used if the robot is unavailable.

SCREEN (12)                           'Sets the graphics screen
VIEW PRINT 26 TO 30                   'Sets the text area
row = 200                             'This is the default row
DIM Object(0 TO 1000) AS INTEGER      'This is for cutting and pasting
col = 320                             'This is the default column
Rowmin = 0                            'This is the minimum row available
Colmin = 0                            'This is the minimum column available
Rowmax = 390                          'This is the maximum row available
Colmax = 639                          'This is the maximum column available
Object.Color = 13                     'Sets the default color
Radius = 5                            'Sets the default radius
DIM Active(0 TO 7)                'The boolean for an active channel
VIEW (Colmin, Rowmin)-(Colmax, Rowmax)     'Sets the viewport
LINE (Colmin, Rowmin)-(Colmax, Rowmax), 15, B        'Puts a box around the
                                                     '  viewport
Key.Pressed$ = " "                    'Clears the variable
CIRCLE (col, row), Radius, Object.Color  'Draws the cursor (a circle)
PAINT (col, row), Object.Color        'Fills the circle
GET (col - Radius, row - Radius)-(col + Radius, row + Radius), Object
```

```
                                            'Saves the circle to memory
PUT (col - Radius, row - Radius), Object, XOR   'Plots the circle using XOR
                                            '  function
DO WHILE Key.Pressed$ <> CHR$(27)           'While ESC is not pressed
     Key.Pressed$ = INKEY$                  'Read from the keyboard
     Cursor.Row = 26                        'The top text line
     CALL A2D.Readings(Source())            'Read the A/D board
     LOCATE Cursor.Row                      'Put the cursor at the top text line
     FOR Input.Number = 0 TO 3              'The first four inputs
          Percentage.of.noise = VAL(LEFT$(STR$(Source(Input.Number) /
           Threshold(Input.Number) * 100), 5))
          IF Percentage.of.noise >= 100 THEN             'If source exceeds
threshold
               Active(Input.Number) = TRUE
          ELSE
               Active(Input.Number) = FALSE
          END IF
          PRINT Inputs$(Input.Number); " =";
          PRINT Percentage.of.noise;
          LOCATE CSRLIN, 23                 'Sets the cursor column
          PRINT "%"
     NEXT
     LOCATE 26
     FOR Input.Number = 4 TO 7              'Same as above for last 4 inputs
          LOCATE CSRLIN, 50
          Percentage.of.noise = VAL(LEFT$(STR$(Source(Input.Number) /
           Threshold(Input.Number) * 100), 5))
          IF Percentage.of.noise >= 100 THEN
               Active(Input.Number) = TRUE
          ELSE
               Active(Input.Number) = FALSE
          END IF
          PRINT Inputs$(Input.Number); " =";
          PRINT Percentage.of.noise;
          LOCATE CSRLIN, 73
          PRINT "%"
     NEXT

     PUT (col - Radius, row - Radius), Object, XOR         'Plot the circle
     PUT (col - Radius, row - Radius), Object, XOR         'XOR the circle
     IF Active(0) THEN                                     'Increase the radius of
the                                                        '  circle
```

```
                IF (Radius < 30) AND (col + Radius + 1 < Colmax) AND (col -
Radius - 1 > Colmin) AND (row + Radius + 1 < Rowmax) AND                (row -
Radius - 1 > Rowmin) THEN        'Ensure the circle does not
                                  '  go off the screen
                        VIEW (Colmin, Rowmin)-(Colmax, 479) 'View the entire
                                                             '  screen area
                        Radius = Radius + 1                 'Increase the radius
                        CIRCLE (320, 440), Radius, Object.Color    'Draw the circle
                        PAINT (320, 440), Object.Color      'Fill in the circle
                        GET (320 - Radius, 440 - Radius)-(320 + Radius, 440 + Radius),
                          Object                            'Put the circle in memory
                        PUT (320 - Radius, 440 - Radius), Object, XOR     'Erase the
                                                                          '  circle
                        VIEW (Colmin, Rowmin)-(Colmax, Rowmax)     'Return to
                                                              'original viewport
                END IF
        END IF

        IF Active(1) THEN               'Change the color of the circle
                IF Object.Color > 15 THEN
                        Object.Color = 0
                ELSE
                        Object.Color = Object.Color + 1
                END IF
                VIEW (Colmin, Rowmin)-(Colmax, 479)        'See Active(0) for
comments
                CIRCLE (320, 440), Radius, Object.Color
                PAINT (320, 440), Object.Color
                GET (320 - Radius, 440 - Radius)-(320 + Radius, 440 + Radius), Object
                PUT (320 - Radius, 440 - Radius), Object, XOR
                VIEW (Colmin, Rowmin)-(Colmax, Rowmax)
        END IF

        IF Active(2) THEN               'Go down a row
                IF (row + Radius + 1) < Rowmax THEN
                        row = row + 1
                END IF
        END IF

        IF Active(3) THEN               'Go up a row
                IF (row - Radius - 1) > Rowmin THEN
                        row = row - 1
        END IF
    END IF

    IF Active(4) THEN                'Go right a column
        IF (col + Radius + 1) < Colmax THEN
            col = col + 1
        END IF
    END IF

    IF Active(5) THEN                'Go left a column
        IF (col - Radius - 1) > Colmin THEN
            col = col - 1
        END IF
    END IF

    IF Active(6) THEN                'Plot the circle
        PUT (col - Radius, row - Radius), Object, PSET
    END IF

    IF Active(7) THEN  'Decrease the radius of the circle  'See Active(0) for
                                                           '  comments
        IF (Radius > 1) AND (col + Radius + 1 < Colmax) AND (col -
Radius - 1 > Colmin) AND (row + Radius + 1 < Rowmax) AND          (row -
Radius - 1 > Rowmin) THEN
            VIEW (Colmin, Rowmin)-(Colmax, 479)
            Radius = Radius - 1
            CIRCLE (320, 440), Radius, Object.Color
            PAINT (320, 440), Object.Color
            GET (320 - Radius, 440 - Radius)-(320 + Radius, 440 + Radius),
              Object
            PUT (320 - Radius, 440 - Radius), Object, XOR
            VIEW (Colmin, Rowmin)-(Colmax, Rowmax)
        END IF
    END IF
LOOP
SCREEN (0)                           'Return to text screen
CLS
END SUB

SUB Control.the.Robot
'This subroutine is the main controller for the SCORBOT portion of the
'program
```

```
CLS
OPEN "com2:9600,n,8,1,bin,cs0,cd0,ds0,rb8192" FOR RANDOM AS #1 'Opens the
COM2 port.
PRINT #1, "noquiet"   'Tells the robot to relay all information to RS-232
DIM Active(0 TO 7)    'Active says whether or not to activate part of the
Active(1) = FALSE     '  robot.
Active(2) = FALSE
Active(3) = FALSE
Active(4) = FALSE
Active(5) = FALSE
Active(6) = FALSE
Active(7) = FALSE
Active(0) = FALSE
DIM sensor(0 TO 7) 'Sensor is a Boolean variable that says whether or not to
sensor(1) = true          'pay attention to the sensor (basically a contingency
sensor(2) = true          'device)
sensor(3) = true
sensor(4) = true
sensor(5) = true
sensor(6) = true
sensor(7) = true
sensor(0) = true
Base. = 0                 'User-defined "home" position
Shoulder = 0
Elbow = 0
Pitch = 0
Roll = 0
'
Base.Max = 24000             'These are all values adjusted from the
Base.Min = -18530            '  Scorbot manuals.
Shoulder.Max = 16900
Shoulder.Min = -100
Elbow.Max = 20700
Elbow.Min = -31700
Pitch.Max = 15700
Pitch.Min = -15700
Roll.Max = 15700
Roll.Min = -15700
Gripper.Max = 32767
Gripper.Min = -32768
Jam.Check = 40               'Value for how long robot checks to find
```

```
                        ' that an axis has jammed. Normal is 10-25.
Program.Speed = 3          'This is a subjective value that reduces
                        ' jerkiness in the robot. Initial value
                        ' is 15; 2 or 3 is recommended for this
                        ' program.
ON COM(2) GOSUB Speak
PRINT #1, "delp Posit"          'Delete Position "Posit"
SLEEP 2                         'Wait 2 seconds--safety feature
PRINT #1, "y"                   'Verify deletion
PRINT #1, "defp Posit"          'Define the current location
PRINT #1, "here Posit"          'Name it.
PRINT #1, "setpv Posit 1 "; Base.      'Set the initial robot values
PRINT #1, "setpv Posit 2 "; Shoulder
PRINT #1, "setpv Posit 3 "; Elbow
PRINT #1, "setpv Posit 4 "; Pitch
PRINT #1, "setpv Posit 5 "; Roll
PRINT #1, "move Posit"                 'Move the robot there
PRINT #1, "close"                      'Close the gripper
PRINT #1, "let par 101="; Base.Max       'Reset axis limitation parameters
PRINT #1, "let par 102="; Shoulder.Max
PRINT #1, "let par 103="; Elbow.Max
PRINT #1, "let par 104="; Pitch.Max
PRINT #1, "let par 105="; Roll.Max
PRINT #1, "let par 106="; Gripper.Max
PRINT #1, "let par 121="; Base.Min
PRINT #1, "let par 122="; Shoulder.Min
PRINT #1, "let par 123="; Elbow.Min
PRINT #1, "let par 124="; Pitch.Min
PRINT #1, "let par 125="; Roll.Min
PRINT #1, "let par 126="; Gripper.Min
PRINT #1, "let par 186="; Jam.Check
PRINT #1, "let par 187="; Jam.Check
PRINT #1, "let par 188="; Jam.Check
PRINT #1, "let par 189="; Jam.Check
PRINT #1, "let par 190="; Jam.Check
PRINT #1, "let par 191="; Jam.Check
PRINT #1, "let par 300="; Program.Speed
PRINT #1, "init control"             'Activate new parameters
PRINT #1, "close"
PRINT #1, "~"                        'Activate "Manual" control
closed = true
letter$ = CHR$(4)        'Keyboard idle character (why CHR$(4))?? Why not?
```

```
'
CLS
PRINT "Press ESC to return to the menu."
PRINT
PRINT
row = CSRLIN                    'Saves the current row number in memory
DO UNTIL letter$ = CHR$(27)  'Until ESC is pressed
    letter$ = INKEY$
    CALL A2D.Readings(Source())  'Read the A/D card
    LOCATE row                   'Place the cursor in the correct row
    FOR Input.Number = 0 TO 7
        Percentage.of.noise = VAL(LEFT$(STR$(Source(Input.Number) /
Threshold(Input.Number) * 100), 5))
        PRINT Inputs$(Input.Number); " =";
        PRINT Percentage.of.noise;
        LOCATE CSRLIN, 23       'Change the column of the cursor
        PRINT "% of Threshold value";
        LOCATE CSRLIN, 44       'Change the column of the cursor
        IF Percentage.of.noise > 100 THEN
            PRINT "     (ACTIVATED)  "
            Active(Input.Number) = true
        ELSE
            PRINT "  (NOT ACTIVATED)  "
            Active(Input.Number) = FALSE
        END IF
    NEXT
    IF letter$ = "~" THEN   'Toggle the Manual Mode (the mode we want)
        PRINT #1, "~"
        letter$ = CHR$(4)
    END IF

    IF (Active(0) AND Active(1) AND sensor(0) AND sensor(1)) THEN
        PRINT #1, "1"       'Move base right
    ELSE
        IF (Active(0) AND sensor(0)) THEN
            PRINT #1, "2"   'Move shoulder up
        ELSE
            IF (Active(1) AND sensor(1)) THEN
                PRINT #1, "w"  'Move shoulder down
            END IF
        END IF
    END IF
```

```
IF (Active(2) AND Active(3) AND sensor(2) AND sensor(3)) THEN
        PRINT #1, "q"     'Move base left
ELSE
        IF (Active(2) AND sensor(2)) THEN
                PRINT #1, "4"   'Move pitch up
        ELSE
                IF (Active(3) AND sensor(3)) THEN
                        PRINT #1, "r"   'Move pitch down
                END IF
        END IF
END IF
IF (Active(4) AND Active(5) AND sensor(4) AND sensor(5)) THEN
        IF closed THEN
                PRINT #1, "y"     'Open gripper
                closed = FALSE
        ELSE
                PRINT #1, "6"     'Close gripper
                closed = true
        END IF
ELSE
        IF (Active(4) AND sensor(4)) THEN
                PRINT #1, "3"     'Move elbow up
        ELSE
                IF (Active(5) AND sensor(5)) THEN
                        PRINT #1, "e"   'Move elbow down
                END IF
        END IF
END IF
IF (Active(6) AND Active(7) AND sensor(6) AND sensor(7)) THEN
ELSE
        IF (Active(6) AND sensor(6)) THEN
                PRINT #1, "t"       'Move roll left
        ELSE
                IF (Active(7) AND sensor(7)) THEN
                        PRINT #1, "5"  'Move roll right
                END IF
        END IF
END IF
Active(1) = FALSE                                       'Resets
Active(2) = FALSE
Active(3) = FALSE
Active(4) = FALSE
```

```
     Active(5) = FALSE
     Active(6) = FALSE
     Active(7) = FALSE
     Active(0) = FALSE
     IF (VAL(letter$) >= 1 AND VAL(letter$) <= 8) THEN    'Was a sensor
          sensor(VAL(letter$) - 1) = NOT sensor(VAL(letter$) - 1)   'toggled?
          letter$ = CHR$(4)
     END IF

LOOP       'Was ESC pressed?

PRINT #1, "~"                    'Turn off manual mode
CLOSE #1
VIEW PRINT 1 TO 24
CLS
END SUB

SUB Home.The.Robot
'This subroutine returns the robot to its home Posit

CLS
PRINT "Homing the Robot--this takes a little while."
OPEN "com2:9600,n,8,1,bin,ds0,cs0,cd0" FOR RANDOM AS #1        'Opens the
                                                               ' COM2 port.
PRINT #1, "home"
CLOSE #1
END SUB

SUB Menu (Menu.Choice)
'This subroutine drives the menu for the main program

Menu.Choice = -1                                       'The default
DO UNTIL Menu.Choice < 5 AND Menu.Choice >= 0          'Make sure the
                                                         '  choices are valid
CLS
PRINT "Robotic Control using Muscular Signals"
PRINT
PRINT "Menu Selection:"
PRINT
PRINT "0. Exit the System"
PRINT "1. Home Robot"
PRINT "2. Reset Muscle Threshold Values"
PRINT "3. Activate the System (Robot)"
PRINT "4. Activate the System (Graphics)"
PRINT
PRINT "Your choice -->";
INPUT Menu.Choice
LOOP

END SUB

SUB Reset.Muscle.Values
'This subroutine resets the random noise values along with the
'Threshold values that activate the controllers

CLS
PRINT "This portion will stabilize the computers common references."
PRINT
PRINT "Stabilize your readings and then press the [SPACE BAR]."
PRINT "The computer will wait three seconds and then save the new"
PRINT "reference points."
PRINT
PRINT
DIM Threshold.Temp(0 TO 7)
row = CSRLIN                         'Save the current row number in memory
DO UNTIL INKEY$ = " "   'Until the SPACE BAR is pressed
        CALL A2D.Readings(Noise())          'Read from the A/D converter
        LOCATE row           'Place the cursor on the right row
        FOR Input.Number = 0 TO 7
                PRINT Inputs$(Input.Number); "  = ";
                PRINT USING "###.#"; Noise(Input.Number);
                PRINT "    "
        NEXT
LOOP
PRINT
PRINT

Start.seconds$ = RIGHT$(TIME$, 2)         'This section "waits" 3 seconds
Print.Seconds$ = "-1"
End.Seconds$ = STR$(VAL(RIGHT$(TIME$, 2)) + 3)
DO UNTIL VAL(Print.Seconds$) = 0
        Current.seconds$ = RIGHT$(TIME$, 2)
        Print.Seconds$ = STR$(VAL(End.Seconds$) - VAL(Current.seconds$))
        CALL A2D.Readings(Noise())
        LOCATE row
        FOR Input.Number = 0 TO 7
                PRINT Inputs$(Input.Number); "  = ";
                PRINT USING "###.#"; Noise(Input.Number);
                PRINT "   "
        NEXT
        PRINT
        PRINT "Time left ="; Print.Seconds$; " seconds"
LOOP
CLS
PRINT "Noise values stored.  Activate one of the muscle groups and"
PRINT "press its corresponding number to store the voltage which will cause"
PRINT "activation of the robot.  Press ESC to end."
PRINT
row = CSRLIN                            'Save the current row in memory
Key.Pressed$ = " "
DO UNTIL Key.Pressed$ = CHR$(27)        'Until ESC is pressed
        Key.Pressed$ = INKEY$           'Read from the keyboard
        CALL A2D.Readings(Source())
        LOCATE row                 'Get the correct row
        FOR Input.Number = 0 TO 7
                Percent.of.noise(Input.Number) =
VAL(LEFT$(STR$(Source(Input.Number) / Noise(Input.Number) * 100), 5))
                PRINT Inputs$(Input.Number); "  = ";
                PRINT USING "###.#"; Source(Input.Number);
                PRINT "   ";
                LOCATE CSRLIN, 35
                PRINT "% of noise (>100% to activate) =";
                PRINT USING "###"; Percent.of.noise(Input.Number);
                PRINT "   "
        NEXT
        PRINT
        LOCATE CSRLIN
        PRINT "Currently set values are:"
        FOR Input.Number = 0 TO 7
                PRINT Inputs$(Input.Number); "  =";
                PRINT USING "###.#"; Threshold(Input.Number);
                PRINT "   ";
                Percent.of.noise(Input.Number) =
VAL(LEFT$(STR$(Threshold(Input.Number) / Noise(Input.Number) * 100), 5))
                LOCATE CSRLIN, 35                 'Sets the correct column
                PRINT "% of noise (>100% to activate) =";
```

```
            PRINT USING "###"; Percent.of.noise(Input.Number);
            PRINT "   "
      NEXT
      Number.Pressed = VAL(Key.Pressed$) - 1          'Reads from the keyboard
      IF (Number.Pressed >= 0 AND Number.Pressed <= 7) THEN
            Threshold(Number.Pressed) = Source(Number.Pressed)
      END IF
LOOP
END SUB
```