A PATTERN RECOGNITION BASED ARM MOVEMENT CLASSIFIER WITH INCREASED FEATURE SPACE

by

John Alan Wieler

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Electrical Engineering

Winnipeg, Manitoba

ISBN 0-315-47923-X

A PATTERN RECOGNITION BASED ARM MOVEMENT CLASSIFIER WITH INCREASED FEATURE SPACE

BY

JOHN ALAN WIELER

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

In pattern recognition based prosthetic control, the addition of new features and the refinement of the feature space is important. This thesis investigates a new feature, muscle activation delay time (MADT), as a function of the coordinating activity of the lower brain. As well, an investigation is carried out into increasing the discriminant power of a linear discriminant function by using six or eight channels of electromyographic signals, as opposed to four as has been previously studied.

EMG signals were observed, digitized, and stored for eight subjects. Six arm movements were studied: humeral rotation medially/laterally, elbow flexion/extension, and wrist supination/pronation.

It is found that eight channels of data allow the classifier to be 99% successful in motion classification. Six channels allow 97.6% successful classification, while four channels of data allow 92.5% success in classification. It is also found that the MADT feature provides little added discriminant power.

ACKNOWLEDGEMENTS

I wish to acknowedge the following people for their support during the course of my research. Dr. Garry Smith for his patience, Jack Sill for his technical support, Mr. and Mrs. John E. Wieler for their moral guidance and support. I would like to thank Dr. Ed Shwedyk especially for his guidance, patience, and motivating influence.

Sola Deo Gloria

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I
INTRODUCTION

A completely lifelike and fully functional artificial replacement for missing limbs has been sought for many years. While popular fiction such as 'The Six Million Dollar Man' has offered tantalizing glimpses of a prosthesis that is stronger and faster than the ordinary human limb, the reality is that man is not easily rebuilt, and replacements fall far short of the original both in appearance and function.

Despite the inadequacies of prosthetic technology past and present, progress is being made. With the advent of microprocessors, prostheses may be controlled with a degree of speed and accuracy not previously possible. However, despite the power of the silicon chip the fundamental problems of the man-machine interface still have to be solved. This thesis addresses the problems of voluntary prosthesis control and proposes a scheme that allows upper limb prosthesis control to be linked to the user's own motor control processes.

1.1 <u>BACKGROUND</u>

At the nexus of present prosthesis control technology is myoelectric control. Myoelectric control uses the characteristics of the electromyographic (EMG) signal generated by muscles to control prosthetic action. Myoelectric control has evolved from the less sophisticated background of mechanical and electromechanical prosthesis control.

Prostheses in their most primitive form were simply pieces of wood or metal hooks strapped to the arm. Later developement allowed for the opening and closing of pincers by means of pulleys and wires attached to the body by cumbersome harnesses. The grasping action was actuated by gross shoulder movements. This form of prosthesis, while unsightly and mechanically crude, is still used due to its simplicity.

Electro-mechanical prostheses allow the user to initiate actions such as wrist pronation or elbow flexion-extension by closing switches located in the shoulder socket of the prosthesis or on straps across the chest. Various movements of the shoulder or upper body open or close the switches allowing electric motors in the prosthesis to start or stop an action. Electro-mechanical prostheses work well due to the relative simplicity of the electro-mechanical circuitry, and are functionally and aesthetically much more sophisticated than the wholly mechanical designs.

Electro-myographic prostheses use the characteristics of the EMG signals to control the electric motors which power the arm. Electro-myographic prostheses bypass mechanical switches completely, allowing the operator to control prosthetic action by controlling the action of a specific muscle or group of muscles.

1.2 EMG SIGNALS

Due to muscle physiology, all muscles produce minute, random electrical signals. The greater the force output of the muscle, the larger the average variance of the electromyographic (EMG) signal. The EMG signal of a muscle, particularly skeletal muscle, can be observed by electrodes attached to the skin above the muscle. By specifying a threshold value for the averaged EMG signal variance the EMG signal may be used as a switch, turning prosthesis actions on or off depending on the value of the variance.

1.3 MYOELECTRIC PROSTHETIC CONTROL

1.3.1 Proportional Control

Various schemes for myoelectric control have emerged over the last twenty years. Proportional control allows the speed or force of prosthetic action to be proportional to state of contraction of the control muscle[1]. A drawback of the proportional control scheme is that one muscle can only control one degree of freedom of a prosthesis. A multifunction prosthesis would require several separate control muscles. Unfortunately, in the upper arm the number of muscles which may be used is limited, which makes a multi-function prosthesis that uses proportional control difficult to realize.

1.3.2 <u>Three State Control</u>

Three state control allows a muscle to control two degrees of freedom of a prosthesis[2]. Two EMG signal thresholds are established. If the EMG signal variance falls below the lower threshold, no action is performed. This is known as the rest state. If the EMG signal variance falls between the two thresholds, a primary action is initiated, and if the variance greater than the second threshold, a secondary action, usually the opposite of primary action, is initiated. For example, for an above the elbow amputee the action performed by the prosthesis may be hand open-close. This action would be controlled by the biceps muscle in the upper arm. Hand opening would be the primary action, and hand closing the secondary action. While three state control requires operator training, it has been shown to be essentially error free[3].

1.3.3 <u>Five State Control</u>

Five state control allows the operator to control five prosthetic actions with a single control muscle. Five state control is essentially the same as three state control except that four thresholds are used instead of two. Using four thresholds allows average EMG variance to fall into one of five states instead of one of three. Five state control is more prone to error due to the greater muscle control needed by the operator to succesfully chose one of five levels of muscle activity.

Research by by the author has shown that even after training operator error was still too high for five level control to be practical. Training was carried out over a two month period, three times a week. All training was carried out in a highly controlled enviroment, with all distractions removed in order to let the subjects concentrate entirely on the training. The experimental method is described by McKenzie[4]. In spite of the carefully controlled enviroment operator error was as high as 30% at the end of the training period. The high error rate indicated that in an uncontrolled enviroment practical five level control would not be feasible.

1.4 <u>PATTERN RECOGNITION BASED MYOELECTRIC CONTROL</u>

Both proportional control and three and five state control require the operator to use the control muscle in a way that is not natural to the action being performed. These muscle actions, especially in the case of three and five state control must be learned before the operator can utilize the prosthesis proficiently, and demand operator concentration even after training. Prosthesis control would be easier and more natural if control muscles were to use their own muscle use patterns to mimic the desired arm movements One way of achieving the goal of 'natural control' is a pattern recognition prosthetic control scheme.

Pattern recogntition ascribes class membership to a piece of data or data vector whose class is unknown by using a priori knowledge of the class of other like data vectors. In the case of prosthesis control, a pattern recognition based control scheme would attempt to identify a prosthesis action desired by the operator by collecting various EMG signal parameters from the synergistic muscles and finding for which arm movement similar signal characteristics have been found. Of course such a scheme requires a priori knowledge of arm action and EMG signal characteristics to make an intelligent decision. The optimal methods of making a decison given a priori knowledge of the class of data vectors, and assuming Gaussian distributions of data vectors in one class are linear or quadratic discriminant functions. A linear discriminant function assumes that the variance of the data vectors of all classes is identical, while a quadratic discriminant feature does not make this assumption.

1.4.1 Synergistic Muscles

The pattern recognition approach to prosthesis control is based on the fact that certain muscle assist in arm action, but are not the prime movers. These muscles are known as synergistic muscles. In above elbow amputation, the prime movers for certain upper limb actions may be destroyed while leaving the synergistic muscles intact. If the amputee tries to initiate an action that is impossible due to limb loss, the synergistic muscles will respond in a normal manner.

The function of synergistic muscles is to balance and help control the muscle that is the prime mover in a specific action. For instance, the muscles of the shoulder are synergistic to elbow flexion, where the biceps is the prime mover. Synergistic muscle action is controlled unconciously by the brain. The control of synergistic action continues even after amputation and loss of prime functions.

By studying the EMG signals from synergistic muscles it is possible to establish patterns linking specific arm movements with synergistic muscle EMG signal characteristics. If the patterns linking EMG signal characteristics and arm movements are well enough defined, arm movements can be recognized by observing the EMG signal characteristics. This type of prosthesis control uses the body's own muscle activation patterns to control prosthesis movements that are completely native to the synergistic muscles own actions. In this way, pattern recognition based prosthesis control is much more natural than other myoeletric based prosthesis control schemes. To initiate an action, the operator simply thinks of completing the action as if the operator still had a functioning complete limb. The synergistic muscles respond as they would in a person with an arm, and the appropriate action is initiated by the prosthesis.

1.4.2 Previous Studies

Pattern recognition based prosthesis control was pioneered by Wirta at the V.A. Hospital of Philadelphia[5]. Later studies were conducted by Saridis[6], Brown[7], Dening[8], and others[9]. Saridis used two channels of EMG data to build a classifier. The data was collected from a pair of electrodes spaced evenly around the upper arm. Brown used four electrode pairs situated over the biceps, triceps, anterior deltoid, and posterior deltoid, as well as four electrode pairs spaced evenly around the upper arm for his study. Saridis used the EMG signal variance, zero crossings, and several higher moments of the EMG signal as components of his feature vector, while Brown used variance, zero crossings, and the absolute value of the third moment as features.

Chapter II
EXPERIMENTAL DESIGN

2.1 GOAL

The goal of this thesis was to investigate two aspects of pattern recognition based prosthetic control: electrode placement and feature extraction.

Electrode placement was studied in order to find the placement for electrodes which would lead to the best classifier. Feature extraction was studied in order to find a feature of the EMG data that would add good discriminant power to the classifier. In the past almost all data used in pattern recognition based studies used data that was statistically related to the distribution of the amplitude of the EMG signal[10]. Unfortunately, the EMG signal is well modeled as a Gaussian noise process[11]. This means statistically only the mean and variance of the signal are independent, with all higher moments becoming zero or dependent on the second moment[12]. Because of this dependence, higher moments add little information to an EMG signal based classifier.

2.2 <u>ELECTRODE PLACEMENT</u>

In order to find the best combination of muscle electrode sites data from eight electrode sites was collected. The first four sites were chosen to be equiaxial about the mid-point of the upper arm. The second four channels were defined as being place equiaxially about the shoulder.

The electrode sites were chosen to extract the maximum synergistic information from the muscles in the upper arm and shoulder. As well, in above elbow amputation, there is usually remnant musculature remaining, especially of the larger muscles such as the biceps and triceps. Of course, musculature is vital to the practical application of a limb controller. The electrode sights were also chosen so that past studies could be replecated.

Care was taken during electrode application so that electrode placement was as standard as possible. The most well defined electrode positions over the biceps and triceps were applied first. Following this, the electrodes between the biceps and triceps were applied, midway between each of the first channels. The electrodes over the deltoid were applied so that all were equiaxial about the shoulder. The channel numbers assigned to the various electrode sites are shown in Figure 1.

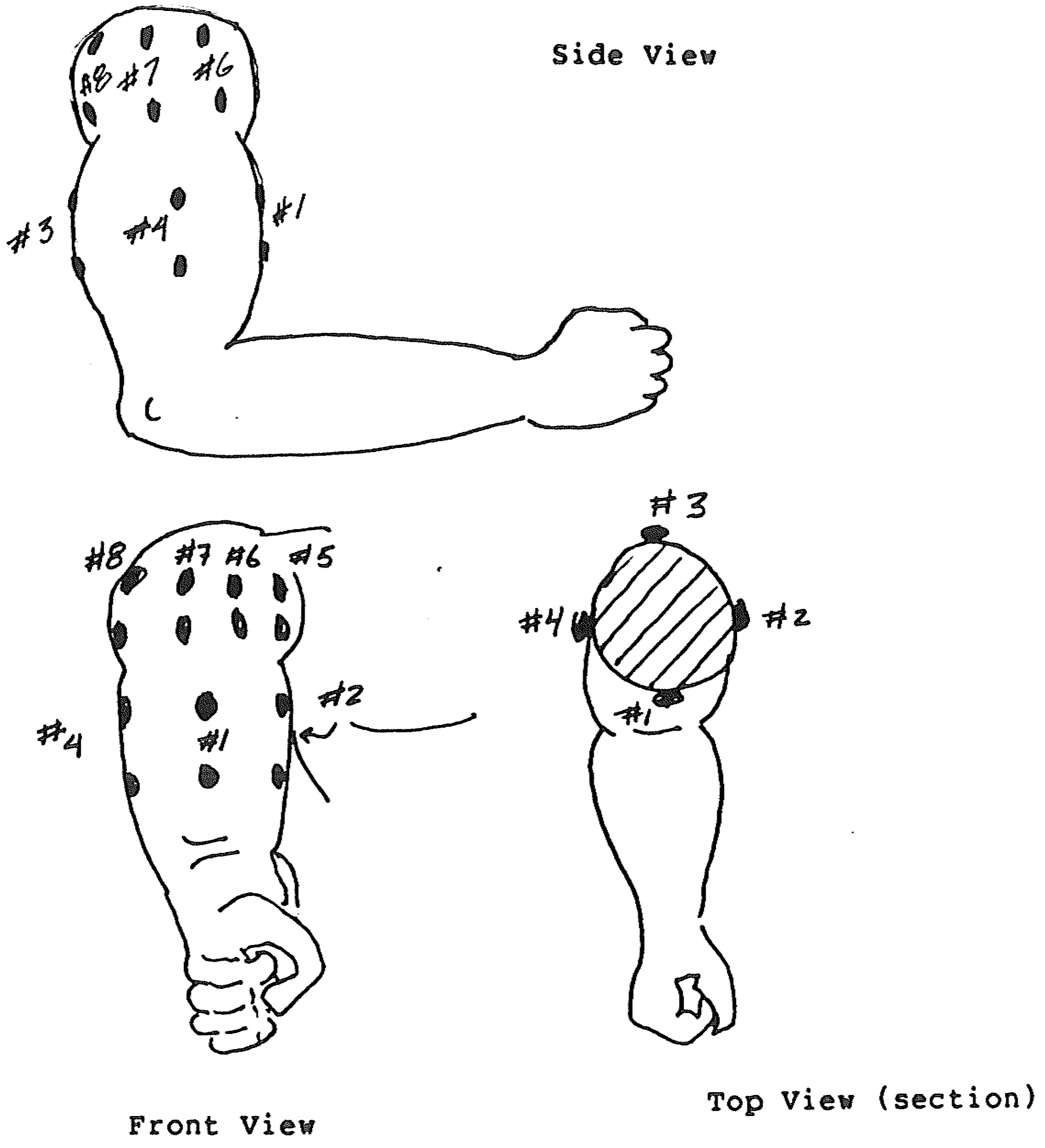


Figure 1: Electrode Positions

2.3 CLASSIFIER FEATURES

It was decided to use EMG variance and zero crossings as features, since these parameters gave good results in both Saridis' and Browns studies. A third feature, muscle activation delay time (MADT), was chosen as a measure to represent muscle sequencing in arm movement. MADT was defined as the time elapsed between the initiation of muscle action and the achievment of a final steady state EMG variance. It was hypothesized that the sequence of muscle activation in the upper arm and shoulder would be different for different actions. If the hypothesis were correct, muscle sequence information would be an important discriminant feature. As well, muscle activation delay time is indepedent of EMG signal parameters, being a result of the co-ordinating. activities of the brain.

After initial processing the variance and MADT data were normalized. The variance data was normalized in order to remove errors induced by amplifier gain changes between channels. MADT data was normalized to remove the effect of the reaction time of the experimental subject. The process of normalization is defined in a following section.

2.4 UPPER LIMB ACTIONS

The six actions chosen to be classified were: humeral rotation medially (towards the body), humeral rotation laterally (away from the body), elbow flexion, elbow extension, wrist pronation, and wrist supination. The actions were chosen for their usefulness in everyday activities as well as the desirability of having a prosthesis re-create these actions controllably. During the experiment each action would be repeated ten times. The rest state, measured with the arm in the standard position, would also be repeated ten times, giving a total of 70 feature vectors for each subject. Table 1 lists the muscles involved in all actions[13].

TABLE 1

Muscles Involved in Upper Limb Actions

Humeral Rotation Medially

1. Pectoralis Major
2. Latissimus Dorsi
3. Teres Major
4. Subscapularis
5. Anterior Fibres of Deltoid

Humeral Rotation Laterally

1. Infraspinatus
2. Teres Minor
3. Deltoid (Posterior)

Elbow Flexion

1. Brachialis
2. Biceps
3. Brachioradialis

Table 1 cont.

Elbow Extension

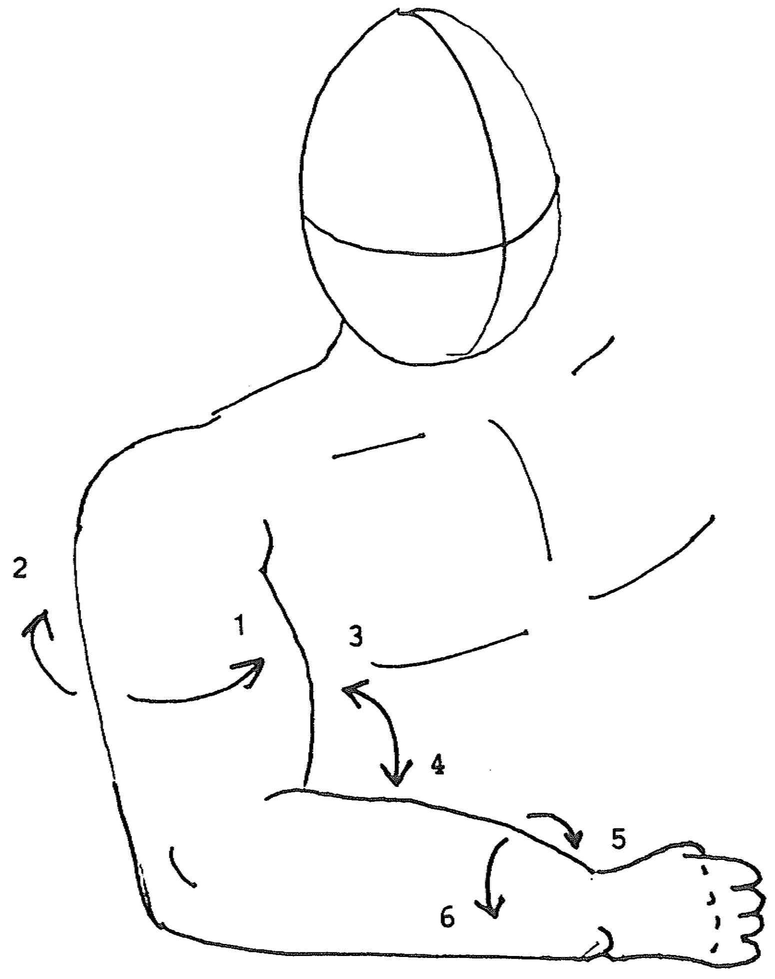
1. Triceps
2. Anconeus

Wrist Pronation

1. Pronator Quadratus
2. Pronator Teres

Wrist Supination

1. Supinator
2. Biceps



1: Humeral Rotation Medially

2: Humeral Rotation Laterally

3: Elbow Flexion

4: Elbow Extension

5: Wrist Pronation

6: Wrist Supination

Arm in Standard Position

Figure 2: Arm Actions Defined

It should be noted that the experiment was initially designed to find a classifier design which could be used most succesfully with above-elbow amputations. Assuming this design criteria obviated the need for studying the humeral rotation actions. However, in a quest for a more generalized classifier these actions were included in the experiment. Nine subjects were studied during the experiment. All were male, ranging in age from 18 to 45. While the handedness of the subjects was not used as a selection criteria, all actions were performed with the right hand.

2.5 <u>EXPERIMENTAL REPRODUCIBILITY</u>

Experimental reproducibility was a very important factor in designing the experiment. The first step in ensuring the reproducibility of the experiment was to ensure the subjects arm was positioned the same way throughout the experiment, so that all arm actions had the same resting position. The arm position chosen was one with the upper arm positioned next to the body, the elbow flexed ninety degrees, the forearm rotated so that the back of the hand is facing away from the body, and the wrist in the neutral position. This arm position is also known as the standard position. With the arm in the standard position all actions could be initiated.

To reduce secondary movement in the subject's arm during an action, a motion isolation jig was used. The motion isolation jig consisted of a handle attached by a screw thread to a bar, which in turn was attached to a table mounted support. By using the motion isolation jig, arm movement was reduced to a minimum.

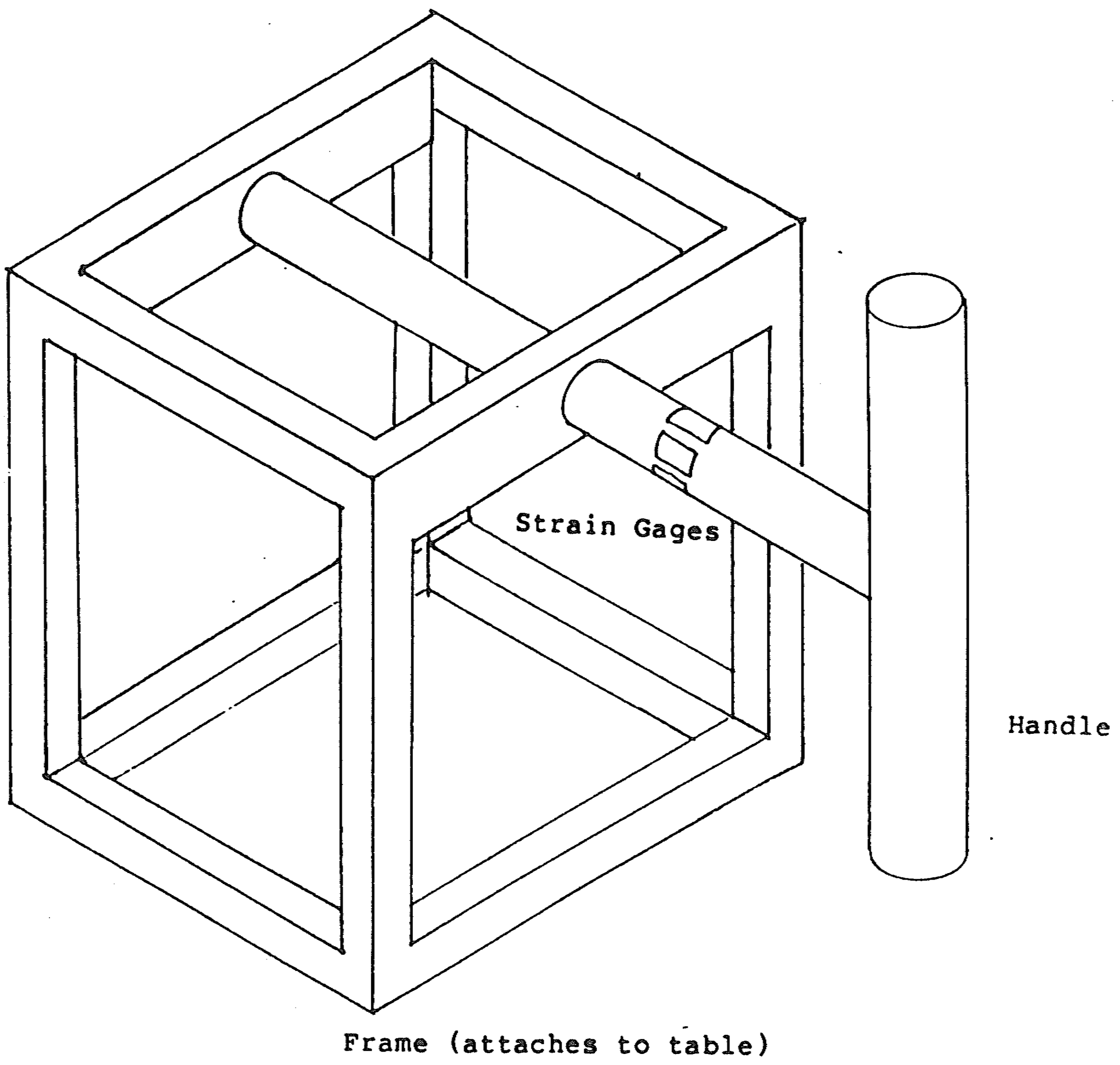


Figure 3: Motion Isolation Jig

Strain gages were mounted on the bar of the motion isolation jig to measure the forces exerted by a subject during an action. The strain gage signal was amplified and displayed as a measure of force by the deflection of a trace on an oscilliscope. By observing the oscilliscope during an action the force produced by the subject could be controlled and reproduced over multiple trials.

By using isometric arm movements, a standard arm position, and controlling force production accurate and reproducible trials were achieved for all subjects.

2.6 <u>DATA COLLECTION</u>

Every trial (or run) was composed of two parts. The first part of the trial was the rest state. During the rest state data was recorded but the subject was not performing any action. The rest state lasted 250 msec. The second part of the trial was the active state, which was initiated by a beep. The beep, lasting 250 msec. was sounded to indicate to the subject to initiate an action. The active state was taken to start at the begining of the beep, since an action could be initiated before the end of the beep, but only by a subject with exceptional reflexes. The active period lasted 1750 msec., thus data collection lasted a total of two seconds for each trial.

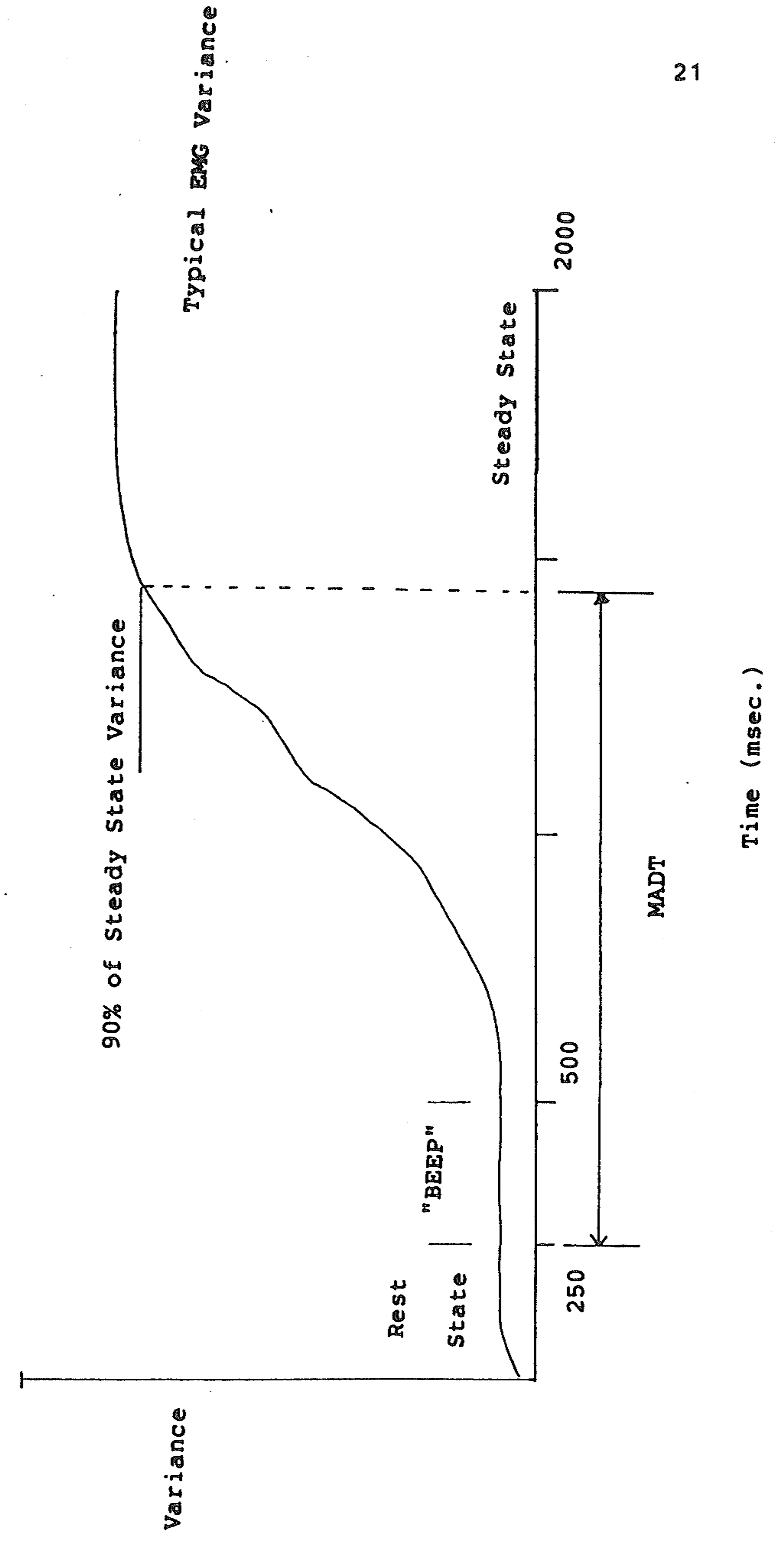


Figure 4: Graphical Interpetation of Experimental Trial

2.6.1 Electrode Attachment

The EMG signals from all eight sites were collected by pairs of 5 mm. silver semi conical electrodes (Grass Instruments E6-SH). The inner surfaces of the electrodes were coated with electrode paste (Medi Trace EEG-Sol 14-016) to improve conductivity. The electrodes were attached to the skin using surgical tape (3M Bioderm) which was found to be well suited to this purpose, showing both good adhesion and flexibility.

2.6.2 Amplification and Filtering

The raw EMG signals from the electrodes were passed through an amplifier-filter section. The amplifier gain was set at 5000, with the high pass filter 3 dB point set to 20 Hz. and the low pass filter 3 dB point set to 500 Hz. The filter settings reflected the need to reduce low frequency noise caused by electrode movement and to avoid aliasing error due to digitization.

After initial amplification and filtering, the EMG signals were passed through a 60 Hz notch filter to remove unwanted 60 Hz noise. The notch filter had a Q of 30, with 40 dB rejection at the center frequency. The notch filter worked extremely well, and after installation 60 Hz noise was virtually nonexistent, without degradation of the EMG signal. It was important that the 60 Hz filter have a sharp notch, since previous studies by Scott[14] have shown that the maximum spectral power density of the EMG signal occurs around 60 Hz. A schematic diagram of the filter is included in the appendices[15].

2.6.3 Digitization

The final step in acquiring the EMG signal was digitization. To digitize all eight channels concurrently, an IBM-XT clone equipped with a 16 channel analog-to-digital conversion board was used. The analog-to-digital conversion board used four analog-to-digital converter chips and four four-to-one analog mulitplexers to achieve 16 channel capability. Of course, only eight channels of the sixteen available channels were used. Because only four channels could be sampled concurrently, an interleaved sampling scheme was used, in which the first four channels were sampled, followed by the second four channels. The sampling rate for any channel was 1000 Hz, making the sampling rate of the A/D chips 2000 Hz, still well within the capabilities of the A/D board.

2.7 SIGNAL PROCESSING

2.7.1 Scaling

The input signal to the A/D board had range of plus/minus one volt, and was digitized to a value of 0 to 4095, with minus one volt corresponding to 0 and plus one volt corresponding to 4095. Two bytes were used to store the sample value. After each trial, the sample values were adjusted so that the range of values would be -2048 to 2047, with a zero volt input corresponding to a sample value of zero. The gain of the initial amplifiers was adjusted so that maximum signal strength would be very close to plus/minus one volt.

Every trial produced four thousand bytes of data per channel, or 32 Kbytes per trial. All samples were first stored in RAM, where they could be studied by means of a program that would display the samples graphically. The graphical display of the EMG data was very useful in highlighting amplifier and filter misfunction, since noise in the sampled data could be observed in relation to the EMG signals[16]. After review, the data was stored on hard disk. It should be noted that due to an error in disk storage, the data for three rest state trials of one subject are identical. However, this is not seen as having an undue effect on the overall experiment.

2.7.2 Data Storage

During a set of trials, the subject would perform each action ten times. Including processed data, each set of trials would produce 2.46 megabytes of data. In order to archive all data and save disk space, after processing all data would be transferred from the hard disk to a streaming tape backup system.

2.8 PARAMETER EXTRACTION

After digitization and storage of the EMG data, the signal parameters were extracted. Variance, zero crossing, and muscle activation delay time parameters were extracted for each channel for all trials.

Both variance and zero crossings were calculated using the last 50 msec. of EMG data, that is, the last 50 samples of EMG data. The calculations were made at this point to ensure that the the EMG signal was at a steady state, with force production as constant as possible.

Variance was calculated over the 50 sample window using the formula:

$$\frac{1}{50}\sum(x - \bar{x})^2$$

The calculation of variance was simplified by the assumption that the EMG signal has a zero mean. The zero crossing parameter was calculated by simply counting the number of sign changes in the 50 sample period.

The muscle activation delay time was calculated by finding the time required for the EMG variance to rise from the rest state value to 90% of its steady state value. When calculating the muscle activation delay time, variance for the period between the initiation of action and the steady state had to be calculated. To calculate the variance a 50 msec. moving window estimator was used. The variance of the EMG signal during some point in time was calculated by finding the variance of the a 50 sample window centered on the point. The window size was chosen heuristically. The 50 sample size was adequate for fairly accurate estimation of the average variance, while tracking changes in variance well. A larger window size also increased the calculation time considerably.

Since the subject was required to respond to an audible signal, the time delay between the signal and the point where 90% of the steady state variance was reached was also a reflection of of the subject's reaction time. To reduce the effect of reaction time the MADT data was normalized.

2.9 <u>NORMALIZATION</u>

2.9.1 <u>MADT Normalization</u>

MADT data normalization was straight forward. The muscle activation delay time was first found for all eight channels. The shortest muscle activation delay time was found and given the value zero, and all other muscle activation delay times were found relative to the shortest by subtracting the value of the shortest from the other times.

2.9.2 Variance Normalization

The variance data was normalized in order to remove inter-subject differences and errors caused by amplifier gain changes. The first step in normalizing variance was recording the EMG signals for a maximum force output for each action (as measured by the strain gages) by asking the subjects to exert as much force as possible during the actions. The maximum EMG data was processed to find the steady state variance for all actions for each channel, with the largest variance of all the actions for each channel being stored for reference as the maximum variance of that channel. During data collection subjects were asked only to produce approximately 50% of the maximum force output (as measured by the strain gages).

The normalized variance was calculated by finding the trial variance as a percentage of maximum variance. For instance, if the trial variance for some action and channel were 30% of the maximum variance of that channel, the normalized trial variance would have a value of 30. Normalizing the variance data in this way allowed any channel to be gain independent.

After all data had been normalized, it was sorted by channel and stored in channel files. Each channel file contained all normalized parameters, and was in a format that was readable by SAS-PC, a statistical software package that was used for the linear discriminant functions.

Chapter III

CLASSIFIER DESIGN

After all data had been collected, processed,and normalized, it was used to create a linear discriminant classifier. A linear discriminant classifier ascribes a feature or collection of features known as a feature vector to a particular class to which the feature belongs.

3.1 FEATURE VECTORS

Each parameter of every channel of EMG data is considered a feature, and every feature is a component of a feature vector. A feature vector is therefore a collection of features collected during a single trial, and represents a point in feature space. A feature vector may also be thought of as an observation. Using eight channels with three parameters per channel makes the feature space 24 dimensional.

By using a collection of vectors whose classification is known a priori, a mean class vector and a class distribution covariance matrix are found.

Assuming that all class distributions are equal has the effect of linearizing the between class decision boundaries. If each class is assumed to have a unique distribution the decision boundaries will have a quadratic form. A more in depth review of linear classifiers can be found in reference by Tou and Gonzalez[17], and Andrews[18]. A brief review of the linear classifiers is found in Appendix E.

3.2 <u>CLASSIFIER TESTS</u>

All discriminant functions for the EMG data were found by using the SAS/STAT and SAS/PC software packages running on an IBM/PC clone[19]. By having eight channels of data available, various combinations of channels and features could be studied. In order to test the effect of electrode placement on classifier performance several combinations of channel data were used. The effect of MADT on classifier performance was investigated by comparing the performance of otherwise identical classifiers with and without the MADT data.

3.2.1 <u>Comparison to Previous Studies</u>

In order to test the performance of the classifier against previous experimental data [Brown], the muscle activation delay time data was not included when finding the first and second discriminant functions. Brown utilized two electrode placement schemes in his study. Brown's first scheme placed electrode pairs on the biceps, triceps, anterior deltoid, and posterior deltoid. To replicate this scheme, variance and zero crossing data from channels 1,3,5, and 7 were used to build the classifier. Brown also used four electrode sites equally distributed axially around the upper arm, so in building the second classifier, the variance and zero crossing data from the corresponding channels (1,2,3,4) were used. To further replicate Brown's experiment, for the first and second classifier designs a pooled covariance matrix was specified in order to obtain a linear classifier.

3.2.2 <u>Classification using 8 Channels of Data</u>

The third classifier was designed using data from all eight channels. All variance, zero crossing, and MADT data was included in this classifier. The effect of MADT data on classification rates was investigated by the fourth classifier, whose design was essentially the same as the third classifier, excepting the exclusion of the MADT data from all channels.

3.2.3 <u>Classification using 6 Channels of Data</u>

As an intermidiate step between four and eight channel classifiers, a six channel classifier was developed. The six channels were chosen by pooling the feature data from all subjects and analyzing the discriminant power of the features by means of a stepwise discriminant function. The six channels whose features provided the greatest discriminant power were chosen from the results of the stepwise discriminant function, which listed the features in terms of discriminant power. The channels chosen were 1,3,4,6,7, and 8.(Table 2) Again, the classifier was designed with and without the MADT data.

TABLE 2

Results of Stepwise Discriminant Analysis

Rank	Feature	Channel
1	Variance	1
2	Variance	3
3	Zero Crossings	3
4	Variance	7
5	Zero Crossings	6
6	Variance	8
7	Variance	4
8	Variance	5
9	Zero Crossings	2
10	Zero Crossings	4
11	Zero Crossings	8
12	Zero Crossings	7
13	MADT	1
14	Zero Crossings	1
15	Variance	6
16	Zero Crossings	5

Table 2 - cont.

17	MADT	8
18	MADT	5
19	Variance	2
20	MADT	3
21	MADT	4
22	MADT	7
23	MADT	6
24	MADT	2

3.2.4 Half and Half Test

In all four primary classifier designs, the data that was used to design the classifier was also used to test the classifier. This may have biased the test results towards a overly optimistic classification rate. In order to test the classifiers rigorously, a half and half test was performed. In a half-and-half test half the data is used to find the classifier design and the other half of the data is used to test the classifier[20]. All results obtained by using the various classifiers can be found in chapter 4.

3.3 MADT DATA CLASSIFICATION

In order to further test the discriminant function of the MADT data, a classifier was designed using only the eight channels of MADT data. Unfortunately, it was found that when using only MADT data in the classifier design, classification was only 46% - 69% correct, which was unacceptably low. The stepwise linear discriminant function confirmed the poor discriminant power of the MADT data. It was theorized that if the MADT data was transformed from the time domain into the discrete sequence domain, basically ignoring the actual time information in favor of the sequence information, the discriminant power of the data would be increased.

3.3.1 Muscle Activation Sequence Discrimination

To transform the MADT data from the time domain to sequence domain, all activation times for an observation were found. The sequence of activation times was then extracted and stored as a muscle activation sequence (MAS) vector. The result of all processing was a separate MAS vector for every observation. Each MAS vector had eight components, each component representing a channel. The value of MAS vector component reflected the sequence in which the corresponding channel had been activated with respect to all other channels during one observation. For example, if channel two had the smallest muscle activation delay time and channel one had the longest muscle activation delay time of all channels during a given observation, the second component of the MAS vector, representing channel two, would have a value of 1, while the first component of the MAS vector would have a value of 8, since channel one had the longest muscle activation delay time and so would be last in the sequence of muscle activation delay times of all channels. If two channels had identical muscle activation delay times, the lower number channels would be given the lower activation sequence number.

In order to test the discriminatory power of the MAS data a new classifier was developed. Since the MAS data was discrete, using a linear or quadratic classifier was not deemd appropriate, so a nearest neighbor classifier was developed.

To classify any feature vector, the relative distance between the vector and a class has to be known. A function of distance between vectors is known as a metric. The metric chosen for measuring the distance between MAS vectors is known variously as the city block, taxi, or Manhattan metric. The taxi metric, Dxy, between two vectors X and Y, with components (x1,x2,,...xn) and (y1,,y2,...yn)

$$Dxy = \sum_{i=1}^{n} |(xi - yi)|$$

If X and Y are identical, Dxy will be zero.

A MAS vector was classified by finding the sum of the distances between it and all members of a class. The class-sum was found for all classes and the vector was classified as belonging to the class which had the lowest class-sum associated with it. Unfortunately, the results obtained from this classifier were as poor as those obtained from the linear discriminant function. A discrete Bayesian approach to classifying the MAS vector was also proposed, but it was found that the feature space was too sparsely populated to provide meaningfull results, even for a reduced vector length.

Chapter IV

RESULTS AND DISCUSSION

4.1 FOUR CHANNEL CLASSIFIER RESULTS

To compare the initial results of the discriminant function with previous results, the first classifiers were built using only four channels of EMG data and no MADT data. Using a reduced feature space allowed comparison to Brown's data. The first classifier was built using only the normalized variance and zero crossing information from channels 1,3,5, and 7. The results are shown in comparison to Brown in Table 3. The average correct classification is quite high, 92.5%, which compares favorably with Brown. The second classifier was built using the normalized variance and zero crossing data from the upper arm channels (1,2,3,and 4). In this case the correct classification rate was slightly lower at 90%. The results for the second classifier are shown in Table 4.

TABLE 3

Average Correct Classification Rate Ch. 1,3,5,7, No MADT

Subject	Avg. Classification (%)
ABK	86
EDS	98
FAP	90
JAW	98
PAK	94
REZ	93
RJP	94
RWW	87
B1*	78
B2*	72
B3*	84
Average (excluding B1-B3)	92.5

* = Data from Brown

TABLE 4

Average Classification Rate Ch. 1,2,3,4, No MADT

Subject	Avg. Classification (%)
ABK	90
EDS	97
FAP	94
JAW	97
PAK	91
REZ	86
RJP	77
RWW	89
B1*	66
B2*	73
B3*	73
Average (excluding B1-B3)	90

4.2 EIGHT CHANNEL CLASSIFIER RESULTS

After initial evaluation, a classifier was built using all available data. Using all eight channels and the normalized variance, zero crossings, and MADT data made the feature space 24 dimensional. The average correct classification rate for the complete data classifier was 99%. This figure may be somewhat misleading, for reasons that will be addressed in a following section, but the result is excellent nonetheless, and an improvement on the four channel classifiers. The results are shown in Table 5. Table 6 shows the results obtained from a classifier built using eight channels, but without the MADT data. Again results are excellent, with an average correct classification rate of 98.6%.

TABLE 5

Average Classification Rate, All Channels, with MADT

Subject	Avg. Classification (%)
ABK	100
EDS	100
FAP	97
JAW	100
PAK	100
REZ	98
RJP	100
RWW	97
Average	99

TABLE 6

Average Classification Rate, All Channels, without MADT

Subject	Avg. Classification (%)
ABK	100
EDS	100
FAP	100
JAW	96
PAK	97
REZ	99
RJP	97
RWW	100
Average	98.6

4.3 SIX CHANNEL CLASSIFIER RESULTS

Tables 7 and 8 show the results obtained from using the six channels with the greatest discriminant power in the classifier design. The classifier design which included MADT data was marginally better than the classifier which did not. However, both designs proved to be more accurate than either four channel classifiers, and nearly as accurate as the eight channel classifiers. These results would indicate that channel selection is more important in increasing the classification rate than the inclusion of MADT data in the classifier design.

TABLE 7

Average Classification Rate, 6 Channels, with MADT

Subject	Avg. Classification (%)
ABK	94
EDS	100
FAP	94
JAW	99
PAK	99
REZ	99
RJP	99
RWW	97
Average	97.6

TABLE 8

Average Classification Rate, 6 Channels, No MADT

Subject	Avg. Classification (%)
ABK	91
EDS	100
FAP	90
JAW	97
PAK	97
REZ	97
RJP	94
RWW	96
Average	95.3

4.4 HALF AND HALF TEST

In order to more rigorously test classifier design, it was decided to perform a 'half and half' test. To this end one subject performed 20 trials for each action, instead of 10 as the other subjects. The first set of ten trials were used to build the classifier, while the last set were used as test vectors to test the classifier. Unfortunately, a disk failure destroyed the last trial, so that only 9 sets of trials were classified. The result are shown in Table 9. The result of the half and half classifier was 73%. The actions are described as follows:

Action #	Description
1	Humeral Rotation Laterally
2	Humeral Rotation Medially
3	Elbow Flexion
4	Elbow Extension
5	Wrist Supination
6	Wrist Pronation
7	Rest State

It should be noted that the classifier for the half and half test was based on six channels of data because of the sudden and catastrophic failure of two channels of amplification. The channels used in the classifier design were 1, 2, 3, 4, 5, and 7. The channel configuration was chosen so that both four channel classifiers could be designed from the data.

The half and half test was re-run, using only the variance and zero crossing information. The results, shown in Table 10, were very similar to the previous results, actually better by 3% (76%). It is evident from the correct classification of all actions except the forearm pronation supination misclassification that without the wrist supination-pronation errors, results could have been very comprable to the results obtained from the eight subjects in Tables 5,6,7,and 8. The reason for the misclassifications is not clear, and unfortunately confuses the issue of classifier validity.

TABLE 9

Classification Table, Half and Half Test, with MADT

	To Action						
From Action	1	2	3	4	5	6	7
1	7	1	0	0	1	0	0
2	0	7	0	0	0	2	0
3	1	1	7	0	0	0	0
4	0	0	0	8	0	1	0
5	7	0	0	0	1	0	1
6	0	2	0	0	0	7	0
7	0	0	0	0	0	0	9
Total	15	11	7	8	2	10	10

Average correct classification = 73%

TABLE 10

Classification Table, Half and Half Test, No MADT

From Action	To Action 1	2	3	4	5	6	7
1	8	0	0	0	0	1	0
2	0	7	0	1	0	1	0
3	0	0	9	0	0	0	0
4	0	0	0	9	0	0	0
5	5	0	0	0	4	0	0
6	0	4	0	3	0	2	0
7	0	0	0	0	0	0	9
Total	13	11	9	13	4	4	9

Average correct classification = 76%

4.5 MADT CLASSIFICATION

To test the discriminant power of the MADT data, a classifier was designed using only the MADT data from all eight channels. The results are poor, as shown in Table 11. It is obvious that MADT data has poorer discriminant power than the combination of variance and zero crossing data.

TABLE 11

Average Classification Rate, 8 Channels, Only MADT

Subject	Avg. Classification (%)
ABK	50
EDS	47
FAP	46
JAW	56
PAK	40
REZ	49
RJP	40
RWW	69
Average	49.6

TABLE 12

Average Classification Rate, MAS Vector Classifier

Subject	Avg. Classification (%)
ABK	49
EDS	46
FAP	57
JAW	44
PAK	44
REZ	47
RJP	50
RWW	50
Average	48.4

4.6 MAS CLASSIFICATION

Table 12 shows the results obtained when the muscle activation sequence vectors were classified by a nearest class classifier. These results are slightly poorer than those obtained from the MADT data. Again, results indicate that this parameter could not be the sole feature in a discriminant function.

4.7 CLASSIFIER CONSIDERATIONS

Classification rates approaching 100% can be attributed to several factors in the experiment. First, classification rates were excellent because great care was taken to ensure experimental reproducibility. Because all trials were isometric and equal in force reproducibility was enhanced. Another contributing factor in the success of the experiment was normalization of the variance data. All other things being equal, normalization increased classification rates.

Secondly, the method of evaluating the classifier design contributed to the classifier success rate. Using the same data to both design and evaluate a classifier may produce an overly optimistic view of classifier performance when the sample size is small. Unfortunately the sample size was limited by the ability to collect and store large volumes of data. As well, individual classifiers were designed for each subject. Using a pooled data base to design classifiers would have allowed the use of larger number of samples but was not considered appropriate. If ever implemented in a prosthetic device, a linear classifier would be tailored to suit the individual. Since there is no need for a universal classifier it would be ludicrous to demand a classifier to perform accurate classification with any data but the data it was designed for. The results of the half and half test show that for all but one action (wrist supination) classification is good. Despite the fact that the wrist supination data was not well classified, the half and half classifier results indicate to some extent the fundamental soundness of all classifier designs and results.

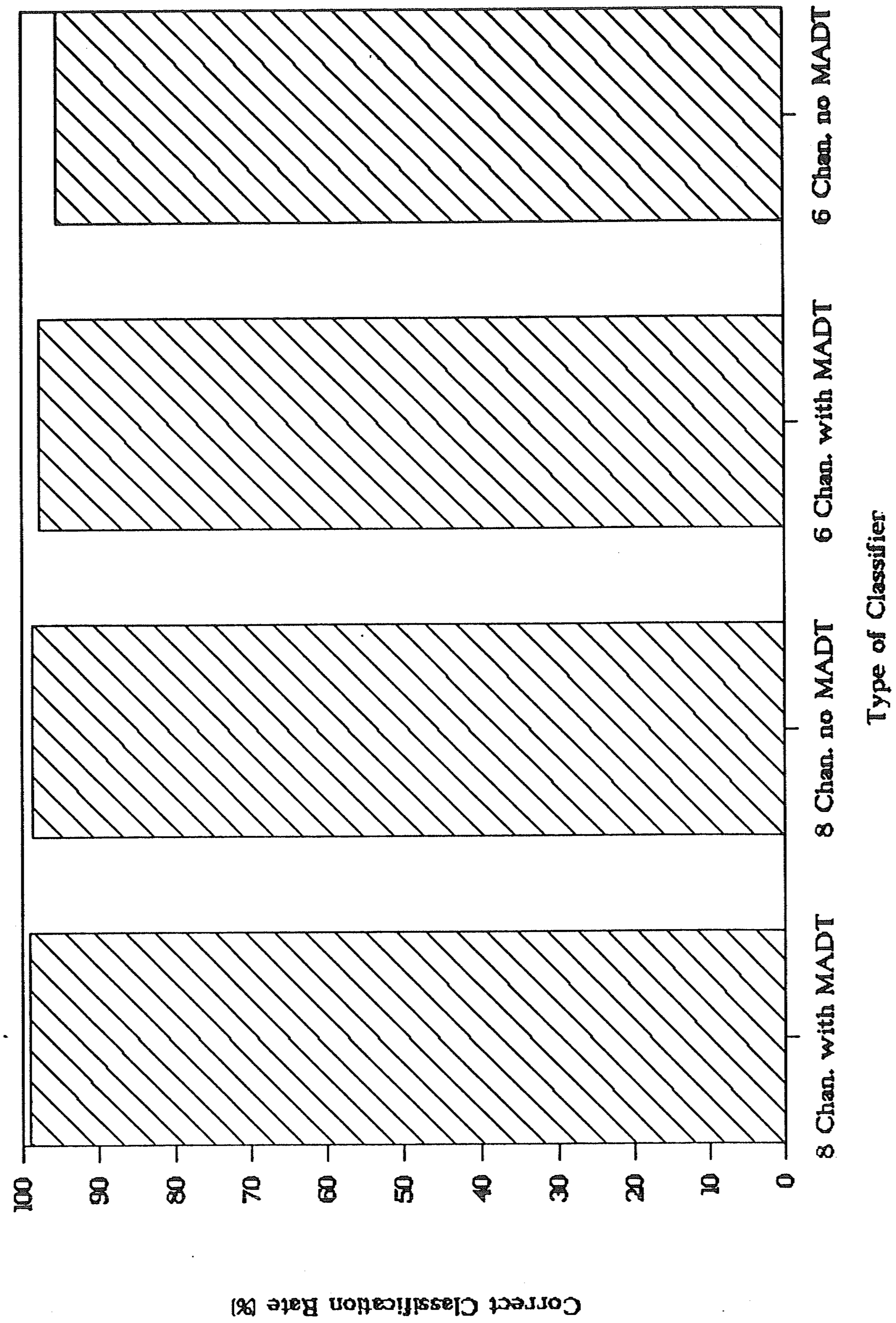


Figure 5: Eight and Six Channel Classifier Success Rates

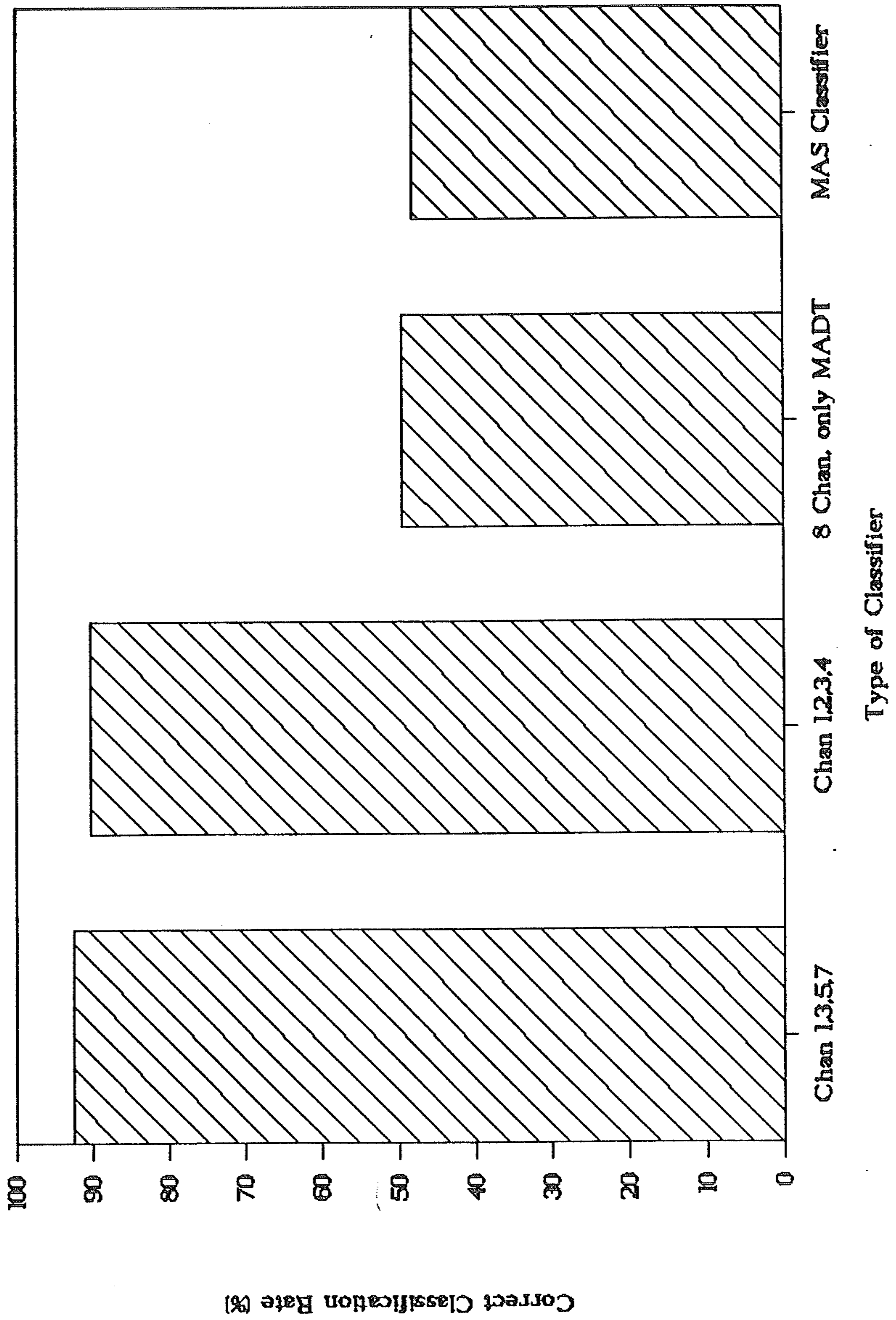


Figure 6: Other Classifier Success Rates

4.8 MADT DATA

MADT data played a seemingly insignificant role in classifier success. Classifier success rates with and without MADT data were not significantly different, and classification using only MADT data produces relatively poor results. MAS vectors used with a discrete classifier also showed poor results, but this may be attributed to the sparse population of the feature space. With a sequence vector length of eight, the feature space is has 8 factorial discrete points, while only 70 MAS vectors were produced by each subject. Obviously if muscle activation sequencing is to be studied in more depth a much larger amount of data must be collected solely for this purpose. It is the author's belief that owing to the success of classifiers utilizing EMG signal parameters as features, MAS and MADT features will probably be used only as an ancilliary feature.

Chapter V

CONCLUSIONS

5.1 ELECTRODE PLACEMENT

The difference in classification rates between the four, six, and eight channel classifiers shows that increasing the number of channels of EMG data in classifier design increases the correct classification rate. The best average classification rate of the four channel classifiers was 92.5%, the best six channel classification rate was 97.6%, and the best eight channel classification rate was 99%. The difference between the six and four channel classifier is significantly greater than between the six and eight channel classifiers, indicating that the use of the stepwise discriminant function to discern the channels with the greatest discriminant power allowed the six channel classifier to be very nearly as powerful as the eight channel classifier. The results of the Half and Half test confirm the applicability of the classifier design, and also show that not any six channels of data will give the optimal classifier.

5.2 MADT

The use of the stepwise discriminant function and linear discriminant function has shown that MADT data has relatively low but significant discriminatory power. The results, while not good enough to allow MADT to be used as the sole feature in a classifier, are encouraging. The concept of muscle activation sequencing must be explored further to refine the discriminant power of this feature.

5.3 PRACTICAL CONSIDERATIONS

In a practical sense an eight channel classifier does carry with it some inherent drawbacks. The task of amplifying and filtering eight channels of EMG data in the limited space of a prosthesis is daunting. As well, signal digitization, processing, and classification must be carried out in real time, requiring hardware and software that would increase the cost of a prosthesis beyond a resonable level. Finally, fitting six or eight electrode pairs to a prosthesis is difficult, and signal quality from these electrodes is hard to ensure. Even two electrodes cause chafing and pressure sores in modern myoelectric limbs. Perhaps if the basic pattern recognition approach could be meshed with a pre-programmed motion control technique feedback[21] greater succes could be achieved while using fewer channels. However, in order for such an amalgamated approach to succeed, more research into natural patterns of arm movements must be carried out[22].

It should be noted that the results presented can not be directly be compared to results from the five level tracking study as carried out by McKenzie. McKenzie's experimental design dictated fairly rapid changes in muscle activity, making error rates dependent on the co-ordination of the subject. The results presented were obtained in a more controlled environment, leaving some question as to the results which would be obtained in an envrionment similar to McKenzies.

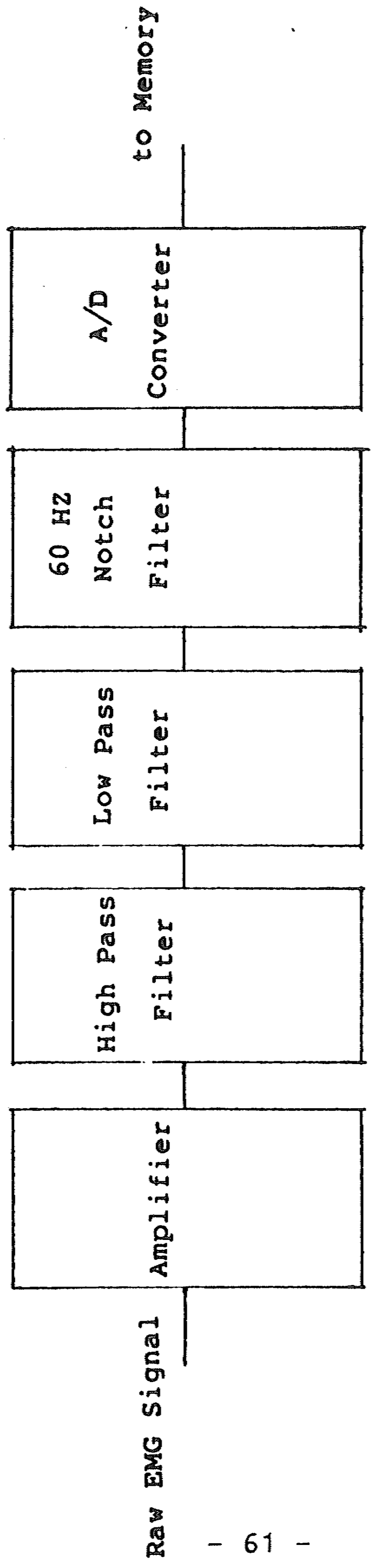
5.4 <u>RECOMMENDATIONS</u>

Electrode placement has been shown to have a significant effect on classifier performance. However, practical considerations limit the number of channels that can be used in a prosthesis. MADT and MAS need to be developed and studied in greater detail. Much more data needs to be collected and studied. Measurement and normalization schemes must be developed and refined. New classifiers must be investigated and tested. There is promise that the intrinsic muscle sequencing patterns of the brain hold information that would make prosthetic control more natural, and perhaps reduce the number of channels that are required for adequate control.

Pattern recognition based classifiers are not easily realizable, if not infeasible. Thought and action must be guided towards the goal of meshing the theoretical and practical aspects of prosthetics and prosthetic control to produce a tool that would benefit a wide variety of the functionally handicapped.

Appendix A

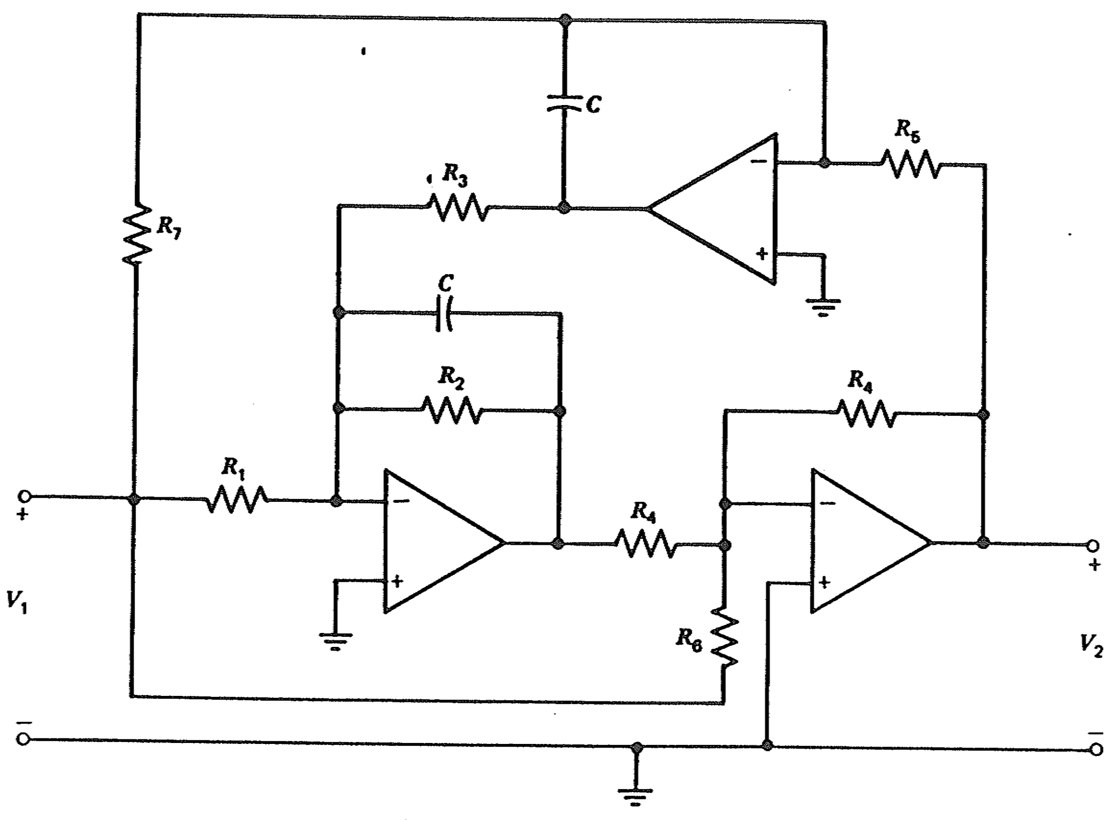
AMPLIFIER BLOCK DIAGRAM



Appendix B

60 HZ FILTER SCHEMATIC

Biquad Band-Reject Filter Design



General circuit

R1, R2 = 40K

R3, R4, R5, R6, R7 = 2.66K

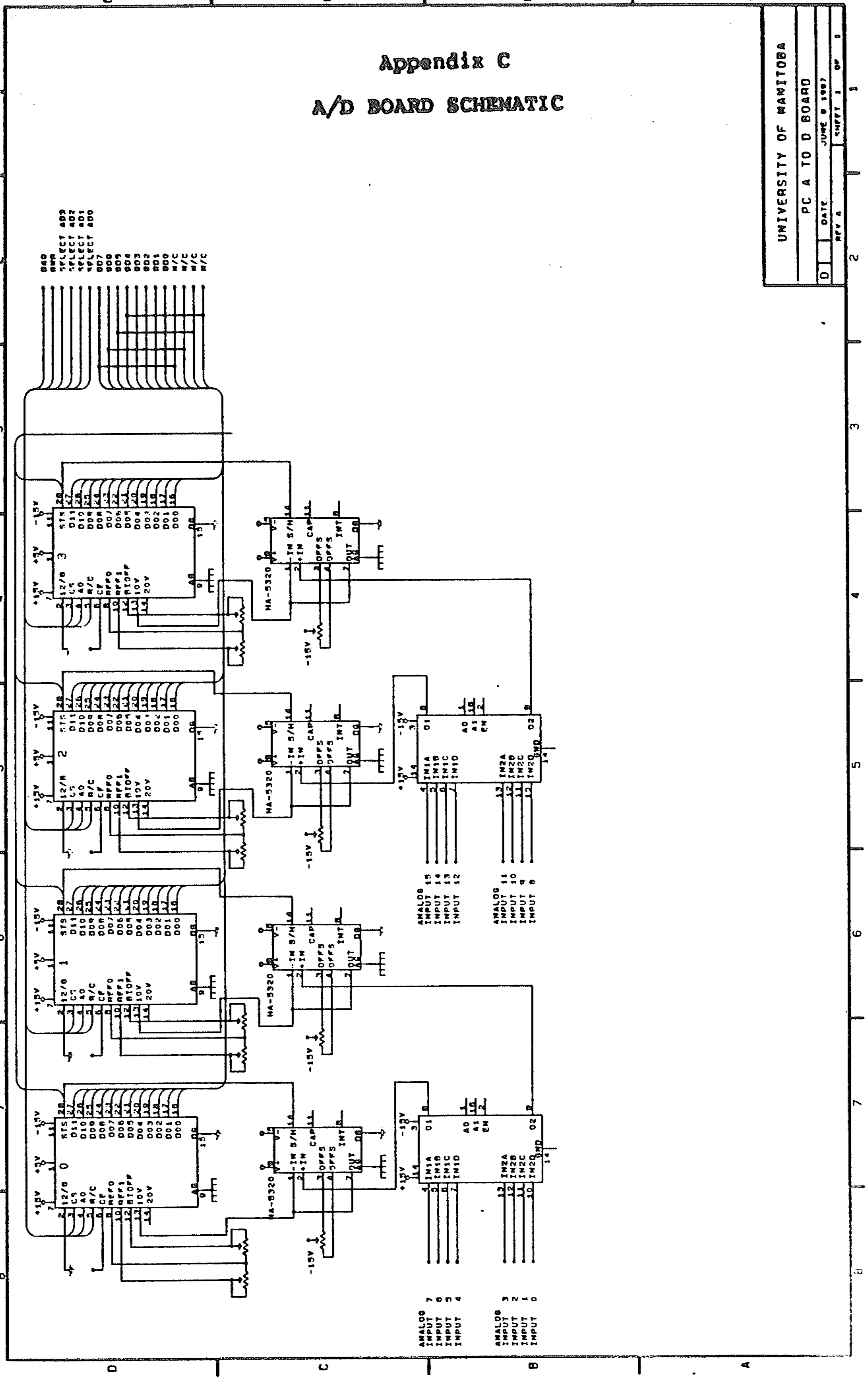
C = 0.01*E-6 F

Q = 37

Fc = 60 HZ

Appendix C

A/D BOARD SCHEMATIC



Appendix D

PROGRAM LISTINGS

```
program atod(input,output);

{this program is designed to run the a/d board and do
8 channel data collection.  Data will be stored in eight arrays
and will then be dumped to disk. }

const
   base = $0100 ;  {mux base}
   PPI  = 97    ;  {PPI address}
   TCB  = $0043 ;  {Timer Control Byte address}
   TLA  = $0042 ;  {Timer Load Adress}
   count = 2000 ;  {number of array elements}

type
   name = string[80];
   string80 = string[80];
   data = array[1..count] of integer;
   realdata = array[1..count] of real;
   datafile = file of data;
   andat = record
           beginvar:real;
           endvar  :real;
           zeros   :integer;
           delta   :integer;
           end;
   varfile = array[1..8] of andat;
   anfile=file of varfile;

var
   filenames:array[1..8] of name;
   varfilename:name;
   tempname:name;
   i,j:integer;
   outfiles:array[1..8] of datafile;
   datain:array[1..8] of data absolute $3000:$0000;
   vardat:array[1..8] of data;
   alldat:varfile;
   allfiles:anfile;
   dee:byte;    {global variable for inline code statements}
   test:integer;

{$I plot.pas}
{*******************************************************************}
{this procedure normalizes the input data from the range 0-4095 to
the range -2047 to 2048

procedure fix_data(var indata:data);

 var
    i:integer;

begin
for i := 1 to 2000 do
   begin
```

```
      indata[i] := indata[i] - 2048;
      if (indata[i] < -2048) or (indata[i] > 2048) then
         indata[i] := 0;
      end;
   end;
   {**********************************************************************}

   {this procedure allows a scaling factor to be used in the screen display
   of data}

   procedure scale(var datain:data;var scale_factor:integer);

   var
      i:integer;
      max:integer;
      min:integer;

   begin
   max := 0;
   min := 0;
   for i:= 1 to 2000 do
      begin
      if datain[i] > max then
         max := datain[i];
      if datain[i] < min then
         min := datain[i];
      end;
   if max > abs(min) then
      scale_factor := max
   else
      scale_factor := abs(min);
   if scale_factor = 0 then
      scale_factor := 1;
   end;
   {**********************************************************************}

   {this procedure sets up the hardware timer on the computer motherboard
   to allow a 2000 Hz square wave output}

   procedure set_up_timer;

   const
      hi = $02 ;
      lo = $54 ;

   var
      test:byte;

   begin
   test := port[PPI];
   test := test OR $03;
   test := test AND $FD;  {set lo bit hi, 2nd bit lo disable speaker}
   port[PPI] := test;
   port[TCB] := $B6; {set timer square wave , input 2 bytes, lo first}
   port[TLA] := lo; {2000 Hz. square wave }
```

```
port[TLA] := hi;
for i:= 0 to 7 do
   port[i+$0100] := 0;
dee := 0;
end;
{**********************************************************************}

{this procedure is written in machine code for speed sakde.  It
runs the hardware of the A/D board and allows the data to be
stored in higher main memory}

procedure data_aq;
begin
writeln('data collection begins.......');
inline
  ($BF/$00/$00/                    {MOV DI,0}
   $BB/$00/$30/                    {MOV BX,3000}
   $BA/$62/$00/           {TESTLOW  MOV DX,0062}
   $EC/                            {IN AL,DX}
   $24/$20/                        {AND AL, 20}
   $74/$02/                        {JZ TESTHI}
   $EB/$F6/                        {JMP TESTLO}
   $EC/                   {TESTHI   IN AL,DX}
   $24/$20/                        {AND AL,20}
   $74/$FB/                        {JZ TESTHI}
   $A0/DEE/                        {MOV AL,DEE}
   $BA/$FE/$00/                    {MOV DX,00FE}
   $B9/$04/$00/                    {MOV CX,4}
   $42/                   {AGAIN    INC DX}
   $42/                            {INC DX}
   $EE/                            {OUT DX,AL}
   $E2/$FB/                         {LOOP AGAIN}
   $B0/$00/                        {MOV AL.0}
   $BA/$18/$01/                    {MOV DX, 0118}
   $EE/                            {OUT DX,AL}
   $BA/$20/$01/                    {MOV DX,0100}
   $EC/                   {EOC      IN AL,DX}
   $24/$01/                        {AND AL,01}
   $75/$FB/                        {JNZ EOC}
   $BA/$10/$01/                    {MOV DX,0110}
   $8E/$C3/                        {MOV ES,BX}
   $B9/$04/$00/                    {MOV CX,3}
   $EC/                   {NEXTREAD IN AL,DX}
   $8A/$E0/                        {MOV AH,AL}
   $42/                            {INC DX}
   $EC/                            {IN AL,DX}
   $D1/$C8/                        {ROR AX,1}
   $D1/$C8/                        {ROR AX,1}
   $D1/$C8/                        {ROR AX,1}
   $D1/$C8/                        {ROR AX,1}
   $25/$FF/$0F/                    {AND AX, 0FFF}
   $26/$89/$05/                    {MOV ES:[DI],AX}
   $42/                            {INC DX}
   $8C/$C0/                        {MOV AX,ES}
   $05/$fa/$00/                    {ADD AX,00fa}
```

```
    $8E/$C0/                        {MOV ES,AX}
    $E2/$E3/                        {LOOP NEXTREAD}
    $8C/$C0/                        {MOV AX,ES}
    $8C/$C3/                        {MOV BX,ES}
    $2D/$E8/$33/                    {SUB AX,33E8}
    $74/$11/                        {JZ NEXT}
    $B8/$00/$30/                    {MOV AX,3000}
    $8B/$D8/                        {MOV BX,AX}
    $B0/$00/                        {MOV AL,0}
    $A2/DEE/                        {MOV DEE,AL}
    $47/                            {INC DI}
    $47/                            {INC DI}
    $EB/$08/$90/                    {JMP DOWN}
    $EB/$96/                {TRICK2  JMP TESTLOW}
    $B0/$01/                {NEXT     MOV AL,1}
    $A2/DEE/                         {MOV DEE,AL}
    $8B/$C7/                {DOWN     MOV AX,DI}
    $2D/$a0/$0f/                   {SUB AX,0000}
    $7F/$27/                       {JG ENDER}
    $8B/$C7/                       {MOV AX,DI}
    $3D/$f4/$01/                   {CMP AX,500}
    $74/$09/                       {JE TURN_ON}
    $8B/$C7/                       {MOV AX,DI}
    $3D/$E8/$03/                   {CMP AX,1000}
    $74/$0C/                       {JE TURN_OFF}
    $EB/$E2/                {TRICK1 JMP TRICK2}
    $BA/$61/$00/          {TURN_ON  MOV DX,97}
    $EC/                           {IN AX,DX}
    $0D/$03/$00/                   {OR AX,1}
    $EE/                           {OUT DX,AX}
    $EB/$F4/                       {JMP TRICK1}
    $BA/$61/$00/         {TURN_OFF  MOV DX,97}
    $EC/                           {IN AX,DX}
    $0D/$03/$00/                   {OR AX,03}
    $25/$FD/$00/                   {AND AX,$FD}
    $EE/                           {OUT DX,AX}
    $EB/$E7/                       {JMP TRICK1}
    $90);                 {ENDER    NOP}

end;
{*********************************************************************}

{this procedure analyses the data and extracts the variance, zero crossi
and MADT data and passes the values back to the calling routine}

procedure anchan(var input_array:data;
                 var out_stuff  :andat;
                 var out_var    :data);

var
i,j:integer;
sigma:real;
temp:real;
temp2:real;
temp3:real;
```

```
zero:integer;
last:boolean;
this:boolean;
done:boolean;
first_stand_dev:real;
sec_stand_dev:real;
test_stand_dev:real;
delta:real;
ninety_point:real;
tau:integer;

begin
sigma := 0;
{first go to the end of the array,and find the zero crossings
 and the steady state variance for the last 100 ms.}
last := false;
if input_array[1899] > 0 then
   last:= true;
sigma := 0;
zero := 0;
for i:= 1900 to 2000 do
   begin
   if input_array[i] >0 then
      this := true
   else
      this := false;
   if this <> last then
      zero := zero + 1;
   temp := input_array[i];
   temp := temp*input_array[i];
   sigma := sigma + temp;
   last := this;
   end;
sigma := sigma/100;
sec_stand_dev := sqrt(sigma);
out_stuff.endvar := sec_stand_dev;
out_stuff.zeros := zero;

{now find the standard deviation of the steady state of the first part}

sigma := 0;
for i:= 1 to 100 do
   begin
   temp := input_array[i];
   temp := temp * input_array[i];
   sigma := sigma + temp;
   end;
sigma := sigma/100;
first_stand_dev := sqrt(sigma);
out_stuff.beginvar := first_stand_dev;
{find delta and find the 90% change value}

delta := first_stand_dev - sec_stand_dev;
ninety_point := first_stand_dev - (delta * 0.9);
sigma := 0;
```

```
 done := false;
 for i := 1 to 51 do
    begin
    temp := input_array[i];
    temp := temp * input_array[i];
    sigma := sigma + temp;
    end;
 temp := sigma;
 for i := 26 to 1974 do
    begin
    sigma := sigma/50;
    test_stand_dev := sqrt(sigma);
    out_var[i] := round(test_stand_dev);
    temp2 := input_array[i-25];
    temp2 := temp2 * input_array[i-25];
    temp := temp - temp2;
    temp3 := input_array[i+26];
    temp3 := temp3 *input_array[i+26];
    temp := temp + temp3;
    sigma := temp;
    if not done then
       begin
       if  (delta < 0) and (test_stand_dev > ninety_point) then
          done := true;
       if (delta > 0) and (test_stand_dev < ninety_point) then
          done := true;
       if done  then
          tau := i ;
       end;
    end;
 for i := 1 to 25 do
    out_var[i] := 0;
 for i := 1976 to 2000 do
     out_var[i] := 0;
 if not done then
     begin
     tau := 0;
     writeln('this run didnt work out right');
     end;
 out_stuff.delta := tau;
 end;

 {*********************************************************************}

 {this procedure writes the header at the beginning of the program}


 procedure write_header;
 begin
 clrscr;
 textmode(2);
 gotoxy(20,2);
 writeln('E.M.G. DATA AQUISITION AND ANALYSIS SOFTWARE SYSTEM');
 gotoxy(20,3);
 writeln('****************************************************');
```

```
end;

{****************************************************************************}

{this procedure plots the raw EMG or variance data on the screen}

procedure show_points(var plot_data:data;scale_factor:integer);

var
  i,j:integer;
  y_cord:integer;
  real_y:real;
begin
hires;
for i := 1 to 4 do
   begin
   draw(89,(25 + (i-1)*50),589,(25 + (i-1)*50),15);
   for j := 1 to 500 do
      begin
      real_y := plot_data[((i-1)*500) + j];
      real_y := ((real_y*20)/scale_factor) ;
      y_cord := round(real_y) * (-1);
      draw(j+89,(25 + (i-1)*50) +y_cord,j+89,(25 +(i-1)*50),15)
      end;
   end;
writeln('scale factor = ',scale_factor:5);
readln;
end;
{****************************************************************************}

{this procedure runs the lower data aquisition procedures}

procedure data_input;

var
   i:integer;

begin
clrscr;
set_up_timer;
writeln('HIT ENTER TO START DATA AQUISITION');
readln;
delay(1000);
data_aq;
writeln('DATA AQUISITION COMPLETE, NOW SCALING ');
for i := 1 to 8 do
   fix_data(datain[i]);
writeln('HIT ENTER TO RETURN TO MAIN MENU');
readln;
end;
{****************************************************************************}

{this procedrue calls the lower screen display procedures}

procedure view_emg;
```

```
var
   reply:char;
   repnum:integer;

begin
reply := '1';
repeat
   clrscr;
   write_header;
   window(5,8,80,24);
   gotoxy(1,1);
   write('ENTER CHANNEL TO BE DISPLAYED  (0 TO QUIT) : ');
   readln(reply);
   repnum:= integer(reply) - 48;
   if not(reply in ['0'..'8']) then
      begin
      sound(440);
      delay(500);
      nosound;
      gotoxy(1,41);
      write(': ');
      end
   else
      begin
      if reply <> '0' then
         begin
         show_points(datain[repnum],2048);
         end;
      end;
   until reply = '0';
write_header;
end;
{**********************************************************************}

{this procedure plots the raw EMG or variance data on a hard copy
 plotter}

procedure plot_emg;

var
   reply:char;
   repnum:integer;
   fileid:string[80];
   string_out:string[20];

begin
reply := '1';
repeat
   clrscr;
   write('ENTER CHANNEL TO BE PLOTTED  (0 TO QUIT) : ');
   readln(reply);
   repnum:= integer(reply) - 48;
   if not(reply in ['0'..'8']) then
      begin
      sound(440);
      delay(500);
      nosound;
      gotoxy(1,41);
      write(': ');
      end
   else
      begin
      if reply <> '0' then
         begin
         write('ENTER FILE IDENTIFIER : ');
         readln(fileid);
         writeln;writeln('WHEN PLOTTER IS READY HIT RETURN');
         readln;
         plot(datain[repnum],fileid,2048,true);
         string_out := ' SP0' + chr(3);
         write(aux,string_out);
         end;
      end;
   until reply = '0';
end;
{*********************************************************************}

{this procedure stores the raw EMG data in disk files seperated by
channel}

procedure store_emg;

var
   filename:string[75];

begin
clrscr;
writeln('PLEASE ENTER DRIVE DESIGNATION AND BASE FILENAME');
write('  REMEMBER, BASE FILENAME HAS 7 OR LESS CHARACTERS : ');
readln(filename);
writeln;writeln('  EMG FILES WILL STORED AS :');
for i := 1 to 8 do
   begin
   filenames[i] := filename + CHR(48+i) + '.dat';
   WRITELN(FILENAMES[I]);
   end;
readln;
for i := 1 to 8 do
  begin
  assign(outfiles[i],filenames[i]);
  rewrite(outfiles[i]);
  write(outfiles[i],datain[i]);
  close(outfiles[i]);
  end;
writeln;writeln('ALL FILES STORED, HIT RETURN TO CONTINUE');
readln;
end;
{*********************************************************************}
```

```
{this procedure retrieves the raw EMG data from disk files}

procedure get_emg;

var
   filename:string[75];
   tempname:string[75];
   fil:file;
   good:boolean;
   quit:boolean;

begin
clrscr;
quit:= false;
repeat
   gotoxy(1,1);
   write('ENTER EMG FILE IDENTIFIER (HIT ! TO ESCAPE) : ');
   readln(tempname);
   if tempname[1] = '!' then
      begin
      good := true;
      quit := true;
      end;
   if not quit then
      begin
      filename := tempname + '1.dat';
      assign(fil,filename);
      {$I-}
      reset(fil);
      {$I+}
      good := (IOresult = 0);
      if not good then
        begin
         sound(440);
         delay(500);
         nosound;
         end
      else
         begin
         for i := 1 to 8 do
            begin
            filename := tempname + chr(48 + i) + '.dat';
            assign(outfiles[i],filename);
            reset(outfiles[i]);
            read(outfiles[i],datain[i]);
            close(outfiles[i]);
            end;
         end;
      end;
  until good;
  if not quit then
     begin
     writeln('ALL DATA RETRIEVED, PRESS RETURN TO CONTINUE');
     readln;
     end;
```

```
end;
{*********************************************************************}

{this procedure analyses the data from one trial for all eight
channels.  Originally, the computer did not have an 8087 math
co-processor and so was very slow, that is why the coffee message
exists}

procedure analyse;

var
   i:integer;

begin
clrscr;
writeln('THIS MAY TAKE A WHILE, GO GET SOME COFFEE');
for  i := 1 to 8 do
   begin
   anchan(datain[i],alldat[i],vardat[i]);
   writeln('CHANNEL ',i:3,' DONE');
   end;
writeln('DATA ANALYSIS COMPLETE');
writeln('HIT RETURN TO CONTINUE');
readln;
end;
{*********************************************************************}

{this procedure allows the user to view the average variance data
as well as the parameters extracted from any channel of data}

procedure view_results;

var
   reply:char;
   repnum:integer;
   scale_factor:integer;

begin
reply := '1';
repeat
   clrscr;
   write_header;
   window(5,8,80,24);
   gotoxy(1,1);
   write('ENTER CHANNEL TO BE DISPLAYED  (0 TO QUIT) : ');
   readln(reply);
   repnum := integer(reply) - 48;
   if not(reply in ['0'..'8']) then
      begin
      sound(440);
      delay(500);
      nosound;
      gotoxy(1,41);
      write(': ');
      end
      else
         begin
         if reply <> '0' then
            begin
            scale(vardat[repnum],scale_factor);
            show_points(vardat[repnum],scale_factor);
            clrscr;
            write_header;
            window(5,8,80,24);
            gotoxy(1,1);
            writeln('VARIANCE DATA FOR CHANNEL ',reply:3);writeln;
            writeln('INITIAL S.D. = ',alldat[repnum].beginvar:6:0);writeln;
            writeln('STEADY STATE S.D. = ',alldat[repnum].endvar:6:0);write
            writeln('   "     "   ZERO CROSSINGS = ',alldat[repnum].zeros:5
            writeln('DELTA = ',alldat[repnum].delta:5,' milliseconds');writ
            writeln('HIT RETURN TO CONTINUE');
            readln;
            end;
         end;
      until reply = '0';
   write_header;
   end;
   {*********************************************************************}

   {this procedure allows the user to make a hardcopy on a plotter of
   the average variance over the whole run for any channel of data}

   procedure plot_results;

   var
      reply:char;
      repnum:integer;
      scale_factor:integer;
      fileid:string[80];
      string_out:string[20];

   begin
   reply := '1';
   repeat
      clrscr;
      write('ENTER CHANNEL TO BE PLOTTED  (0 TO QUIT) : ');
      readln(reply);
      repnum := integer(reply) - 48;
      if not(reply in ['0'..'8']) then
         begin
         sound(440);
         delay(500);
         nosound;
         gotoxy(1,41);
         write(': ');
         end
      else
         begin
         if reply <> '0' then
            begin
            scale(vardat[repnum],scale_factor);
            write('ENTER FILE IDENTIFIER : ');
            readln(fileid);
            write('WHEN PLOTTER IS READY HIT RETURN');
            readln;
            plot(vardat[repnum],fileid,scale_factor,false);
            string_out := ' SP0' + CHR(3);
            write(aux,string_out);
            end;

         end;
      until reply = '0';
   end;
   {*******************************************************************}

   {this procedure stores all data on disk for further processing}

   procedure store_results;

   var
      filename:string[75];

   begin
   clrscr;
   writeln('PLEASE ENTER DRIVE DESIGNATION AND BASE FILENAME');
   write('  REMEMBER, BASE FILENAME HAS 7 OR LESS CHARACTERS : ');
   readln(filename);
   writeln;write(' VARIANCE DATA FILES WILL STORED AS :');
   varfilename := filename + '.PRO';
   writeln(varfilename);
   assign(allfiles,varfilename);
   rewrite(allfiles);
   write(allfiles,alldat);
   close(allfiles);
   writeln;writeln('VARIANCE DATA FILE STORED, HIT RETURN TO CONTINUE');
   readln;
   end;
   {*******************************************************************}

   {this procedure displays a main menu with which the user can
   select a desired action}

   procedure main_menu;

   var
      reply:char;
      done:boolean;
      quit:boolean;


   begin
   quit := false;
   clrscr;
   textmode(2);
   gotoxy(20,2);
```

```
 writeln('E.M.G. DATA AQUISITION AND ANALYSIS SOFTWARE SYSTEM');
 gotoxy(20,3);
 writeln('***************************************************');
 repeat
    window(5,8,80,24);
    clrscr;
    gotoxy(1,1);
    writeln;
    writeln('PLEASE ENTER OPTION LETTER');
    writeln('[A] DATA AQUISITION');
    writeln('[B] VIEW EMG DATA ');
    writeln('[C] PLOT EMG DATA ');
    writeln('[D] STORE EMG DATA FILES');
    writeln('[E] RETRIEVE EMG DATA FILES');
    writeln('[F] ANALYSE EMG DATA');
    writeln('[G] VIEW RESULTS OF ANALYSIS');
    writeln('[H] PLOT RESULTS OF ANALYSIS ');
    writeln('[I] STORE RESULTS OF ANALYSIS ');
    write('[J] QUIT                             : ');
    done := false;
    repeat
       readln(reply);
       reply := upcase(reply);
       if  not (reply in['A'..'J']) then
          begin
          sound(440);
          delay(250);
          nosound;
          gotoxy(35,12);
          write(': ');
          end
       else
          done := true;
    until done;
    case reply of
                 'A':data_input;
                 'B':view_emg;
                 'C':plot_emg;
                 'D':store_emg;
                 'E':get_emg;
                 'F':analyse;
                 'G':view_results;
                 'H':plot_results;
                 'I':store_results;
                 'J':quit := true
    end;
until quit = true;
end;
{*********************************************************************}

{MAINLINE}

begin
clrscr;
for i:= 1 to 8 do
```

```
     begin;
     for j := 1 to 2000 do
        begin
        datain[i,j] := 0;
        vardat[i,j] := 0;
        end;
     end;
  main_menu;
  end.
```

```
program quik(input,output);
{this program evaluates the emg files in batches}


const
   base = $0100 ;  {mux base}
   PPI  = 97    ;  {PPI address}
   TCB  = $0043 ;  {Timer Control Byte address}
   TLA  = $0042 ;  {Timer Load Adress}
   count = 2000 ;  {number of array elements}

type
   name = string[80];
   name_array = array [1..30] of name;
   string80 = string[80];
   string2=string[2];
   data = array[1..count] of integer;
   realdata = array[1..count] of real;
   datafile = file of data;
   andat = record
           beginvar:real;
           endvar  :real;
           zeros   :integer;
           delta   :integer;
           end;
   varfile = array[1..8] of andat;
   anfile=file of varfile;

var
   filenames:array[1..8] of name;
   varfilename:name;
   tempname:name;
   i,j:integer;
   outfiles:array[1..8] of datafile;
   datain:array[1..8] of data absolute $3000:$0000;
   alldat:varfile;
   allfiles:anfile;
   dee:byte;    {global variable for inline code statements}
   test:integer;

{*******************************************************************}

{this procedure analyses one channel of EMG data, compromising
an array of 2000 elements.  It extracts the variance, zero crossing,
and MADT data and returns the values in the data structure out_stuff}

procedure anchan(var input_array:data;
                 var out_stuff  :andat);

var
i,j:integer;
sigma:real;
temp:real;
temp2:real;
```

```
temp3:real;
zero:integer;
last:boolean;
this:boolean;
done:boolean;
first_stand_dev:real;
sec_stand_dev:real;
test_stand_dev:real;
delta:real;
ninety_point:real;
tau:integer;

begin
sigma := 0;

{first go to the end of the array,and find the zero crossings
 and the steady state variance for the last 100 ms.}

last := false;
if input_array[1899] > 0 then
   last:= true;
sigma := 0;
zero := 0;
for i:= 1900 to 2000 do
   begin
   if input_array[i] >0 then
      this := true
   else
      this := false;
   if this <> last then
      zero := zero + 1;
   temp := input_array[i];
   temp := temp*input_array[i];
   sigma := sigma + temp;
   last := this;
   end;
sigma := sigma/100;
sec_stand_dev := sqrt(sigma);
out_stuff.endvar := sec_stand_dev;
out_stuff.zeros := zero;

{now find the standard deviation of the steady state of the first part}

sigma := 0;
for i:= 1 to 100 do
   begin
   temp := input_array[i];
   temp := temp * input_array[i];
   sigma := sigma + temp;
   end;
sigma := sigma/100;
first_stand_dev := sqrt(sigma);
out_stuff.beginvar := first_stand_dev;
{find delta and find the 90% change value}
```

```
 delta := first_stand_dev - sec_stand_dev;
 ninety_point := first_stand_dev - (delta * 0.9);
 sigma := 0;
 done := false;
 for i := 1 to 101 do
    begin
    temp := input_array[i];
    temp := temp * input_array[i];
    sigma := sigma + temp;
    end;
 temp := sigma;
 for i := 51 to 1949 do
    begin
    sigma := sigma/100;
    test_stand_dev := sqrt(sigma);
    temp2 := input_array[i-50];
    temp2 := temp2 * input_array[i-50];
    temp := temp - temp2;
    temp3 := input_array[i+51];
    temp3 := temp3 *input_array[i+51];
    temp := temp + temp3;
    sigma := temp;
    if not done then
       begin
       if  (delta < 0) and (test_stand_dev > ninety_point) then
          done := true;
       if (delta > 0) and (test_stand_dev < ninety_point) then
          done := true;
       if done  then
          tau := i ;
       end;
    end;
if not done then
    begin
    tau := 0;
    writeln('this run didnt work out right');
    end;
out_stuff.delta := tau;
end;

{*********************************************************************}

{This procedure finds how many runs a subject completed by counting
the number of files with the correct format}

procedure find_run_number(wild_file:name; var run_number:integer);

var
   i:integer;
   wild:name;
   num:string[2];
   fil:file;
   done:boolean;

begin
```

```
 done := false;
 i := 0;
 repeat
    i := i + 1;
    wild:= wild_file;
    if i < 10 then
       begin
       str(i:1,num);
       wild  := wild + '0'+ num;
       end
    else
       begin
       str(i:2,num);
       wild := wild + num;
       end;
    wild := wild + '1.dat';
    assign(fil,wild);
    {$I-}
    reset(fil);
    {$I+};
    done := ioresult <> 0;
until done;
i := i -1;
run_number := i;
end;
{*********************************************************************}

{This procedure allows the user to enter the initials of the
subject(s) whose data is to be processed}


procedure read_in_names(var inputnames:name_array;var num_names:integer)

begin
num_names := 1;
window(3,5,80,24);
gotoxy(1,1);
clrscr;
writeln('enter 3 letter file identifiers (stop to quit)');
repeat
   inputnames[num_names] := '';
   readln(inputnames[num_names]);
   num_names := num_names + 1;
until(inputnames[num_names-1] = 'stop');
num_names := num_names -2;
end;

{*********************************************************************}

{this procedure analyses the data for a complete set of runs for
one subject}

procedure do_one_name(one_name:name);

var
```

```
    i,j,k:integer;
    run_number:integer;
    filename:name;
    kname:name;
    jname:name;
    outname:name;
    action:array[1..7] of string2;
    small:string2;

 begin
 action[1] := 'HI';
 action[2] := 'HO';
 action[3] := 'EF';
 action[4] := 'EE';
 action[5] := 'WP';
 action[6] := 'WS';
 action[7] := 'RS';
 filename:= one_name;
 filename:= filename + 'HI';
 find_run_number(filename,run_number);
 for i := 1 to run_number do
    begin
    for j := 1 to 7 do
       begin
       jname := one_name + action[j];
       if i < 10 then
          begin
          str(i:1,small);
          jname := jname + '0' + small;
          end
       else
          begin
          str(i:2,small);
          jname := jname + small;
          end;
       for k := 1 to 8 do
          begin
          kname := jname;
          kname := kname +chr(k+48) + '.dat';
          assign(outfiles[k],kname);
          reset(outfiles[k]);
          read(outfiles[k],datain[k]);
          close(outfiles[k]);
          writeln('processing ',kname);
          anchan(datain[k],alldat[k]);
          end;
       kname := jname;
       kname := kname + '.pro';
       assign(allfiles,kname);
       rewrite(allfiles);
       write(allfiles,alldat);
       close(allfiles);
       end; {end j}
    end; {end i}
end;
```

```
{*********************************************************************}

{this procedure repeatedly calls the previous and analyses all
data for all subjects}

procedure do_all_names;

var
   inputnames:name_array;
   num_names:integer;
   i:integer;
   reply:char;

begin
repeat
   read_in_names(inputnames,num_names);
   clrscr;
   gotoxy(1,1);
   writeln('are these the files you want ? y/n');
   writeln;
   for i:= 1 to num_names do
      begin
      writeln(inputnames[i]);
      end;
   gotoxy( 36,1);
   readln(reply);
   reply := upcase(reply);
until reply <> 'N' ;
gotoxy(1,1);
writeln('processing these files                          ');
writeln('**********************');
window(40,5,80,24);
clrscr;
gotoxy(1,1);
writeln('process information');
writeln('******************');
window(40,8,80,20);
gotoxy(1,1);
for i := 1 to num_names do
   begin
   do_one_name(inputnames[i]);
   end;
end;

{*********************************************************************}

{MAINLINE}

begin
clrscr;
writeln;
writeln('                QUIK DATA ANALYSIS PROGRAM');
writeln('*****************************************************************
do_all_names;
end.
```

```
 program max(input,output);

 {this program evaluates the max emg files in batches.
 it basically emulates the quik program but ignores
 the zero and MADT data}



const
   count = 2000 ;  {number of array elements}

type
   name = string[80];
   name_array = array [1..30] of name;
   string80 = string[80];
   string2=string[2];
   data = array[1..count] of integer;
   datafile = file of data;
   andat = record
           beginvar:real;
           endvar  :real;
           zeros   :integer;
           delta   :integer;
           end;
   varfile = array[1..8] of andat;
   anfile=file of varfile;
   norm_array = array[1..8] of integer;

var
   filenames:array[1..8] of name;
   varfilename:name;
   tempname:name;
   i,j:integer;
   outfiles:array[1..8] of datafile;
   datain:array[1..8] of data absolute $3000:$0000;
   alldat:varfile;
   allfiles:anfile;
   dee:byte;    {global variable for inline code statements}
   test:integer;

{*********************************************************************}

{this procedure finds the maximum steady state variance for
all channels and stores them in the norm_array data structure}

procedure find_max(var max_array:norm_array);


var
   i,j:integer;
   sigma:real;
   int_sigma:integer;
   temp:real;

begin
```

```
for i:= 1 to 8 do
   begin
   sigma := 0;
   for j := 1800 to 2000 do
      begin
      temp := datain[i,j];
      temp := temp * temp;
      sigma := sigma + temp;
      end;
  sigma := sigma/200;
  sigma := sqrt(sigma);
  int_sigma := round(sigma);
  max_array[i] := int_sigma;
  end;
end;

{*********************************************************************}

{this procedure allows the user to enter the initials of the subject(s)
whose maximum data is to be processed}

procedure read_in_names(var inputnames:name_array;var num_names:integer)

begin
num_names := 1;
window(3,5,80,24);
gotoxy(1,1);
clrscr;
writeln('enter 3 letter file identifiers (stop to quit)');
repeat
   inputnames[num_names] := '';
   readln(inputnames[num_names]);
   num_names := num_names + 1;
until(inputnames[num_names-1] = 'stop');
num_names := num_names -2;
end;

{*********************************************************************}

{this procedure analyses the maximum data for one subject}

procedure do_one_name(one_name:name);

var
   i,j,k:integer;
   run_number:integer;
   filename:name;
   kname:name;
   jname:name;
   outname:name;
   action:array[1..6] of string2;
   small:string2;
   max_array:norm_array;
   old_array:norm_array;
   which_action:norm_array;
```

```
 begin
 for i:= 1 to 8 do
    begin
    max_array[i] := 0;
    old_array[i] := 0;
    which_action[i] := 0;
    end;

 action[1] := 'HI';
 action[2] := 'HO';
 action[3] := 'EF';
 action[4] := 'EE';
 action[5] := 'WP';
 action[6] := 'WS';
 filename:= one_name;
 for j := 1 to 6 do
    begin
    jname := one_name + action[j] + 'MX';
    for k := 1 to 8 do
       begin
       kname := jname;
       kname := kname +chr(k+48) + '.dat';
       assign(outfiles[k],kname);
       reset(outfiles[k]);
       read(outfiles[k],datain[k]);
       close(outfiles[k]);
       writeln('processing ',kname);
       end;
    find_max(max_array);
    for i := 1 to 8 do
       begin
       if max_array[i] > old_array[i] then
          begin
          old_array[i] := max_array[i];
          which_action[i] := j;
          end;
       end;
    end; {end j}
for i := 1 to 8 do
   begin
   alldat[i].endvar := old_array[i];
   writeln('channel ',i:2,' max ',alldat[i].endvar:6:0,' action ',action
   end;
kname := one_name;
kname := kname + 'MAX.pro';
assign(allfiles,kname);
rewrite(allfiles);
write(allfiles,alldat);
close(allfiles);
end;


{*********************************************************************}
```

```
{this procedure calls the routines which analyze the maximum data
for all subjects}

procedure do_all_names;

var
   inputnames:name_array;
   num_names:integer;
   i:integer;
   reply:char;

begin
repeat
   read_in_names(inputnames,num_names);
   clrscr;
   gotoxy(1,1);
   writeln('are these the files you want ? y/n');
   writeln;
   for i:= 1 to num_names do
      begin
      writeln(inputnames[i]);
      end;
   gotoxy( 36,1);
   readln(reply);
   reply := upcase(reply);
until reply <> 'N' ;
gotoxy(1,1);
writeln('processing these files                              ');
writeln('**********************');
window(40,5,80,24);
clrscr;
gotoxy(1,1);
writeln('process information');
writeln('*******************');
window(40,8,80,20);
gotoxy(1,1);
for i := 1 to num_names do
   begin
   do_one_name(inputnames[i]);
   end;
end;

{*******************************************************************}

{MAINLINE}

begin
clrscr;
writeln;
writeln('               MAXIMUM EMG DATA NORMALIZATION STUFF');
writeln('*****************************************************************

do_all_names;
end.
```

```
program normal(input,output);

{this program reads in the *.pro files and normalizes the data
 for variance and delta.  It then re-arranges the data so that
 it can be added to the ch*.pro files * = 1-8}

type
   name = string[80];
   name_array = array [1..30] of name;
   string80 = string[80];
   string2=string[2];
   string4=string[4];
   andat = record
           beginvar:real;
           endvar  :real;
           zeros   :integer;
           delta   :integer;
           end;
   chdata = record
           variance :string4;
           zeros    :string4;
           delta    :string4;
           action   :string4;
           end;
   varfile = array[1..8] of andat;
   anfile=file of varfile;
   chdarray=array[1..8] of chdata;
   chfile = text;
   chfilearray = array[1..8] of chfile;

var
   filenames:array[1..8] of name;
   bigdat:varfile;
   varfilename:name;
   tempname:name;
   i,j:integer;
   channeldata:chdarray;
   channelfile:chfilearray;
   alldat:varfile;
   allfiles:anfile;

{*********************************************************************}

{this procedure reads in all the 'max' data  in order to find the larges
steady state variance for each action}

procedure read_in_biggest(var bigdat:varfile;varfilename:name);

var
   temp:name;
   i:integer;

begin
temp := varfilename;
temp := temp + 'MAX.pro';
```

```
assign(allfiles,temp);
reset(allfiles);
read(allfiles,bigdat);
for i:= 1 to 8 do
   begin
   writeln(bigdat[i].endvar:6:0);
   end;
close(allfiles);
end;


{***********************************************************************}

{this procedure normalizes the variance data according to the
largest reading for each channel.}

procedure norm_var(var alldat:varfile);


var
   i:integer;
   biggest:real;

begin
biggest:=0;
for i := 1 to 8 do
   begin
   alldat[i].endvar := (alldat[i].endvar/bigdat[i].endvar) * 100;
   alldat[i].beginvar := (alldat[i].beginvar/bigdat[i].endvar) * 100;
   end;
end;

{***********************************************************************}

{this procedure normalizes the MADT data.  If the MADT of a particular
channel is less than 250 msec. it is assigned the value of -1, in
other words, that channel was not activated}

procedure norm_delta(var alldat:varfile);

var
   i:integer;
   closest:integer;

begin
closest:=2000;
for i := 1 to 8 do
   begin
   if ((alldat[i].delta - 250) < closest) AND ((alldat[i].delta - 250) >
      closest := alldat[i].delta;
   end;
for i := 1 to 8 do
   begin
   alldat[i].delta := (alldat[i].delta-closest);
   if alldat[i].delta < 0 then
```

```
      alldat[i].delta  := -1;
    end;
 end;

 {*********************************************************************}

 {this procedure outputs all the data in a channel by channel form so
 that it can be read by the SAS program}

 procedure out_data(var alldat:varfile;
                         action:integer;
                 var channelfile:chfilearray);

 var
    tempchdata:chdarray;
    i:integer;
    outline:string[16];

 begin
 for i:= 1 to 8 do
    begin
    str(round(alldat[i].endvar):4,tempchdata[i].variance);
    str(alldat[i].zeros:4,tempchdata[i].zeros);
    str(alldat[i].delta:4,tempchdata[i].delta);
    str(action:4,tempchdata[i].action);
    with tempchdata[i] do
       outline :=  variance + zeros + delta + action;
    writeln(channelfile[i],outline);
    end;
 end;

 {*********************************************************************}

 {this procedure opens the channel files for the addition of the normaliz
 data}

 procedure open_channel_files(var channelfile:chfilearray);

 var
    i:integer;
    channelname:name;

 begin
 for i := 1 to 8 do
    begin
    channelname := 'ch' + chr(i+48) + '.nor';
    assign(channelfile[i],channelname);
    rewrite(channelfile[i]);
    end;
 end;

 {*********************************************************************}

 {this procedure finds the number of runs each subject accomplished
 for all actions during the experiment}
```

```
procedure find_run_number(wild_file:name; var run_number:integer);

var
   i:integer;
   wild:name;
   num:string[2];
   done:boolean;
   fil:file;

begin
done := false;
i := 0;
writeln('finding run number');
repeat
   writeln(i);
   i := i + 1;
   wild:= wild_file;
   if i < 10 then
      begin
      str(i:1,num);
      wild  := wild + '0'+ num;
      end
   else
      begin
      str(i:2,num);
      wild := wild + num;
      end;
   wild := wild + '.PRO';
   assign(fil,wild);
   {$I-}
   reset(fil);
   {$I+};
   done := (ioresult <> 0);
until done;
i := i -1;
run_number := i;
end;
{*********************************************************************}

{this procedure allows the user to enter the initials of the subject(s)
whose data is to be normalized}


procedure read_in_names(var inputnames:name_array;var num_names:integer)

begin
num_names := 1;
window(3,5,80,24);
gotoxy(1,1);
clrscr;
writeln('enter 3 letter file identifiers (stop to quit)');
repeat
   inputnames[num_names] := '';
   readln(inputnames[num_names]);
   num_names := num_names + 1;
until(inputnames[num_names-1] = 'stop');
num_names := num_names -2;
end;

{********************************************************************}

{this procedure normalizes and stores  all the data  for one subject}

procedure do_one_name(one_name:name;channelfile:chfilearray);

var
   i,j,k:integer;
   run_number:integer;
   filename:name;
   kname:name;
   jname:name;
   outname:name;
   action:array[1..7] of string2;
   small:string2;

begin
action[1] := 'HI';
action[2] := 'HO';
action[3] := 'EF';
action[4] := 'EE';
action[5] := 'WP';
action[6] := 'WS';
action[7] := 'RS';
filename:= one_name;
filename:= filename + 'HI';
{find_run_number(filename,run_number);}
run_number := 9;
for i := 1 to 7 do
   begin
   for j := 1 to run_number do
      begin
      jname := one_name + action[i];
      if j < 10 then
         begin
         str(j:1,small);
         jname := jname + '0' + small;
         end
      else
         begin
         str(j:2,small);
         jname := jname + small;
         end;
      kname := jname;
      kname := kname + '.PRO';
      assign(allfiles,kname);
      reset(allfiles);
      read(allfiles,alldat);
      close(allfiles);
      writeln('processing ',kname);
```

```
        norm_var(alldat);
        norm_delta(alldat);
        out_data(alldat,i,channelfile);
        close(allfiles);
        end;
     for j:=1 to 8 do
        flush(channelfile[j]);
     end; {end i}
  end;
  {*******************************************************************}

  {this procedure closes all data files after processing is complete}

  procedure close_channel_files (var channelfile:chfilearray);

  var
     i:integer;

  begin
  for i:= 1 to 8 do
     begin
     close(channelfile[i]);
     end;
  end;

  {*******************************************************************}

  {this procedure call the normalization routines for all subjects}

  procedure do_all_names;

  var
     inputnames:name_array;
     num_names:integer;
     i:integer;
     reply:char;

  begin
  open_channel_files(channelfile);
  repeat
     read_in_names(inputnames,num_names);
     clrscr;
     gotoxy(1,1);
     writeln('are these the files you want ? y/n');
     writeln;
     for i:= 1 to num_names do
        begin
        writeln(inputnames[i]);
        end;
     gotoxy( 36,1);
     readln(reply);
     reply := upcase(reply);
  until reply <> 'N' ;
  gotoxy(1,1);
  writeln('processing these files                        ');
```

```
writeln('*********************');
window(40,5,80,24);
clrscr;
gotoxy(1,1);
writeln('process information');
writeln('*******************');
window(40,8,80,20);
gotoxy(1,1);
for i := 1 to num_names do
   begin
   read_in_biggest(bigdat,inputnames[i]);
   do_one_name(inputnames[i],channelfile);
   end;
close_channel_files(channelfile);
end;
{*******************************************************************}

{MAINLINE}


begin
clrscr;
writeln;
writeln('                 DATA NORMALIZATION PROGRAM');
writeln('***********************************************************

do_all_names;
end.
```

Appendix B

LINEAR DISCRIMINANT FUNCTION REVIEW

t	a subscript to distinguish groups
S_t	the covariance matrix between group t
$\lvert S_t \rvert$	the determinant of S_t
S	the pooled covariance matrix
x	a vector containing the variables of an observation
m_t	a vector containing means of the variables in group t
q_t	the priori probability for group t

The generalized squared distance from x to group t is

$$D_2^t = g_1(x,t) + g_2(t)$$

where

$$g_1(t,x,) = (x-m_t)' \, S_t^{-1}(x-m_t) + \ln|S_t|$$

if the within group covariance matrices are used, or

$$g_1(x,t) = (x-m_t)' \, S^{-1}(x-m_t)$$

if the pooled covariance matrix is used; and

$$g_2(t) = -2\ln(q_t)$$

if the prior probabilities are not all equal, or

$$g_2 = 0$$

if the prior probabilities are all equal.

An observation is classified into group u if setting t = u produces the smalles value of D_t^2.

REFERENCES

[1] S.C. Jacobsen, D.F. Knutti, R.T. Johnson, H.H. Sears, "Developement of the Utah Artificial Arm", IEEE Transactions on Biomedical Engineering, Vol. BME-29, No.4, pp. 249-269, April 1982.

[2] J.E. Paciga, D.A. Gibson, R.Gillespie, "Clinical Evaluation of UNB 3-State Control of Prostheses", Bulletin of Prosthetics Research, BPR 10-36, Vol. 18, No.2, pp. 3-11, Fall 1981.

[3] R.N. Scott, J.E. Paciga, P.A. Parker, "Operator Error In Multistate Myoelectric Control Systems", Medical and Biological Engineering and Computing, Vol. 16, pp. 296-301, 1978.

[4] M.M. McKenzie, "Evaluation of Three Digital EMG Processors", M.Sc. Thesis, University of Manitoba, 1985.

[5] R.W. Wirta, D.R. Taylor, F.R. Finley,"Pattern Recognition Arm Prosthesis", Bulletin of Prosthetics Research Vol. BPR 10-30, Fall 1978.

[6] G.N.Saridis, T.P. Gootee, "EMG Pattern Analysis and Classification for a Prosthetic Arm", IEEE Transactions of Biomedical Engineering, Vol. BME-29, No. 6, pp. 403-412, June 1982.

[7] W.J. Brown,"An Investigation of a Nine State EMG Pattern Recognition Classifier for a Prosthetic Arm", M.Sc. Thesis, University of Manitoba, 1985.

[8] D.C. Dening, F.G. Gray, R.M. Haralick, "Prosthesis Control Using a Nearest Neighbor Electromyographic Pattern Classifier", IEEE Transactions of Biomedical Engineering, Vol. BME-30, No. 6, pp. 356-360, June 1983.

[9] P.C. Doerschuk, D.E. Gustafson, A.S. Willsky, "Upper Extremity Limb Function Discrimination Using EMG Signal Analysis", IEEE Transactions on Biomedical Engineering, Vol. BME-30, No. 1, pp. 18-28, Jan. 1983.

[10] D. Graupe, W.K. Cline, "Functional Seperation of EMG Signals via ARMA Identification Methods for Prosthesis Control Purposes", IEEE Transactions of Systems, Mand, and Cybernetics, Vol SMC-5, No. 2, pp. 252-258, March 1975.

[11] N. Hogan, R.W. Mann, "Myoelectric Signal Processign:

Optimal Estimation Applied to Electromyography-Part 1: Derivation of the Optimal Myoprocessor", IEEE Transactions of Biomedical Engineering, Vol. BME-27, No. 7, pp. 382-394, July 1980.

[12] A.L. Edwards, "Multiple Regression and the Analysis of Variance and Covariance", W.H. Freeman, New York, 1985.

[13] J.V. Basmajian, Primary Anatomy, Eight Edition, Williams and Wilkins, Baltimore, Md.,1982.

[14] R.N. Scott, "Myoelectric Energy Spectra", Medical and Biological Engineering and Computing, Vol. 5, pp. 303-305, 1967.

[15] D.E.Johnson, J.L. Hilburn, Rapid Practical Design of Active Filters, Wiley-Interscience, 1975.

[16] Turbo Pascal Vs. 3.0 Reference, Borland, Scotts Valley, California, 1985.

[17] Tou, Gonzalez, Pattern Recognition Principles, Addison-Wesley Publishing Company, Reading, Mass., 1974.

[18] Andrews, Introduction to Mathematical Techniques in Pattern Recognition, Wiley-Interscience, N.Y., 1972.

[19] SAS/STAT Guide for Personal Computers, Version 6 Edition, SAS Institute Inc. Cary, North Carolina, 1985.

[20] Duda, Hart, Pattern Classification and Scene Analysis, Wiley-Interscience, N.Y., 1973.

[21] D.T. Gibbons, M.D. O'Riain, S. Phillipe-Auguste, "An Above-Elbow Prosthesis Employing Programmed Linkages", IEEE Transactions on Biomedical Engineering, Vol. BME-34, No. 7, pp. 493-498, July, 1987.

[22] R. Safaee-Rad, Functional Human Arm Motion Study with a New 3-D Measurement System (VCR - Pipez - PC), M.Sc. Thesis, University of Manitoba, 1987.