Digital Processing of Somatosensory Evoked Potentials Applied to the Early Diagnosis of Multiple Sclerosis

By

KIM BYRON ROBERTS
B.A.Sc. UBC 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in
THE FACULTY OF GRADUATE STUDIES

Department of Electrical Engineering

We accept this thesis as conforming to the required standard

University of British Columbia
May 1984
© Kim Roberts, 1984

86.

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date May 16 1984

ABSTRACT

This study investigated several digital techniques for the extraction of parameters from Somatosensory Evoked Potentials (SEPs). The higher frequency smaller peaks in the SEP were emphasised using a digital filter to remove the low frequency components. The filtered SEPs were compared to a Standard Wave using Dynamic Time Warping techniques which produced a Cost Function to measure abnormality. A quantification of the dispersion of the SEPs was also computed. These parameters were evaluated as to their sensitivity and reliablility when used to classify Normal and Multiple Sclerosis SEPs. They proved to be clinically significant diagnostic tools.

Table of Contents

ACKNOWLEDGEMENTS

I would like to thank Peter Lawrence and Andrew Eisen for their advice and for their support, both academic and financial.

Cathy Eisen arranged and coordinated the group of MS patients.

Maureen Hoirch and Stella Kyriakidis provided valuable assistance and advice.

A special thank you to the forty-nine people who volunteered to come to the lab and have the tests done, without whom there would have been no data.

List of Tables

List of Figures

1) INTRODUCTION

Sensory stimuli induce electrical events recordable over the cerebral cortex. These cerebral evoked potentials are classified as somatosensory (meaning body sensation) when peripheral sensory nerves are stimulated. The recording of Somatosensory Evoked Potentials (SEPs) is not new. They were first recorded without the aid of computer averaging [Dawson,1947]. With the advent of sophisticated electronic equipment in the 1970's, the recording of cerebral evoked potentials became popularized. Within the last five to ten years, the field has blossomed and the techniques are now routine in many clinical neurophysiological laboratories throughout the world.

The Somatosensory Evoked Potential can be electrically or mechanically elicited by exciting peripheral nerves or their end-organs. The resulting impulses are conducted, by large-diameter sensory fibres, into the central nervous system, where they are carried predominantly by the dorsal column-lemniscal systems [Eisen, 1982a]. The SEP, measured between two scalp electrodes, contains peaks that are thought to be generated by sequential subcortical and cortical structures: the cuneate nucleus, the medial lemniscus, the thalamus, the thalamo-cortical connections, and the Somatosensory Cortex. [Kritchevsky et al, 1978; Anziska et al, 1978; Desmedt and Cheron, 1980; 1981]. The last is thought to produce the large

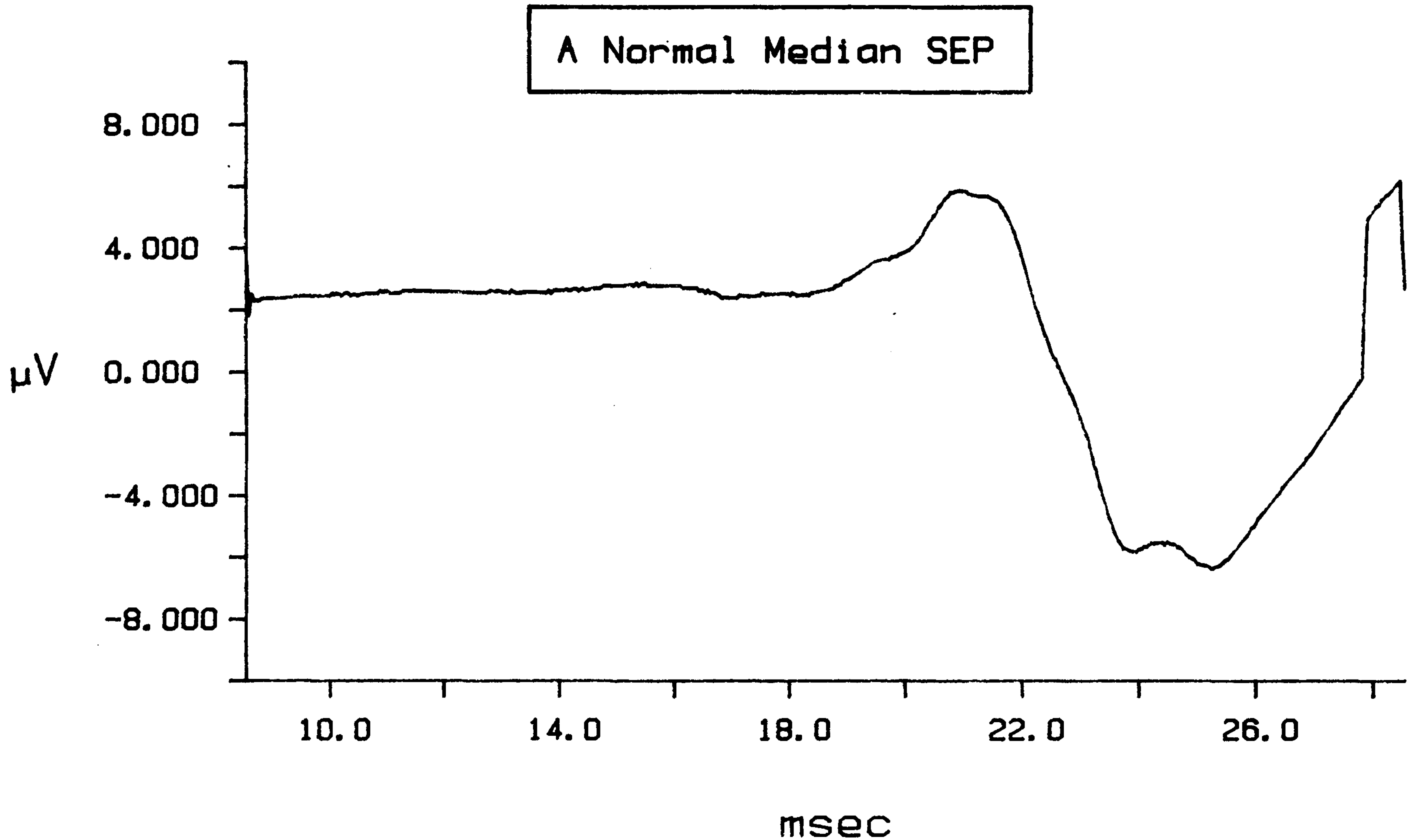

A Normal Median SEP
8.000
4.000
0.000
-4.000
-8.000
μV
10.0
14.0
18.0
22.0
26.0
msec

Figure 1.1

peak about 20 milliseconds after the stimulation. [Allison et al, 1980]. However, these suggested generators of the SEP are volume currents that pass through layers of different conductivities and complicated geometries, making it very difficult to resolve the sources [Kaufmann, 1981]. Although some invasive measurements have been done [Larson, 1968; Celesia, 1979], much more needs to be done in humans before any firm conclusions can be drawn as to the source of the components of the SEP.

In Somatosensory Evoked Potentials, the latency of the early components, that is, the time taken for impulses to reach the brain after application of a stimulus, is used as the chief means of diagnosing abnormality. The variability in inter-peak latencies, which is held to be indicative of the conduction times through various parts of the peripheral and central nervous system, is used to a lesser extent. Short latency components are considered as those with latencies of under 25 milliseconds when stimulating an arm, and less than 45 milliseconds when stimulating a leg. These components are very stable, being unaffected by drowsiness, sleep, or light anasthesia [Abrahamian et al, 1963; Goff et al, 1966]. They are, however, affected by deeper anaesthesia. Medium, and long latency components are much less stable.

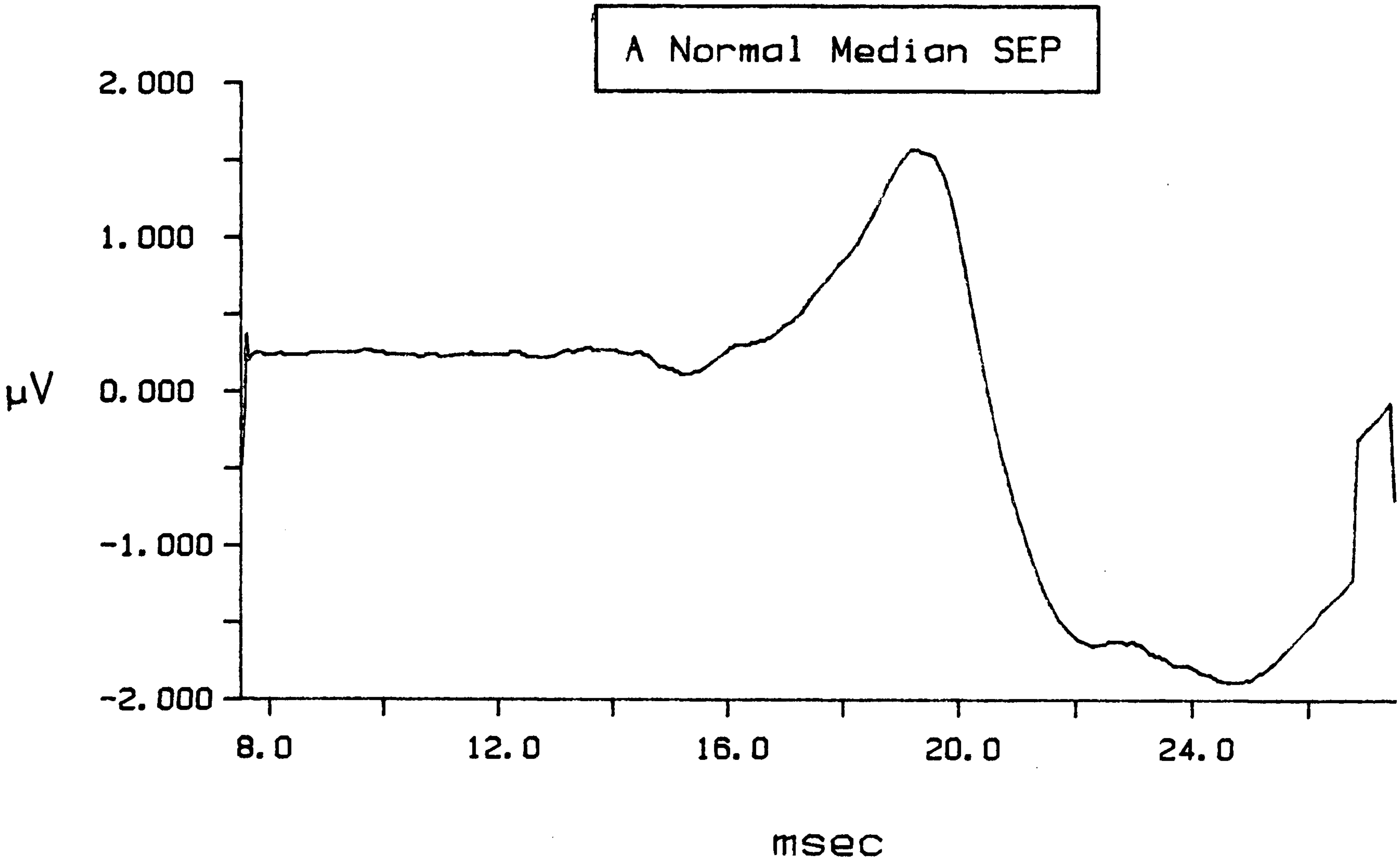

A Normal Median SEP
2.000
1.000
0.000
-1.000
-2.000
μV
8.0
12.0
16.0
20.0
24.0
msec

Figure 1.2

Possible parameters of the SEP (which are not normally used) include the overall shape of the waveform, amplitude of various major peaks, the smoothness or degree of synchronization of the response, and the presence of the higher frequency smaller peaks. The amplitude of the SEP is variable between subjects and therefore of limited value in absolute terms, although a 50 or greater percentage difference between left and right limbs in a given individual is usually regarded as being abnormal. Similarly, the shape of the SEP is dependent upon stimulation and recording sites, and probably other, so far undetermined, factors. Figures 1.1 to 1.5 show typical normal median nerve SEP waveforms measured between a pair of scalp electrodes. These figures are indicative of the wide variation in shape that is observed in normals.

Abnormal SEPs are particularly associated with the position sense loss [Giblin, 1964].

Amongst the various diseases in which evoked potential recording have been found useful, Multiple Sclerosis (MS) ranks amongst the highest. This disease, which has an incidence of about 100 per 100,000 is presently of unknown cause. MS causes a loss of the myelin covering of central nervous system fibres. Consequently, conduction in these fibres is slowed and eventually fails. The interrupted conduction produces distortion of impulse

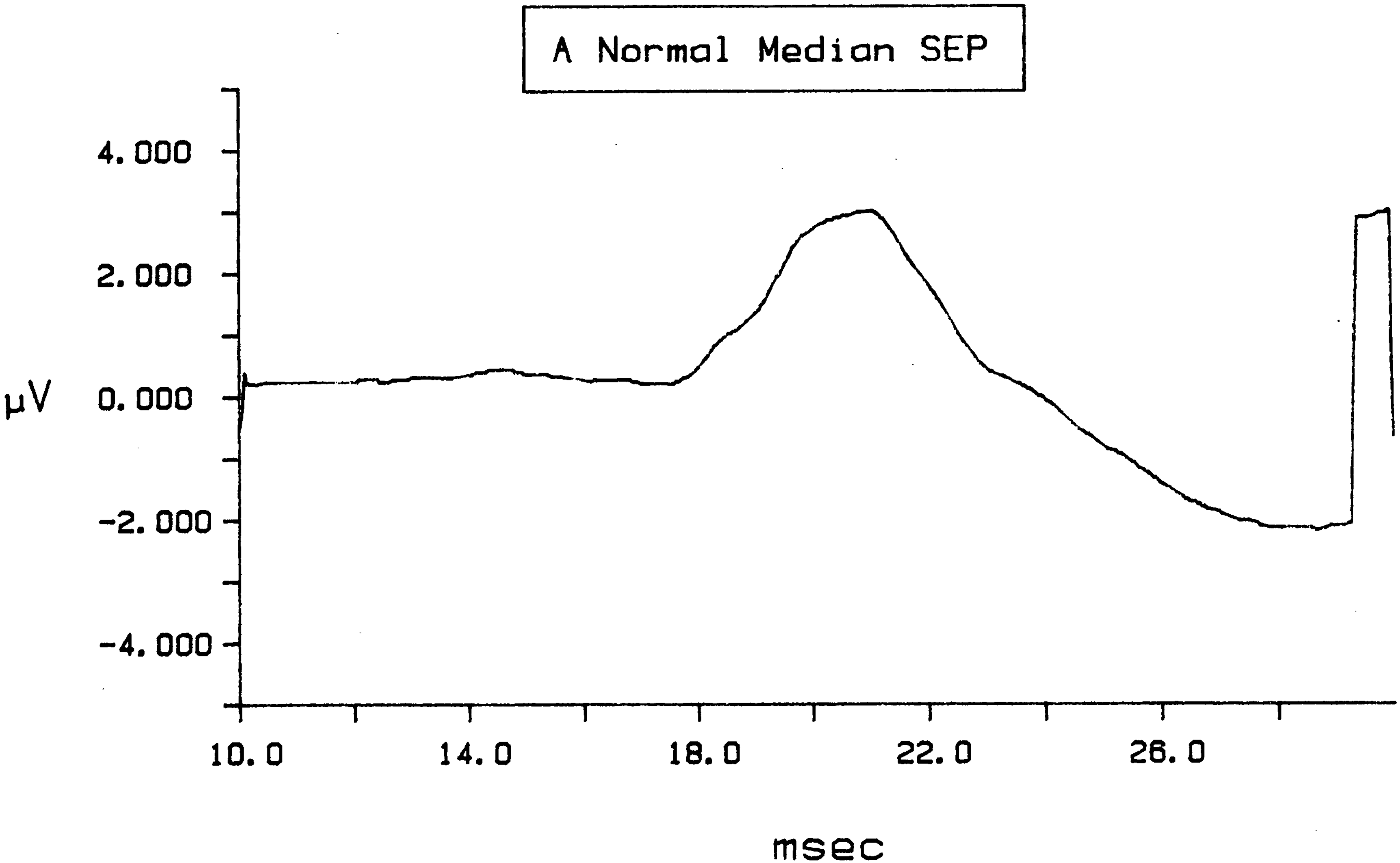
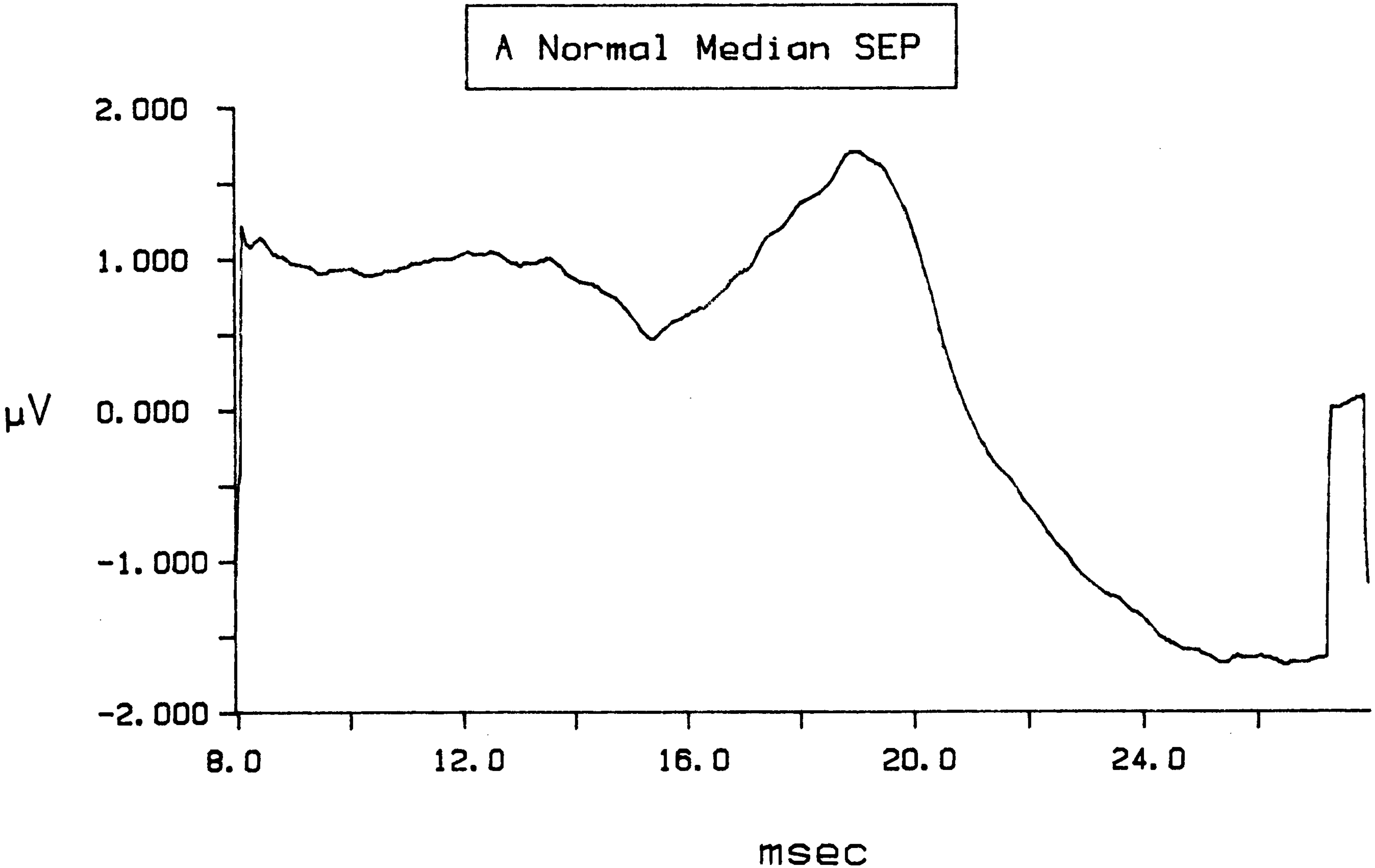

A Normal Median SEP
4.000
2.000
0.000
-2.000
-4.000
µV
10.0
14.0
18.0
22.0
26.0
msec

Figure 1.3

traffic which is interpreted in the brain as a host of inappropriate sensations and can result in impaired motor function [Eisen, 1983].

Multiple Sclerosis causes abnormalities of cerebral evoked potentials. There have been many studies reporting the use of Somatosensory Evoked Potentials in MS [Chiappa, 1982]. In cases where there is definite MS, and disease referable to the nerves being tested, the diagnostic accuracy of these tests approaches 100%. However, in suspected MS, a stage of the disease in which there is a real need for a good laboratory test, the diagnostic accuracy is poor, usually less than 40%, and if the patient is asymptomatic in the nerve system being stimulated the diagnostic accuracy is very poor.

The primary purpose of this work was to develop a more subtle and objective physiological test for MS, which, when used in combination with other laboratory tests, might allow earlier diagnosis. While there is currently no effective treatment for MS, the ability to diagnose the disease in its early stages will, undoubtably, be essential to any potential therapy.



A Normal Median SEP
2.000
1.000
μV
0.000
-1.000
-2.000
8.0
12.0
16.0
20.0
24.0
msec

Figure 1.4

This thesis addresses the following aspects:

1) The design of research facilities for the Neuromuscular Diseases Unit.

2) The isolation of features of the Somatosensory Evoked Potential that can be used as a diagnostic test for Multiple Sclerosis.

3) The evaluation of the sensitivity and reliability of the isolated features in diagnosing Multiple Sclerosis.

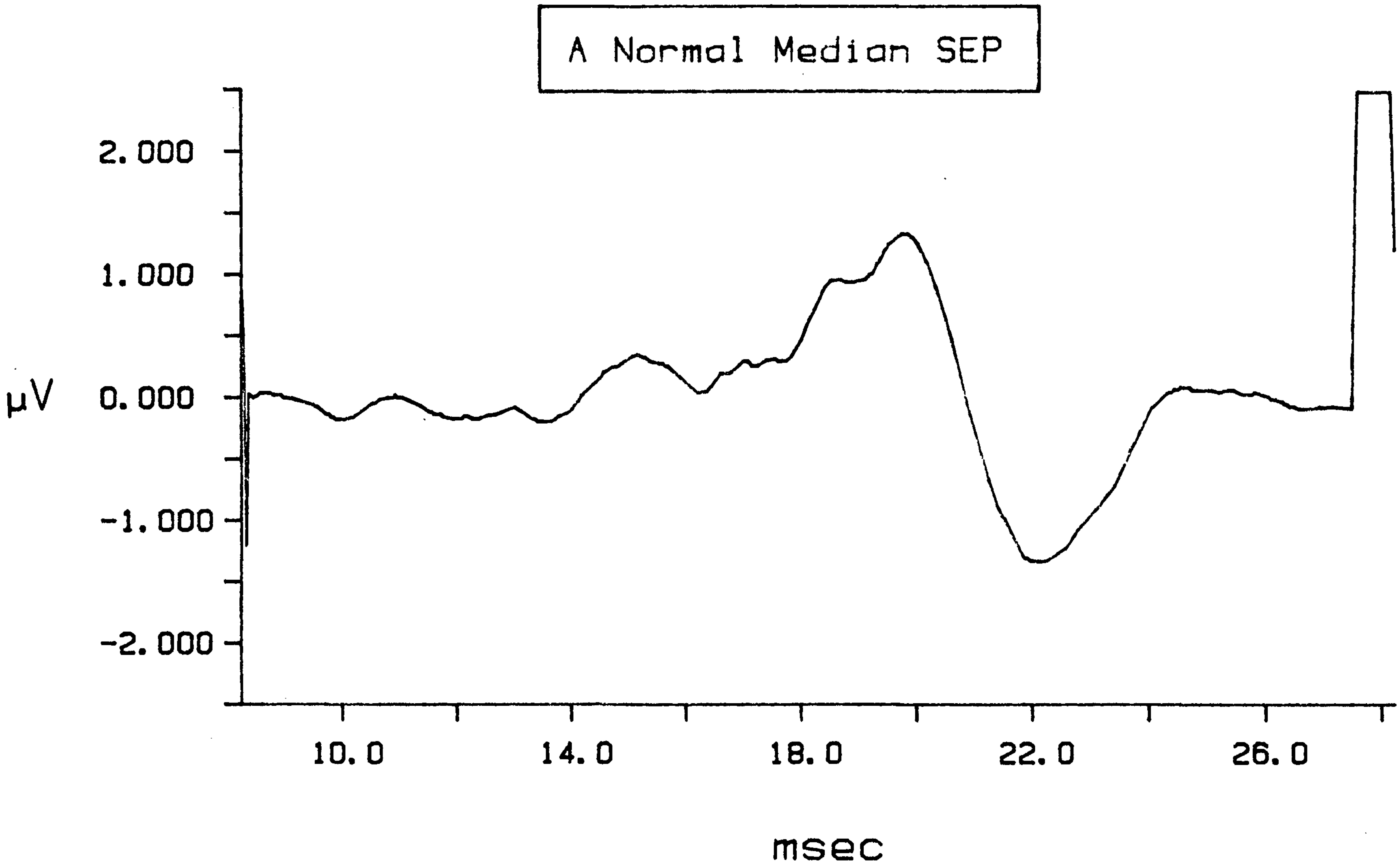

A Normal Median SEP
2.000
1.000
0.000
-1.000
-2.000
μV
10.0
14.0
18.0
22.0
26.0
msec

Figure 1.5

2)

EXAMINATION OF

SOMATOSENSORY EVOKED POTENTIALS

USING DIGITAL FILTERING

2.1) Introduction

Many of the early peaks of the SEP are inconsistent or of such small amplitude as to make them difficult to measure. A high-pass filter can be used to enhance the early latency peaks by removing the lower frequency components and thereby making the early peaks easier to measure [King and Green, 1979; Rossini et al, 1981; Lueders et al, 1981; Lueders et al, 1983; Maccabee et al, 1983]. Cephalic bipolar recording is popular in the routine clinical setting, and so it is of interest to explore the value of high-pass filtering of waves recorded with this technique.

Although high-pass analog filtering was investigated in this laboratory, it had several serious drawbacks. A 48 dB/octave Krohn-Hite Butterworth analog filter was inserted after a DISA 15C01 amplifier. The filter's sharp cut-off made it possible to remove components of the SEP below 300 Hz. However, several inherent features of the analog filter severely distorted the signal. When the impulse-like stimulus artifact passed through the filter it produced a slowly decaying 300 Hz ringing superimposed on the SEP (Figure 2.19). The nonlinear phase characteristic of the analog filters meant that different frequency components were delayed by different amounts. The amount of ringing and phase shift varied with the roll-off frequency settings of the filters [Doyle, 1983; Doyle, 1981; Scherg, 1982; Wastell, 1979; Dawson, 1973] (Figure 2.20). This made identification of peaks and measurement of latencies difficult and inaccurate.

Digital filtering eliminated all of these problems. Through use of a sweep delay, the stimulus artifact is not digitized and so does not even enter the digital filter. The digital filter was designed to introduce no delay. One SEP may be digitally filtered in several different ways without destroying the original data. This allows several different features to be highlighted from one recording.

In this chapter it will be shown that several components of the SEP recorded using cephalic bipolar techniques can be made more easily identifiable by the use of a digital filter.

2.2 Methods

Twenty-four normal subjects volunteered for this study. There were fourteen females (ages 28 ± 12)and ten males (ages 24 ± 3).

Square current pulses from a DISA 15E25 stimulator were delivered through silver discs separated longitudinally by about 40 mm, positioned along the median nerve at the wrist. Stimulus duration was 1.0 ms. This long duration stimulus was chosen because it selectively excites afferent, predominantly group Ia, nerve fibers (Veale et al, 1973), which mediate the SEP (Gandevia et al, 1983). The stimulus intensity was adjusted to induce a visibly minimal muscle twitch. The stimuli were delivered at a rate of 4 to 5 Hz. Scalp needle electrodes were used for recording. They were positioned at C3 or C4 (International 10-20 system) contralateral to the side of stimulation and at Fpz [Jasper, 1958].

The signal was amplified by a Disa 15C01 EMG amplifier passing a band from 10 Hz to 10,000 Hz Next the signal was low pass filtered using a Krohn-Hite 3342 Butterworth filter set with a cut off frequency of 2500 Hz and a roll off of 48 dB/octave. The filtered signal was sampled at 20 microsecond intervals for

20 ms and then quantized to eight bit accuracy. The sampling rate of 50 KHz, which was ten times the Nyquist minimum sampling rate, was used to reduce the quantizing error.

Longer epochs sampled at 10 KHz were also recorded to observe the later components of the SEP. In 50% of the volunteers the noise was also measured by recording with no stimulus.

In order to reduce the variation in latency due to arm length, and to avoid the stimulus artifact, the start of the sampling was delayed after the stimulation according to the formula: **D=0.15L-3.0** where L was the length of the subjects arm measured from wrist to C7 and D was the delay in milliseconds. This formula was derived from results published in [Eisen and Nudleman, 1978]. In their figure 3 the regression line relating the arm length to the onset latency of the cervical somatosensory response had the equation Y=0.15X+0.98 with X in centimeters and Y in milliseconds. The r value for this regression was 0.84 . From laboratory data it was decided that starting the epoch 4.0 milliseconds before the onset of the SEP would select the most informative region of the wave. Therefore, the delay becomes D=0.15L-3.0 .

The signal next passed through an eight bit analog to digital converter in the Disa digital control unit. The converter clipped the signal at $\pm$ 8 microvolts. If one trace had more than two such clipped regions the signal was rejected and was not used in the averaging. The 1024 eight bit values composing one sweep were added to the contents of the memory of the averager. Typically 1024 individual sweeps were added into the memory.

The 1024 eight bit values, comprising the averaged wave, and the 64 bytes of switch-setting information, were passed to the laboratory PDP-11/23. The program EMG (Documented in Appendix 1) stored the data and further subject information in duplicate data files and added an entry to the index of waves recorded.

The program FILTER (Documented in Appendix 2) was used to read in a selected data file and digitally filter it. FILTER first removed any DC bias from the data and then applied a Blackman window (Figure 2.1) [Oppenheim + Schafer,1975] between samples 10 and 960 of the 1024 samples comprising the wave. The Blackman window was chosen because of its very low side lobes(Figure 2.2). The Fast Fourier Transform [Cooley + Tukey, 1965; Brenner, 1980] of the windowed data was calculated. The components between 300 and 2500 Hz were selected by windowing the spectrum. The spectral window used had a square lower end and a one quarter width Blackman upper frequency end (Figure 2.3). This window provided a sharp cutoff at the low frequency end

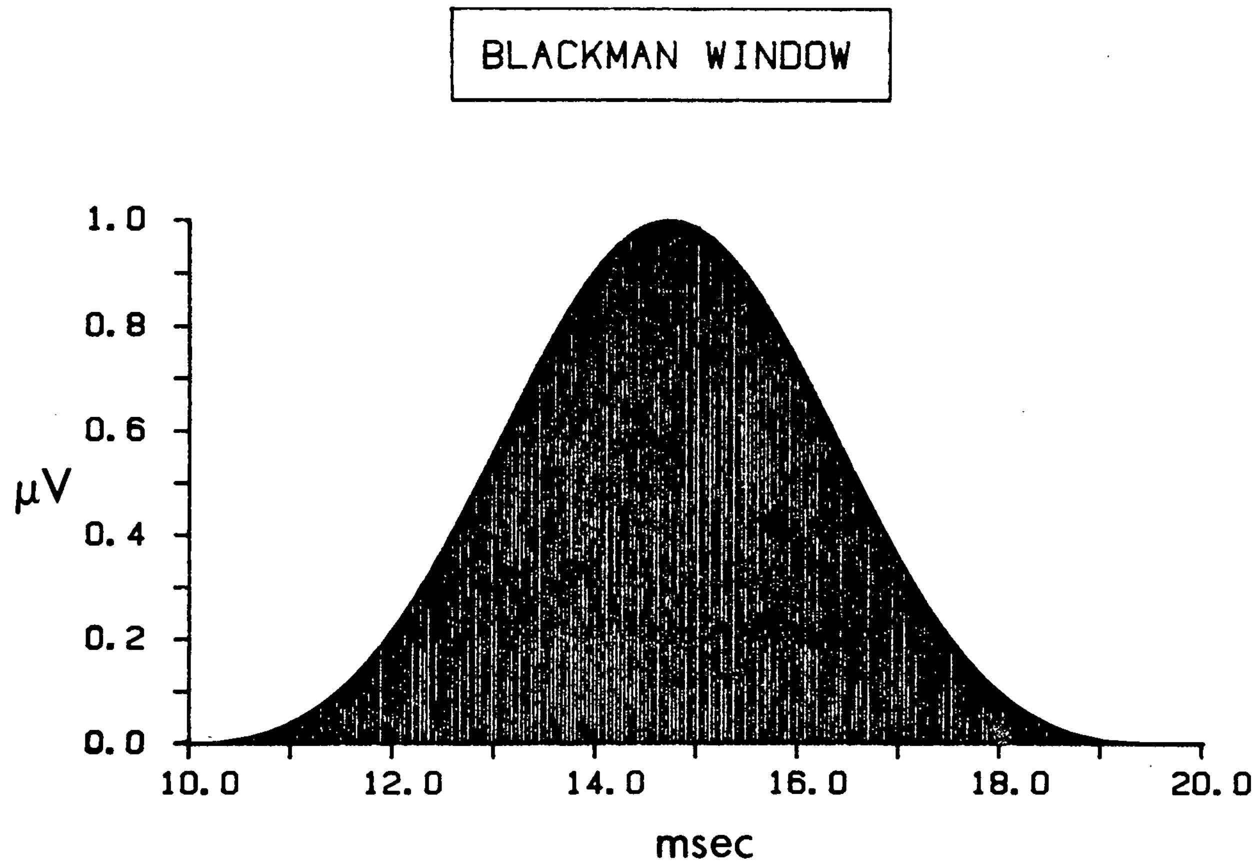


Figure 2.1 The waveform is multiplied by a Blackman window before being transformed.

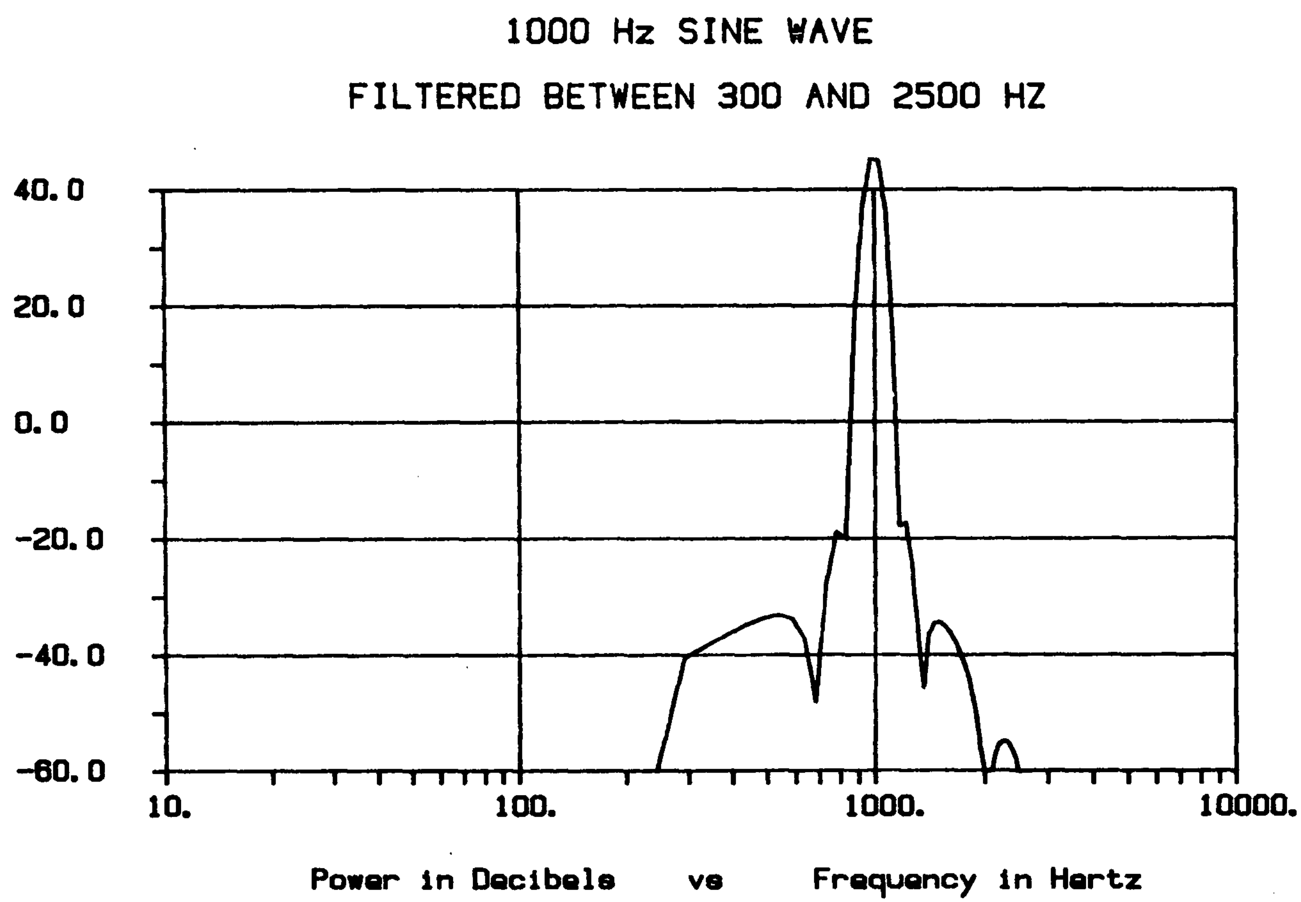


Figure 2.2 The Blackman window produces very low side lobes as demonstrated by this spectra of a filtered sine wave.

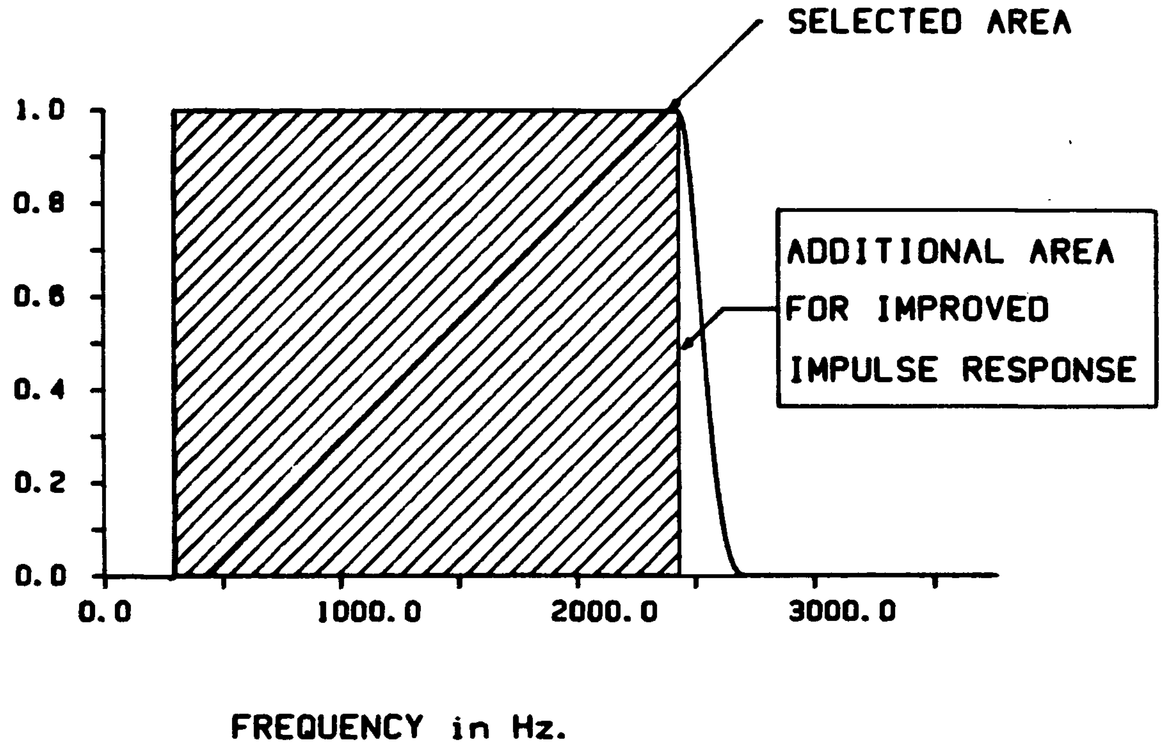


Figure 2.3 The spectral window passed frequencies between 300 and 2500 Hz.

while reducing the ringing that a sharp high frequency cut-off would produce (Figures 2.4, and 2.5). The Inverse Fast Fourier Transform was then applied to produce the filtered waveform that was stored in an EMG data file. The program EMG could be used to plot, display or manipulate the filtered signal.

2.3) Results

After digital filtering, the SEP evoked by median nerve stimulation had a stereotype morphology which (see Figures 2.6, 2.7, and 2.8) consisted of a series of ripples. Their peak and interpeak latencies are summarized in Tables I and II. Four negative (N16, N18, N19, and N20) and four positive (P15, P17, P18, and P19) peaks were invariably recognized. An early negative peak, N14, and a later positive peak, P21, were also frequently seen (Table I). Negative to negative and positive to positive interpeak latencies between N16 and P21 averaged 1.3 $\pm$ 0.22 ms (range 1 - 1.7 ms). These peaks are only rarely recognizable in the original SEP. The SEPs in figures 2.7-2.9 are exeptional in that the peaks can be recognized without filtering. These were chosen for the figures to show the correspondance between peaks of the filtered and unfiltered SEPs.

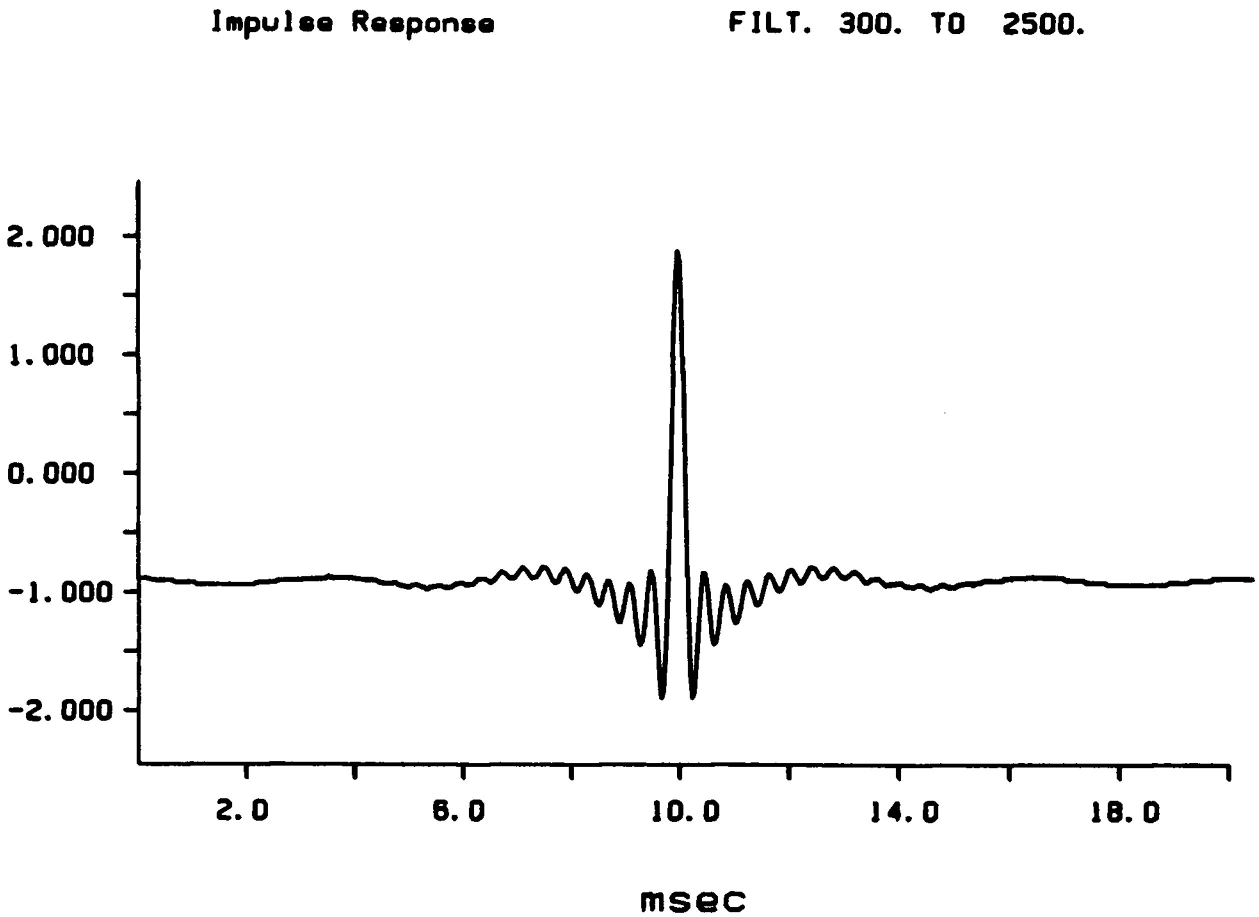


Figure 2.4 The impulse response of the digital filter was noncausal and symmetric.

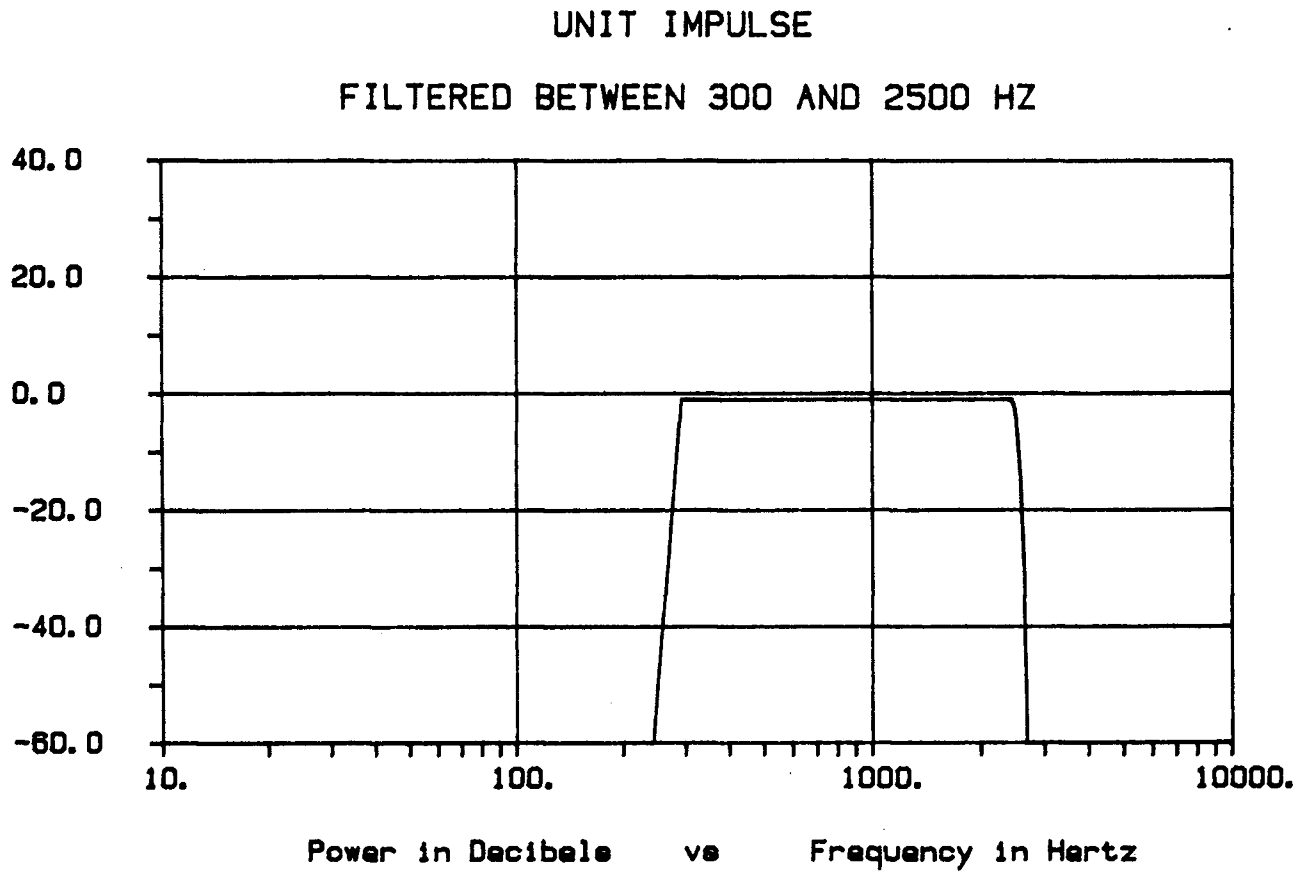


Figure 2.5 The frequency response of the digital filture to a unit impulse.

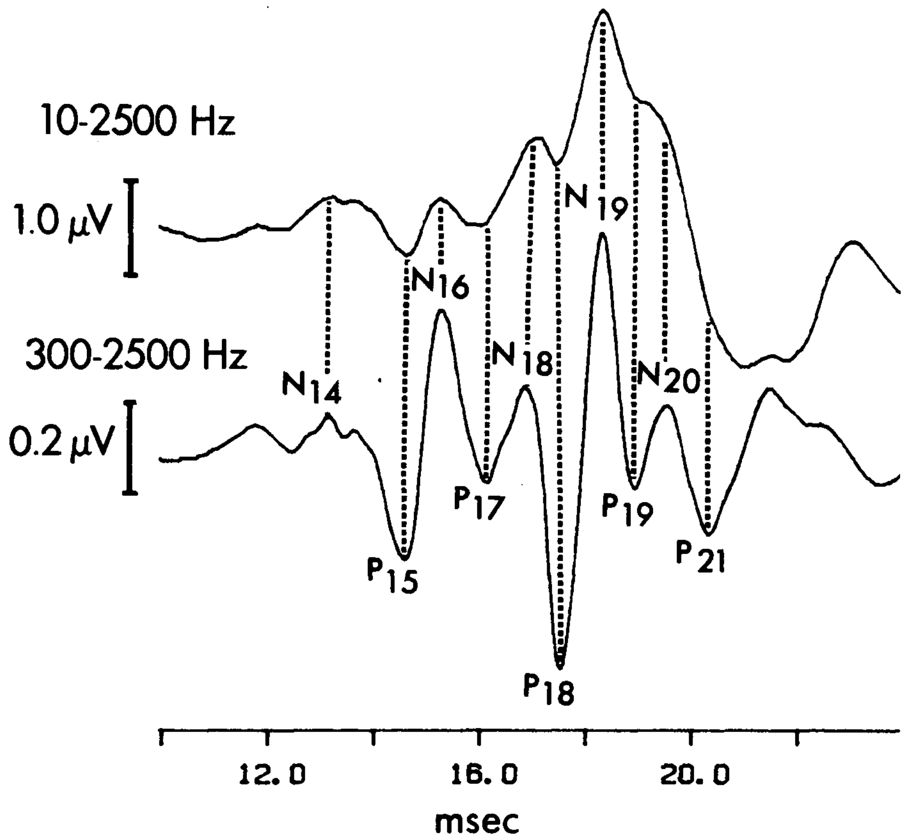


Figure 2.6 A normal SEP unfiltered (top) and filtered (bottom). The peaks of the filtered signal are recognizable in the unfiltered SEP from this person. This is not usually the case.

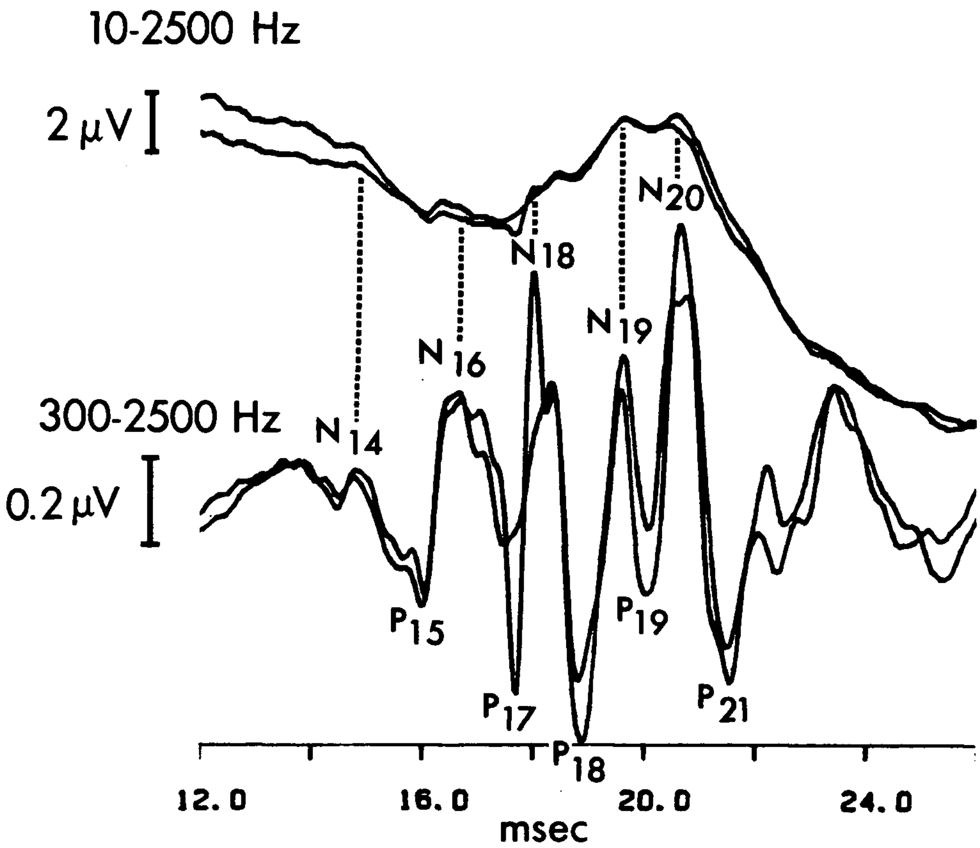


Figure 2.7 Two separate recordings of the SEP from one person are shown unfiltered (top) and filtered (bottom). The differences between two superimposed waveforms are an indication of the noise present in the recordings.

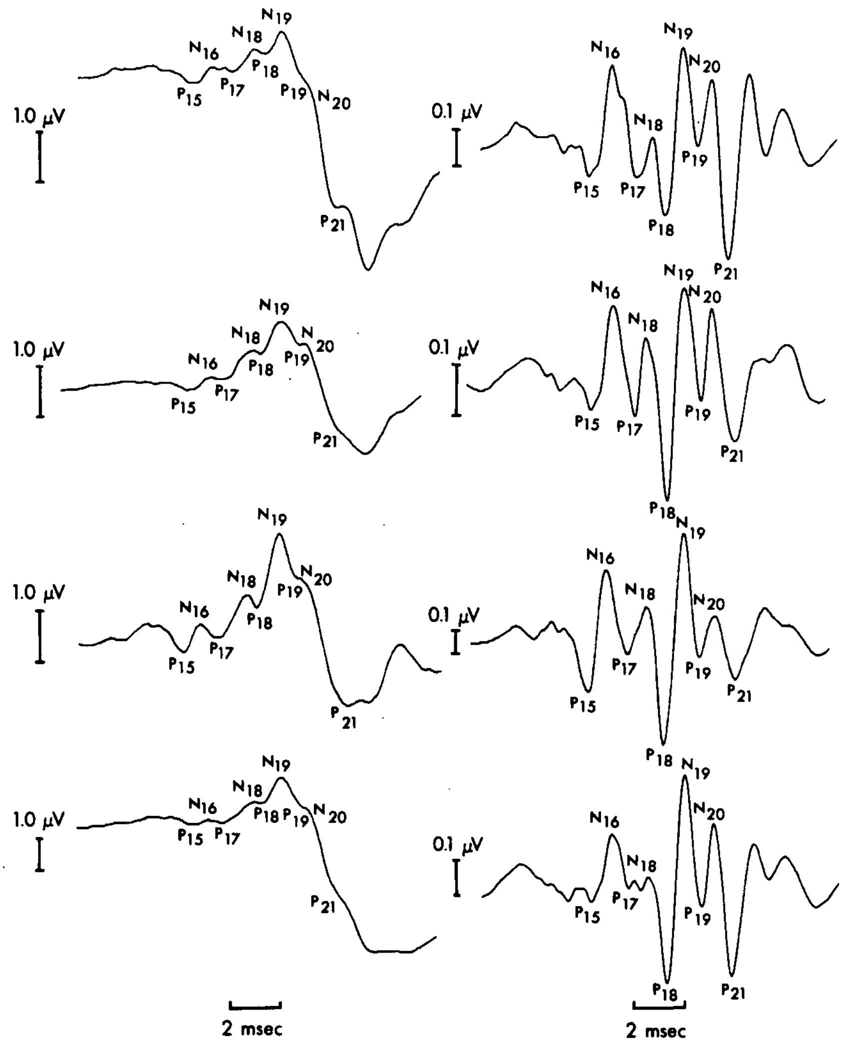


Figure 2.8 Four normal SEPs unfiltered (left) and filtered (right). The time axes have been shifted to line up the peaks and demonstrate their consistency.

Table I   Latencies (ms) of Peaks of Filtered SEPs

PEAK	NUMBER PRESENT	MEAN	SD	RANGE
N14	16	13.8	0.85	12.8 - 14.9
P15	24	15.3	0.82	13.7 - 16.9
N16	24	16.2	0.82	14.7 - 17.9
P17	24	17.0	0.89	15.1 - 18.5
N18	24	17.6	0.80	15.8 - 19.0
P18	24	18.4	0.82	16.6 - 20.9
N19	24	19.1	0.86	17.9 - 20.9
P19	24	19.5	0.88	17.9 - 21.2
N20	20	20.1	0.80	18.5 - 21.6
P21	14	20.7	0.96	19.8 - 22.4

Table II Interpeak Latencies (ms)

INTERVAL	# MEASURABLE	MEAN	SD	RANGE
		NEGATIVE PEAKS		
N14 - N16	16	2.4	0.4	1.9 - 3.0
N16 - N18	24	1.4	0.2	0.9 - 1.8
N18 - N19	24	1.5	0.2	1.1 - 1.9
N10 - N20	20	1.0	0.2	0.8 - 1.4
		POSITIVE PEAKS		
P15 - P17	20	1.7	0.2	1.3 - 2.2
P17 - P18	20	1.2	0.2	0.7 - 1.4
P18 - P19	24	1.3	0.1	1.0 - 1.5
P19 - P21	15	1.2	0.3	0.5 - 1.4

When a wider modified Blackman window (Figure 2.9) was used on longer sweeps (100 ms), it was apparent that the peaks extracted by digital filtering were restricted to the region of the "N19" component. (see Figures 2.10, 2.11). Similar identifiable peaks were not seen in association with later SEP components.

2.4) Discussion

The amplifier and analog filters were tested by putting a 0.1 ms wide pulse into the amplifier and measuring the impulse response. Figure 2.12 shows that the decay of the 2500 Hz ringing is such that it was 35 dB down from the signal after 5 ms. In the data collected the start of sampling was delayed by 6 to 8 milliseconds so the stimulus artifact was not a problem.

The frequency response of the amplifier and analog filters was measured to be flat ($\pm$0.3 dB) from 25 to 2000 Hz. The response was down 3 dB at 2500 Hz and down 44 dB at 3800 Hz.

To measure the amplifier, A/D, and enviromental noise, together with stimulus artifact, the recording electrodes and a ground were placed in a tray of saline solution that was then stimulated. 1024 epochs were averaged to produce a representative sample of the noise from all nonbiological sources

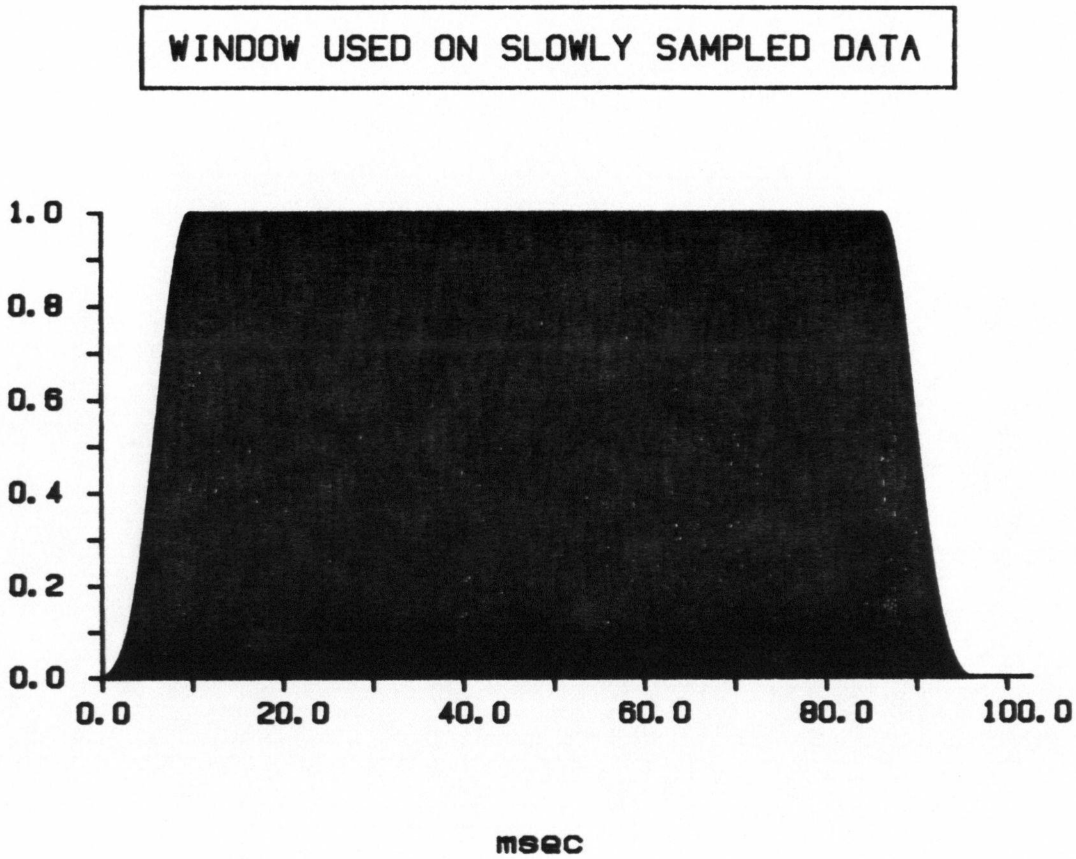


Figure 2.9 A modified Blackman window was used with longer waveforms.

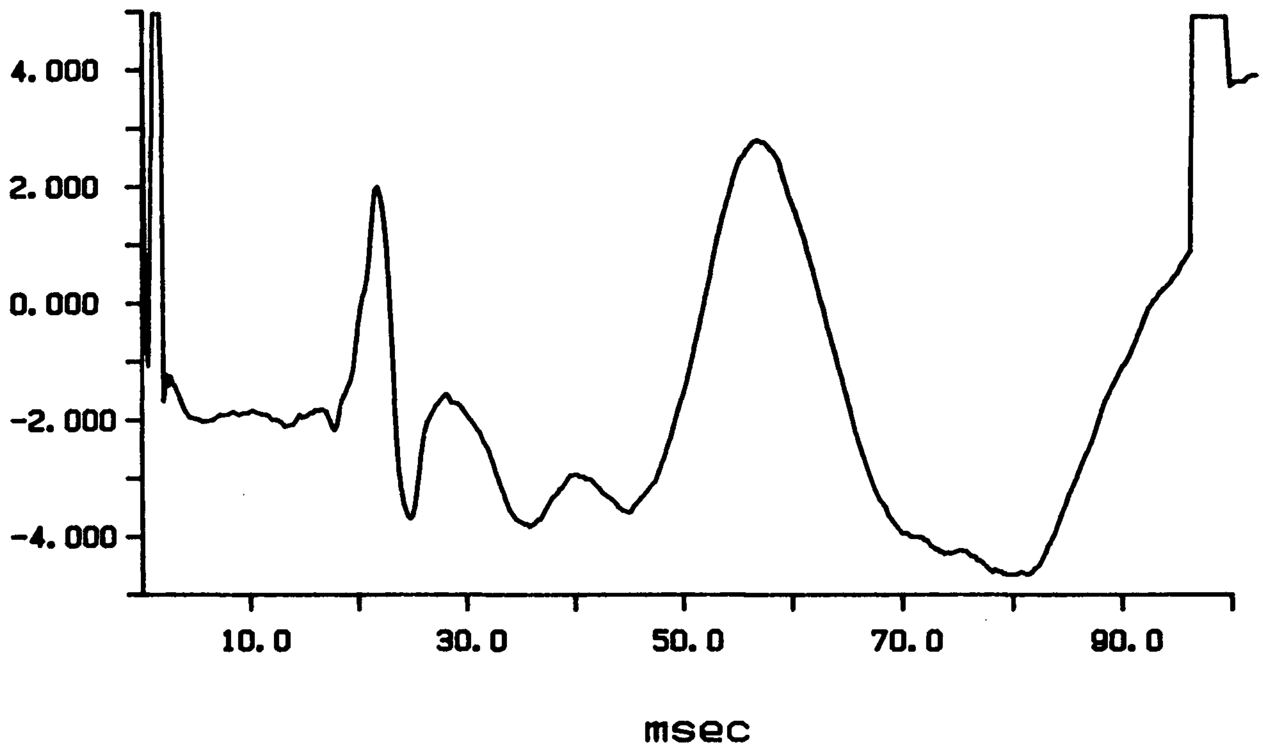


Figure 2.10 A normal SEP recorded with a longer (100 ms) epoch.

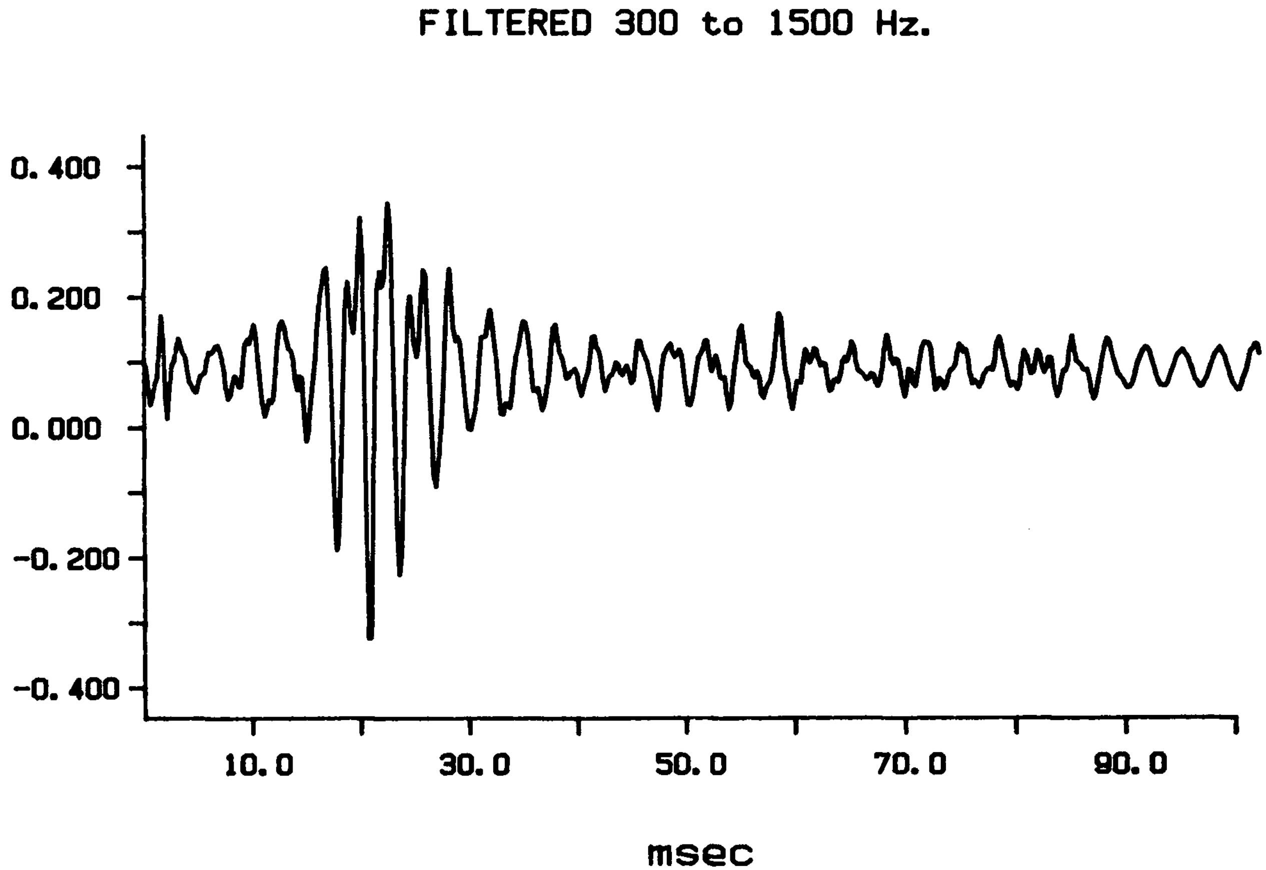


Figure 2.11 The result if digitally filtering the SEP of figure 2.10. The signal rises and falls around the 20 ms point.

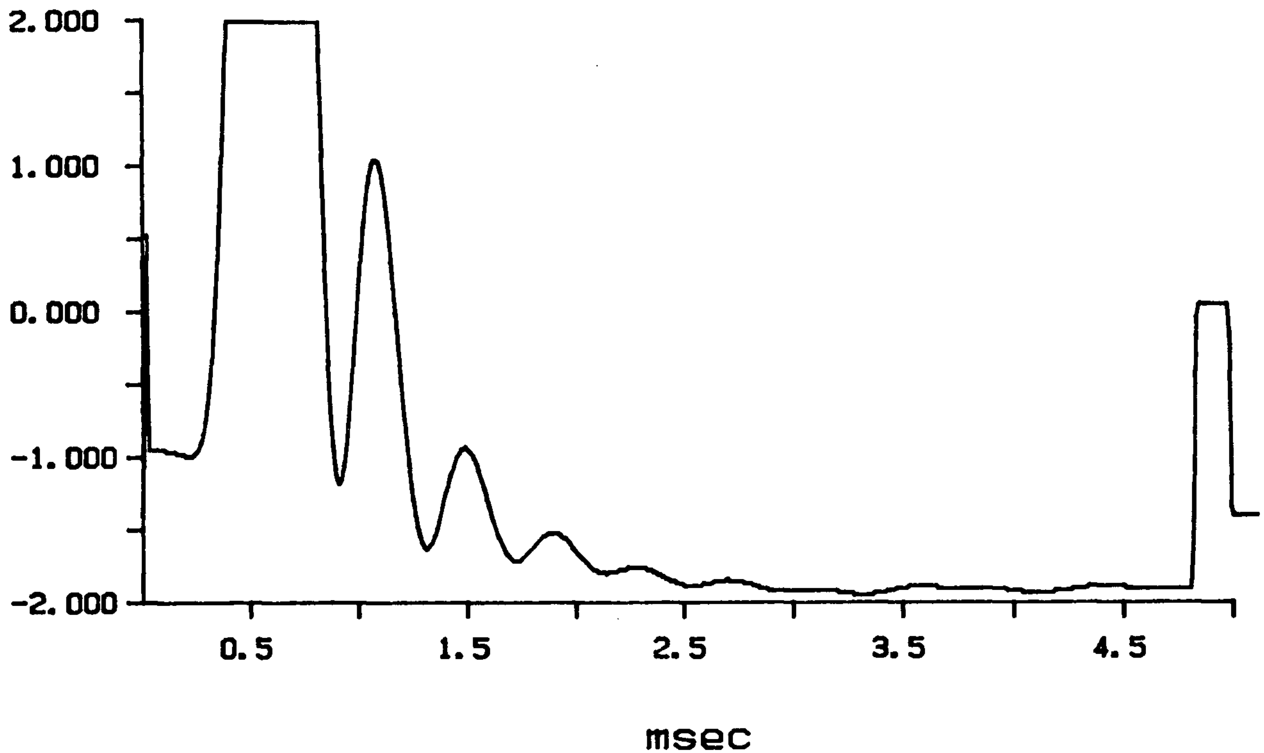


Figure 2.12 The impulse response of the DISA 15C01 amplifier and the Krohn-Hite 3342 filter used to record the SEPs.

which is shown in Figures 2.13 and 2.14. The typical SEP recording started eight ms after the stimulus. In the region of the typical recording the stimulus artifact is not present and the noise was 34 dB below the signal.

The total noise was measured, in 50% of the subjects, by averaging 1024 epochs of the SEP with the stimulator turned off (Figure 2.15). The total noise was typically 30 dB below the SEP after averaging 1024 epochs. If a subject was abnormally noisy the recordings were redone.

Figure 2.16 compares the spectra of: a median SEP, the same measurement with no stimulus, and the nonbiological noise. At frequencies above 1000 Hz the SEP of a normal subject was dominated by noise. Figure 2.17 shows the spectra of the SEP before and after filtering and demonstrates the sharp cutoff of the digital filter (greater than 50 dB per octave).

The digital filtering introduced no phase shift in the components of the filtered wave because the frequency selection window was strictly real and symmetric about the Nyquist frequency. The noncausal symmetric impulse response shown in Figure 2.4 illustrates this.

The Blackman Window changed the relative amplitude at different points of the waveform, depending upon where they were

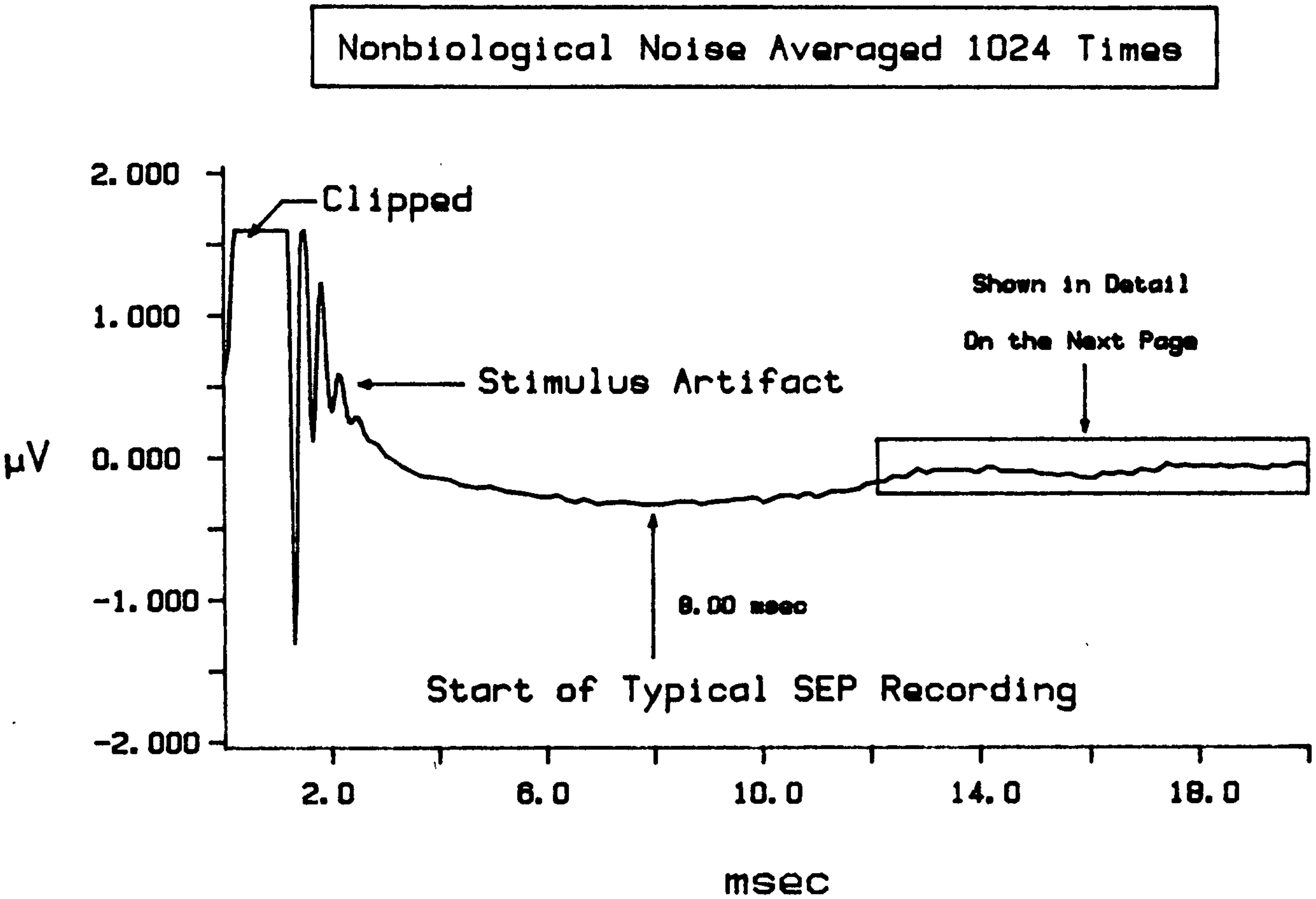


Figure 2.13 An estimate of the nonbiological noise present in an SEP recording was measured.

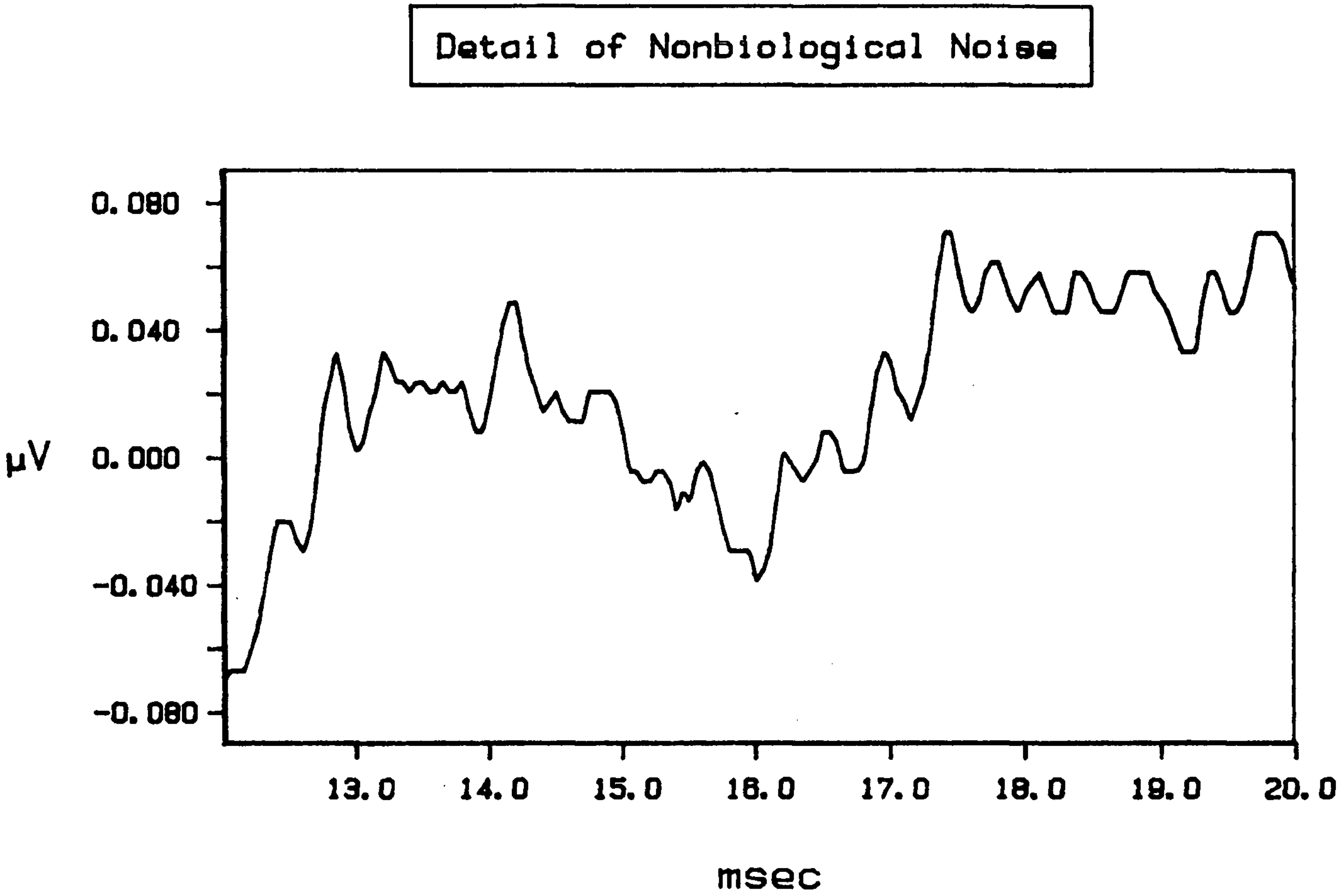


Figure 2.14 A detailed look at a section of figure 2.13.

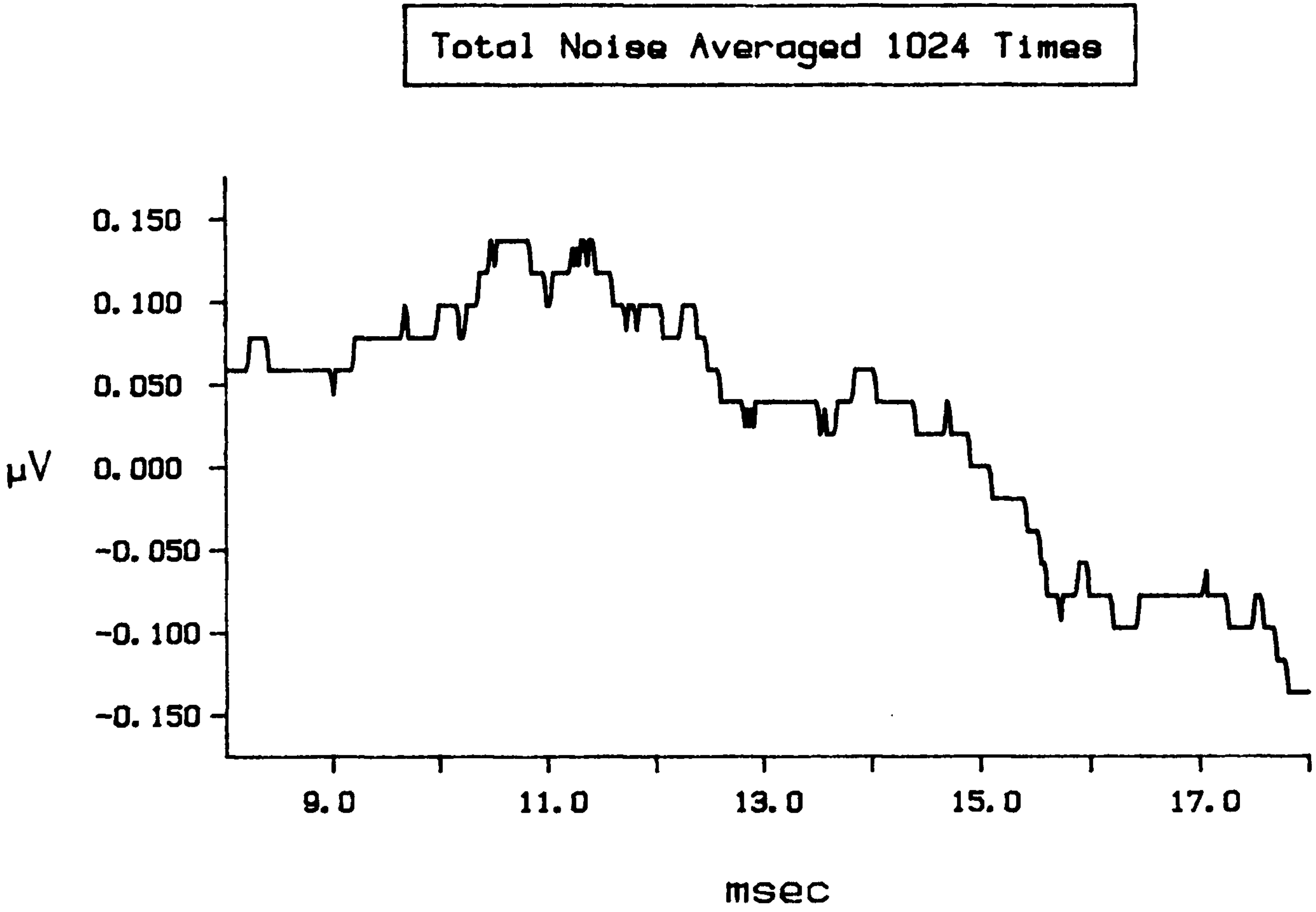


Figure 2.15 The total noise present in an SEP recording was estimated by recording the potential with no stimulus.

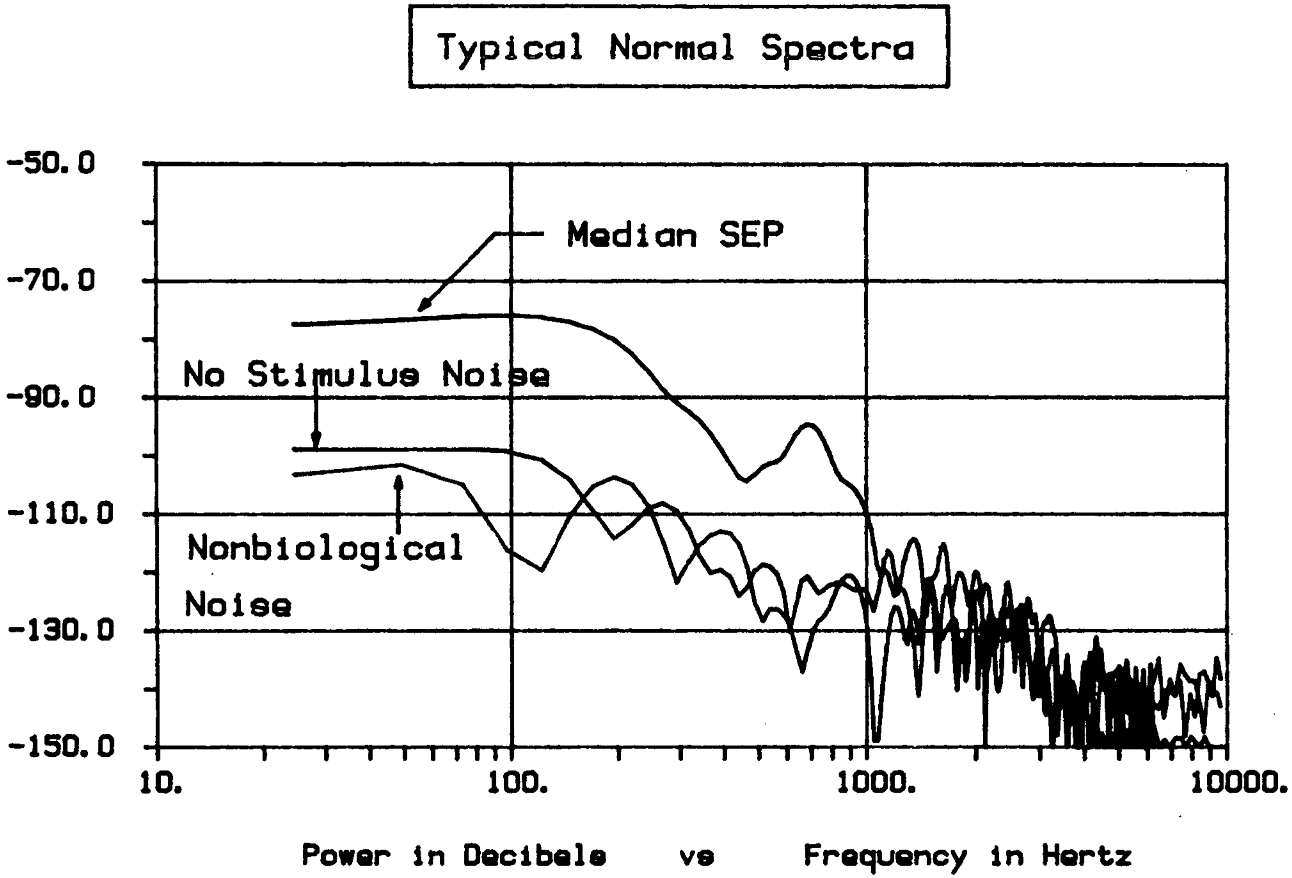


Figure 2.16 The spectra of an SEP and of noise measurements. At high frequencies the normal SEP becomes dominated by noise.

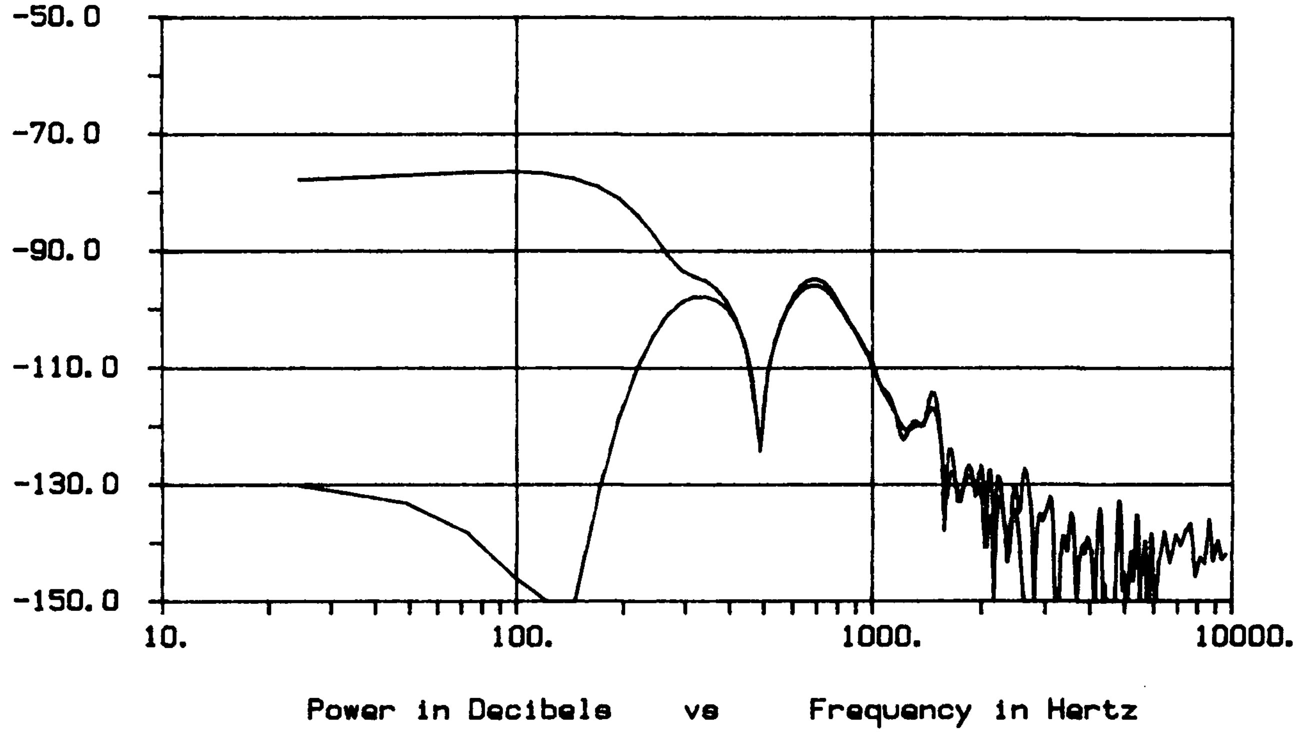


Figure 2.17 The spectra of a filtered and unfiltered SEP. The digital filter has a cutoff rate of greater than 50 dB per octave.

in the window. For this reason a modified Blackman Window (Figure 2.9) was used with 100 ms long sweeps when one wished to compare the amplitudes of different sections of the wave. The flat portion of the Modified Blackman Window allowed accurate comparisons of the amplitude of different peaks. The much longer sweep time overcame the effect of the larger window side lobes. When using the longer sweep times the digital filter band-pass used was 300 to 1500 Hz and the Krohn-Hite low pass filter was set to 1500 Hz.

When employing the Modified Blackman Window, it was apparent that the potentials extracted by digital filtering were restricted in distribution (see Figures 2.10, 2.11). They rose and fell fairly abruptly around "N19" with onsets at about 16 ms.

If the beginning or end of the wave is not close to the DC median, then multiplying by the Blackman window will produce distortions at the frequencies that are being considered. Figure 2.22 shows the Blackman window itself filtered between 300 and 2500 Hz. While these ripples were more than 30 dB below the signal in the central region, they are noticeable at the edges of many of the filtered waves.

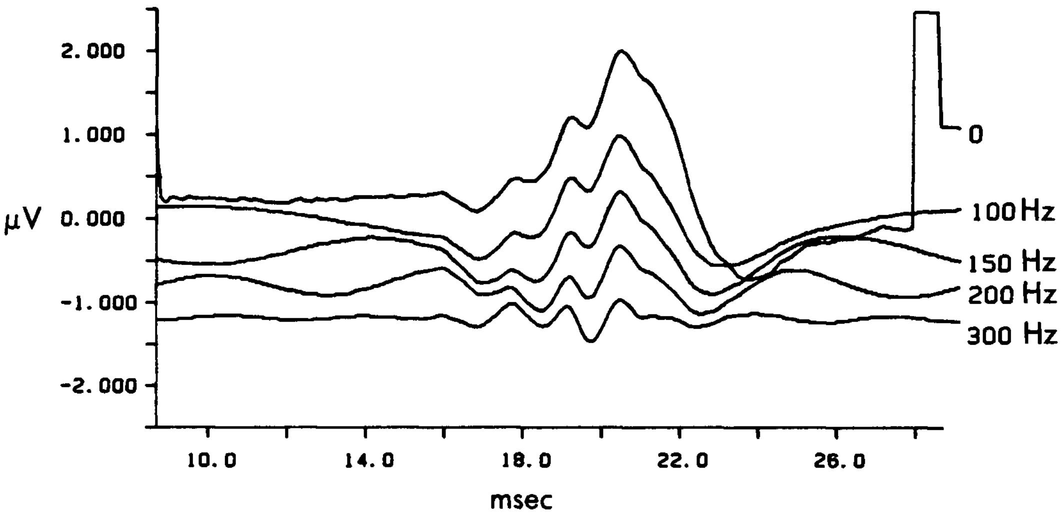


Figure 2.18 An SEP highpass filtered with various cutoff frequencies. The peaks remain aligned.

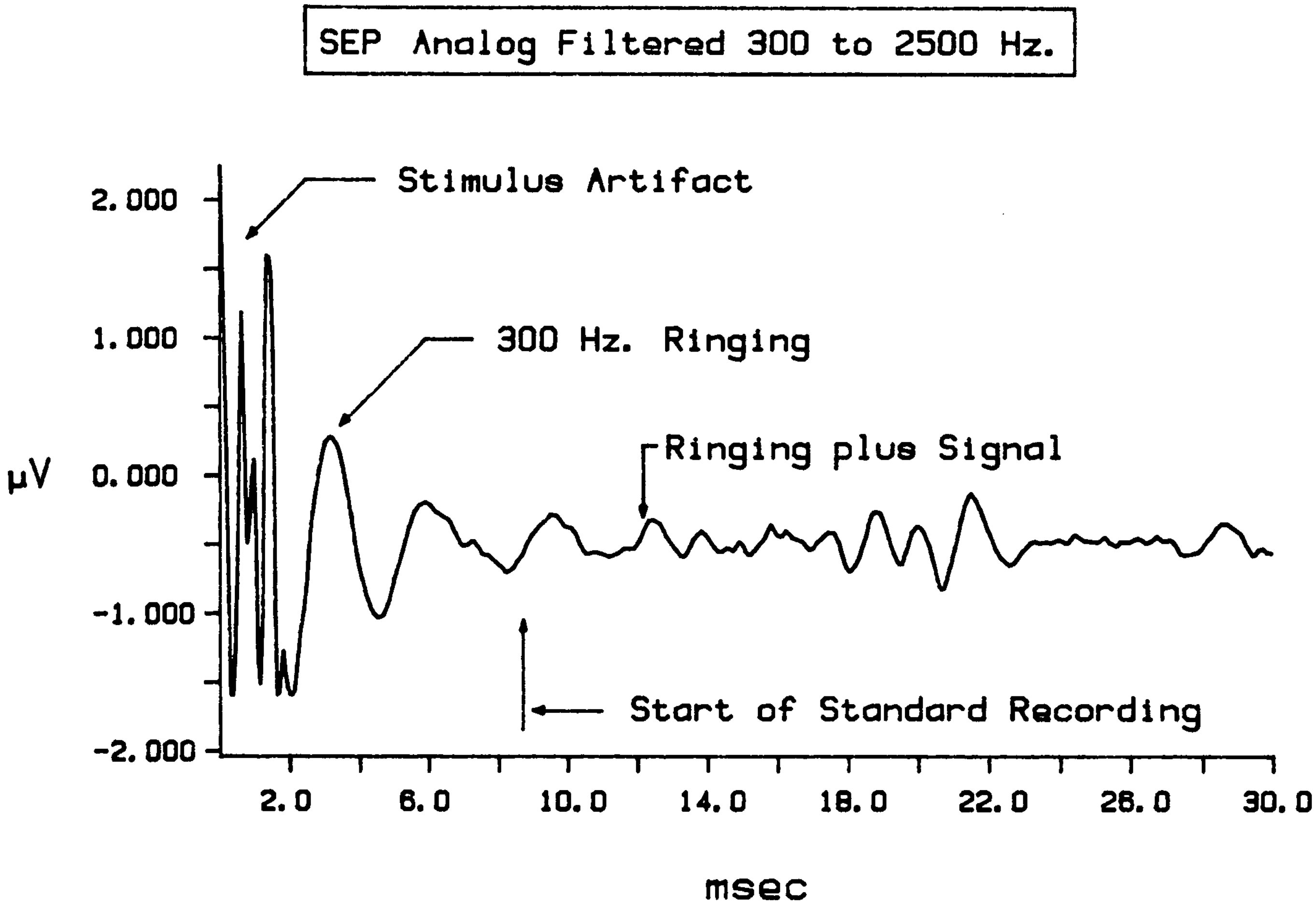


Figure 2.19 An analog highpass filter produced substantial ringing from the stimulus artifact.

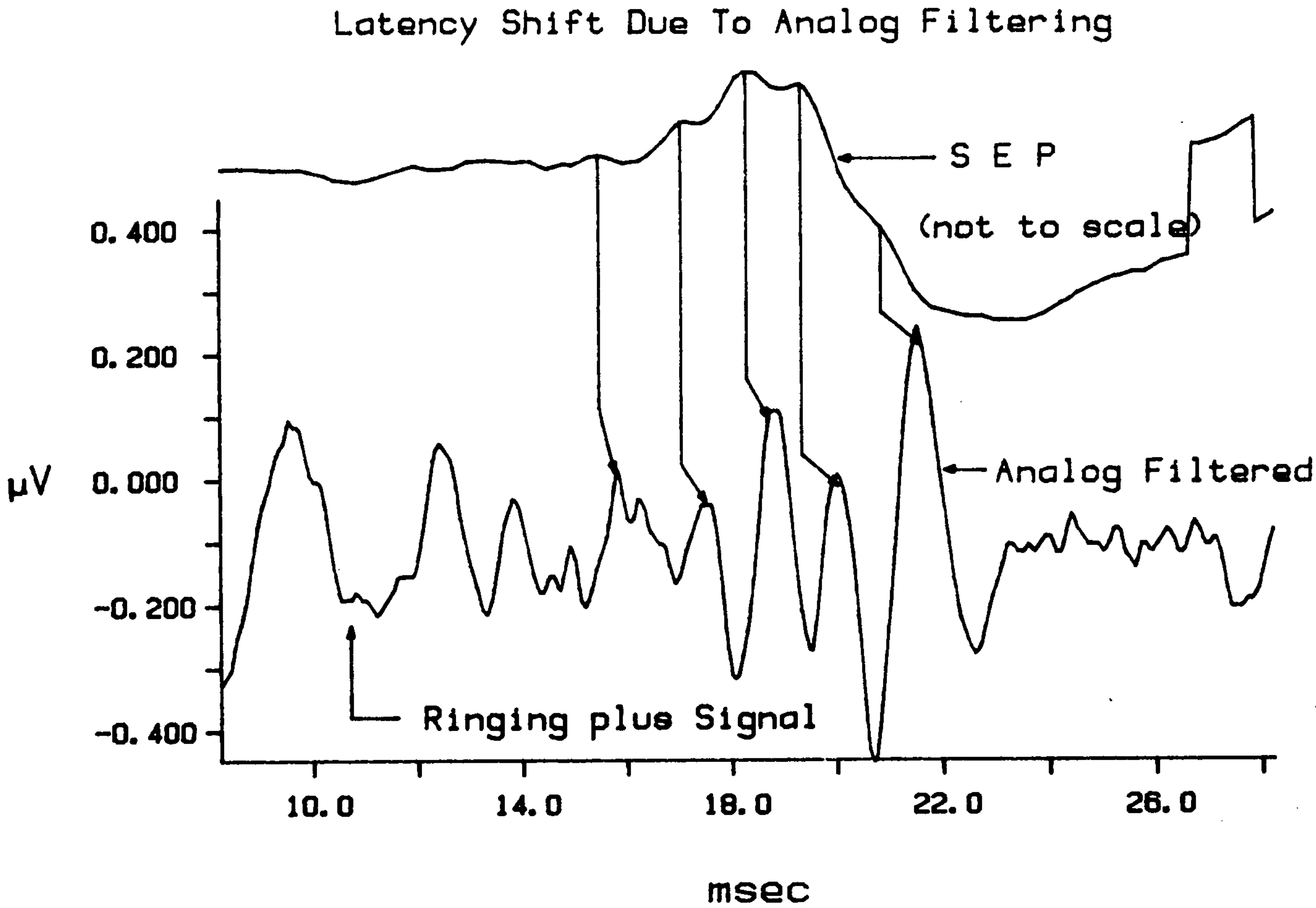


Figure 2.20 An analog highpass filter produced phase shift.

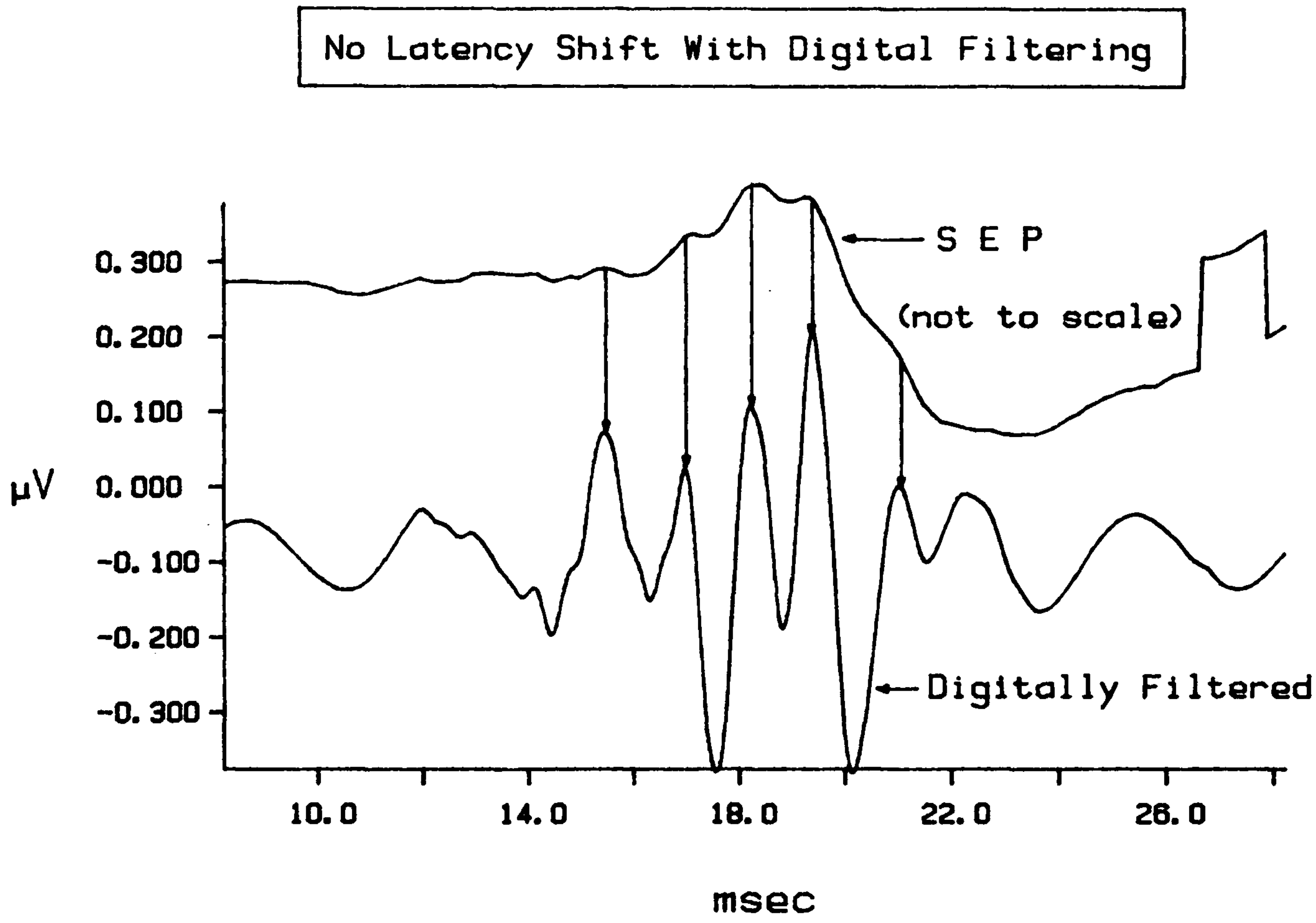


Figure 2.21 The digital filter produced no ringing and no phase shift.

In this study, digital filtering revealed several SEP components. For the following reasons they were not considered artifactual:

1. The same time locked, but less clearly identifiable components were recorded in most subjects before digital filtering (see figs 2.7, 2.8, and 2.9).

2. There was a latency shift of corresponding components in different subjects in keeping with variable limb length. On the other hand there was constancy of component latencies recorded at different times in a given individual.

3. When the Blackman window was shifted to the right or left, component peaks remained true to latency.

4. Analog filtering with a similar pass band has also allowed the extraction of similar small amplitude, high frequency components, but does induce phase shift (King and Green, 1979; Rossini et al, 1981; Luders et al,1983; Maccabee et al, 1983).

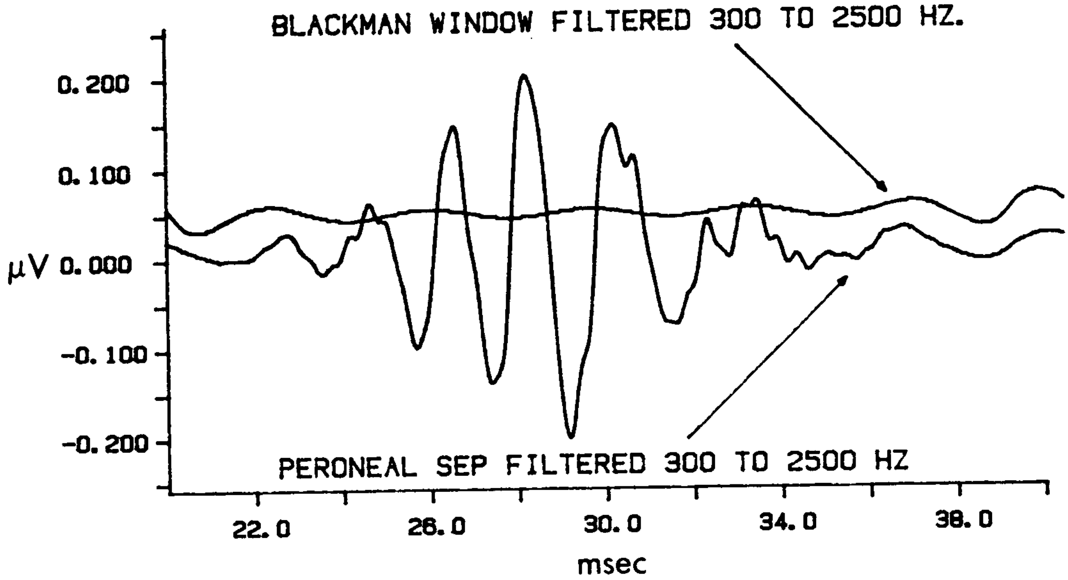


Figure 2.22 The edges of the filtered signals contain artifact due to windowing.

2.5) CONCLUSION

Digital filtering using the Blackman Window and a pass band of 300 to 2500 Hz was applied to Somatosensory Evoked Potentials recorded with cephalic bipolar montages. The digital filtering eliminated the problem of ringing from the stimulus artifact and the problem of phase shift variations of peak latencies.

Relative amplitude comparison was not accurate using the Blackman window; however, the use of longer sweep times and a modified window allowed accurate amplitude measurements when such were desired. The Blackman window introduced a small amount of higher frequency distortion at the edges of some waves. This artifact was easily recognisable and was not large enough to pose a problem (30 dB down).

Four consistent negative and corresponding positive peaks with latencies of about 16, 18, 19, and 20 ms were elicited with median nerve stimulation. The interpeak latencies measured 1.3 $\pm$ 0.2 ms. These peaks are rarely identifiable in unfiltered SEPs. Similar peaks have been elicited by lower limb stimulation where they are never visible in the unfiltered SEPs. The presence of the high frequency peaks in the SEP raises questions about the current theories of the origin of the SEP [Eisen, 1984]. The next chapter of this work uses these peaks to detect Multiple Sclerosis.

3) DYNAMIC TIME WARPING OF THE FILTERED SEP

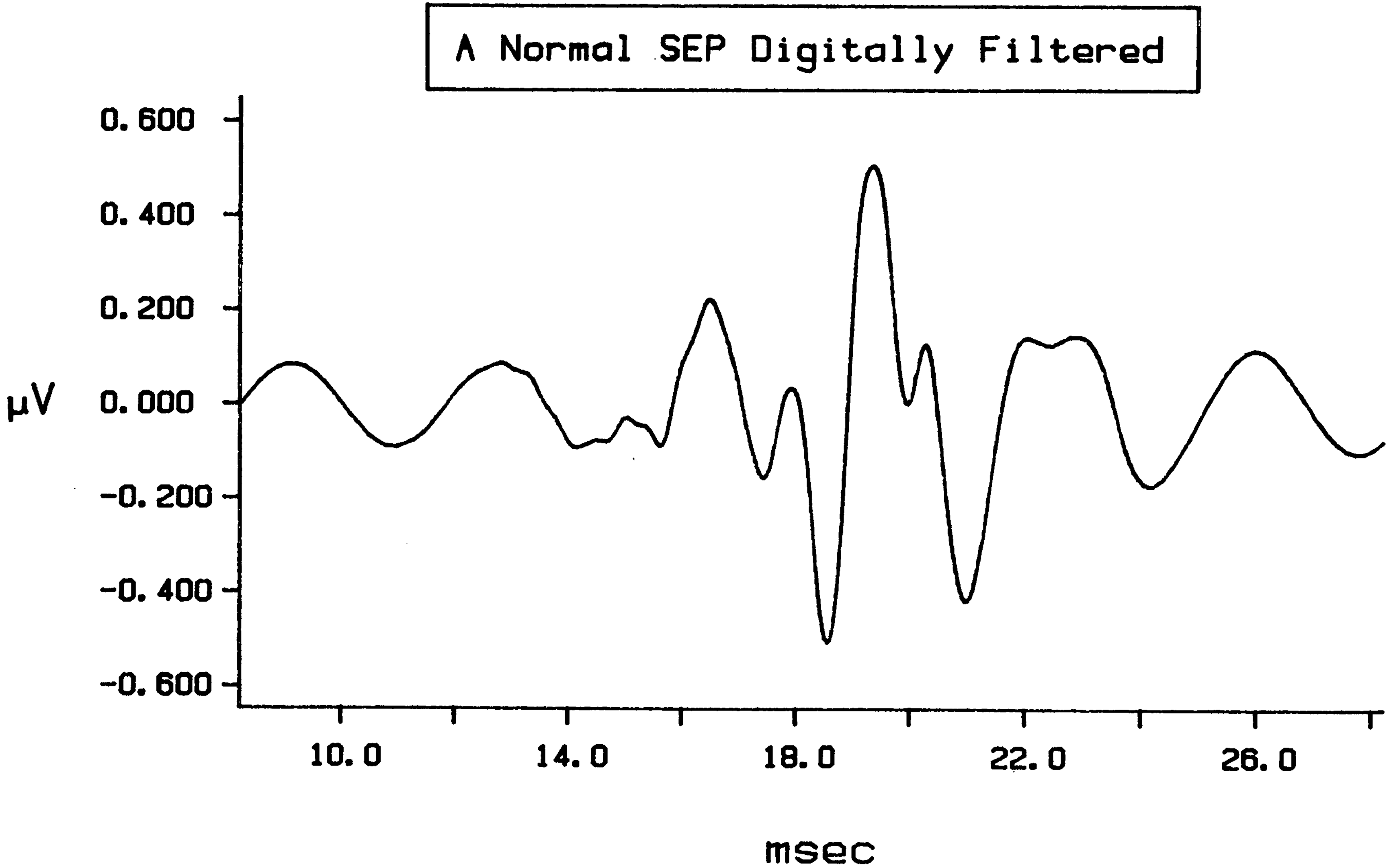
3.1) Introduction

The digitally filtered Somatosensory Evoked Potentials, as described in section 2, have a distinctive pattern of peaks (Figures 2.6, 2.7, 2.8, 3.1 - 3.5). However, not all peaks are strongly present in all waves, and there is variation in their latencies. This makes the accurate computer identification of the peaks a nontrivial problem.

Evoked Potential Waveforms have been represented as linear combinations of Pricipal Components [Suter, 1970; Aunon, 1981; Van Rotterdam, 1970]. This analysis is useful for quantifying changes in one person's response due to varying stimuli. Techniques such as Latency Corrected Averaging [Woody, 1967; Aunon, 1978; Aunon, 1981] have been used for correcting variations in the latencies of features of the evoked potential from one person. However, these techniques are not applicable to the problem of identifying corresponding features in many different people.

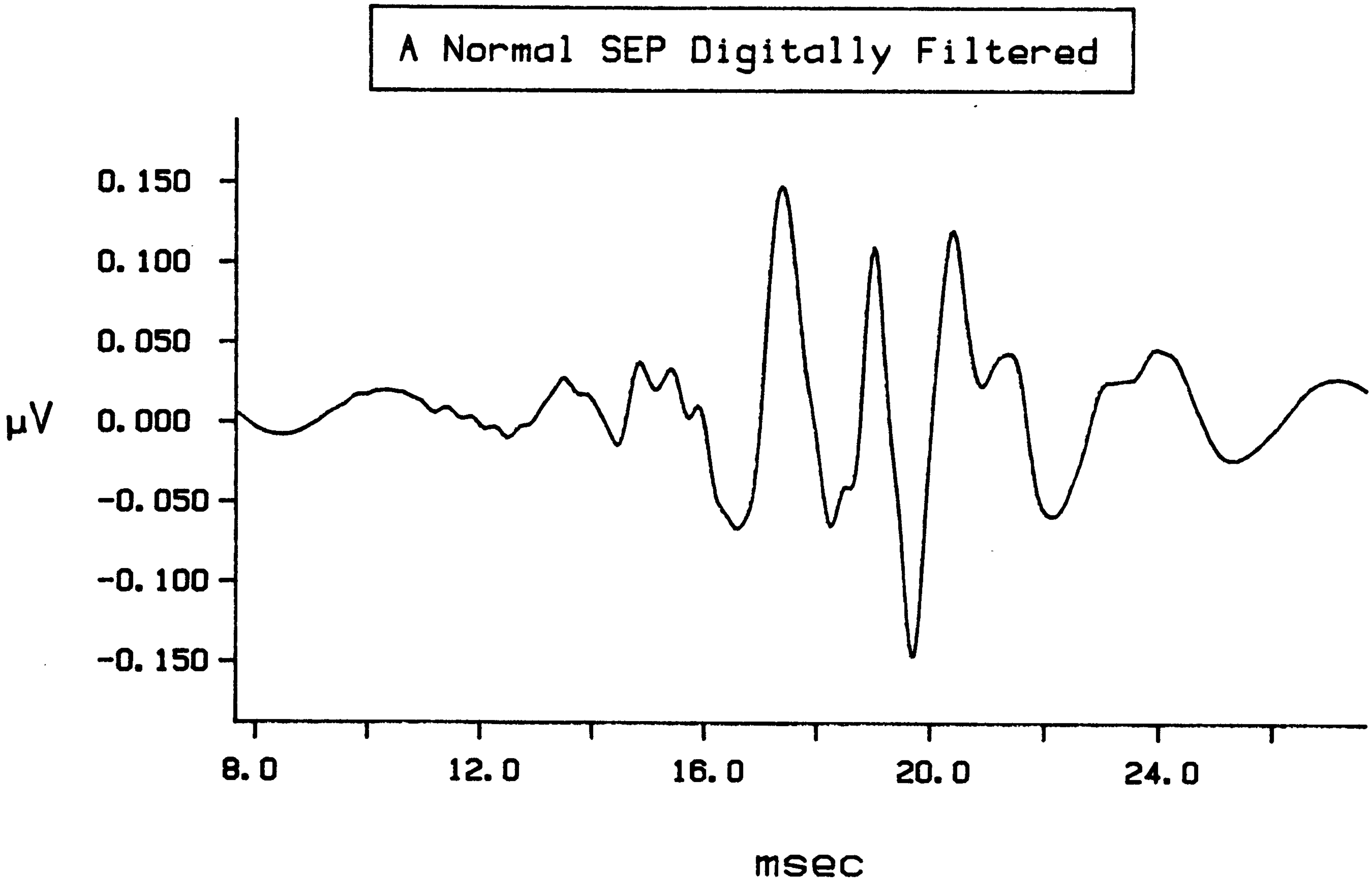
The method used in the present work to solve this problem is Dynamic Time Warping which can find an optimal time adjustment (shifting, stretching and contracting) between waveforms.

<u>Dynamic Time Warping</u> is the use of the Dynamic Programming Algorithm [Bellman, 1957; Whittle,1982] to find from given waves **a** and **b** the warping function **w** that minimizes a specified cost.



A Normal SEP Digitally Filtered
0.600
0.400
0.200
0.000
-0.200
-0.400
-0.600
µV
10.0
14.0
18.0
22.0
26.0
msec

Figure 3.1



A Normal SEP Digitally Filtered
μV
0.150
0.100
0.050
0.000
-0.050
-0.100
-0.150
8.0
12.0
16.0
20.0
24.0
msec

Figure 3.2

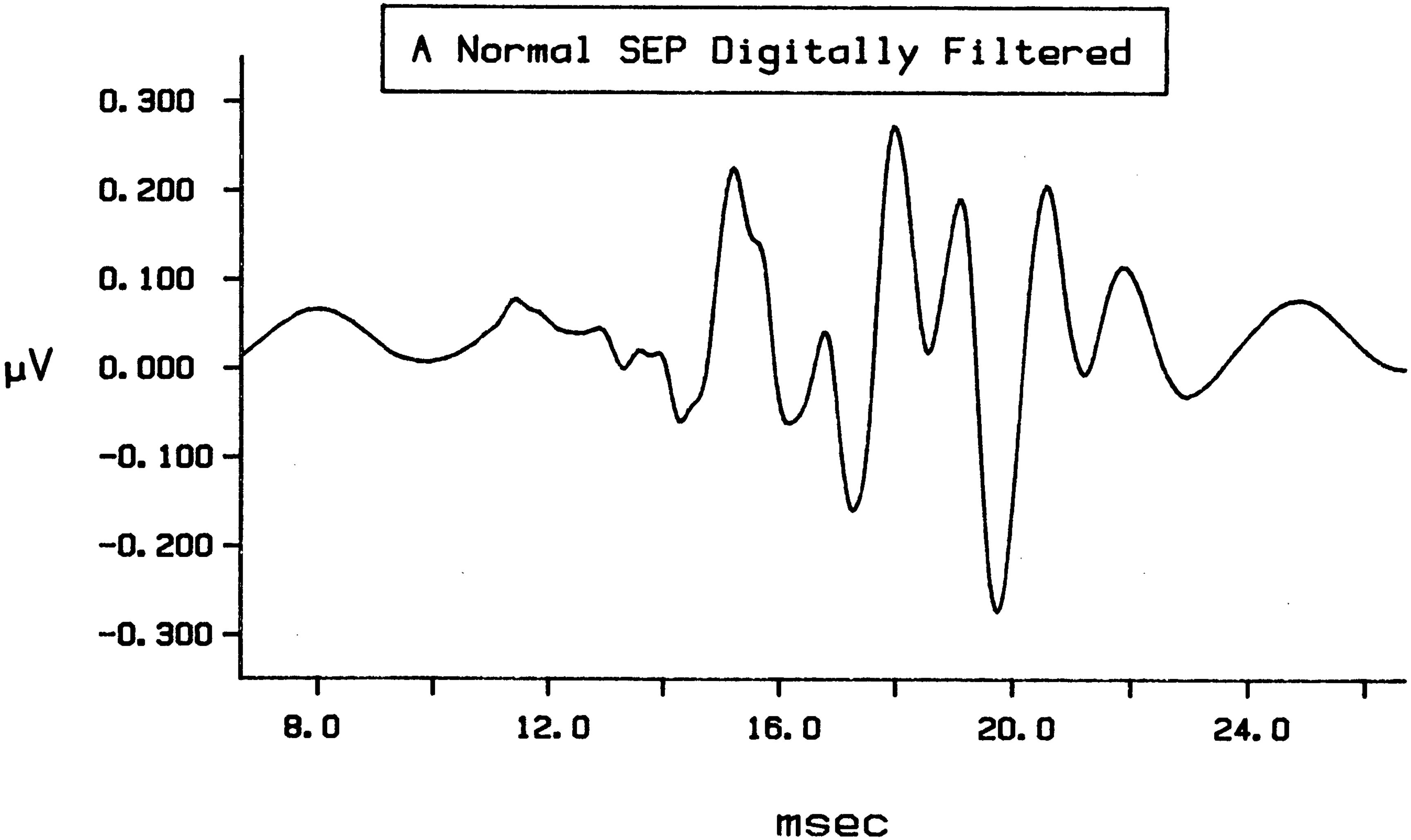

A Normal SEP Digitally Filtered
μV
0.300
0.200
0.100
0.000
-0.100
-0.200
-0.300
8.0
12.0
16.0
20.0
24.0
msec

Figure 3.3

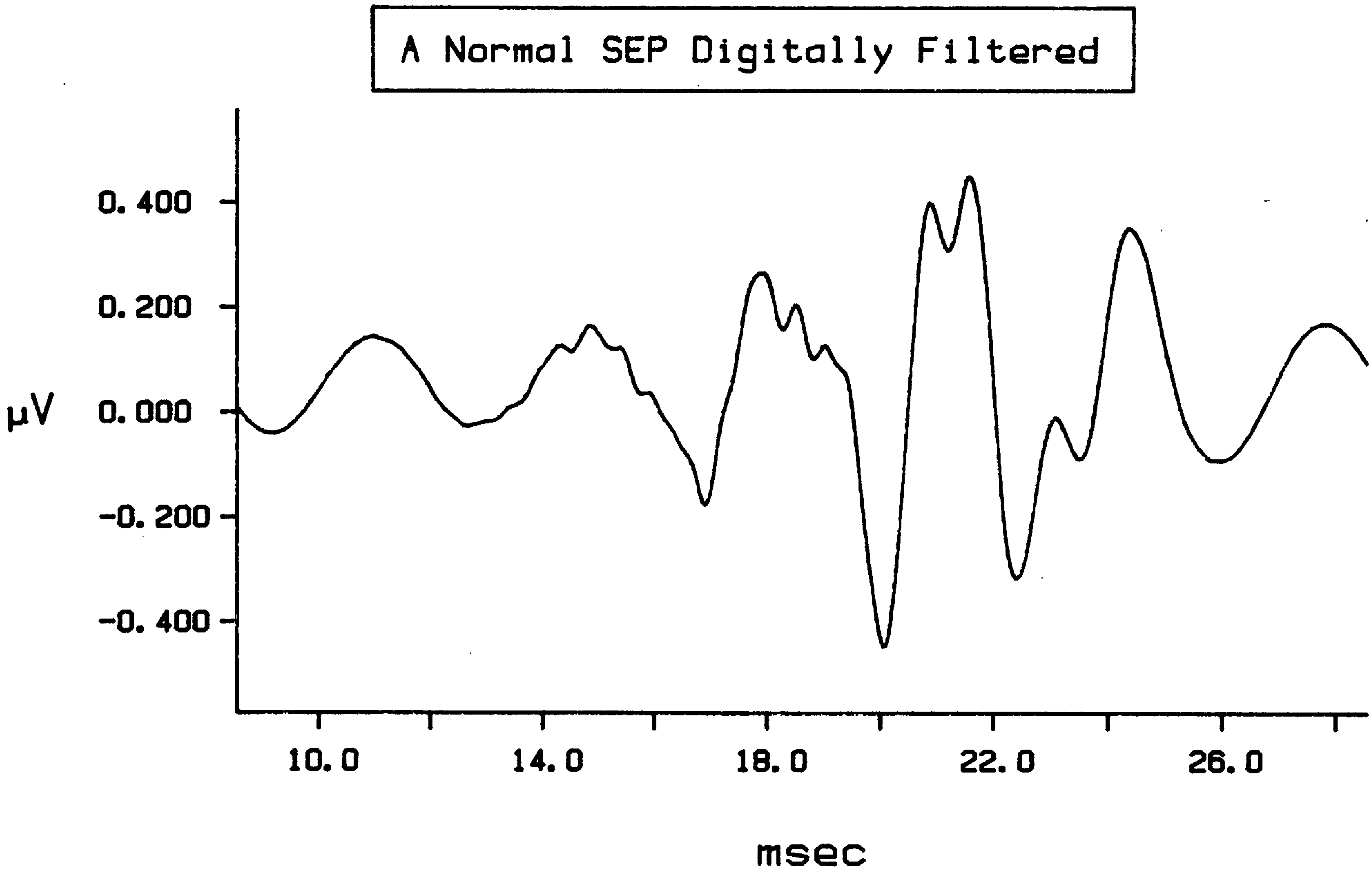

A Normal SEP Digitally Filtered
0.400
0.200
µV
0.000
-0.200
-0.400
10.0
14.0
18.0
22.0
26.0
msec

Figure 3.4

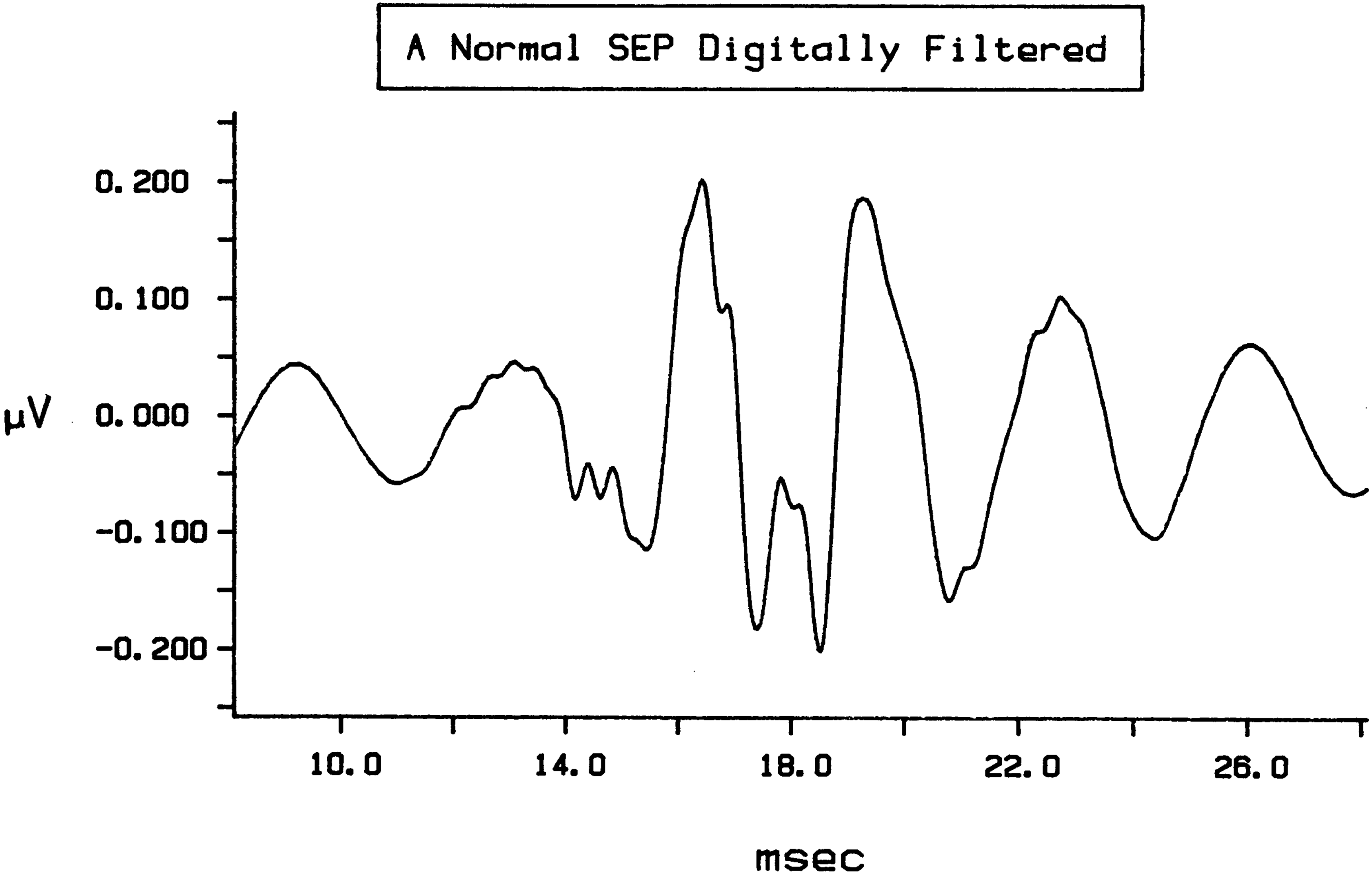

A Normal SEP Digitally Filtered
µV
0.200
0.100
0.000
-0.100
-0.200
10.0
14.0
18.0
22.0
26.0
msec

Figure 3.5

T is used to represent the number of locations at which each wave is sampled. The warping function w is a monotonic integer mapping from {1...T} to {1...T} with 0 being used to denote an undefined value. It is useful to visualize **w** as a path contained in the two-dimensional array { 1...T ; 1...T }, with **t** being the horizontal parameter and **w[t]** the vertical parameter (Figure 3.12). When the path goes outside the array, **w[t]** is undefined and given the symbolic value of 0. The tail of a path is the subset of the path from some **t** on to the end at T. w is a warping function in that when **w[t]** is used to replace **t** it produces a warped time axis. Thus, a[w[t]] is a function of **t** that is a warped version of a[t].

Warping is composed of three operations. The first, shifting, is produced when **w**[1.0]$\neq$1.0 meaning that the whole wave has been shifted along the time axis. The second, a jump, is produced when **w**[t+1.0] > **w**[t]+1.0 and so the time axis has been stretched near the points **t** and t+1. The other form of operation is a flat spot where **w**[t+1] = **w**[t] resulting in the time axis being contracted between **t** and t+1. A warping function is composed of a shift and any combination of jumps and flat spots.

A cost function is a real valued function of two waves **a** and **b** and a warping **w**. The inverse warping function w^{-1} is the function such that, wherever **w** is defined and strictly monotonic, $w^{-1}[w[t]] = t$, and wherever w^{-1} is defined and strictly monotonic, $w[w^{-1}[t]] = t$.

3.2) Design of the Cost Function

The cost function that is to be minimized is a function of the waves **a** and **b**, and the warping function **w**. The cost function used in this work is based on the correlation <u>Q</u> between wave **b**[t] and the warped wave **a**[**w**[t]]. A high degree of correlation produces a low value of the cost function, and conversly, a poor correlation produces a high value of the cost function. There is also a penalty function <u>P</u> that is a function of **w**. <u>α</u> is the coefficient of the penalty that controls directly how expensive the penalty is.

The cost function $C(a,b,w,\alpha)$ =

$$\left\{ \begin{array}{ll} \dfrac{1}{Q(a,b,w) - \alpha P(w)} & [Q(a,b,w) > \alpha P(w)] \\ \text{Infinity} & [Q(a,b,w) <= \alpha P(w)] \end{array} \right\}$$

As the solutions have to be found numerically, the cost function must be designed such that it can be calculated for the Dynamic Programming Algorithm. A necessary condition for this is that the cost of any tail from some t on to the end T can be calculated knowing only the tail and not the rest of the path. This condition arises from the way in which the Dynamic Programming Algorithm works. It is desirable to be able to compute this cost in a small number of operations as it will be calculated many millions of times.

The minimization of the cost function was translated to be a maximization of the denominator of the ratio:

$$Q(a,b,w) - \alpha P(w)$$

This allowed the contributions of the tails to be calculated from any t if the contributions to P and Q of the tail could be calculated. The cost function itself could not be calculated for each tail and so could not be used directly with the Dynamic Programming algorithm.

3.2.1) The Correlation Quotient Q

The simple correlation quotient

$$Q = \sum_{t=1}^{T} a[w[t]]b[t]$$

is summed over t and not w[t]. This function is not symmetric with respect to a and b because

$$\sum_{t=1}^{T} a[w[t]]b[t] \neq \sum_{t=1}^{T} b[w^{-1}[t]]a[t]$$

As an overriding principle it was desired that the cost function and the solutions be symmetric between **a** and **b**. Specifically, if warping **a** to b produced a cost C and a warping function **w**, the warping of b to a should produce the same cost C and the inverse of the warping function $\mathbf{w}^{-1}$.

The inclusion of the sub sum $\sum_{w[t]>s>w[t-1]} a[s]b[t]$

made the function symmetric by summing all of the positions in a jump, just as all of the positions in a flat spot are summed.

This produced the modified correlation quotient

$$Q = \sum_{t=1}^{T} \left[a[w[t]]b[t] + \sum_{w[t]>s>w[t-1]} a[s]b[t] \right]$$

The correlation quotient utilized the absolute microvolt scale of the waves, as opposed to the typical correlation that would be divided by the RMS value of each wave. This acted as a safety feature in that very small waves that are mostly background noise would produce a large cost, even if their shape happened to be close to that of a normal wave.

It should be noted that any DC offset in the waves was removed prior to warping to prevent the DC from biasing the correlation.

3.2.2) Constraints and the Penalty Function

The path **w** is constrained to be monotonically increasing, which is a sufficient condition for the existence of a unique inverse. As the functions range over discrete time rather than continuous time, discontinuities in the inverse are not a problem. This allows sections of the paths to have slope zero (flat spots).

Dynamic Time Warping has been used in speech recognition [Rabiner, et al, 1978; Myers + Rabiner, 1981; Myers et al, 1980; Brown + Rabiner, 1982]. In speech applications, the start and end of the word had been identified before the Time Warping was applied. This allowed the endpoints of the warping function to be constrained to be (0,0) and (**T**,**T**). These endpoint constraints were not possible in our application because, while the variations in onset latency due to arm-length were corrected when the SEP's were recorded, that correction was merely a least squares error adjustment and not an identification of the onset of the wave. Therefore, different waves did not necessarily start at the same point. Similarly, the finish of different waves did not match exactly. This meant that there could be no endpoint constraints. The endpoints had to be able to be shifted to correct for any variation in alignment.

In the speech recognition applications a very severe continuity constraint was used. The warping function was constrained to have jumps with a height of at most 1.0, and flat spots of length at most 1.0. This seemed a harsh and arbitrary constraint that would tend to produce solutions that were constraint-bound rather than globally optimal. It was decided that a more elegant method of producing reasonable warping functions would be to include a penalty for discontinuities in the cost function.

The penalty function could be of the form:

$$P(w) = \sum_{t=1}^{T-1} | w[t+1] - w[t] - 1 |^n$$

The parameter n determines how severely large jumps are to be penalized relative to small jumps. As n approaches infinity, the penalty function becomes the constraint used in speech recognition.

For n=1:

$$P(w) = \sum_{t=1}^{T-1} | w[t+1] - w[t] - 1 |$$

This computationally very simple penalty function did not strongly penalize large jumps, and so the warping functions were often severely discontinuous. Increasing the penalty coefficient, α , enough to prevent large jumps eliminated the small variations as well.

For n=2:

$$P(w) = \sum_{t=1}^{T-1} (w[t+1] - w[t] - 1)^2$$

This penalty function is not symmetric in that the flat spots are penalized linearly rather than proportionally to the square of their length.

Using the identity:

$$x^2 = \sum_{r=1}^{x} 2r-1$$

and the formula for the length of the flat spot from t on:

$$\inf\{s-t-1 \text{ such that } s \in \{1..T]; \; s>t; \; w[s] \neq w[t]\}$$

produced the symmetric penalty function $P = \sum_{t=1}^{T-1} p(w,t)$

$$p(w,t) = \begin{cases} (w[t+1]-w[t]-1)^2 & [w[t+1]>w[t]] \\ 2\inf\{s-t-1 \mid s \in \{t..T];w[s] \neq w[t]\}-1 & [w[t+1]=w[t]] \end{cases}$$

This function solved the problem of severe discontinuities by penalizing large jumps or long flat sections very severely. A disadvantage of the squared penalty, that all such nonlinear penalties share, is that the dynamic programming algorithm may not produce the optimal solution. The penalty for a section with slope zero is the square of the length of the section. The previous length of the flat spot affects the penalty for continuing the flat spot. This means that locally optimal paths may not be globally optimal, i.e.; the best path from (s,j) to (t,k) may not be a subsection of the best path from (r,i) through (s,j) to (t,k). This may result in paths that, when traced backwards, have a zero slope section starting one place too soon. This is not a major problem as the algorithm is robust and the paths found will be very close to optimal.

α is the coefficient of the penalty that controls directly how expensive it is to warp. A small (0.001) value of α produced extreme warpings. A large (0.1) value of α produced warping functions that were merely shifts with no jumps or flat spots at all. The apropriate value, $\alpha = 0.0075$, was selected by trial and error.

3.2.3) Summary of Cost Function Used

The cost function $C(a,b,w,\alpha)$ =

$$\begin{cases} \dfrac{1}{Q(a,b,w)-\alpha P(w)} & [Q(a,b,w)>\alpha P(w)] \\ \text{Infinity} & [Q(a,b,w)<=\alpha P(w)] \end{cases}$$

where the correlation quotient

$$Q = \sum_{t=1}^{T}\left[a[w[t]]b[t] + \sum_{w[t]>s>w[t-1]} a[s]b[t]\right]$$

and the penalty function $P = \sum_{t=1}^{T-1} p(w,t)$

$$p(w,t) = \begin{cases} (w[t+1]-w[t]-1)^2 & [w[t+1]>w[t]] \\ 2\inf\{s-t-1 \mid s\in[t..T];w[s]\neq w[t]\}-1 & [w[t+1]=w[t]] \end{cases} .$$

3.3) Implementation

As the number of possible paths **w[t]** increases exponentially with T it is not feasable to compare all possible paths to find the best one. The Dynamic Programming Algorithm finds the optimal path using only a number of operations proportional to T^3. By assuming that the solution will have no jumps larger than some constant, the number of operations can be reduced to being proportional to T^2. In the case at hand, where the number of samples in a wave equals one thousand, even this is too slow as it would take over an hour to warp one pair of waves on the laboratory PDP-11/23. However, due to the windowing, only the center regions of the waves are of interest. This allows one to look at only the central 600 samples. As the sampling frequency is ten times the Nyquist frequency and the digitally filtered data is quantized with fourteen bit accuracy, the sampling rate can be reduced by a factor of six without losing much information. This makes T=100 and allows the warping to be done in under a minute. Warpings with T=200 were computed and compared to the T=100 warpings. There was not a significant difference between the two.

The Dynamic Programming Algorithm considers the warping functions as paths through a square array. The algorithm starts at the end (t=T) and column by column finds the lowest cost tails starting in each row of that column. In this application the algorithm found the maximum values for the denominator of the cost function evaluated over the tails. The Pascal implementation of the algorithm is documented in Appendix 3.

3.4) The Standard Wave

A standard wave was constructed to serve as a template. Unknown waves can be warped to this template to determine if they fit the pattern of normal SEPs.

The standard wave was constructed to be a representative mean of the set of 24 normal digitally filtered waves. For each $i \in \{1..T\}$ the normal waveforms a_i were warped to every wave a_j $j \in \{1..T\}$. Let the resulting cost be represented by C_{ij}.

The mean cost $M_j = \sum_{i=1}^{T} C_{ij}/T$ was calculated for each j along with the variance

$$V_j = \sum_{i=1}^{T} C_{ij}^2/T \quad - M_j^2 \quad .$$

The means and variances were scaled by the cost of the wave a_j warped to itself.

$V'_j = V_j/C_{jj}$

$M'_j = M_j/\mathrm{SQRT}(C_{jj})$

This scaling was done to remove the variation due to the amplitude of the waves. After one especially large and two especially small waves were eliminated from consideration, the wave with the smallest scaled mean and variance was chosen as the <u>median wave</u> of the set.

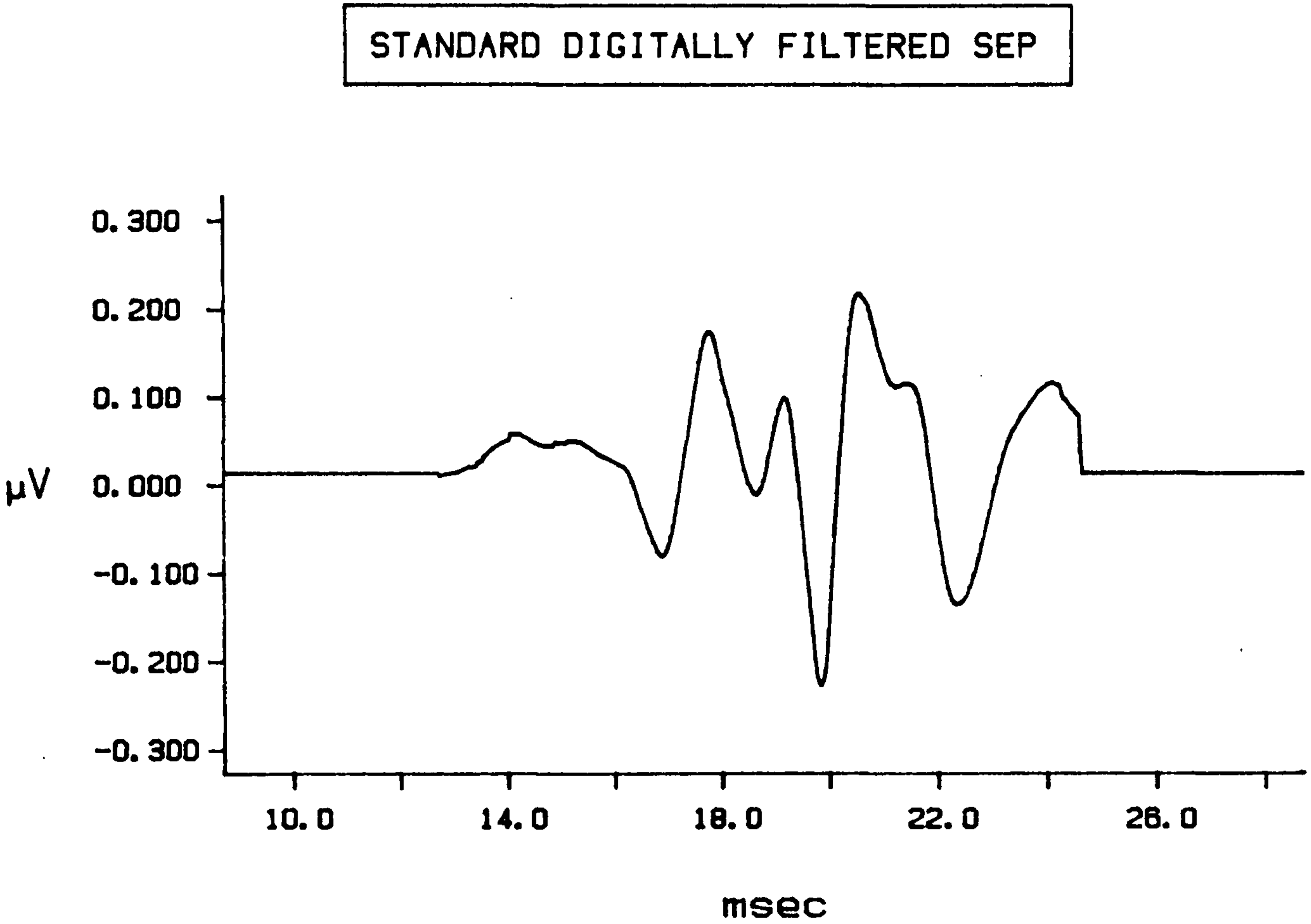


Figure 3.6 The standard wave was constructed to serve as a template to which unknown signals are compared.

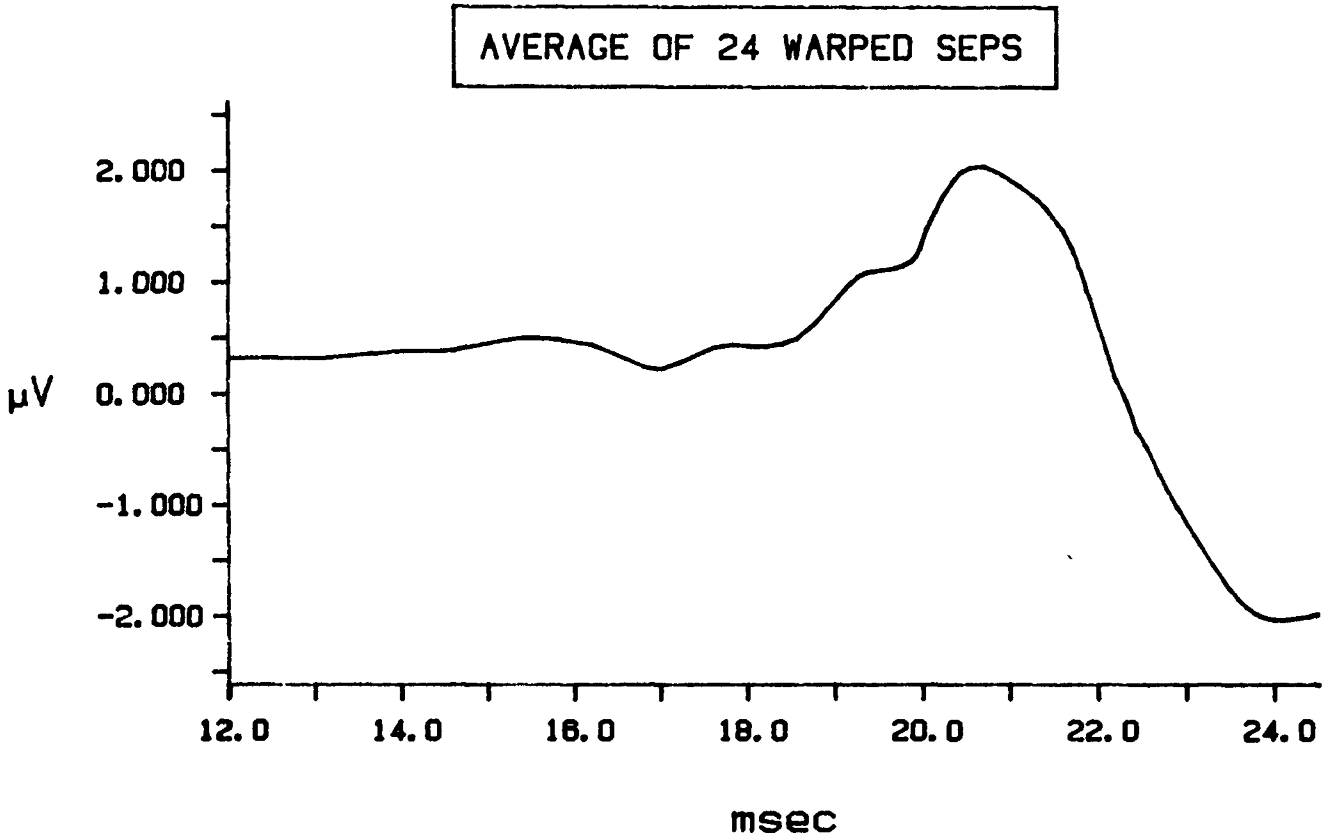


Figure 3.7 This average SEP was constructed by applying the warping functions generated by the filtered SEPs to the original SEPs, and averaging together the results.

Let w_i represent the warping function produced by warping wave a_i to the median wave.

s(t) is the <u>standard wave</u> where

$$s(t) = \sum_{i=1}^{T} a_i[w_i[t]]/T$$

I.e. all of the normal waves were each warped to the median wave and averaged pointwise to produce the standard wave.

To check the validity of the process of construction of the standard wave, each member of the set of normal waves was warped to the standard wave. When the mean and variance of the costs were estimated and scaled as before, they were smaller than those of the median wave, validating the averaging process. While no claim is made that the standard wave is the optimum template, it is better than any of the waves in the set of normals.

The standard wave is shown in Figure 3.6. Figure 3.7 shows the average of the SEPs warped with the corresponding warping functions. In Figures 3.8 and 3.9 warped waves are plotted below the standard wave. Notice how the peaks line up. Figure 3.10

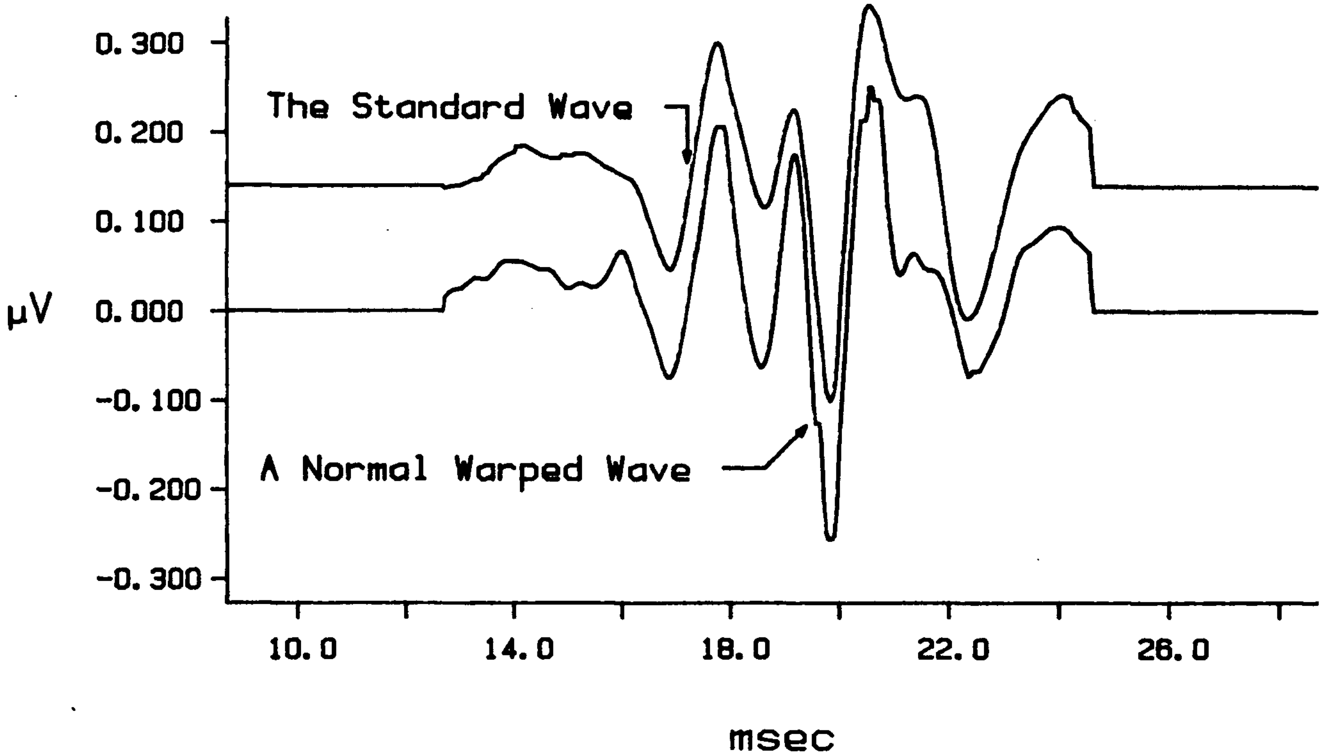


Figure 3.8 A normal wave that has been warped to the Standard Wave. Notice how the peaks line up.

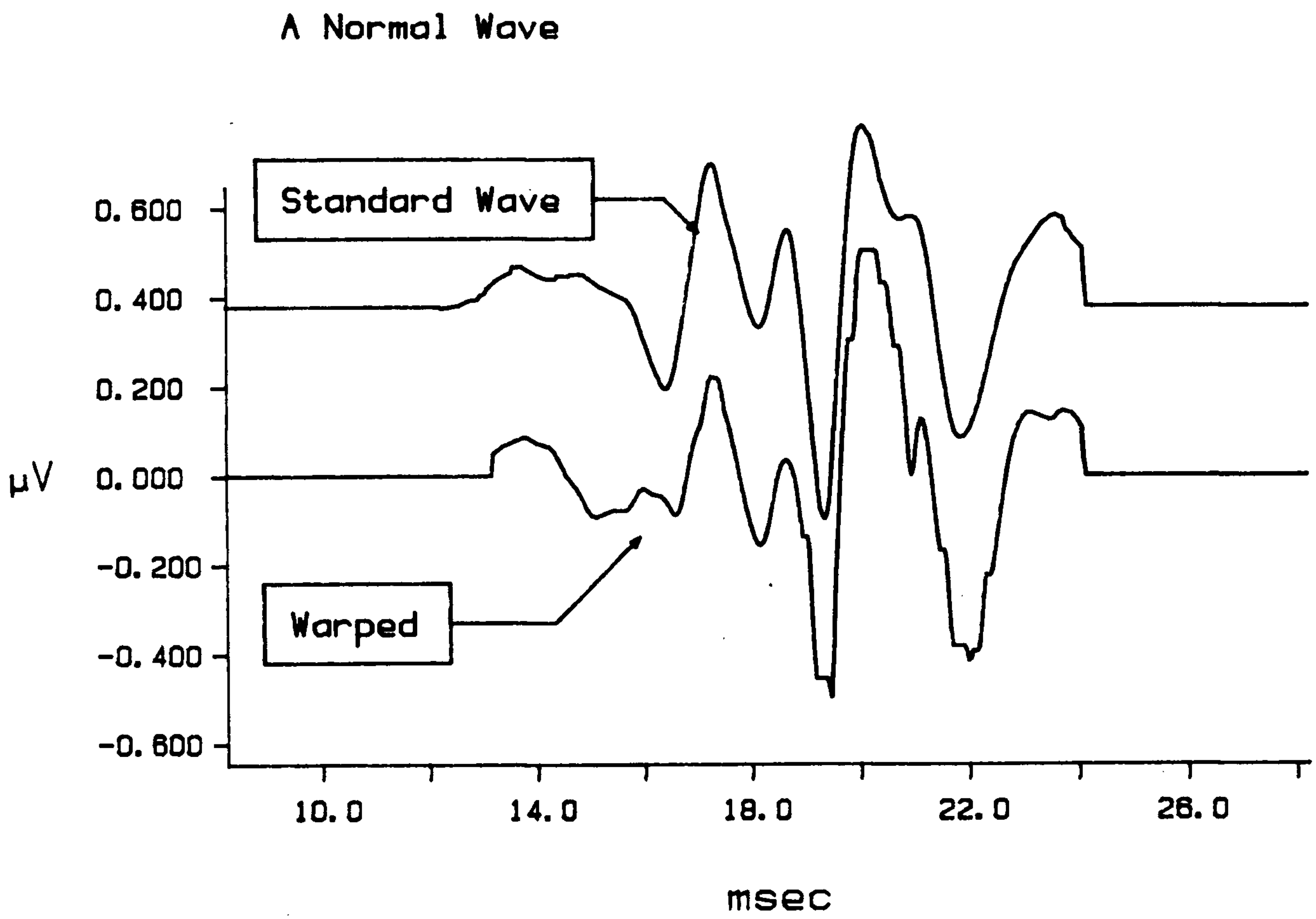


Figure 3.9 Another normal wave that has been warped to the Standard Wave.

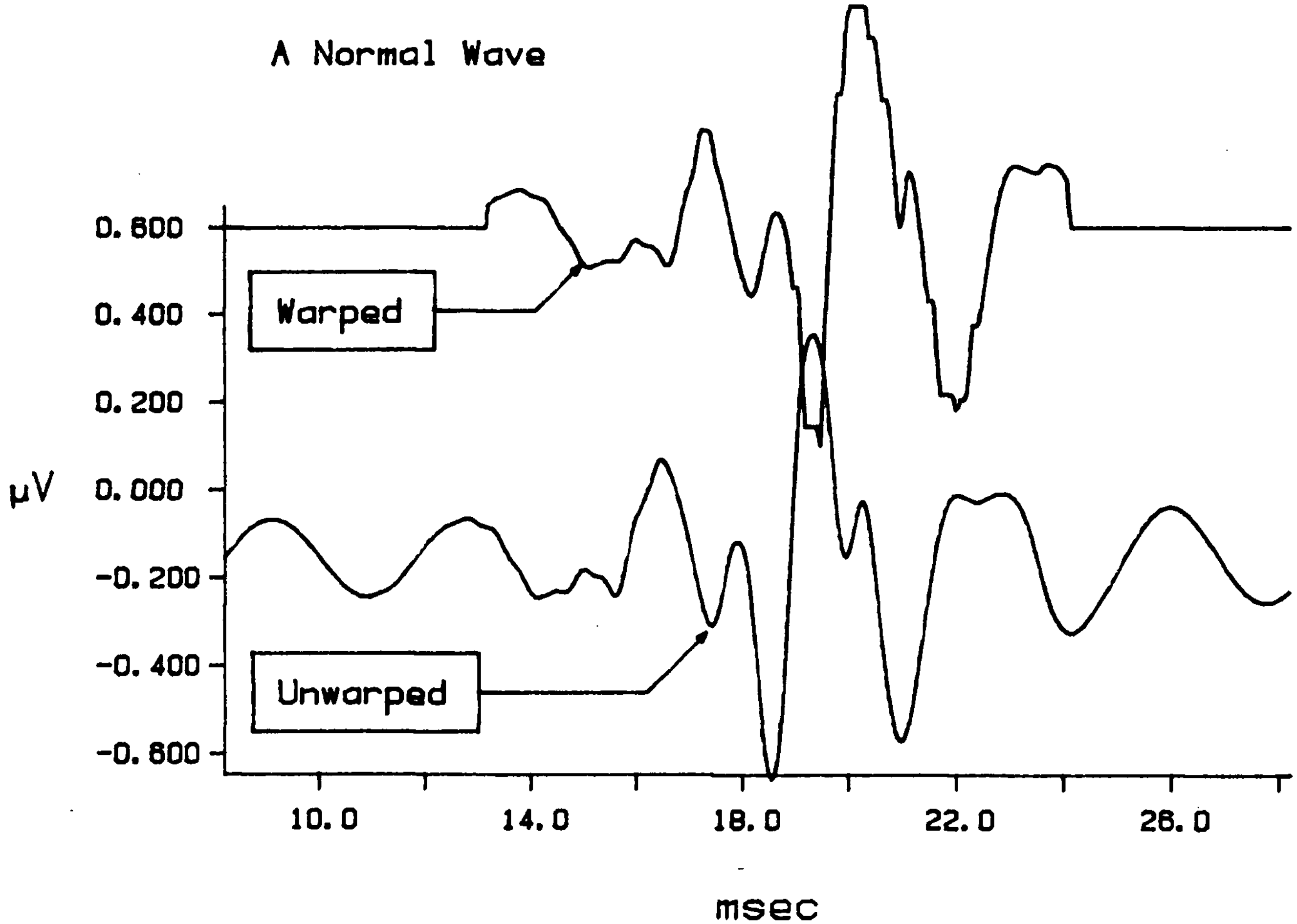


Figure 3.10 Comparing waves before and after warping. This wave is also shown in figure 3.9.

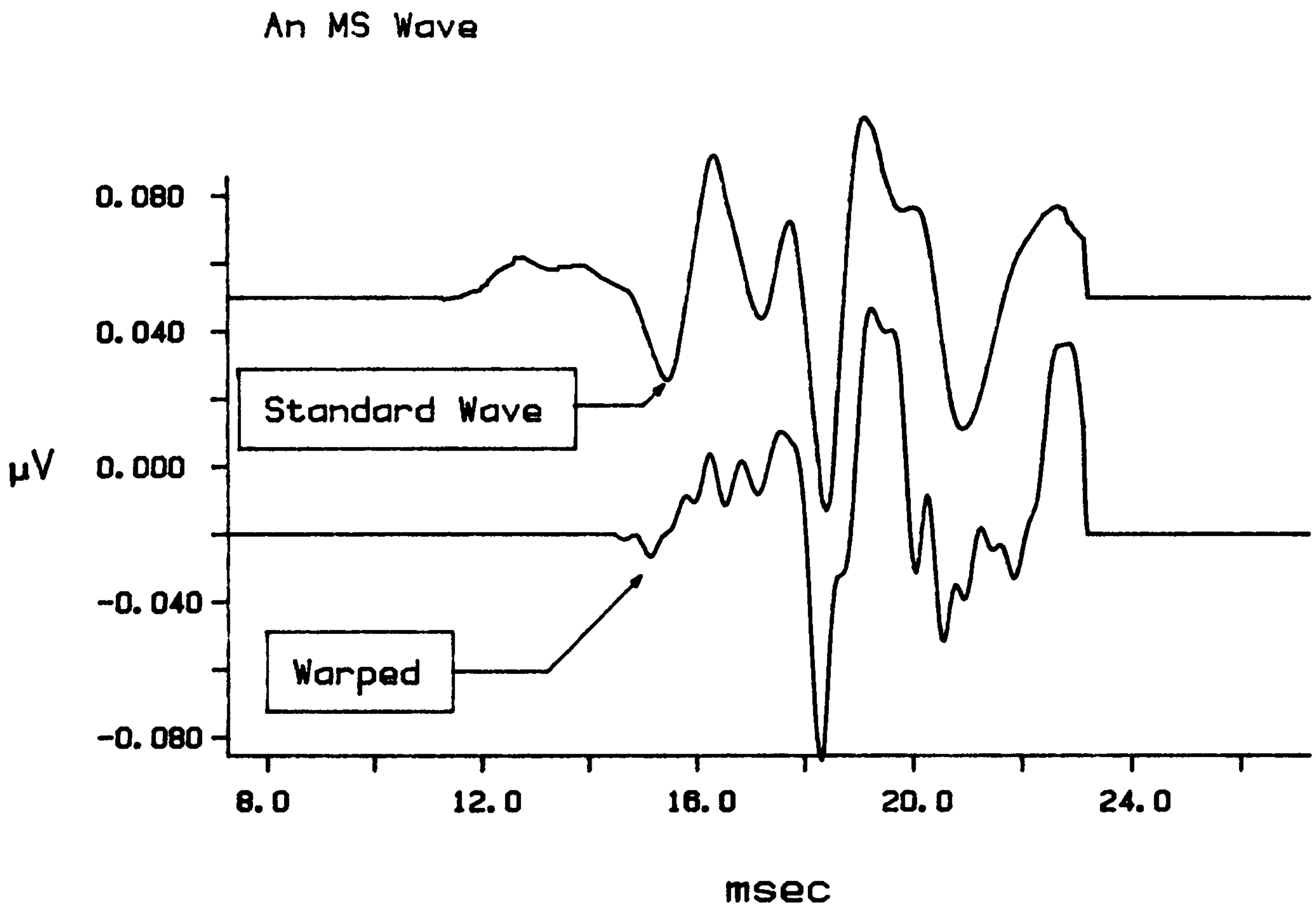


Figure 3.11 An MS wave that has been warped to the Standard Wave. Note how severe warping has lined up some peaks of the irregularly shaped wave.

compares the warped wave of figure 3.9 to what it was before warping. Figure 3.11 shows an MS wave warped to the standard wave. Figures 3.12 to 3.17 are graphs of the warping functions.

3.5) Clinical Test

A clinical test was developed using the Dynamic Time Warping Algorithm:

1) The patients SEP would be recorded using the techniques described above.

2) The SEP would be digitally filtered between 300 and 2500 Hz.

3) The filtered SEP would be warped to the Standard Wave.

4) The resulting Cost Function value would be compared to the mean of the values of the normal waves

Twenty-five Multiple Sclerosis patients volunteered for this experiment. Based on a constellation of laboratory tests over a period of several years, each of the patients had been diagnosed as having Multiple Sclerosis. Twenty-four of the volunteers were classified as Definite MS and one was classified as Probable MS.

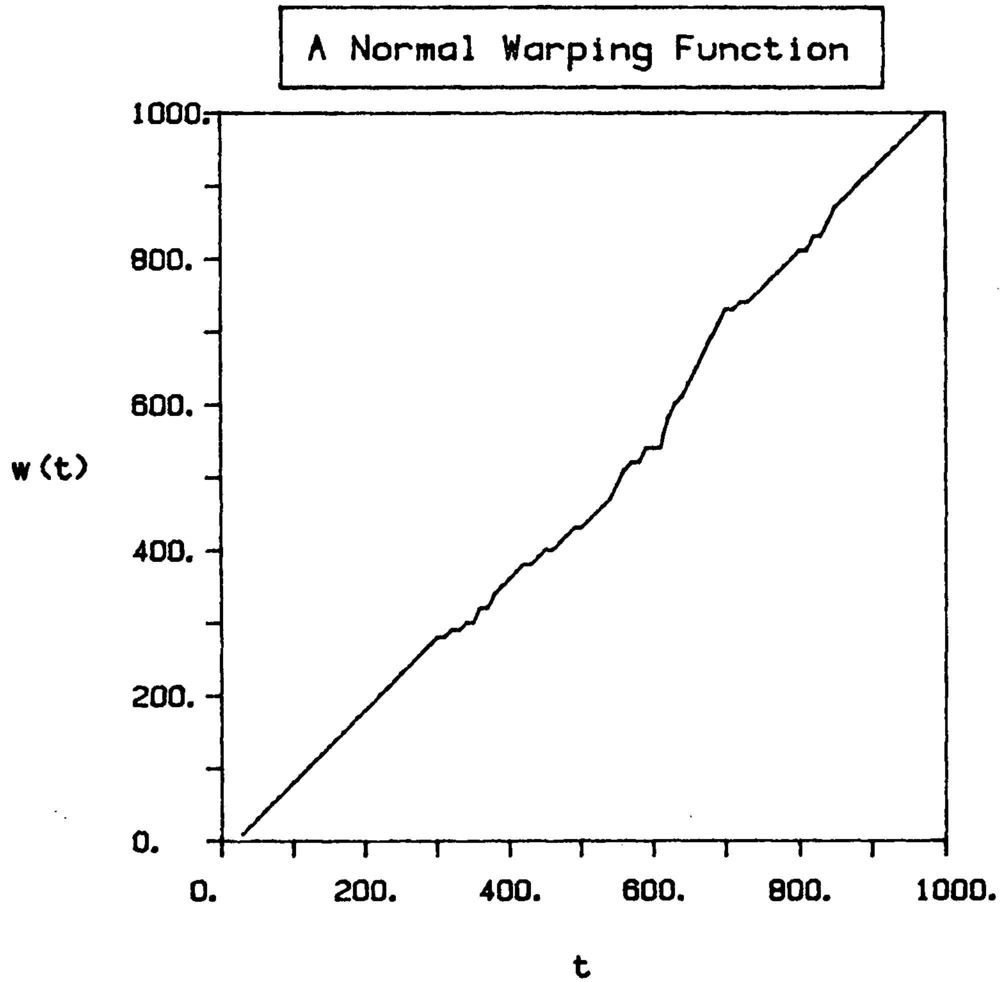


Figure 3.12 An example of the warping function generated by a normal waveform.

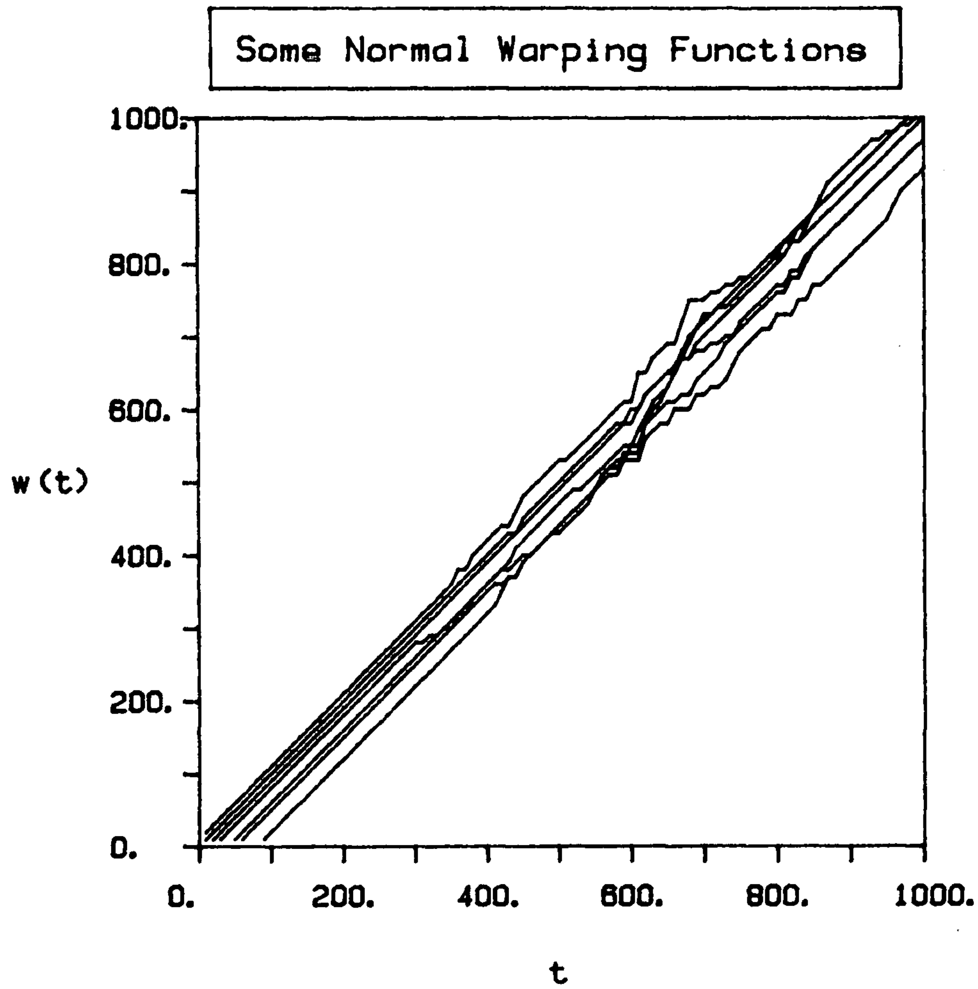


Figure 3.13 A comparison of several normal warping functions.

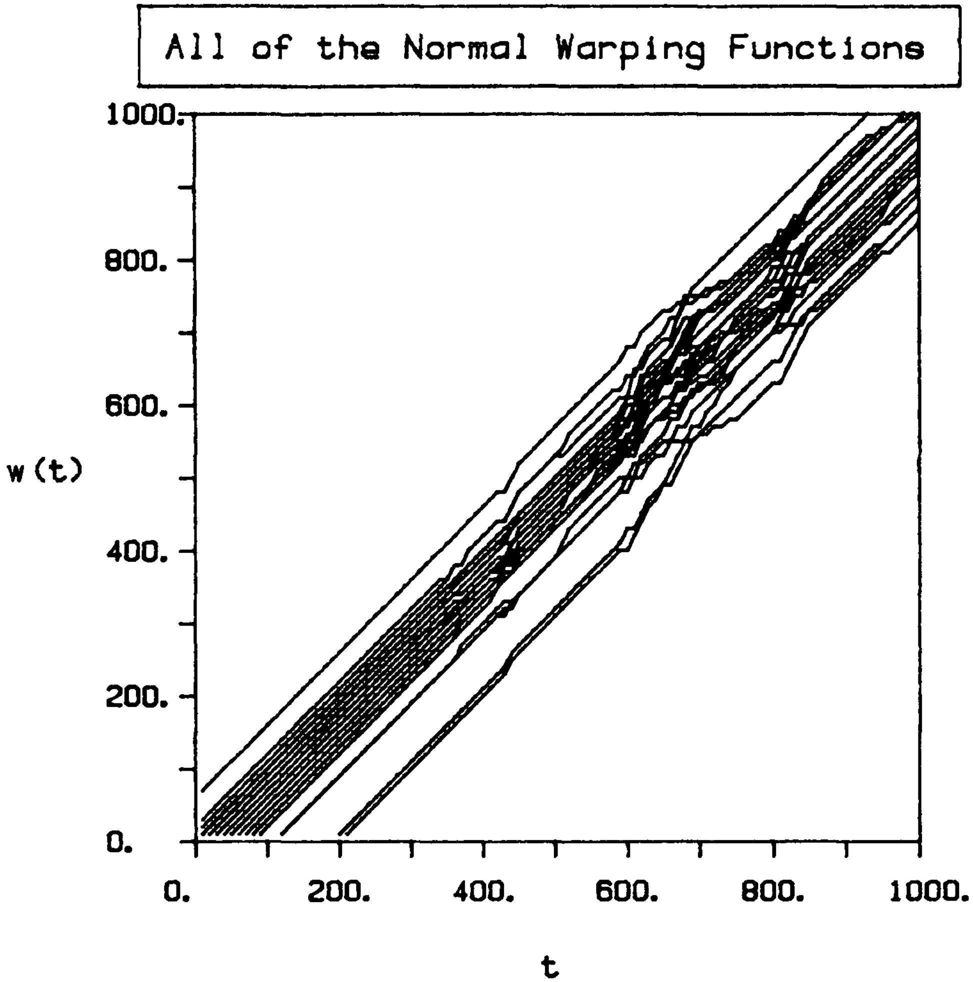


Figure 3.14 All twenty-four normal warping functions superimposed.

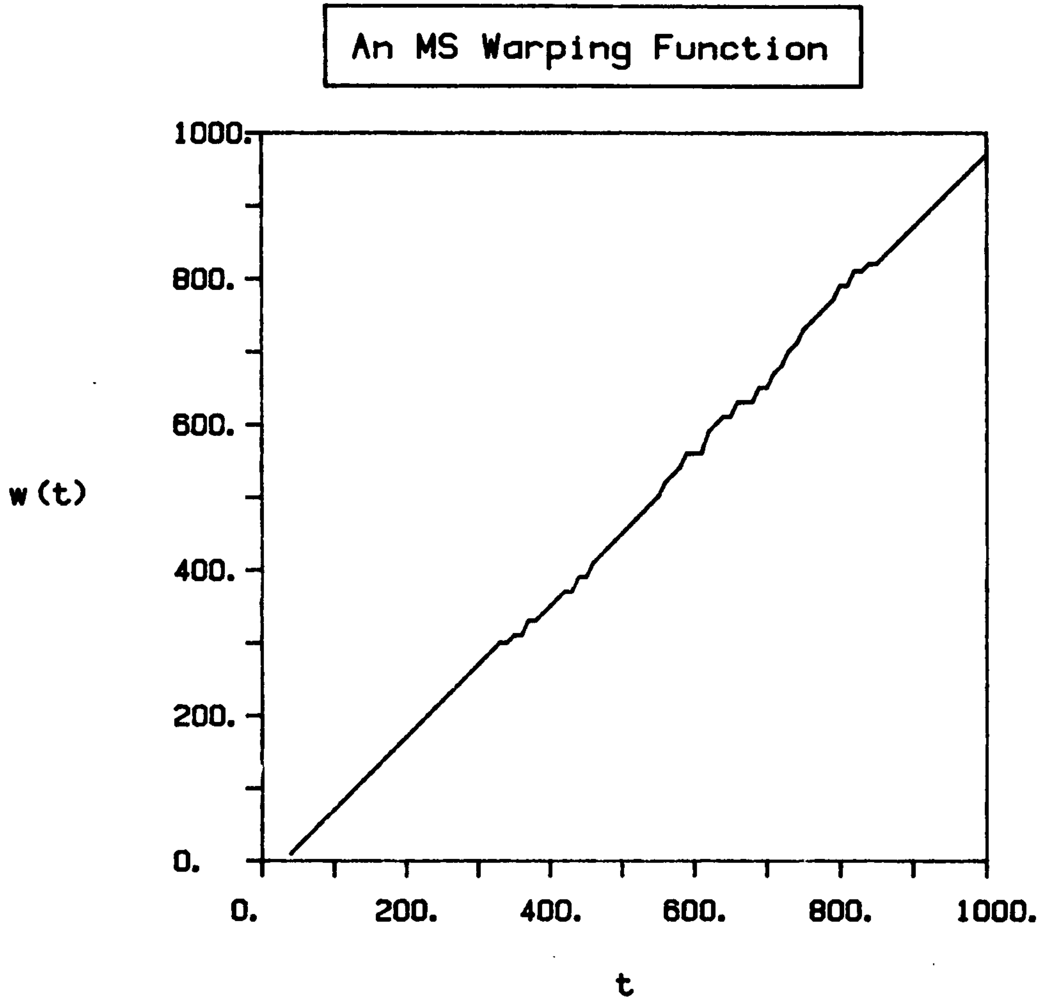


Figure 3.15 An example of an MS warping function. There is little apparent difference between the MS and normal functions.

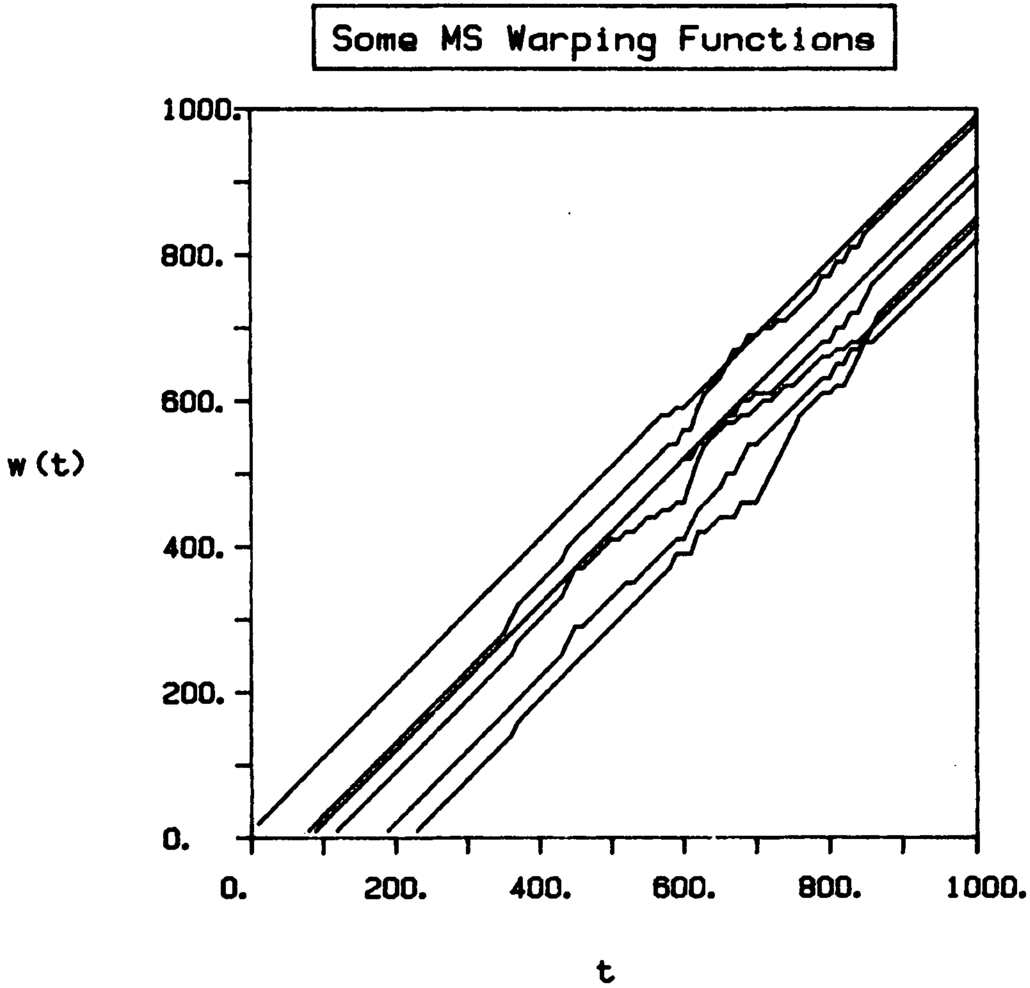


Figure 3.16 Several MS warping functions.

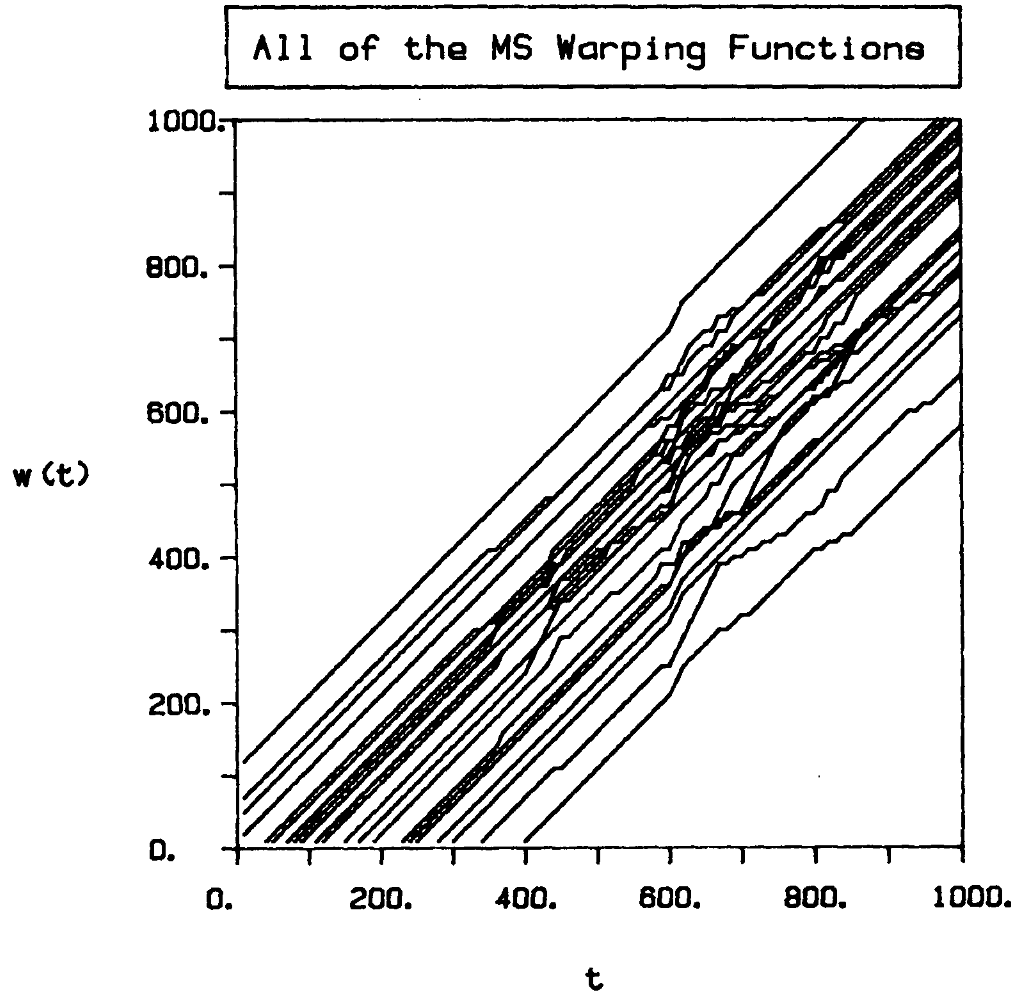


Figure 3.17 All twenty-five MS warping functions. Variations in latency cause a wide variety of shifts.

Only six of the twenty-five had symptoms in the median nerve system that was to be stimulated. The rest had spinal, optic or other systems affected. There were nineteen females (ages 39 $\pm$ 11) and six males (ages 45$\pm$9). Their SEPs were recorded using the same technique as described in section 3.2 for the Normals.

3.6) Results

The mean of the MS group was 5.089 compared to 1.305 for the Normals. The standard deviation was 4.277 for the MS group and 0.501 for the Normal group. With the limit of normality placed at three standard deviations above the Normal mean, this test identified fourteen out of the twenty five MS patients as being abnormal. None of the Normals were so identified.

Table III The Cost Function Values

Normal Results	M S Results	
2.456	5.627	***
1.036	0.903	
0.525	1.217	
1.218	0.906	
2.174	0.738	
1.073	12.215	***
0.918	3.040	***
1.216	3.165	***
1.120	1.952	
1.454	2.291	
1.270	1.675	
0.529	1.881	
0.772	9.134	***
1.270	13.382	***
1.673	3.971	***
2.263	0.701	
0.894	6.357	***
1.184	9.578	***
1.150	3.428	***
1.046	1.785	
1.506	9.587	***
2.128	1.632	
1.066	13.972	***
1.376	7.282	***
	10.812	***

*** More than three Standard Deviations above the Normal Mean.

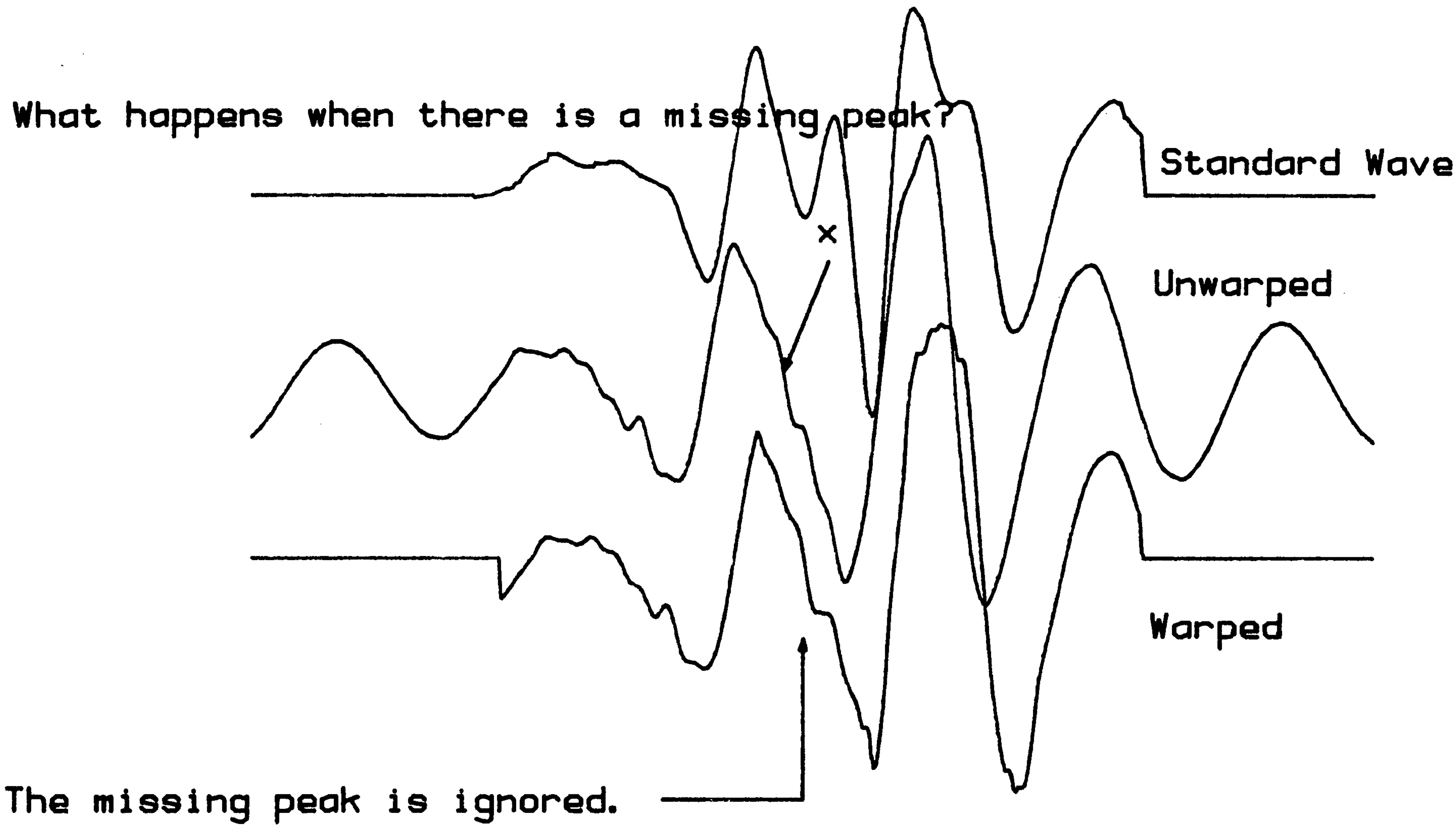

What happens when there is a missing peak?
Standard Wave
x
Unwarped
Warped
The missing peak is ignored.

Figure 3.18

3.7) Discussion

The use of the penalty function rather than severe continuity constraints as used in speech recognition increased the computation time by a factor of four. In a clinical setting this warping would be done at most twice per patient, and therefore the difference between 15 seconds of computation and one minute is not much of a burden.

The Cost Function worked well in identifying MS patients. The Cost Function is rigorously compared to other features of the SEP in chapter 5. It is concluded that this is a very sensitive test for Multiple Sclerosis.

To test the stability of this test over time, one of the Normal volunteers was retested six months after the initial test. The initial Cost Function was 1.073 and the retest result was 1.063. This is a very small change in six months, indicating that in Normals the test is stable. One would expect major changes over time with MS patients as the disease progressed or remitted. The results gained from ongoing clinical use of this test with the MS patients will elucidate this.

4) MEASUREMENT OF THE DISPERSION OF THE SEP

4.1) Introduction

Abnormalities of smoothness or lack of synchronization (Dispersion) in the Somatosensory Evoked Potential are common in disease. It has long been known that peripheral mixed or sensory nerve action potentials can have a significant amount of desynchronization indicating disease, even though the latency, and therefore conduction velocities, are normal with amplitude being only slightly reduced. This reflects the fact that a significant number of myelinated axons are conducting at variable speeds, although there are still a few large-diameter, fast-conducting fibres capable of normal conduction. It has been shown previously [Roberts, Lawrence + Eisen, 1983; Eisen, Lawrence, Roberts + Hoirch, 1982] that the desynchronization of the SEP would be a useful parameter.

Conversely, Figure 4.1 shows that even in a confirmed MS patient a very normal median nerve SEP can be observed. Figures 4.2, 4.3 and 4.4 more typically show a shape abnormality and lack of smoothness.

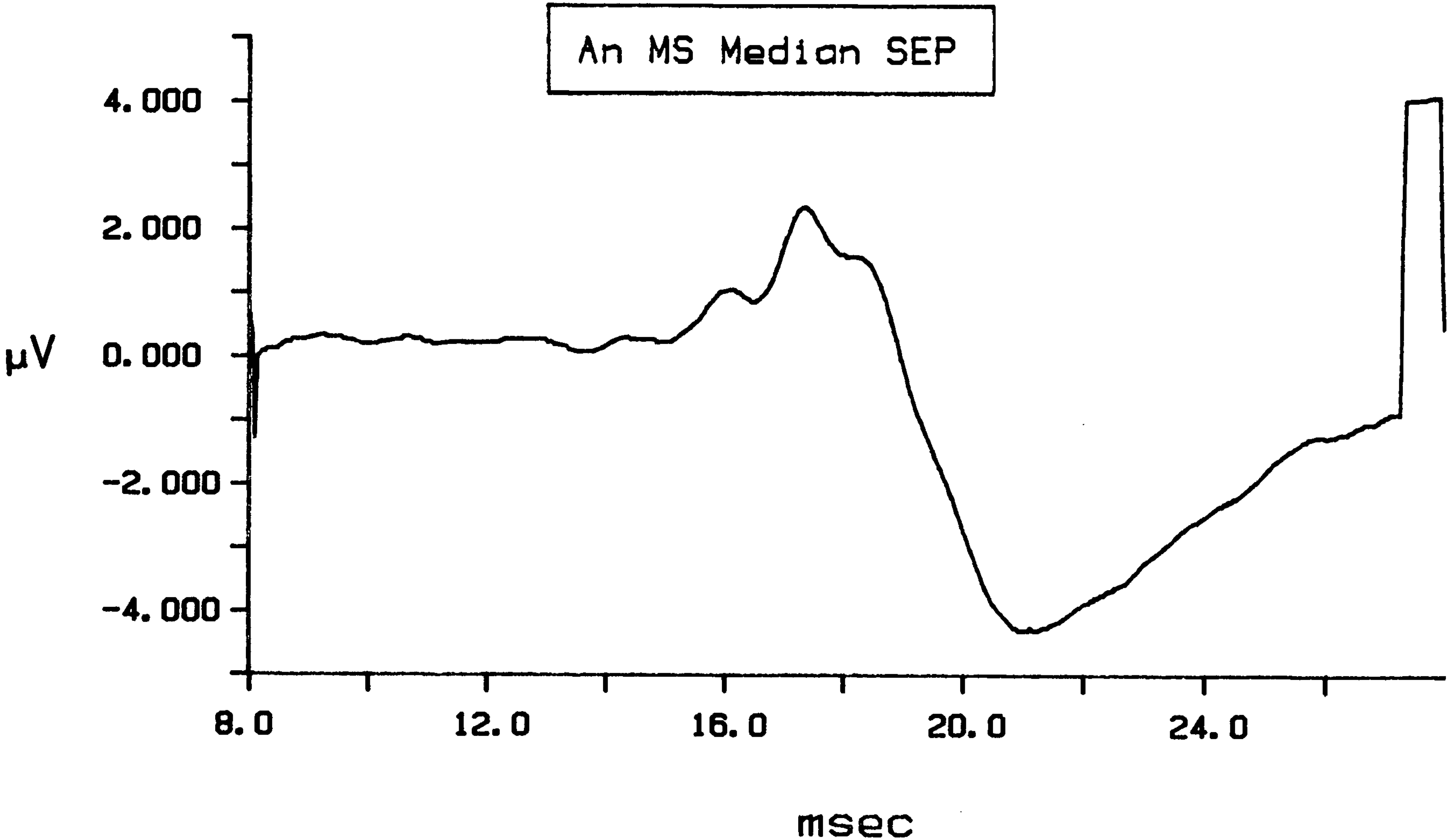

An MS Median SEP
μV
4.000
2.000
0.000
-2.000
-4.000
8.0
12.0
16.0
20.0
24.0
msec

Figure 4.1

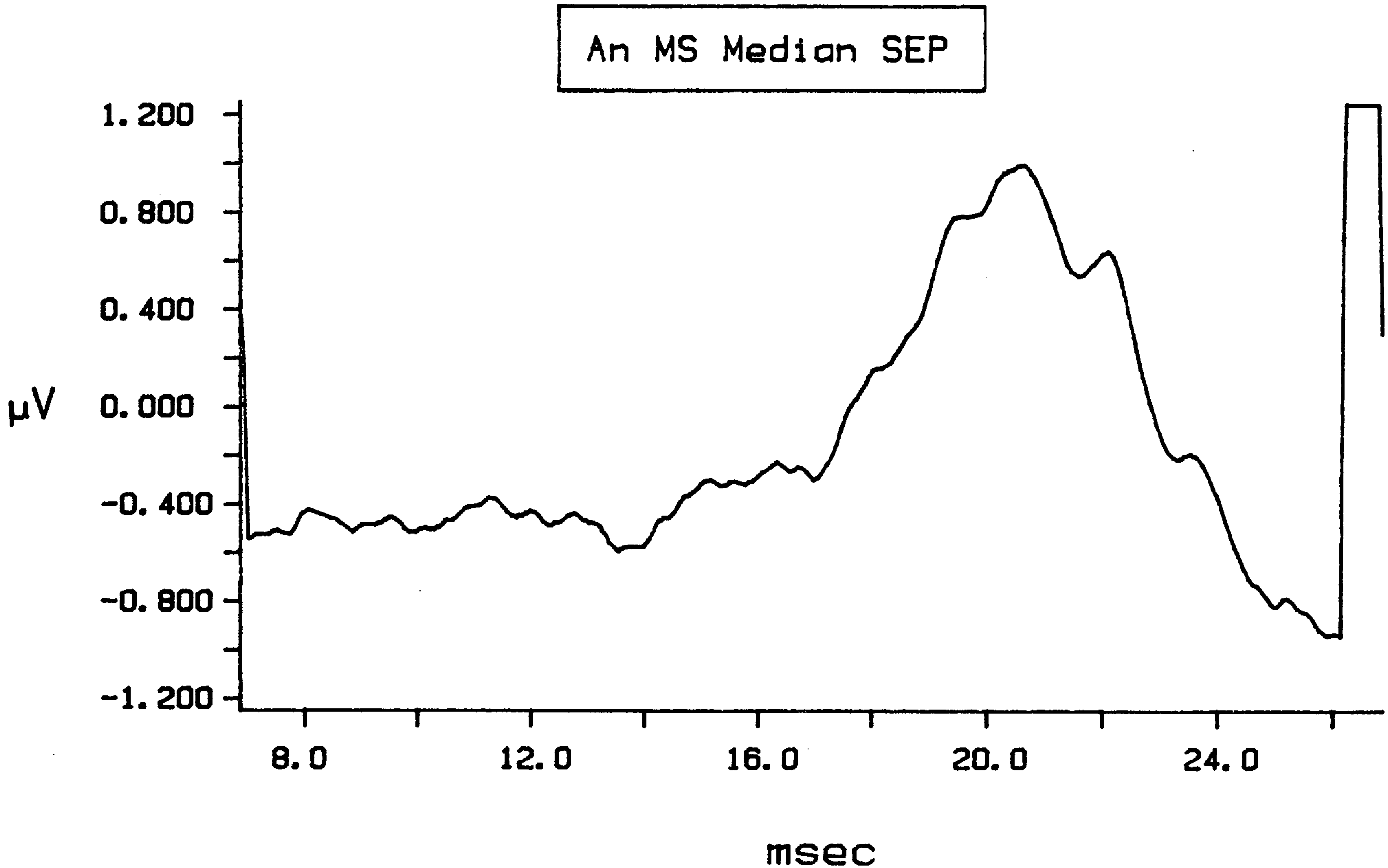

An MS Median SEP
1.200
0.800
0.400
0.000
-0.400
-0.800
-1.200
µV
8.0
12.0
16.0
20.0
24.0
msec

Figure 4.2

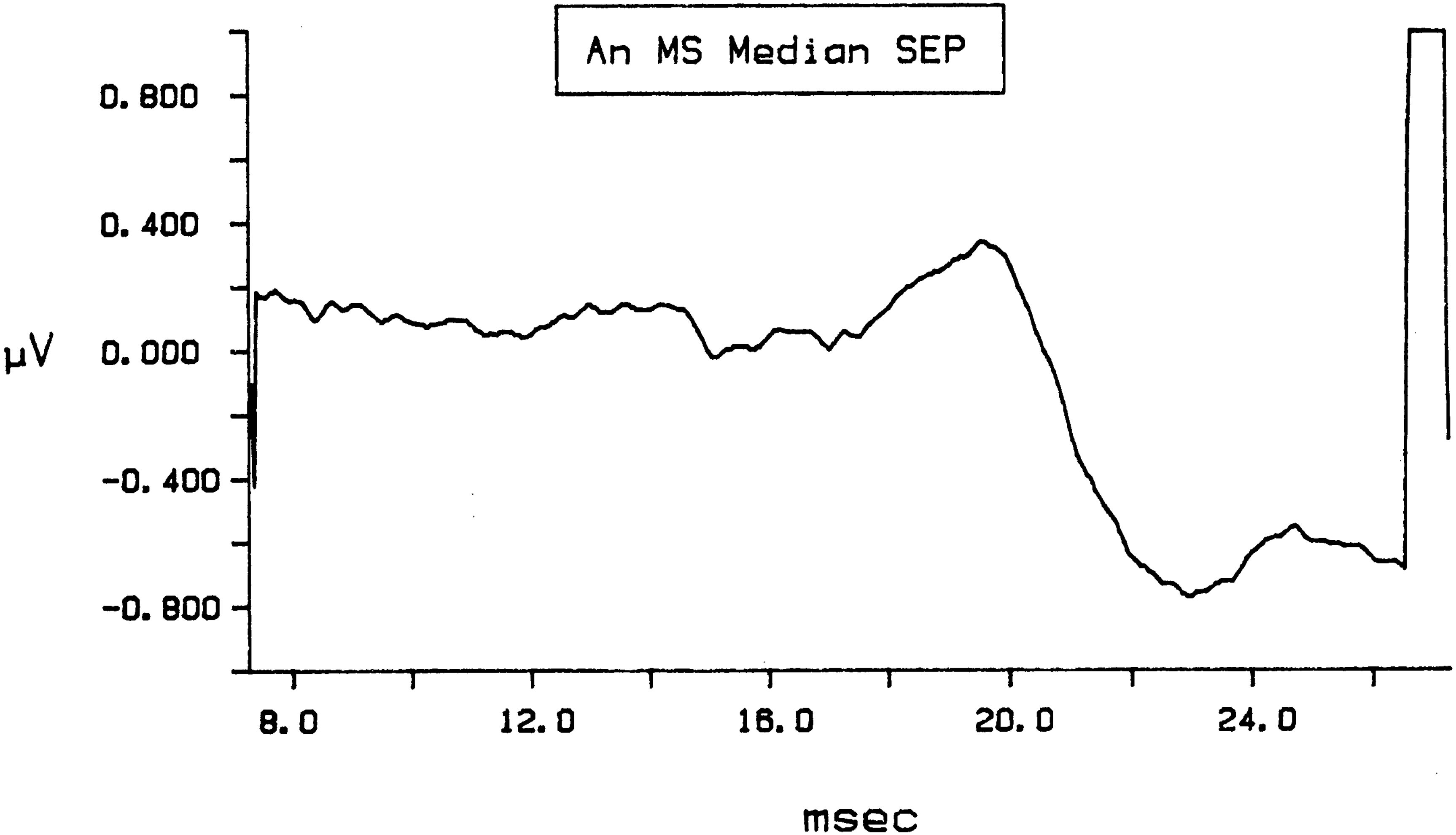

An MS Median SEP
0.800
0.400
0.000
-0.400
-0.800
µV
8.0
12.0
16.0
20.0
24.0
msec

Figure 4.3

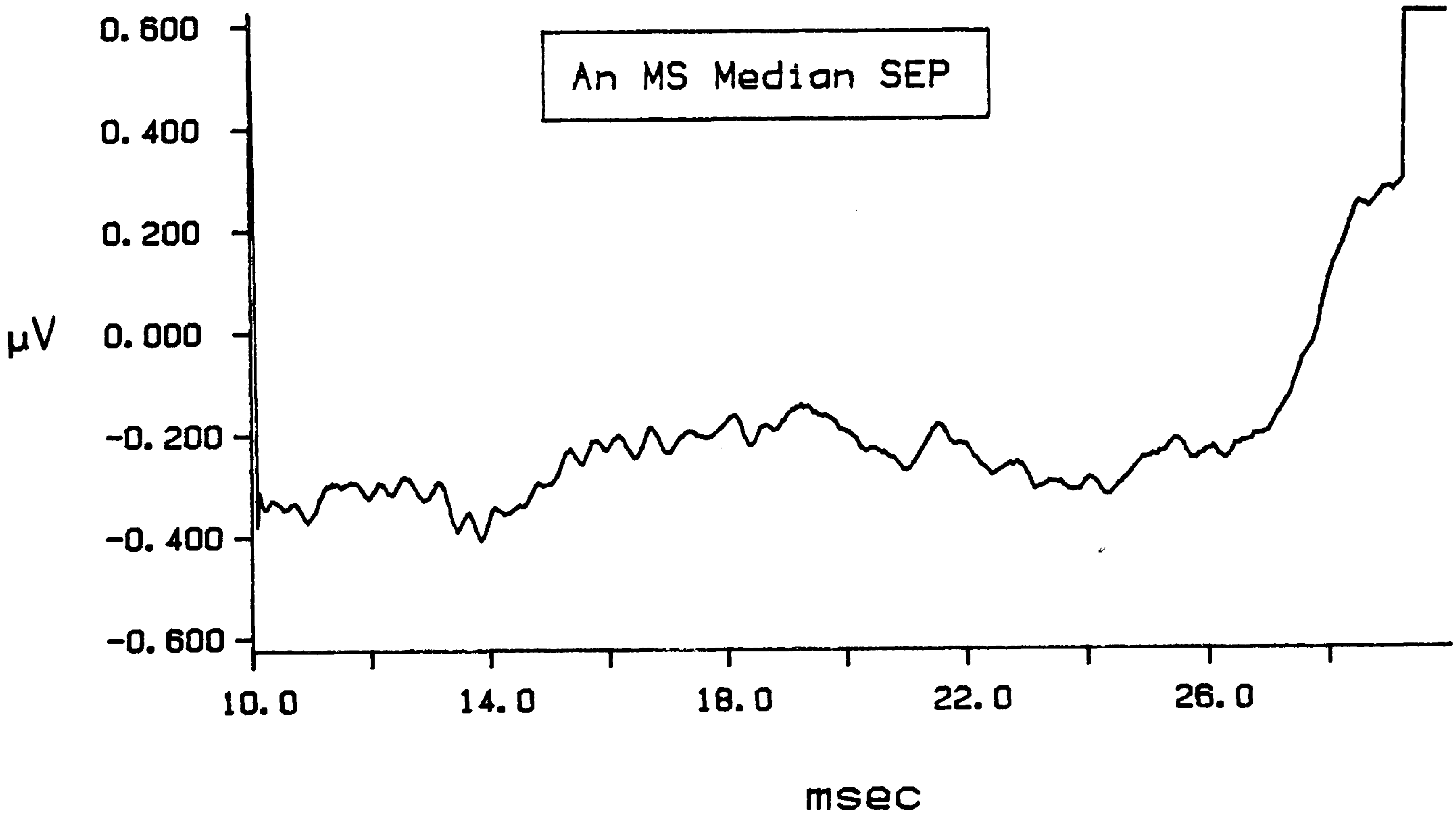

An MS Median SEP
0.600
0.400
0.200
0.000
-0.200
-0.400
-0.600
μV
10.0
14.0
18.0
22.0
26.0
msec

Figure 4.4

Fast Fourier Transform techniques were used to quantify the smoothness of Somatosensory Evoked Potentials. For median nerves the measure used for the evaluation was the energy between 1000 and 2500 Hz.

4.2) Methods

Twenty-five Multiple Sclerosis patients volunteered for this experiment. Based on a constellation of laboratory tests over a period of several years, each of the patients had been diagnosed as having Multiple Sclerosis. Twenty-four of the volunteers were classified as Definite MS and one was classified as Probable MS. Only six of the twenty-five had symptoms in the median nerve system that was to be stimulated. The rest had spinal, optic or other systems affected. There were nineteen females (ages 39 $\pm$ 11) and six males (ages 45$\pm$9). Their SEPs were recorded using the technique described in section 3.2. Normal Somatosensory Evoked Potentials had been recorded as described in sections 2.2. These potentials, recorded from 24 Normal and 25 Multiple Sclerosis volunteers, were examined in the frequency domain.

In the analysis any DC component of the signal was removed. The signal was windowed using the Blackman Window:

$$w(i)=0.42 - 0.5\cos\left(\frac{2\pi i}{N-1}\right) + 0.08\cos\left(\frac{4\pi i}{N-1}\right)$$

where N is the width of the window and i runs between 0 and N-1. Figure 3.1 shows the shape of the Blackman Window.

The windowed data was contained in an array of 1024, 32 bit floating point values. The Fast Fourier Transform algorithm [Cooley+Tukey, 1965] was used to calculate the power spectrum. The power spectrum was then scaled to have a maximum value of one.

The Dispersion Figure was defined to be the sum of the energy in the scaled spectra between 1000 and 2500 Hz.

4.3) Results

Graphs of the Fourier spectra of the SEP's showed that, relative to their maximum value, most Normal waveforms had very little energy above 1000 Hz, while, on average, the MS waveforms had signifiantly more (Figures 4.5 to 4.11)

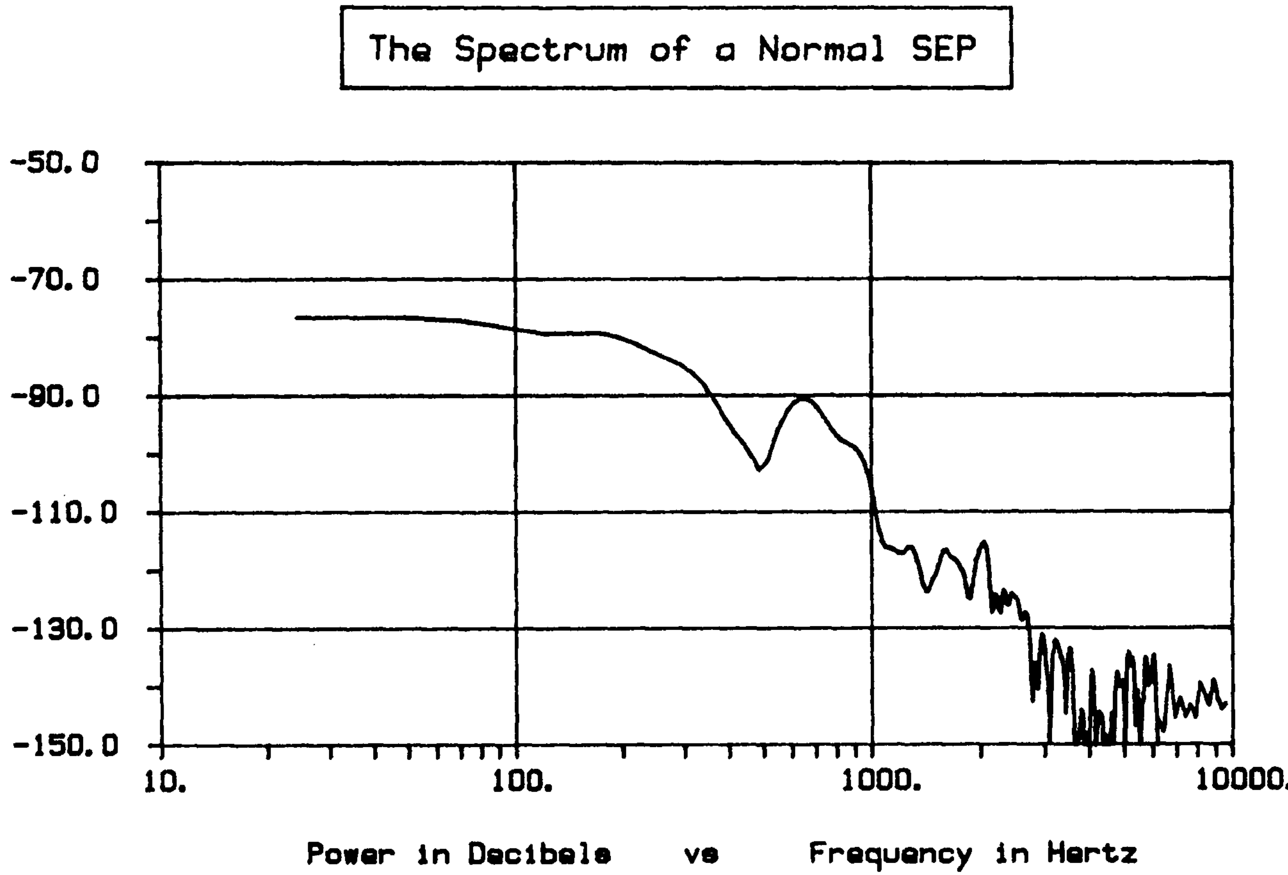

The Spectrum of a Normal SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.5

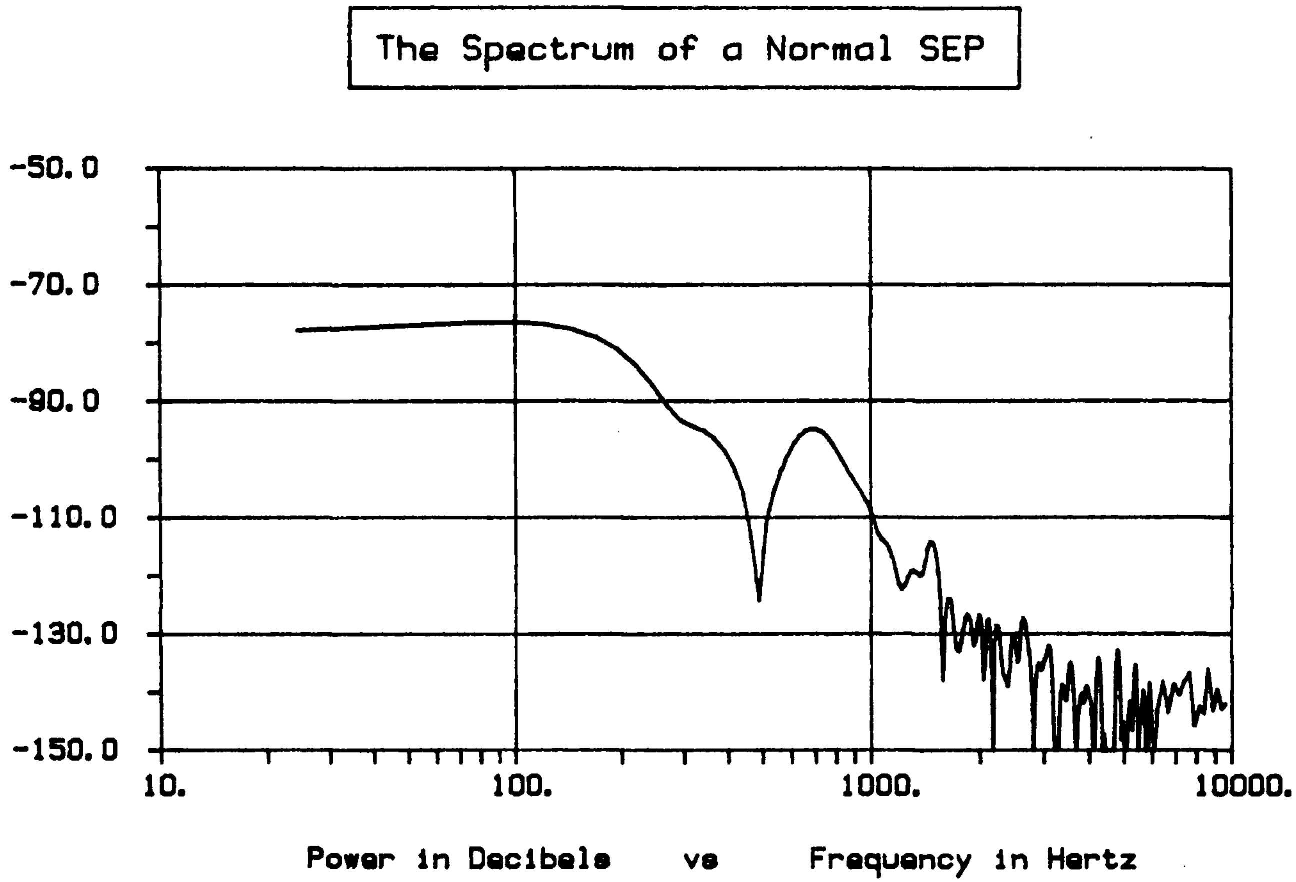

The Spectrum of a Normal SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.6

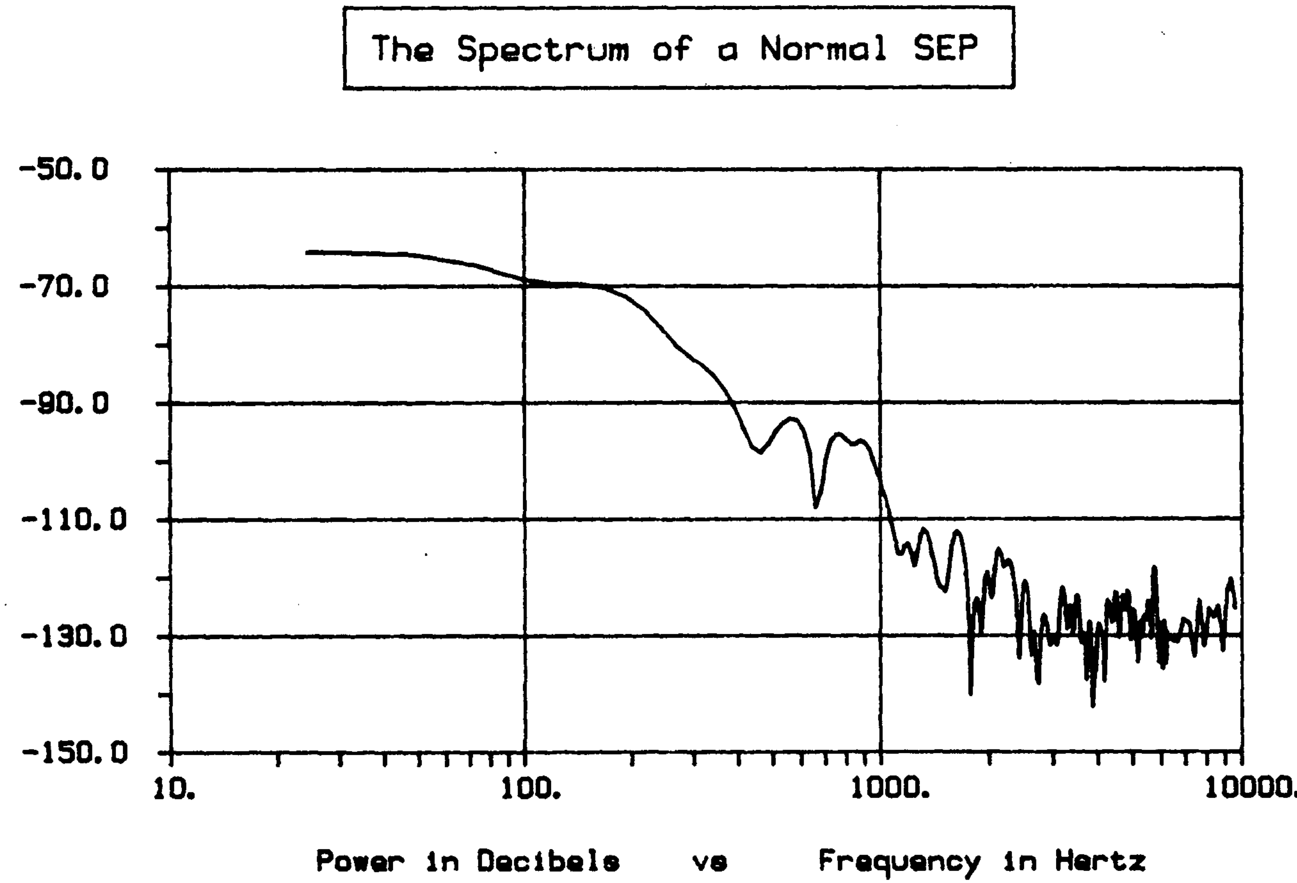

The Spectrum of a Normal SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.7

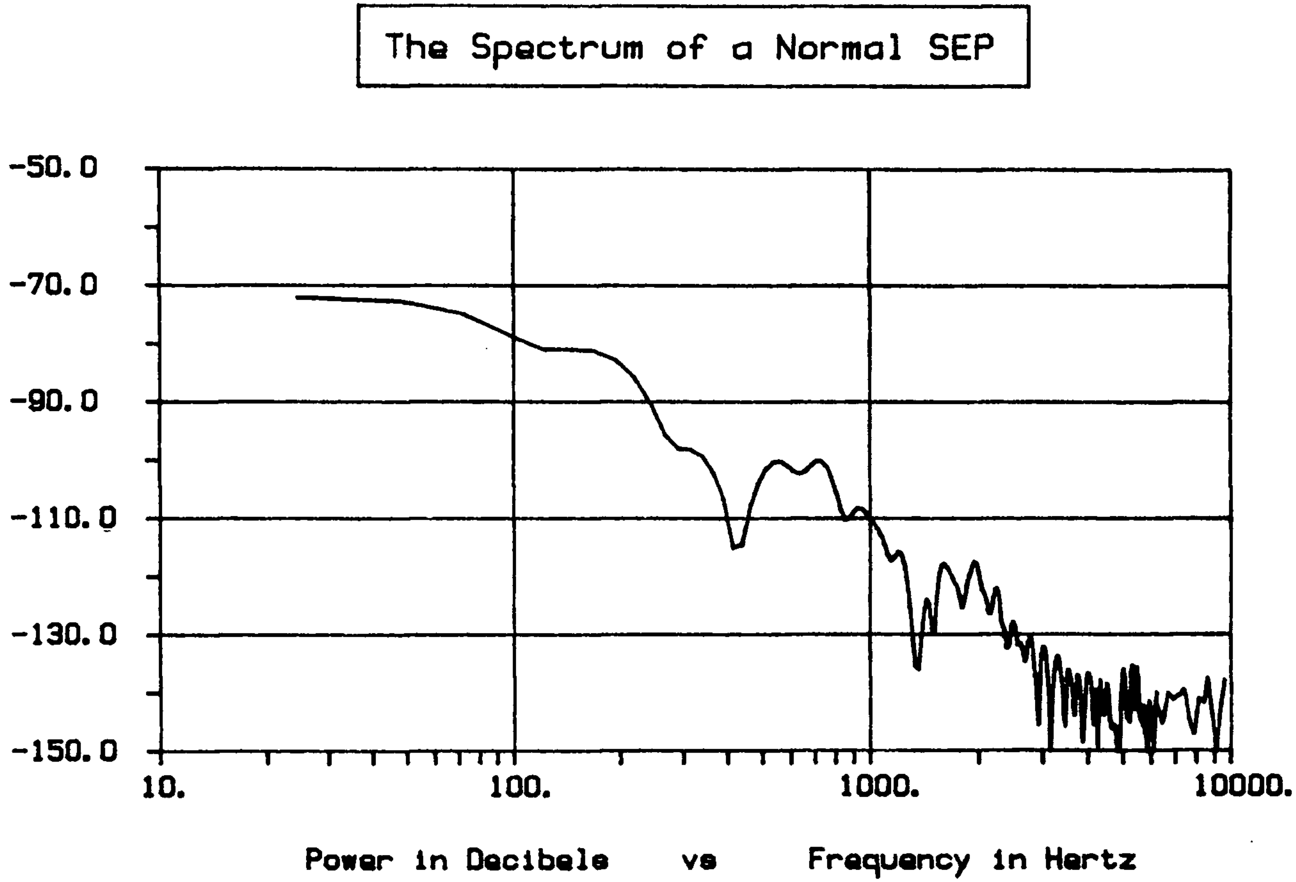

The Spectrum of a Normal SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.8

The mean Dispersion Figure of the set of Normals was 0.0019, with a standard deviation of 0.0020. One of the Normal SEPs had a Dispersion Figure of 0.0090, which is three and one half standard deviations from the mean. The nontypical SEP was re-examined to determine if the recording was excessively noisy, but this was not the case. Setting the outer limit of normality at three standard deviations from the Normal mean judged as abnormal seven of the twenty-five MS SEPs and one of twenty-four Normal SEPs.

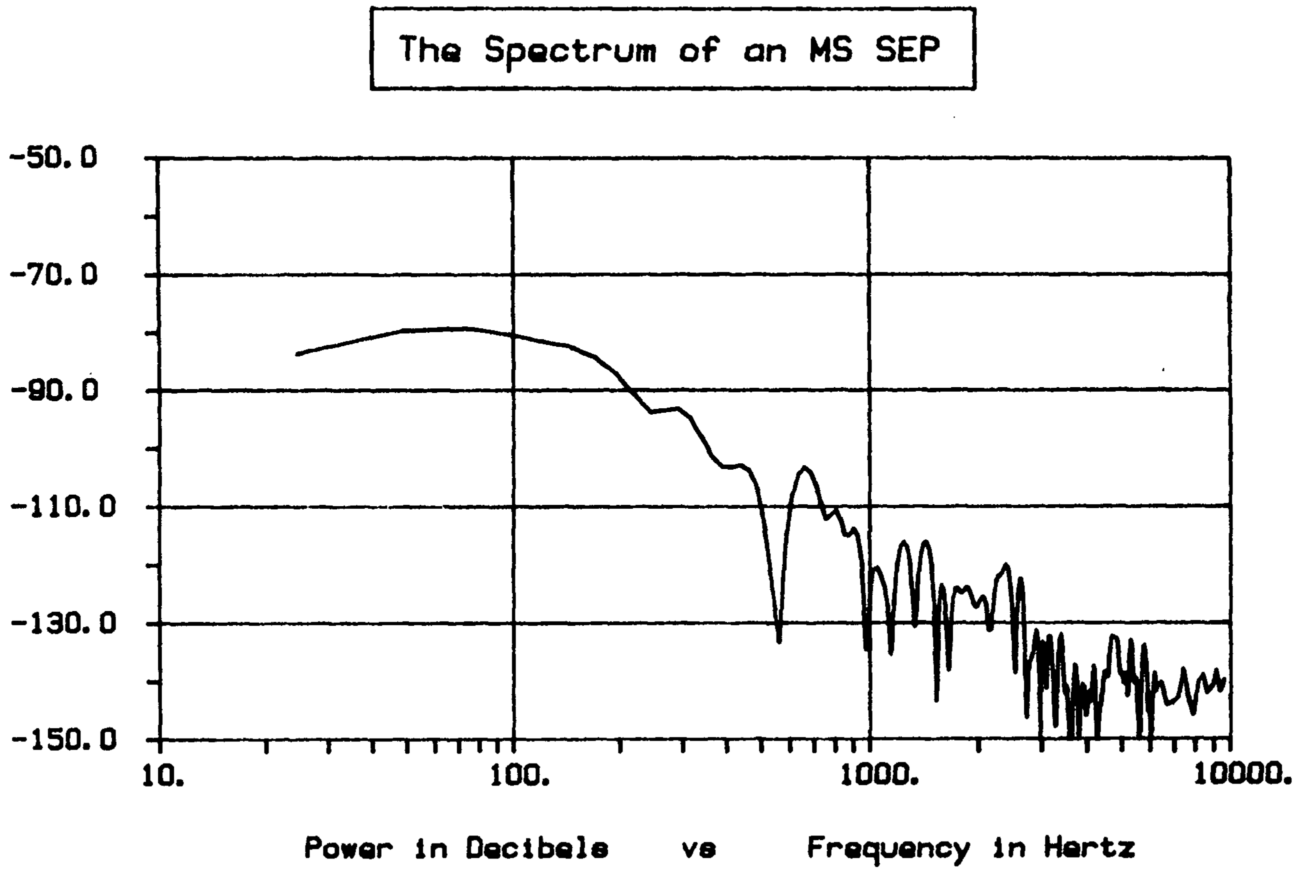

The Spectrum of an MS SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.9

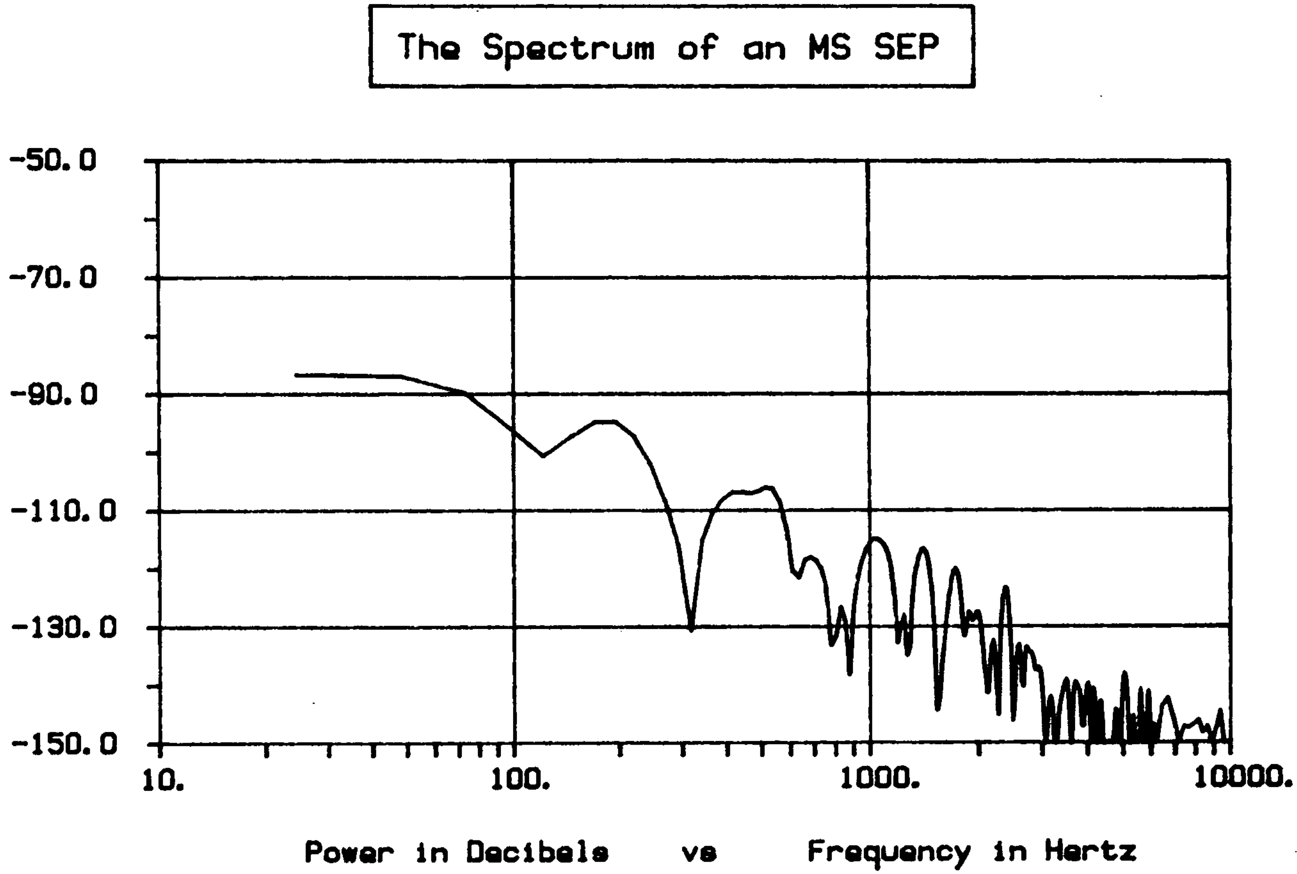

The Spectrum of an MS SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.10

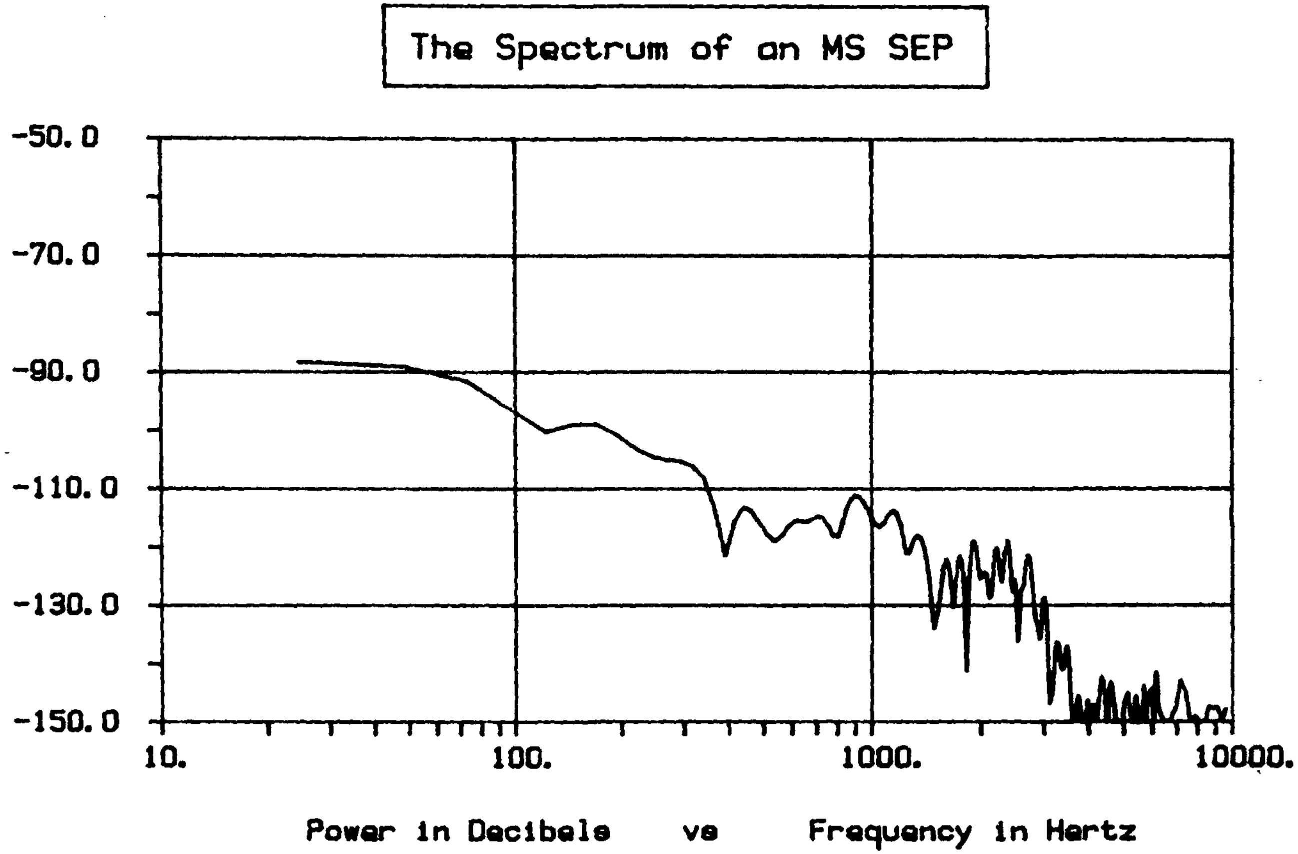

The Spectrum of an MS SEP
-50.0
-70.0
-90.0
-110.0
-130.0
-150.0
10.
100.
1000.
10000.
Power in Decibels vs Frequency in Hertz

Figure 4.11

Table IV

Values of the Dispersion Figure

NORMAL		M S	
0.0010		0.0021	
0.0010		0.0008	
0.0009		0.0003	
0.0048		0.0002	
0.0008		0.0011	
0.0009		0.1212	***
0.0007		0.0020	
0.0003		0.0150	***
0.0020		0.0029	
0.0013		0.0005	
0.0034		0.0006	
0.0022		0.0004	
0.0090	***	0.0000	
0.0004		0.0106	***
0.0002		0.0032	
0.0010		0.0158	***
0.0006		0.0052	
0.0029		0.0261	***
0.0027		0.0007	
0.0043		0.0002	
0.0004		0.0235	***
0.0005		0.0037	
0.0010		0.0304	***
0.0031		0.0009	
		0.0008	

Normal Mean = 0.0019

Normal Standard Deviation = 0.0020

*** More than three standard deviations above the normal mean.

4.4) Discussion

A Blackman data window was chosen because of its very low side lobes; the first sidelobe being down 65 dB. For normal data the power spectrum was down 40-50 dB in the region of interest, compared to the maximum peak. Any substantial sidelobes in this region would have dominated the data. The Blackman window's wide main lobe, although tending to smear the data's frequency spectrum, was tolerated because we were interested in the comparative energy in a band of frequencies.

The frequency band selected here was higher than that selected in previously published works [Roberts, Lawrence + Eisen 1983; Eisen, Lawrence, Roberts, + Hoirch, 1982]. There, the analog filters were set with a low-pass roll-off frequency of 200 Hz. The sampling rate was 2.5 times slower than used here. This meant that the components between 1000 and 2500 Hz. were severely diminished by the filters and not as accurately sampled. In the previous work the median nerve was stimulated at the finger rather than the wrist, the stimulus level was 2.5 times the threshold rather than just enough to induce a visible twitch.

The patient selection was different, in that all of the previous set of patients had symptoms of demyelination in the median nerve, whereas, only six out of twenty-five of the current patients had such. With the recording methods used here, there was not a significant difference between the Normal and MS spectra in the 390 to 1000 Hz band previously used.

It is anticipated that the Dispersion Figure will prove a more subtle test of the dispersion than that previously used, despite the occurrence of one of the normal SEPs outside the range of normality. The enlargement of the set of normal data that the statistics are based on will, we expect, increase the estimate of the deviation of the normals and so include the maverick Normal recordings while still excluding a fair proportion of the MS recordings.

5)

BAYESIAN DECISION MAKER

APPLIED TO FEATURES OF

SOMATOSENSORY EVOKED POTENTIALS

5.1) Introduction

Three different features of the Somatosensory Evoked Potential were measured in a set of Normal volunteers and a set of Multiple Sclerosis volunteers: The latency measurement [Eisen, 1980], the Cost function described in chapter 4, and the Dispersion Figure described in chapter 5. These features needed to be evaluated to determine how sensitive and how reliable indicators of abnormality they were. The optimal thresholds for a judgement of abnormality needed to be determined. The utility of using a combination of features needed to be explored.

A Baysian Decision Maker [Tou + Gonzalez, 1974] was implemented to satisfy these needs. A Baysian Decision Maker, or Pattern Classifier, uses estimates of the distributions of the features, the a priori probability of the results, and estimates of the costs of correct and incorrect decisions, to judge the features of an unknown wave. This judgment is made so as to minimize the expected value of the costs.

5.2) Methods

Twenty-five Multiple Sclerosis patients volunteered for this experiment. Based on a constellation of laboratory tests over a period of several years, each of the patients had been diagnosed as having Multiple Sclerosis. Twenty-four of the volunteers were classified as Definite MS and one was classified as Probable MS. Only six of the twenty-five had symptoms in the median nerve system that was to be stimulated. The rest had spinal, optic or other systems affected. There were nineteen females (ages 39 $\pm$ 11) and six males (ages 45$\pm$9). Their SEPs were recorded using the same technique as used previously for the controls.

A blind test was set up by plotting these SEPs and those of eight normals. Five experienced clinicians, who were not involved in this work and who had never seen this data before, were separately asked to judge which of these median SEPs were abnormal. They had only the graphs of the SEPs to judge from. They were were told that the set contained a mixture of MS and Normals SEPs with no abnormally noisy recordings.

The control and MS SEP's were digitally filtered and then warped to the Standard Wave. The resulting cost functions, along with the latency of the P19 peak, and the Dispersion Figure, were tabulated.

The <u>a priori</u> probability of a wave being MS was set to be 0.50. The cost of mis-diagnosing an MS as Normal (Type A error) was set to be 1.0. The cost of misdiagnosing a Normal as MS (Type B error) was set to be 2001.0. The cost of a correct diagnosis was set as -1.0. This produced a probability threshold of 1000.0. If the probability of the wave occurring, given that is was MS, divided by the probability of the wave occuring, given that it was Normal, was greater than 1000.0 then the wave is judged to be MS. A failure to judge the wave as MS does not imply that the wave is necessarily normal.

The underlying probability distributions were assumed to be normal. The data was tested to determine the reasonableness of this assumption. Table V contains the detailed results of this test. The Normal Cost Function and the Normal Latency were seen to have reasonably normal distibutions. The distribution of the Normal Dispersion Figure appeared less normal. The MS distributions were definitely not normal. They appear to be closer to the average of a uniform distribution and the distribution of the Normals. Even so, when estimating the probabilities, the assumption of normality was used for the MS distributions. This was safe, as the error in the estimates would produce even more conservative judgments. Separate covariance matrices were used for MS and Normal when calculating the probabilities.

The set of MS data records and the set of Normal data records was combined. One by one, each record was separated from the rest of the set. A Baysian Decision Maker [Tou+Gonzalez, 1974] was trained on the rest of the data, and then judged whether the separated record was MS or Normal. The separation was done so that the record being judged would have no influence on the probability estimates used by the Decision Maker.

Table V

Feature	Chi Square	Chi Prob.	Skewness	Kurtosis
NORMAL				
Cost Function	2.147	0.14	0.842	0.283
Latency	0.427	0.51	0.240	0.644
Dispersion	2.783	0.10	2.236	6.145
M S				
Cost Function	4.166	0.04	0.808	-0.763
Latency	14.270	0.00	2.161	4.634
Dispersion	25.294	0.00	4.052	18.096

5.3) Results

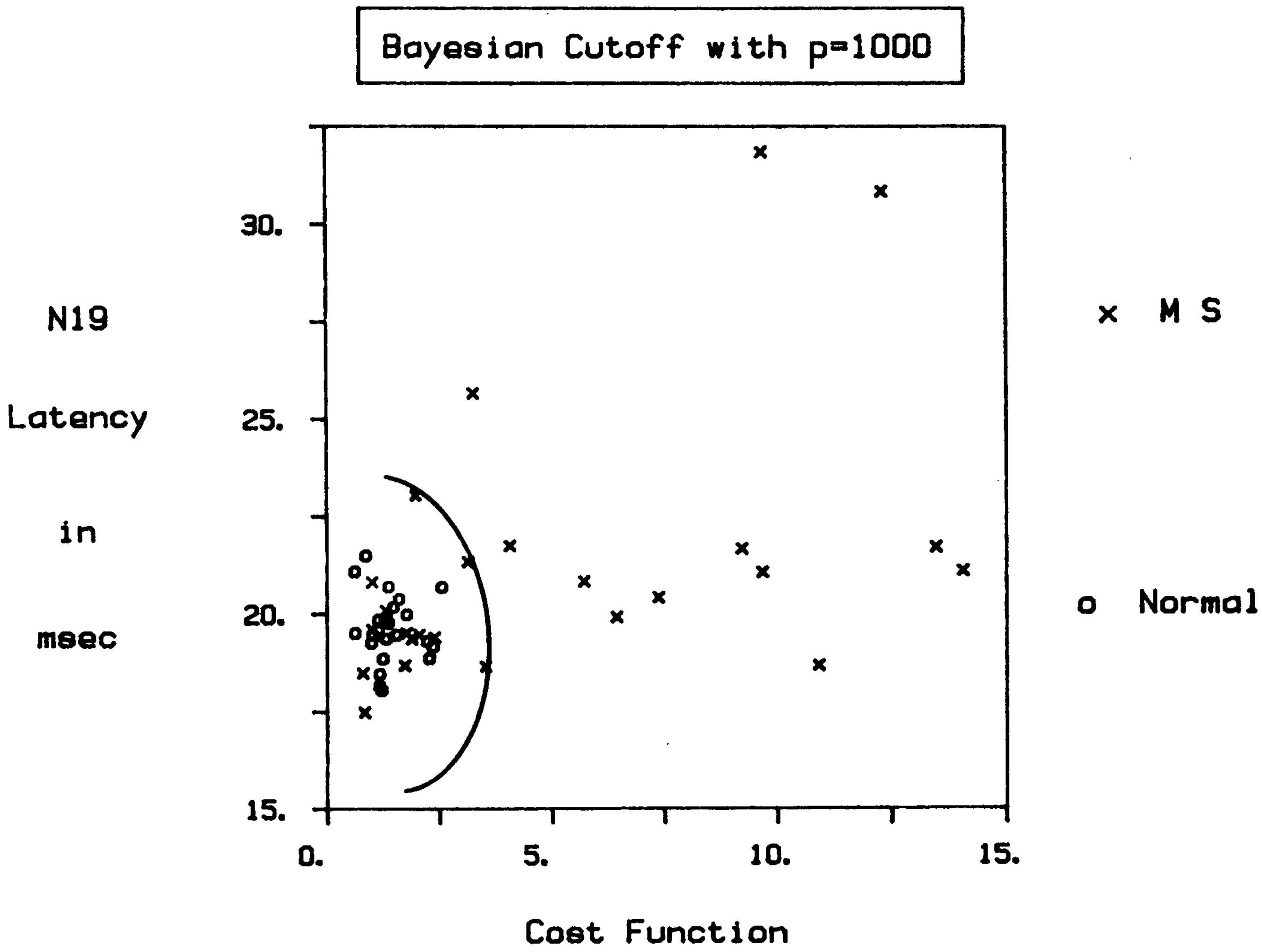
In the blind test three of the five clinicians picked the same five waves as abnormal, all of which were from MS patients. A fourth picked the same five plus one more MS. The fifth clinician had a much more liberal criteria for abnormality and judged sixteen of the SEPs to be abnormal. However, two of these sixteen waves were from normal volunteers, demonstrating the danger of too liberal a criteria of abnormality. Thus, the yield for reliably identifying abnormality in these waves, by experienced clinicians, was 20%.

When judging on the latency measurement alone, 12% of the MS and 100% of the Normals were correctly judged. When judging on the Cost Function alone, 44% of the MS and 100% of the Normals were judged correctly. The Dispersion Figure alone yielded 28% of the MS correctly judged, but misjudged one of the normals. Table VI shows the results from combinations of the features. The best results were produced when judging on both the latency and Cost Function; in that 48% of the MS, and 100% of the normals were judged correctly.

5.4) Discussion

In the Bayesian pattern classifier, the loss matrix was chosen to reflect the enormous psychological damage that would be done to a patient who was misdiagnosed as having MS. Because of the progressive nature of the disease an error in the detection of a patient with MS would be picked up in a later assessment. It is possible that in this investigation the data used to train the pattern classifier may not have been perfectly categorized, as there is no definitive marker for MS. MS requires a fairly large constellation of clinical and laboratory features for its diagnosis. As experience grows with the present patients and as future patients are added to the set the classifier is expected to become more precise.

The misjudging of the one normal when the Dispersion Figure was included was due to the fact that, when that wave is being judged, the Normal covariance matrix is estimated from the rest of the Normals. This estimate is significantly different from that based on all of the Normals. As shown in Figures 5.1, 5.2, and 5.3, when all of the data is used to estimate the statistics the cut-off surface is outside all of the normals. The "leave-one-out method" is an overly pessimistic estimate of the



Bayesian Cutoff with p=1000
N19
Latency
in
msec
30.
25.
20.
15.
0.
5.
10.
15.
Cost Function
x M S
o Normal

Figure 5.1

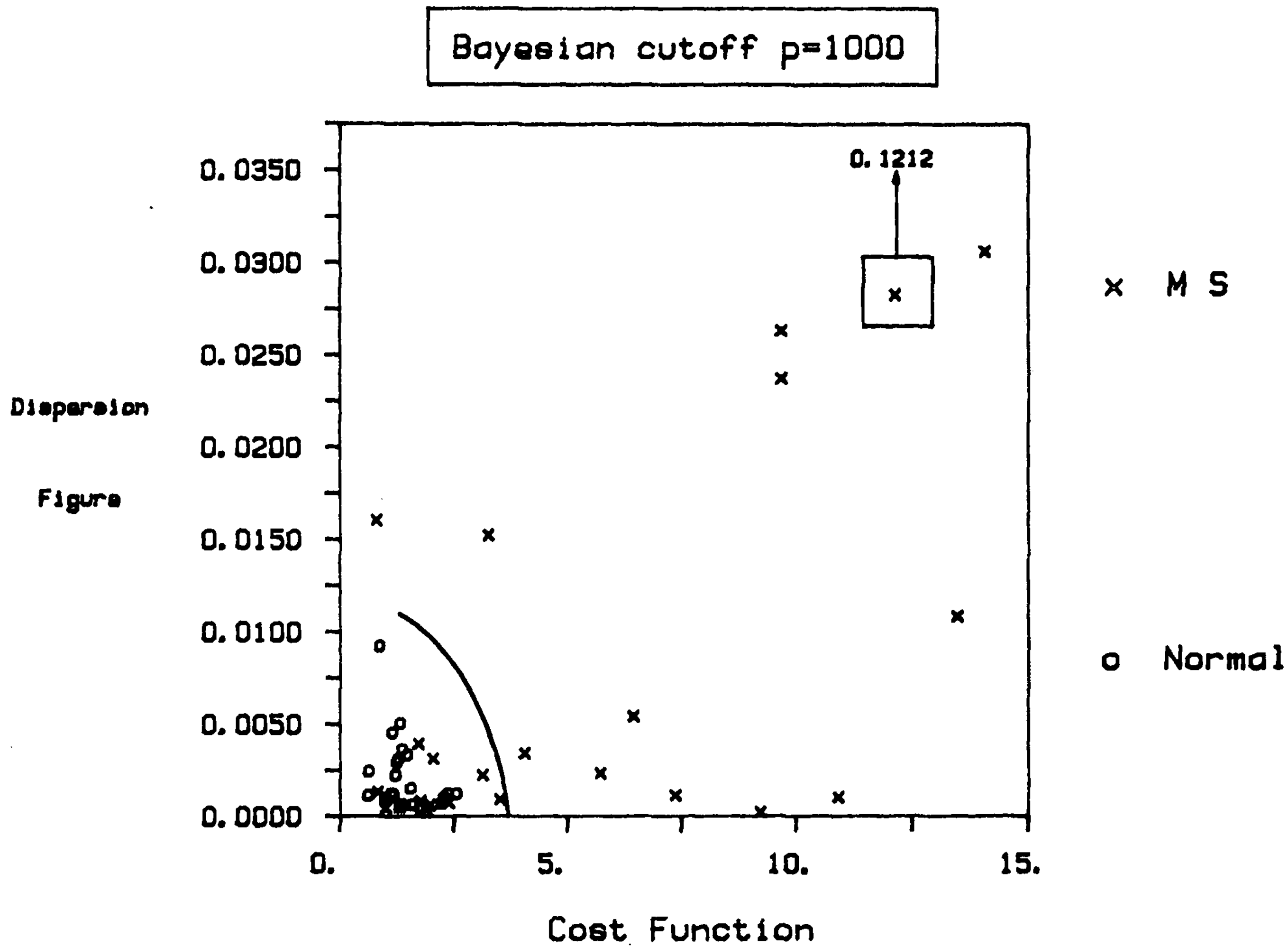

Bayesian cutoff p=1000
0.1212
0.0350
0.0300
0.0250
0.0200
0.0150
0.0100
0.0050
0.0000
Dispersion
Figure
0.
5.
10.
15.
Cost Function
× M S
o Normal

Figure 5.2

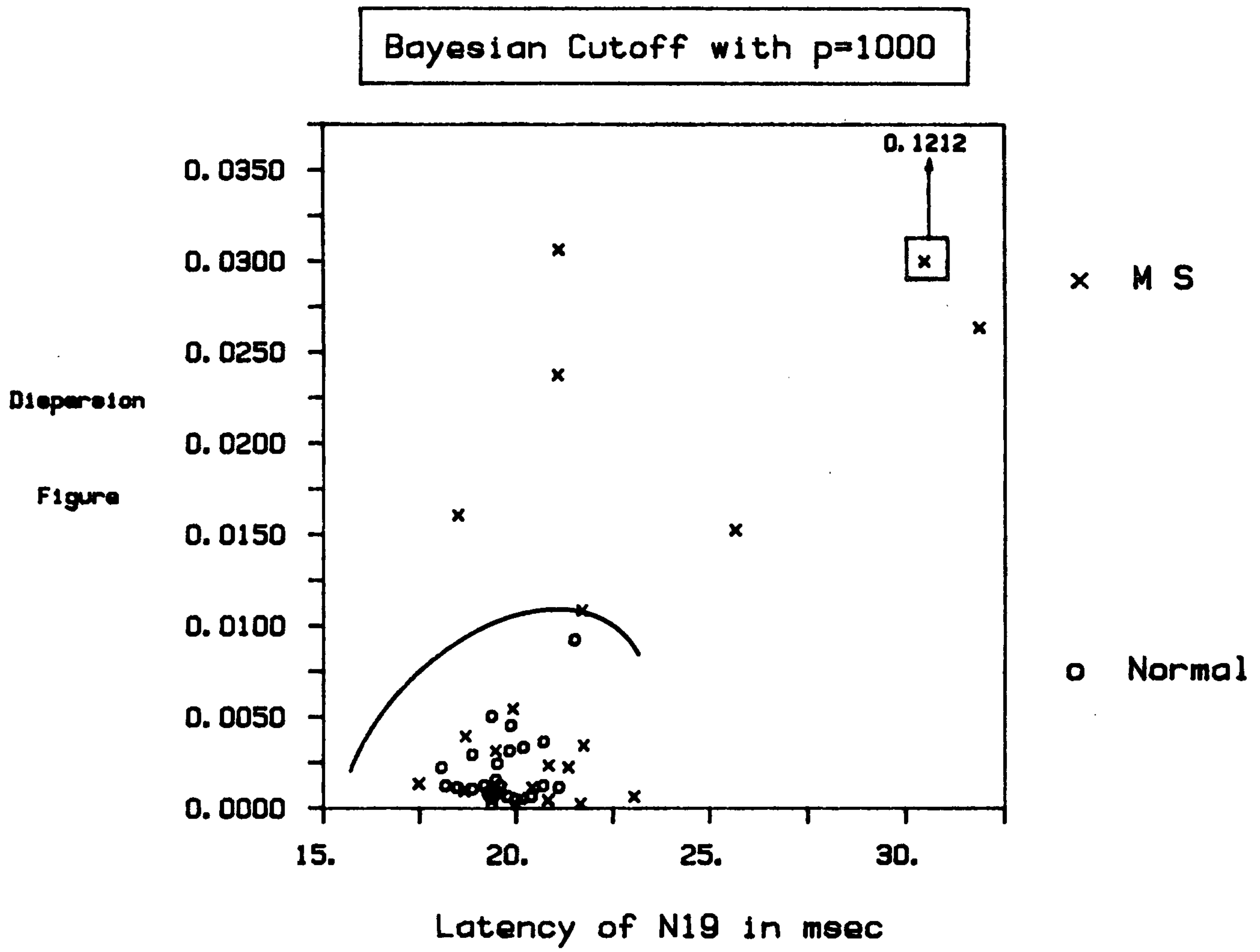

Bayesian Cutoff with p=1000
0.1212
0.0350
0.0300
0.0250
0.0200
0.0150
0.0100
0.0050
0.0000
Dispersion
Figure
15.
20.
25.
30.
Latency of N19 in msec
× M S
o Normal

Figure 5.3

perfomance of a pattern classifier [McGillem, 1981]. Table IV demonstrates that the normal distribution estimate is not a very reliable one. To reliably estimate the distribution of the Dispersion Figure, more samples are needed.

5.5) Conclusions

Both the Cost Function and the Dispersion Figure, when used alone, identified more MS SEPs than the traditional latency measurement and the subjective judgment of experienced clinical personnel. However, the Dispersion Figure caused a misjudgment of one Normal SEP. See the table following for the quantitative details.

Table VI

Features included	**% MS Correct**	**%Normal Correct**
Cost function	44.0	100.0
Latency of N19	12.0	100.0
Dispersion figure	28.0	95.8
Cost and Latency	48.0	100.0
Cost and Dispersion	52.0	95.8
Latency and Dispersion	28.0	95.8
Cost, Latency and Dispersion	52.0	95.8

When the Cost Function was used together with the latency measurement the resulting yield was four times that from using the latency alone and over twice that of the clinicians. Thus, it is concluded that the Cost Function and latency, when used together, are a more sensitive and reliable test for Multiple Sclerosis than any other known feature of the Somatosensory Evoked Potential.

6) CONCLUSIONS

Using digital filtering with a band-pass of 300 to 2500 Hz four negative, and the corresponding positive, peaks of the median SEP were consistently detected in recordings from a group of twenty-four normal subjects. These peaks are only rarely identifiable in unfiltered SEPs, and their presence questions the accepted theories of the origin of the SEP.

A new form of Dynamic Time Warping was developed to be a measure the abnormality of an SEP. The Dynamic Time Warping was applied to the peaks isolated by digital filtering to produce the Cost Function. The values of the Cost Function from a group of twenty-five Multiple Sclerosis patients was significantly different from those of the Normal group.

A measure of the dispersion of the SEP was defined. The Dispersion Figure is the sum of the energy between 1000 and 2500 Hz in the SEP spectra scaled to have a peak value of one. Seven of the Multiple Sclerosis patients had abnormal Dispersion Figures.

A Bayesian Decision Maker was applied to the results of the Cost Function, the Dispersion Figure, and the traditional latency measurement. The latency correctly identified 12% of the MS patients and 100% of the Normals. The Dispersion Figure correctly identified 28% of the MS patients, but misjudged one of the Normals. The Cost Function correctly identified 44% of the

MS and 100% of the Normals. The most accurate combination of features was the Cost Function together with latency which judged 48% of the MS and 100% of the Normals correctly. For comparison, experienced clinicians using just the SEPs correctly identified only 20% of the MS patients. Thus, the Cost Function when combined with the latency measurement is a very sensitive test for the abnormalities in the Somatosensory Evoked Potential.

Further work needs to be done to investigate what other features of the SEP can be isolated by the use of digital filtering. The ongoing clinical use of digital filters will produce a larger pool of data to analyse than that considered in this work.

The Dynamic Time Warping technique presented may prove applicable to signals other than SEPs. A larger group of normal subjects needs to be studied to test the accuracy of the Cost Function. The Cost Function should prove useful in other neurological diseases besides MS.

REFERENCES

Abbruzzese, M., Favale, E. Leandri M., and Ratto, S., New subcortical components of the cerebral somatosensory evoked potential in man. Acta Neurologica Scandinavica, 1978, 48:325-332.

Abrahamian, H.A., et al, Effects of Theopental on Human Cerebral Evoked Responses. Anaesthesiology, 1963, Vol. 24:650-657.

Ackmann, J.J., Elko, P.P., and Wu, S.J., Digital Filtering of On-Line Evoked Potentials. Intern. J. Bio-medical Computing, 1979, Vol 10:291-303.

Allison, T., et al, On the Neural Origin of Early Components of the Human Somatosensory Evoked Potential. In: Progress in Clinical Neurophysiology, Vol. 7: 51-68, 1980, Karger:Basel.

Anziska, B. and Cracco, R.Q. Short latency somatosensory evoked potentials: studies in patients with focal neurological diseases. Electroenceph. Clin. Neurophysiol., 1980, 49:227-239.

Anziska, B., et al, Somatosensory Far Field Potentials: Studies in Normal Subjects and Patients With Multiple Sclerosis. Electroenceph. Clin. Neurophysiol., 1978, Vol 45:602-610.

Aunon, J.I., Computer Techniques for the Processing of Evoked Potentials. Computer Programs in Biomedicine, 1978, Vol 8:243-254.

Aunon, J.I., Sencaj, R.W., Comparison of Different Techniques for Processing Evoked Potentials, Med. Biol. Comput., 1978, Vol 16:642-657.

Aunon, J.I., McGillem C.D, Childers, D.G., Signal Processing in Evoked Potential Research: Averaging and Modeling. CRC Critical Reviews in Bioengineering, July 1981, Vol 5:323-367.

Bellman, Dynamic Programming. Priceton N. J., Princeton University Press 1957.

Boston, J.R., Spectra of Auditory Brainstem Responses and Spontaneous EEG. IEEE Trans. Biomedical Engineering, April 1981, No. 4, Vol. BME-28:334-341.

Brazier, M.A.B., Varieties of Computer Analysis of Electrophysiological Potentials. Electroenceph. Clin. Neurophysiol. Supp. 26, 1967.

Brenner, N., Discrete Fourier Transforms and Applications to Fourier Series and Convolution Integrals. UBC Fourt, University of British Columbia Computer Centre, 1980.

Brown, M.K., Rabiner, L.R., An Adaptive, Ordered, Graph Search Technique for Dynamic Time Warping for Isolated Word Recognition. IEEE Trans. Acoustics, Speech and Signal Processing, vol ASSP-30 No. 4, August 1982.

Buchthal, F., Rosenfelek, A., Behse, F., Sensory potentials of normal and diseased nerves. Peripheral Neuropathy, Vol 1 (Eds) P.J.Dyck P.K.Thomas E.A.Lambert. W.B.Saunders Co. Pp 442-464.

Campbell, J.A., Leandri, M., The Effects of High Pass Filters on Computer-Reconstructed Evoked Potentials. Electroenceph. Clin. Neurophysiol., 1984, 57:99-101.

Carlton, E. H., Katz, S., Is Weiner Filtering an Effective Method of Improving Evoked Potential Estimation? IEEE Trans. Biomedical Engineering, April 1980, No. 4, Vol BME-27:187-192.

Celesia, G. Somatosensory evoked potentials recorded directly from human thalamus and SMI cortical area. Arch. Neurol., 1979, 36:399-405.

Chiappa, K.H., and Ropper, A.L., Evoked Potentials in Clinical Medicine. Part II. N. Engl. J. Med., 1982, Vol 306:1205-1211.

Cooley, J. W. and Tukey, J.W. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 1965, 19:207-301.

Dawson, G.D., Investigations in a patient subject to myoclonic seizure after sensory stimulation. J. Neurology Neurosurgery Psychiatry. Vol 10:141-162, 1947.

Dawson, W.W., Doddington, W., Phase Distortion of Biological Signals: Extraction of Signal From Noise Without Phase Distortion. Electroenceph. Clin. Neurophysiol., 1973, Vol 34:207-211.

Desmedt J.E., Brunko, E., Debecker, J. and Carmeliet, J. The system bandpass required to avoid distortion of early components when averaging somatosensory evoked potentials. Electroenceph. Clin. Neurophysiol., 1974, 37:407-410.

Donchin, E., A Multivariate Approach to the Analysis of Averaged Evoked Potentials. IEEE Trans. Biomedical Engineering, 1966, Vol. BME-13:131-139.

Doyle, D.J., Hyde, M.L., Analogue and Digital Filtering of Auditory Brainstem Potentials. Scandinavian Audiology, 1981 Vol 10:81-89.

Doyle, D.J., Hyde, M.L., Digital Inverse Filtering Methodology for Physiological Signals. Psychophysiology, 1983, Vol 20(5):591-596.

Eisen, A.A. The somatosensory evoked potential. Can. J. Neurol. Sci., 1982a, 9:65-77.

Eisen, A.A., Lawrence, P.D., Roberts, K.B., Hoirch V.M., Dispersion Measurements of the SEP in Suspected Multiple Sclerosis. Neurology, 1982b, Vol 32(2):144.

Eisen, A.A., Nudleman, K., F-wave and cervical somatosensory response conduction from the seventh cortical spinous process to cortex in multiple sclerosis. Canadian Journal of Neurological Science, Vol 5:289-295. 1978.

Eisen, A.A., Odusote, K., Central and peripheral conduction times in multiple sclerosis. Electroencephalography Clinical Neurophysiology. Vol 48:253-265, 1980.

Eisen, A.A., Neurophysiology in Multiple Sclerosis. Neurologic Clinics, Vol 1, No. 3:615-629, 1983.

Fukunaga, K., Krile, T.F., Calculation of Bayes Recogniton Error For Two Multivariate Gaussian Distributions. IEEE Trans. Comp., 1969, Vol 18, No 3:220-229.

Fukushima, T., Mayanagi, Y. and Bouchard G. Thalamic Evoked Potentials to Somatosensory Stimulation In Man. Electroenceph. Clin. Neurophysiol., 1976, Vol 40:481-490.

Giblin, D.R., Somatosensory Evoked Potentials in Healthy Subjects and in Patients with Lesions of the Nervous System. Annals of the New York Academy of Science, Vol 122:93-142.

Glaser, E.M., Ruchkin, D.C., Principles of Neurological Signal Analysis, Academic Press, New York, 1976.

Goff, W.R., et al, Cerebral Somatosensory Responses During Sleep in Man. Electroenceph. Clin. Neurophysiol., 1966, Vol 21:1-9.

Harmon, L.D., Lewis, E.R., Neural Modeling. Advances in Biomedical Engineering and Medical Physics, vol 1, Wiley, New York, 1968.

Hoopen, M.T., Reuver, H.A., Aspects of Average Computations by Aperiodic Stimulation. Med. Biol. Eng., 1972, Vol. 10:621-630.

Jansen, B.H., Bourne, J.R., Ward, J.W., Autoregressive Estimation of Short Segment Spectra for Computerized EEG Analysis. IEEE Trans. Biomedical Engineering, September 1981, Vol. BME-28, No 9:620-638.

Jasper, H.H., The Ten Twenty electrode system of the International Federation. Electroenceph. Clin. Neurophysiol., 1958, Vol. 10:371-375.

Kaufman, L., et al., On the Relation Between Somatic Evoked Potentials and Fields. Intern. J. Neuroscience, 1981, Vol 15:223-239.

Kay, S.M., Marple, S.L.Jr., Spectrum Analysis - A Modern Perspective. Procedings of the IEEE, Vol. 69, No. 11:1380-1419, November 1981.

Kimura, J., Mitsudome, A., Beck, D.O., Yamadu, T., and Dickins, O.S., Field distribution of antidromically activated digital nerve potentials: model for far-field recording. Neurology, 1983, 33:1164-1169.

King, D.W., and Green, J.B. Short latency somatosensory evoked Potentials in Humans. Electroenceph. Clin. Neurophysiol., 1979, 46:702-708.

Kritchevsky, M. and Wiederholt, W.C., Short-latency somatosensory evoked potentials. Arch. Neurol., 1978, 35:706-711.

Landolt, J.P., and Correla, M.J., Neuromathematical Concepts of Point Process Theory. IEEE Trans. Biomedical Engineering, Jan 1978, Vol BME-25:1-12.

Larson, S.J., and Sances, J.A., Averaged evoked potentials in stereotaxic surgery. J. Neurosurg., 1968, 28:227-232.

Lesser, R.P., Lueders, H., Hahn, J. and Klem G. Early somatosensory potentials evoked by median nerve stimulation: intraoperative monitoring. Neurology (Minneap.), 1981, 31:1519-1523.

Lueders, H., Andrish, J., Gurd, A., Weiner, G., and Klem, G. Origin of far field subcortical potentials evoked by stimulation of the posterior tibial nerve. Electroenceph. Clin. Neurophysiol., 1981, 52:336-344.

Lueders, H., Lesser, R., Hahn, J., Little, J. and Klem G. Subcortical somatosensory evoked potentials to median nerve stimulation. Brain, 1983, 106:341-372.

Maccabee, P.J., Pinkhasov, E.I. and Cracco, R.Q., Short latency somatosensory evoked potentials to median nerve stimulation: effect of low frequency filter. Electroenceph. Clin. Neurophysiol., 1983, 55:34-44.

Mauguiere, F., Desmedt, J.E., and Courjon, J., Neural generators of N18 and P14 far-field somatosensory evoked potentials studied in patients with lesions of thalamus or thalamocortical radiations. Electroenceph. Clin. Neurophysiol., 1983(a), 56:283-292.

Mauguiere, F., Desmedt, J.E., and Courjon, J., Astereognosis and dissociated loss of frontal or parietal components of somatosensory evoked potentials in hemispheric lesions: detailed correlations with clinical signs and computerized tomographic scanning. Brain, 1983(b), 106:271-311.

McGill, K.C., et al, On the Nature and Elimination of Stimulus Artifact in Nerve Signals Evoked and Recorded Using Surface Electrodes. IEEE Trans. Biomedical Engineering, Feb. 1982, No 2, Vol. BME-29:129-137.

McGillem C.D, Aunon, J.I., Childers, D.G., Signal Processing in Evoked Potential Research: Applications of Filtering and Pattern Recognition. CRC Critical Reviews in Bioengineering, Oct 1981, Vol 6:225-265.

Mucciardi, A., Goose, E.E., A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties. IEEE Trans. Comp., 1971, Vol 20, No. 9:1023-1031.

Myers, C.S., Rabiner, L.R., A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition. Bell System Technical Journal, Vol 60, No 7, September 1981.

Myers, C.S., Rabiner, L.R., Rosenberg A.E., Performance Tradeoffs In Dynamic Time Warping Algorithms for Isolated Word Recognition. IEEE Trans. Acoustics, Speech, and Signal Processing, Vol ASSP-28, No. 6, December 1980.

Nagy, G., State of the Art in Pattern Recognition. Proc. IEEE, 1968, Vol 56 No 5:836-864.

Nakanishi, T., Tamaki, M., Ozaki, Y. and Arasaki, K. Origins of short latency somatosensory evoked potentials to median nerve stimulation. Electroenceph. Clin. Neurophysiol., 1983, 56:74-85.

Oppenheim, A.V. and Schafer, R.W. Digital signal processing. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.

Pagni, C.A. Somatosensory evoked potentials in thalamus and cortex of man. Electroenceph. Clin. Neurophysiol., 1967, 18:147-155.

Rabiner, L.R. and Gold, B. Theory and application of digital signal processing. Prentice-Hall, Englewood Cliffs, N.J., 1975.

Rabiner, L.R., Rosenberg, A.E., Levison S.E., Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition. IEEE Trans. Acoustics, Speech, and Signal Processing, Vol ASSP-26, No 6, December 1978.

Roberts, K.B., Lawrence P.D., Eisen, A.A., Dispersion of the Somatosensory Evoked Potential (SEP) in Multiple Sclerosis. IEEE Trans. Biomed. Eng., Vol. BME-30, No. 6:360-364. June 1983.

Rossini, P.M., Cracco, R.Q. Cracco, J.B., and Hanse, W.J. Short latency somatosensory evoked potentials to peroneal nerve stimulation: Scalp topography and the effect of different frequency filters. Electroenceph. Clin. Neurophysiol., 1981, 52:540-542.

Rossini, P.M., Treviso, M., DiStefano, E. Di Paolo, B. Nervous impulse propagation along peripheral and central fibers in patients with chronic renal failure. Electroenceph. Clin. Neurophysiol., 1983, 56:293-303.

Ruchkin, D.S., An Analysis of Average Response Computation Based Upon Aperiodic Stimuli. IEEE Trans. Biomedical Engineering, April 1965, Vol. 12:87-93.

Ruchkin, D.S., Walter, D.O., A Shortcoming of the Median Evoked Response. IEEE Trans Biomedical Engineering, May 1975, Vol BME-22:245

Sances, A., et al, Theoretical Investigations and Clinical Application of the Evoked Potential. Conference Proceedings IEEE Eng. in Med. and Biol. Soc. 1979.

Santopierto, R.F., The Origin and Characterization of the Primary Signal Noise, and Interference Sources, in the High Frequency Electrocardiogram. Proc. IEEE, May 1977, Vol. 65:707-713.

Scherg, M., Distortion of the Middle Latency Auditory Response Produced by Analog Filtering. Scandinavian Audiology, 1982, Vol 11:57-60.

Schoenfeld, R.L., A Programming Discipline for Laboratory Computing. IEEE Trans. on Biomedical Engineering, May 1983, Vol. BME-30:257-270.

Sherwood, A.S., Characteristics of Somatosensory Evoked Potentials Recorded over the Spinal Cord and Brain of Man. IEEE Trans. Biomedical Engineering, July 1981, Vol BME-28, No 7:481-485.

Sidman, R.D., Finding The Extended Sources of Evoked Cerebral Potentials: An Application to the N55 Component of the Somatosensory Evoked Potential. Conference Proceedings IEEE Eng. in Med. and Biol. Soc. 1979.

Suter, C.M., Pricipal Component Analysis of Average Evoked Potentials. Exp. Neurol. 1970, Vol 29:317.

Tou, J. T., Gonzalez, R. C., Pattern Recognition Principles. Addison Wesley 1974.

Tsumoto, T., Hirose, N., Nonaka, S., Takahashi, M. Analysis of somatosensory evoked potentials to lateral popliteal nerve stimulation in man. Electroenceph. Clin. Neurophysiol., 1972, 33:379-388.

Van Rotterdam, A., Limitations and Difficulties in Signal Processing by Means of Principle-Component Analysis. IEEE Trans. Biomed. Eng., 1970 Vol BME-17:268-269.

Vas, G.A., Cracco, J.B., Cracco, R.Q., Scalp-recorded short latency cortical and subcortical somatosensory evoked potentials to peroneal nerve stimulation. Electroenceph. Clin. Neurophysiol., 1981, 51:1-8.

Veale, J.L., Mark, R.F. and Rees, S. Differential sensitivity of motor and sensory fibers in human ulnar nerve. J. Neurol. Neurosurg. Psychiat., 1973, 36:75-86.

Wastell, D.G. The application of low-pass linear filters to evoked potential data: filtering without phase distortion. Electroenceph. Clin. Neurophysiol., 1979, 46:355-356.

Weerd, J.P.C., Facts and Fancies About a Posteriori "Weiner" Filtering. IEEE Trans. Biomedical Engineering, March 1981, No. 3, Vol. BME-28:252-257.

Whittle, Optimization Over Time. Chichester New York, Wiley, 1982.

Wolbarscht, M.L., Spekreijse, H., Interference and Noise in Biological Instrumentation. Advances in Biomedical Engineering and Medical Physics, vol 2, Wiley, New York, 1968.

Woody, C.D., Characterization of an Adaptive Filter for the Analysis of Variable Latency Neuroelectric Signals. Med. Biol. Eng., 1967, Vol 5:539-553.

Yamada, T., Kimura J, Young S., and Powes, M. Somatosensory evoked potentials elicited by bilateral stimulation of the median nerve and its clinical application. Neurology (Minneap.), 1978, 28:218-223.

APPENDIX 1) USER'S MANUAL FOR THE PROGRAM EMG

The Main Menu

To run the emg program type:

RUN EMG <RETURN>

If a floppy disk with empty space on it is not in place an error message will be produced. (Just follow the instructions in the error message.) The program will set up its files and then present you with a menu. The EMG program is organized into levels with this main menu being the top level. From here you can select any procedure that you want the computer to execute. When the computer is finished executing the procedure that you selected, it will return to this menu to allow you to make another choice.

Some of the procedures have lower level menus where you can again choose what action that you want the computer to perform. While these lower level menus operate in exactly the same way as the main menu, they only let you choose the actions allowed within the given procedure. If you want to choose other actions you have to return to the main menu. Anytime that you have a choice of actions, if you enter ten times the number corresponding to a certain action, detailed information on that action will be printed on the screen. If you ever enter a reply that is not valid an error message will be produced, and then you can try again. To enter characters or numbers into the computer type the appropriate keys and then press the key marked RETURN or ENTER or CR or LINEFEED to signify the end of the line being input. This key is denoted <RETURN> in this manual.

If you choose "1" you will:

Input Data from the DISA

This procedure has its own menu levels where you can choose what you want the computer to do. If you enter a "0" this help procedure will be executed. A "99" will stop the whole program bypassing the main menu. A "9" will return you to the main menu where you can choose other actions. To input data for a new patient from the DISA enter a "2". Then, the program will prompt you for the patient information before getting the data from the DISA. If you are storing more data from the same patient you should enter a "1" so that you will not have to enter the patient information again. If there was no last patient, the computer will ask for the information.

Entering the patient information

The program will prompt you for information regarding the patient.

PATIENT'S NAME?

LAST **first**

Type in the patients name <RETURN>

AGE SEX

Type in age and sex (M/F) under the coresponding headings. Finally type <RETURN>. If you don't type in a reasonable age the program will respond with an error message of the form:

-345 IS NOT A VALID AGE. TRY AGAIN.

EMG NUMBER?

Type in this patients EMG NUMBER in the form 79-1234. A letter will be appended to this number by the program to give each recording a unique label.

DISEASE = ?

Type in a one line description of the patient.

ANY MISTAKES

"Y" <RETURN> is sufficient for a "YES". Any other first letter including just typing <RETURN> is equivalent to typing "NO". If "YES" is typed then all the above questions are asked again otherwise the program carries on.

The program checks that it can communicate with the DISA interface properly. Error messages are produced if a fault is found in the system. for example:

** CONNECT CABLE OR POWER ON 1500 SYSTEM **

After any error message the following prompt will be typed:

TYPE <RETURN> WHEN THE PROBLEM HAS BEEN REMEDIED

You should correct the cause of the error then type <RETURN>. If the error is not corrected the error message and prompt will be printed again. If no error occurs or the error is corrected, the program will print the following prompt:

CHANNEL.... (1,2,3,4) FINISHED RECORDING THIS AVERAGE? THEN TYPE A 9

will be printed. Now you can take as much time as is necessary to record the patients EMG response on any of the four DISA channels. When you are satisfied with the response type in the number of the DISA channel (1,2,3, or 4) containing the patients EMG response or a "9" if you are finished, then type <RETURN>. If you typed a number other than 1,2,3,4 or 9 then the prompt will be repeated. Entering a "9" causes the program to go back to the beginning. A "9" may be entered even if no data has yet been recorded from this patient.

The following prompt will now be printed:

ENTER INFORMATION ON THE WAVE

Type in where the measurements are being recorded from and any other information about this recording.

The program will now read in the response data from the DISA channel you specified. The routines that read in the data from the DISA do some error checking. If an error is found an error message will be printed such as:

** ERROR FROM STATUS ROUTINE **
** ERROR FROM INPUT ROUTINE **
** ERROR FROM EMGOUT ROUTINE **

In addition the prompt:

WHEN PROBLEM HAS BEEN FIXED, TYPE <RETURN>

is printed. Consult your computer programmer to discover and correct the cause of the error message then type <RETURN> to continue executing the program.

The program will ask:

WAS THE X10 SWITCH ON FOR AMP #4

On the back of the DISA there is a switch for each amplifier that is labeled X10 and X1. Enter "Y" if the switch for the specified amplifier was in the X10 position, otherwise enter "N".

If the program detects the use of the delay line, it will check with you to confirm its use:

Is a delay of 8.73 ms correct? (Y,N,QUIT)

If you answer 'Y' the delay time will be used.
If you reply 'N' the delay time will be ignored.
If you answer 'Q' the wave will not be stored and you can redo it if you so wish.

This completes the recording of one response from a patient. The program will now go back and allow you to record another channel from if you so wish.

Compute

This procedure computes the dispersion index as described in Dispersion of the Somatosensory Evoked Potential (SEP) in Multiple Schlerosis by Kim Roberts, Peter Lawrence and Andrew Eisen. The dispersion index is only valid when computed from waveforms recorded at C3 or C4 and stimulated at the median nerve. If the computer does not find the words "C3" (or "C4") and "median" in the wavetype it will display the wavetype and ask the user:

DO YOU REALLY WISH TO CALCULATE THE DISPERSION INDEX FOR THIS WAVEFORM? ENTER "YES" TO CONTINUE.

Do not enter "YES" unless you are sure that the waveform is actually of the correct type. The computer takes a few seconds to do the thousands of calculations involved in computing the dispersion index. A new datafile is created containing all of the data on the waveform including the dispersion index.

Read data

This procedure reads a datafile from the disk storage. All that you have to do is enter the emgnumber of the file that you want read. Remember that the wave letter is part of the emgnumber. If the file is not available the procedure will tell you so and you can then try again with the proper spelling, or else put the correct floppy disk in place. Once the waveform has been read in it is available to be used by other procedures such as graph, subtract, compute and expand.

Graph Data

To be able to graph the waveform you must be at either a VT100 terminal containing a Selenar graphics card, or a compatable copy. If you try to run this procedure on any other kind of terminal you will get absolute garbage! Graphdata will produce on the screen a graph of the current { i.e. the last one read in} emg waveform. If you want to look at another waveform use Readdata to read it in from the disk, and then return to Graphdata. The screen will clear and the wave graphed. When you have finished looking at the graph hit <RETURN>. The procedure will respond:

Do you want a cursor?

If you answer 'Y' you will enter cursor mode. See the next page for details on the cursor. If you answer anything other than 'Y' or once you have left cursor mode, you will be asked:

Do you wish to save this display?

If you answer 'Y' the picture will remain as a backdrop as you return to the main menu. You may do anything that you wish except terminate the execution of the program and the graphics will still be stored. The next time that you pick graph, the program will check that the delay time and sweepspeed of the now current wave are the same as those of the graphed wave. The vertical voltage scale will automaticly be adjusted to match. The program will ask you

HOW FAR WOULD YOU LIKE THIS WAVE OFFSET (+10 to -10)

The wave will be plotted on top of the previous wave(s) offset by the amount that you respond. +1.0 is about the smallest offset that will separate two waves. You may plot as many waves as you wish by repeating this process. When you wish to clear the graphics screen, answer 'N' when asked if you wish to save the display.

CURSOR MODE

In cursor mode you have four cursors: 1, 2 ,3 and 4. Cursors 3 and 4 are horizontal and cursors 1 and 2 are vertical. You can only move one cursor at a time. To select the current cursor,(the one you can move) press "1", "2", "3", or "4". You do not press <RETURN> when in cursor mode until you wish to leave, as <RETURN> returns you to GRAPH.

To move the current cursor press the appropriate arrow key. To speed up the motion of the cursor press ",". To slow down the motion of the cursor press "-".

The number of the current cursor is displayed on the top right of the screen along with the positions of the cursors that you have used. These positions are displayed in absolute coordinates; either time in milliseconds or voltage in microvolts.

You may change the zero position of a pair of cursors by pressing "0" This makes the position of the current cursor 0.00 and adjusts the value of the other cursor on the same axis to correspond. This allows you to measure differences between two points.

Once you have finished with the cursors press <RETURN>.

Plot

Plot will allow you to plot a wave on the Hewlett-Packard plotter.

Make sure that the plotter is connected, turned on, and has pens and paper before pressing "4" to plot.

The wave will be plotted with the left pen. The axis will be plotted with the right pen. The program will ask:

Do you wish to have the patient information plotted? (Y/N)

If you answer "Y" the patient name etc. will be plotted at the top of the page.

The program will ask:

Do you wish to add to this plot? (Y/N)

If you answer "Y" then next time that you plot the wave will be fitted to the axis already on the paper. If you later change your mind you can reset this by choosing RESET at the main menu.

Subtract

This procedure allows you to subtract one waveform from another. The program will ask you for the emgnumber of the wave that you wish to subtract from. Enter the full emgnumber including the wave letter. Then, the program will ask you for the letter of the wave that you wish to subtract. The wave must be from the same patient. The difference between the two waves will become the current wave in the program which you may then graph or expand.

Average

This procedure allows you to average together waveforms to compute the mean wave and the RMS error.

To start an average select the procedure average at the main menu. At the secondary menu that the procedure will display, choose 1 . The current waveform is used to start the average. All future waves that you will include in the average will be tested to make sure that they have the same sweepspeed and delay values as this the first one. The procedure will ask you for the latencies between which you wish the RMS ERROR to be calculated. The computer will compute the average root mean square error between the waves in the range of latencies specified.

You may now do anything else that you wish with the emg program (as long as you do not start a new average) and you will still be able to add another wave when you are ready to. To add a wave select the average procedure at the main menu and then enter a 2 at the secondary menu. The current wave will be scaled to fit the average and added to the sum. The number of waves averaged will be shown.

If you enter the wrong wave and wish to delete it, you should have it as the current wave and pick 3 at the secondary menu. The program will then delete the wave entirely.

When you wish to calculate the mean wave, choose 4 . The mean wave will be computed and become the current wave. The RMS ERROR will be calculated, between the boundaries that you specified, and then printed out.

Display

This procedure types the wave information on the terminal. The procedure will ask you:

DO YOU WANT A BRIEF OR VERBOSE DISPLAY? (B/V/S)

If you respond with 'B' <RETURN>, only the main items describing the current wave will be typed. If you respond 'S' the status of the program variables will be shown. Otherwise the whole data header will be printed.

Expand

If you select expand at the main menu wou will be able to expand the detail of a portion of the current emg waveform. The program will ask you

ENTER THE LATENCIES BETWEEN WHICH YOU WISH TO EXPAND THE WAVE.

Type the latency in milliseconds at which you wish the expanded wave to start, then leave a space and type the latency at which you wish the expanded wave to stop. If the values that you specify are not possible, the program will write '*** error ***' and ask for the latencies again. Both the time and voltage scales are adjusted to make the expanded wave fill the whole screen. The expanded wave becomes the current wave in the program. This allows you to take a detailed look at a portion of the wave using the procedure graphdata. Or, you could even expand a subportion even further by using expand again. The expanded wave is stored in a file for future access. Its emgnumber is the same as that of the original wave except that the fourth character has been changed to an 'X'. I.e. 82-1234B becomes 82-X234B.

Reset

This procedure allows you to reset all of the program variables. This includes error checks and responces from GRAPH, PLOT and AVERAGE.

APPENDIX 2) THE PROGRAM FILTER

```
CONST
  pi=3.141592654;
  sydefaultname='SY:JUNKK.EMG';
  defaultname = 'DY:TEST.EMG';
  dilabel='Dispersion Index';
  ident='V06.01';


TYPE
  c3=ARRAY[1..3] OF char;
  c5 = ARRAY[1..5] OF char;
  c6=ARRAY[1..6] OF char;
  c8=ARRAY[1..8] OF char;
  c9=ARRAY[1..9] OF char;
  c10=ARRAY[1..10] OF char;
  c11=ARRAY[1..11] OF char;
  c13=ARRAY[1..13] OF char;
  c16=ARRAY[1..16] OF char;
  c17=ARRAY[1.,17] OF char;
  c19=ARRAY[1..19] OF char;
  c20=ARRAY[1..20] OF char;
  c28=ARRAY[1..28] OF char;
  c29=ARRAY[1..29] OF char;
  c30=ARRAY[1..30] OF char;
  c32=ARRAY[1..32] OF char;
  c75 = ARRAY[1..75] OF char;
  c80=ARRAY[1..80] OF char;
  waveformtype=ARRAY[1..1024] OF integer;
  r1026=ARRAY[1..1026] OF real;
VAR
  s1,s2,s3,s4:real;
  stim1,stim2:real;
  delay,latency,sweepspeed:real;
  lok,st1ok,st2ok,dok,s1ok,s2ok,s3ok,s4ok,swok:char;
  wavetype : ARRAY[1..80] OF char;
  channelnumber : integer;
  expansionfactor : integer;
  name : ARRAY[1..30] OF char;
  age : integer;
  sex : c6;
  emgnumber : ARRAY[1..10] OF char;
  patienttype : ARRAY[1..80] OF char;
  date : c9;
  di : real;
```

```
error:boolean;
asciifile : text;
asciifilename : c6;
size,i:integer;
data:r1026;

{*************************************************************************
 *                                                                       *
 *                               OKLETTER                                *
 *                                                                       *
 *         Checks to see  if the letter passed it is a valid letter      *
 *         for a filename.  If not an 'A' is returned.                   *
 *                                                                       *
 *                                                                       *
 *                                                                       *
 *************************************************************************}

FUNCTION okletter(a:char):char;
 BEGIN
   IF ((a<='Z') AND (a>='A')) OR
    ((a<='z') AND (a>='a')) OR
    ((a<='9') AND (a>='0'))

    THEN okletter:=a
    ELSE okletter:='A';

 END;
```

```
{***************************************************************************
 *                                                                         *
 *                         CREATEASCIIFILENAME                             *
 *                                                                         *
 *        This  short procedure constructs the asciifilename from the      *
 *     EMG number.                                                         *
 *                                                                         *
 ***************************************************************************}


PROCEDURE createasciifilename;

 BEGIN {createasciifilename}
   asciifilename[1] := okletter(emgnumber[2]);
   asciifilename[2] := okletter(emgnumber[4]);
   asciifilename[3] := okletter(emgnumber[5]);
   asciifilename[4] := okletter(emgnumber[6]);
   asciifilename[5] := okletter(emgnumber[7]);
   asciifilename[6] := okletter(emgnumber[8]);
 END;
{createasciifilename}


{***************************************************************************
 *                                                                         *
 *                            INCREMENTNAME                                *
 *                                                                         *
 *        Increments the file name in alphabetical order  starting         *
 *        at  the sixth  letter.   Does this  by  incrementing the         *
 *        emgnumber.                                                       *
 *                                                                         *
 ***************************************************************************}


PROCEDURE incrementname;
VAR
  i:integer;
  done:boolean;
 BEGIN
   i:=4;
   emgnumber[i]:=okletter(succ(emgnumber[i]));
   createasciifilename;
 END;
```

```
{****************************************************************************
 *                                                                          *
 *                               READFLOPPY                                 *
 *                                                                          *
 *        Reads in  the specified emg  data  file from  the floppy         *
 *        disk.  All of the emg variables are global.                       *
 *                                                                          *
 ****************************************************************************}

PROCEDURE readfloppy(VAR error:boolean;VAR data:r1026);

VAR
  asciifile:text;

  {*************************************************************************
   *                                                                       *
   *                                RDATA                                  *
   *                                                                       *
   *        This procedure is one large read statement that reads in an   *
   *    emg waveform from an asciifile.  There are many different styles   *
   *    of  asciifiles  produced  by different versions of the software.   *
   *    All of the styles  since  May  1981  are  compatable  with  this   *
   *    procedure  as  it  uses  default  values if the newer fields are   *
   *    empty.                                                             *
   *                                                                       *
   *************************************************************************}

  PROCEDURE rdata;

  VAR
    i:integer;
    skip9:c9;
    skip10:c10;
    skip13:c13;
    skip19:c19;
    skip16:c16;
    skip20:c20;
    skip28:c28;
    skip32:c32;
    id:c6;
    tmp:real;
```

```
BEGIN {rdata}

  FOR i:= 1 TO 8 DO readln(asciifile);

  readln(asciifile,skip19,name);
  readln(asciifile,patienttype);
  readln(asciifile,id);

  FOR i:= 1 TO 3 DO readln(asciifile);
  readln(asciifile,skip9,emgnumber,age,skip10,sex,skip10,date);

  FOR i:= 1 TO 5 DO readln(asciifile);

  readln(asciifile,latency,lok,stim1,st1ok,stim2,st2ok,delay,dok);
  FOR i:= 1 TO 5 DO readln(asciifile);
  readln(asciifile,s1,s1ok,s2,s2ok,s3,s3ok,s4,s4ok);
  readln(asciifile);
  readln(asciifile);
  readln(asciifile,skip13,sweepspeed);
  read(asciifile,skip32,channelnumber);
  read(asciifile,skip32);
  IF (skip32='       Digital expansion factor=')
   THEN readln(asciifile,expansionfactor)
   ELSE
    BEGIN
      readln(asciifile);
      expansionfactor:=1
    END;
  readln(asciifile,wavetype);
  readln(asciifile);

  IF (id<'V06.00') OR (id>'V99.99')
   THEN
    BEGIN
      FOR i:= 1 TO 1000 DO
       BEGIN
         read(asciifile,tmp);
         data[i]:=64*tmp;
       END
    END {idok}
   ELSE
    BEGIN
      FOR i:= 1 TO 1000 DO
      read(asciifile,data[i]);
    END;
```

```
    FOR i:=1001 TO 1026 DO        data[i]:=0.0;

    IF NOT eof(asciifile)
     THEN readln(asciifile,skip16);
    IF (skip16=dilabel)
     THEN read(asciifile,di)
     ELSE di:=999.9;
  END;
 {rdata}

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

BEGIN
  writeln;
  writeln;
  writeln;
  writeln('  D I G I T A L      F I L T E R ');
  writeln;
  writeln;
  writeln('Enter the emgnumber of the data file');
  readln(emgnumber);
  writeln;
  createasciifilename;
  reset(asciifile,asciifilename,defaultname,size);

  error:= (size=-1);

  IF  error
   THEN
    BEGIN
      writeln('The file ',asciifilename,'.EMG does not exist on the floppy disk');
      writeln(chr(007B));
    END
   ELSE
   rdata;
END;
```

```
{**********************************************************************
 *                                                                    *
 *                      CONVERTRECORDTOASCII                          *
 *                                                                    *
 *       This  long  boring  procedure  does  the  actual  work  of   *
 *    converting the patient data into the final form.   This  routine *
 *    is one large write statement.                                   *
 *                                                                    *
 **********************************************************************}


PROCEDURE convertrecordtoascii;

CONST
  b5='     ';

VAR
  dataitem:integer;
  tmp:real;

 BEGIN  {convertrecordtoascii}
   writeln(asciifile,'*                                                          *');
   writeln(asciifile,'*                                                          *');
   writeln(asciifile,'*              E M G   P A T I E N T   D A T A             *');
   writeln(asciifile,'*                                                          *');
   writeln(asciifile,'*                                                          *');
   writeln(asciifile,'*                      ',asciifilename,'.EMG                          *');
   writeln(asciifile,'*                                                          *');
   writeln(asciifile);
   writeln(asciifile,'Name of patient:    ',name);
   writeln(asciifile,patienttype);
   writeln(asciifile,ident);
   writeln(asciifile);
   writeln(asciifile);
   writeln(asciifile,'         EMGNUMBER        AGE           SEX            DATE');
   write(asciifile,'         ',emgnumber,'     ',age:7,'           ',sex);
   writeln(asciifile,b5,b5,date);

   writeln(asciifile);
   writeln(asciifile);
   writeln(asciifile,'         LATENCY       STIMULUS1      STIMULUS2       DELAYII');
   writeln(asciifile,'         seconds         amps           amps          seconds');
```

```
  writeln(asciifile);
  write(asciifile,latency:15:6,lok,stim1:15:6,st1ok);
  writeln(asciifile,stim2:15:6,st2ok,delay:16:6,dok);
  writeln(asciifile);
  writeln(asciifile);
  writeln(asciifile,'      SENSITIVITY 1   SENSITIVITY 2   SENSITIVITY 3   SENSITIVITY 4');
  writeln(asciifile,'     volts/division        v/d              v/d              v/d');
  writeln(asciifile);
  writeln(asciifile,s1:15:10,s1ok,s2:15:10,s2ok,s3:15:10,s3ok,s4:15:10,s4ok);
  writeln(asciifile);
  writeln(asciifile);
  writeln(asciifile,'sweepspeed = ',sweepspeed:9:6,swok,' seconds/division');
  write(asciifile,'the data is from channel number ',channelnumber:3);
  writeln(asciifile,'       Digital expansion factor=',expansionfactor:3);

  writeln(asciifile,wavetype);
  writeln(asciifile);

  FOR dataitem:=1 TO 1000 DO
   BEGIN
     tmp:=data[dataitem];
     IF abs(tmp)>32750.0
      THEN tmp:=32750.0*abs(tmp)/tmp;
     write(asciifile,round(tmp):7);
     IF dataitem MOD 10 = 0
      THEN
      writeln(asciifile);
   END;
  writeln(asciifile);
  IF abs(di)>10
   THEN
   writeln(asciifile,' The dispersion index is not defined')
   ELSE writeln(asciifile,'Dispersion index ',di:5:1);
  writeln(asciifile)
END; /* convertrecordtoascii */
```

```
{**********************************************************************
 *                                                                    *
 *                            TOFLOPPY                                *
 *                                                                    *
 *       Writes out the new data file to the floppy disk from the     *
 *       global variables.                                            *
 *                                                                    *
 **********************************************************************}

PROCEDURE tofloppy;

 BEGIN {flop}
   emgnumber[1]:='D';
   emgnumber[2]:='F';
   emgnumber[4]:='1';
   createasciifilename;
   reset(asciifile,asciifilename,defaultname,size);
   writeln;
   WHILE (size<>-1)
   DO
    BEGIN
      write(' EMGNUMBER ',emgnumber);
      writeln(' has already been used');
      incrementname;
      reset(asciifile,asciifilename,defaultname,size);
    END;

   size := 20;
   rewrite (asciifile,asciifilename,defaultname,size);
   WHILE size = -1 DO
    BEGIN
      writeln('The floppy disk is full or not present.  ',chr(007B));
      writeln('Place a new data disk in the drive, then press <RETURN>. ');
      readln;
      rewrite (asciifile,asciifilename,defaultname,size);
    END;

   convertrecordtoascii;
   close (asciifile);
   writeln;
   writeln(' So ',emgnumber,' will be used');
   writeln;
   writeln('DY:',asciifilename,'.EMG has been created');
 END;  {flop}
```

```
{***********************************************************************
 *                                                                     *
 *                              removedc                               *
 *                                                                     *
 *          removedc subtracts from all the data  the  average  OF  the    *
 *     data between windowstart AND windowstop.                        *
 *                                                                     *
 *                                                                     *
 ***********************************************************************}


PROCEDURE removedc(VAR data:r1026;windowstart,windowstop:integer);

VAR
  windowwidth,i:integer;
  sum,average:real;

 BEGIN
   sum:=0.0;
   windowwidth:=windowstop-windowstart+1;


   FOR i := windowstart TO windowstop DO
   sum := sum + data[i];

   average :=sum/windowwidth;

   FOR i := 1 TO 1024 DO
   data[i] := data[i] - average;
 END;
```

```
{**********************************************************************
 *                                                                    *
 *                          blackmanwindow                            *
 *                                                                    *
 *          this PROCEDURE applies   a blackman  window   between     *
 *     windowstart  AND  windowstop, WITH all OF the rest OF the window  *
 *     equal TO zero.                                                 *
 *                                                                    *
 *                                                                    *
 **********************************************************************}

PROCEDURE blackmanwindow(VAR data:r1026;windowstart,windowstop:integer);

VAR
  windowwidth,i:integer;
  a,b:real;

 BEGIN
   windowwidth:=windowstop-windowstart+1;

   FOR i:= 1 TO windowstart DO
   data[i] := 0.0;

   FOR i := windowstart TO windowstop DO
    BEGIN
      a := cos(2.0*pi*(i-windowstart)/(windowwidth-1));
      b := cos(4.0*pi*(i-windowstart)/(windowwidth-1));
      data[i] := data[i]*(0.42-0.5*a+0.08*b);
    END;

   FOR i := windowstop+1 TO 1026 DO data[i] :=0.0;
 END;
```

```
{****************************************************************************
 *                                                                          *
 *                                CONVERT                                   *
 *                                                                          *
 *        Converts  a  floating  point  number  into  a  character         *
 *        string.   Format controls the number of places after the         *
 *        decimal  point.   A negative  format  means  no  decimal         *
 *        point.  The maximum field size allowed is 30 characters.         *
 *                                                                          *
 ****************************************************************************}


PROCEDURE convert(f:real;format:integer;VAR a:c30;VAR n:integer);
VAR
  t,i,count:integer;
  power:real;

 BEGIN
   a:='                              ';
   count:=1;
   IF f<0
    THEN
     BEGIN
       a[count]:='-';
       count:=succ(count);
       f:=-f;
     END;
   power:=1.0;
   WHILE f>9.9999*power DO power:=10.0*power;
   IF format>=0
    THEN  f:=f+4.9*exp10(-1-format);
    REPEAT
     t:=trunc(f/power);
     a[count]:=chr(60B+t);
     f:=f-power*t;
     power:=power/10.0;
     count:=succ(count)
    UNTIL power<0.5;

   IF format>=0
    THEN
     BEGIN
       a[count]:='.';
       count:=succ(count);
     END;
```

```
   power:=0.1;
   FOR i:= 1 TO format DO
    BEGIN
      t:=trunc(f/power);
      a[count]:=chr(60B+t);
      f:=f-power*t;
      power:=power/10.0;
      count:=succ(count);
    END;
   n:=count-1;
 END;


{***********************************************************************
 *                                                                     *
 *                              SELECT                                 *
 *                                                                     *
 *        Windows  the frequency  spectrum  using  window  with  a     *
 *        boxcar  lower end and a one quarter width Blackman upper     *
 *        end.                                                         *
 *                                                                     *
 ***********************************************************************}

PROCEDURE select(VAR data:r1026);

VAR
  windowwidth,m,n,i,stop,start:integer;
  a,b,df,lowerbound,upperbound:real;
  ok:boolean;
  astring:c30;

 BEGIN
    REPEAT
     readln(lowerbound,upperbound);
     ok:= (0.0<=lowerbound) AND (lowerbound<upperbound) AND (upperbound<11000.0);
     IF NOT ok
      THEN writeln(lowerbound:6:1,upperbound:6:1,' are not valid',chr(7))
    UNTIL ok;

   patienttype[30]:='F';
   patienttype[31]:='I';
   patienttype[32]:='L';
   patienttype[33]:='T';
   patienttype[34]:='.';
   patienttype[35]:=' ';
```

```
convert(lowerbound,0,astring,n);
FOR i:=1  TO n+1 DO patienttype[i+35]:=astring[i];
patienttype[37+n]:='T';
patienttype[38+n]:='O';
patienttype[39+n]:=' ';
convert(upperbound,0,astring,m);
IF n+m<40
 THEN
 FOR i:=1  TO m+1 DO patienttype[i+n+40]:=astring[i];

df:=100.0/sweepspeed/1024.0;

stop:=round(lowerbound/df);
start:=round(upperbound/df)+1;

FOR i:= 1 TO 2*stop DO
data[i]:=0.0;

windowwidth:=(start-stop) DIV 4;
start:=start-(windowwidth DIV 8) ;
IF windowwidth>1
 THEN
 FOR i := start TO start + windowwidth DIV 2 DO
  BEGIN
    a := cos(2.0*pi*(i-start+windowwidth DIV 2)/(windowwidth-1));
    b := cos(4.0*pi*(i-start+windowwidth DIV 2)/(windowwidth-1));
    data[2*i-1] := data[2*i-1]*(0.42-0.5*a+0.08*b);
    data[2*i] := data[2*i]*(0.42-0.5*a+0.08*b);
  END;


FOR i:= 2*(start + windowwidth DIV 2) TO 1026 DO
data[i]:=0.0;

END;
{select}
```

```
{****************************************************************************
 *                                                                          *
 *                                 OPTIM                                    *
 *                                                                          *
 *        Removes the factor  of 1024 produced by the Inverse FFT.          *
 *        Adjusts the scaling factor  and the data values so as to          *
 *        fully utilize the integer range in the data  file.   Two          *
 *        safety  bits  are  left to prevent integer  overflow  on          *
 *        addition.  This leaves 14 bits to be used.                        *
 *                                                                          *
 ****************************************************************************}

PROCEDURE optim(VAR data:r1026);
VAR
  i:integer;
  max,min,s,scale:real;

 BEGIN
   max:=-9.9E20;
   min:=9.9E20;
   FOR i:= 1 TO 1024 DO
    BEGIN
      data[i]:=data[i]/1024.0;
      IF data[i]>max
       THEN max:=data[i];
      IF data[i]<min
       THEN min:=data[i];
    END;

   IF max=min
    THEN writeln('error no signal present',chr(007B))
    ELSE
     BEGIN
       scale:=12800/(max-min);

       CASE channelnumber OF
         1:s:=s1;
         2:s:=s2;
         3:s:=s3;
         4:s:=s4;
        END;

       s:=s/scale;
```

```
       CASE channelnumber OF
         1:s1:=s;
         2:s2:=s;
         3:s3:=s;
         4:s4:=s;
        END;

       min:=min;
       FOR i:= 1 TO 1000 DO
       data[i]:=(data[i]-min)*scale-6400.0;

     END{else}
END;
```

```
{**************************************************************************
 *                                                                        *
 *                              TRANSFORM                                 *
 *                                                                        *
 *        Performs  a  real  to  symmetric-complex   Fast  Fourier        *
 *        Transform  on  the data.   If  inverse  is  true then an        *
 *        inverse  FFT is  calculated.   The inverse  comes out  a        *
 *        factor  of 1024 too large.   This factor  is  removed in        *
 *        OPTIM.  Full details on FFT listed in UBC FOURT.                *
 *                                                                        *
 **************************************************************************}

PROCEDURE transform(inverse:boolean; VAR data:r1026);


VAR
  isign,ndim,nodim,iform:integer;


  PROCEDURE four2(VAR data:r1026;VAR ndim,nodim,isign,iform:integer);
  FORTRAN;

 BEGIN
   ndim := 1024;
   nodim := 1;                { see fft documentation FOR details }

   IF inverse
    THEN
     BEGIN
       isign :=  1;
       iform := -1;
     END
    ELSE
     BEGIN
       isign := -1;
       iform := 0;
     END;

   four2(data,ndim,nodim,isign,iform);

 END;
```

```
{***************************************************************************
 *                                                                         *
 *                              MAIN BLOCK                                 *
 *                                                                         *
 *         This program reads in the data from the floppy disk file        *
 *         that the user requests.  The data has the DC removed, is        *
 *         windowed,  and transformed.   A band  of the spectrum is        *
 *         selected  by  the  user.    The  spectrum  is   inversly        *
 *         transformed and the scaling is optimized.  The result is        *
 *         output into a file on the floppy disk.  If a file of the        *
 *         same name already exists the name is incremented until a        *
 *         unique name is found.                                           *
 *                                                                         *
 ***************************************************************************}

BEGIN {main}
  readfloppy(error,data);
  IF NOT error
   THEN
    BEGIN
      writeln;
      writeln;
      writeln('Enter the lower and upper frequency bounds.');
      close(asciifile);
      removedc(data,10,960);
      blackmanwindow(data,10,960);
      transform(false,data);
      select(data);
      writeln;
      transform(true,data);
      writeln;
      optim(data);
      tofloppy;
    END;
END.
```

APPENDIX 3) THE PROGRAM WARP

```
{***********************************************************************
 *                                                                     *
 *                        CREATEASCIIFILENAME                          *
 *                                                                     *
 *         This  short procedure constructs the asciifilename from the *
 *    EMG number.                                                      *
 *                                                                     *
 ***********************************************************************}

PROCEDURE create(VAR asciifilename:c13; emgnumber:c8);

 BEGIN {createasciifilename}
   asciifilename:='             ';
   asciifilename[1] := emgnumber[2];
   asciifilename[2] := emgnumber[4];
   asciifilename[3] := emgnumber[5];
   asciifilename[4] := emgnumber[6];
   asciifilename[5] := emgnumber[7];
   asciifilename[6] := emgnumber[8];
 END;
{createasciifilename}
```

```
{*************************************************************************
 *                                                                       *
 *                                                                       *
 *                                ZERODC                                 *
 *                                                                       *
 *         Removes any DC component from the array passed to it.         *
 *                                                                       *
 *************************************************************************}

PROCEDURE zerodc(VAR x:rvec);
VAR
  sum:real;
  i:integer;

 BEGIN
   sum:=0.0;
   FOR i:=1 TO nsamples DO
   sum:=sum+x[i];
   sum:=sum/nsamples;
   FOR i:=1 TO nsamples DO
   x[i]:=x[i]-sum;
 END;
{zerodc}
```

```
{***************************************************************************
 *                                                                         *
 *                                 INPUTF                                  *
 *                                                                         *
 *       Reads  the standard  EMG  data  file whose  emgnumber is          *
 *       passed it.  The waveform in microvolts is returned.  The          *
 *       function returns a true value  if the data has been read          *
 *       properly.                                                         *
 *                                                                         *
 ***************************************************************************}


FUNCTION inputf(emgnumber:c8;VAR data:rvec):boolean;

CONST
  dilabel='Dispersion Index';
  ident = 'V06.02';
  defaultname='DY:TEST.EMG';

TYPE
  c2=ARRAY[1..2] OF char;
  c4 = ARRAY[1..4] OF char;
  c5 = ARRAY[1..5] OF char;
  c6 = ARRAY[1..6] OF char;
  c9 = ARRAY[1..9] OF char;
  c10=ARRAY[1..10] OF char;
  c13=ARRAY[1..13] OF char;
  c16=ARRAY[1..16] OF char;
  c19=ARRAY[1..19] OF char;
  c20=ARRAY[1..20] OF char;
  c28=ARRAY[1..28] OF char;
  c30=ARRAY[1..30] OF char;
  c32=ARRAY[1..32] OF char;
  c75 = ARRAY[1..75] OF char;
  c80=ARRAY[1..80] OF char;
  c82=ARRAY[1..82] OF char;
  r512=ARRAY[1..512] OF real;
  waveformtype=ARRAY[1..1000] OF integer; { N O T E   1 0 0 0   NOT   1 0 2 4}
  statustype= RECORD
              s:ARRAY[1..4] OF real; {MODIFIED NOV 1983}
              stim1,stim2:real;
              delay,latency,sweepspeed:real;
              lok,st1ok,st2ok,dok,swok:char;
              sok:c4; {MODIFIED NOV 1983}
             END;
```

```
VAR
  wavetype : ARRAY[1..80] OF char;
  status : statustype;
  channelnumber : integer;
  expansionfactor : integer;
  name : c30;
  age : integer;
  sex : c6;
  patienttype : c80;
  date : c9;
  waveform : waveformtype;
  di : real;
  i:integer;
  scale:real;
  asciifile:text;
  asciifilename:c13;
  size:integer;


  {***************************************************************************
   *                                                                         *
   *                                RDATA                                    *
   *                                                                         *
   *        Does the actual reading of the data file.                        *
   *                                                                         *
   ***************************************************************************}


  PROCEDURE rdata;

  VAR
    tmp,i:integer;
    id:c6;
    skip9:c9;
    skip10:c10;
    skip13:c13;
    skip19:c19;
    skip16:c16;
    skip20:c20;
    skip28:c28;
    skip32:c32;
```

```
BEGIN {rdata}

  FOR i:= 1 TO 8 DO readln(asciifile);

  readln(asciifile,skip19,name);
  readln(asciifile,patienttype);
  readln(asciifile,id);

  FOR i:= 1 TO 3 DO readln(asciifile);
  readln(asciifile,skip9,emgnumber,age,skip10,sex,skip10,date);

  FOR i:= 1 TO 5 DO readln(asciifile);

  WITH status DO
   BEGIN
     readln(asciifile,latency,lok,stim1,st1ok,stim2,st2ok,delay,dok);
     FOR i:= 1 TO 5 DO readln(asciifile);
     readln(asciifile,s[1],sok[1],s[2],sok[2],s[3],sok[3],s[4],sok[4]);
     readln(asciifile);
     readln(asciifile);
     readln(asciifile,skip13,sweepspeed);

     read(asciifile,skip32,channelnumber);
     read(asciifile,skip32);
     IF (skip32='       Digital expansion factor=')
      THEN readln(asciifile,expansionfactor)
      ELSE
       BEGIN
         readln(asciifile);
         expansionfactor:=1
       END;

     readln(asciifile,wavetype);
   END;
  {with status}
```

```
   readln(asciifile);

   IF (id<'V06.00') OR (id>'V99.99')
    THEN
     BEGIN
       FOR i:= 1 TO 1000 DO
         BEGIN
           read(asciifile,tmp);
           waveform[i]:=64*tmp;
         END
     END {idok}
    ELSE
     BEGIN
       FOR i:= 1 TO 1000 DO
       read(asciifile,waveform[i]);
     END;


   IF NOT eof(asciifile)
    THEN readln(asciifile,skip16);
   IF (skip16=dilabel)
    THEN read(asciifile,di)
    ELSE di:=999.9;
   close(asciifile);
 END;
{rdata}
```

```
BEGIN
  create(asciifilename,emgnumber);
  reset(asciifile,asciifilename,defaultname,size);

  IF  (size=-1)
   THEN
    BEGIN
      writeln('The file ',asciifilename,'.EMG does not exist on the floppy disk');
      writeln;
      inputf:=false;
    END
   ELSE
    BEGIN

      rdata;
      scale:=status.s[channelnumber]/expansionfactor*1.0E6/32.0/64.0;
      FOR i:= 1 TO nsamples DO data[i]:=waveform[(i-1)*every+start]*scale;
      zerodc(data);
      inputf:=true;
    END;
 END;
{inputf}
```

```
{*************************************************************************
 *                                                                       *
 *                          FETCHSTANDARDWAVE                            *
 *                                                                       *
 *        Fetcheds the standard wave from the file TMPLT.WRP.            *
 *                                                                       *
 *************************************************************************}

FUNCTION fetchstandardwave(VAR x:rvec):boolean;

VAR
  size:integer;
  swfile:FILE OF rvec;

 BEGIN
   fetchstandardwave:=false;
   reset(swfile,swfilename,defaultname,size);
   IF size=-1
    THEN writeln('*** *** ERROR *** *** standard wave file reset problem')
    ELSE
     BEGIN
       x:=swfile^;
       close(swfile);
       fetchstandardwave:=true;
     END;
    {size}
 END;
{fetch}
```

```
{************************************************************************
 *                              W A R P                                 *
 *                                                                      *
 *        From  the two  waves  a and  b  warp  finds  the optimal      *
 *        warping function w and the minimal cost function.            *
 *                                                                      *
 *                                                                      *
 ************************************************************************}

{$A-} { Compiler Switches to turn off array bounds and stack checking}
{$T-}

PROCEDURE  warp(a,b:rvec; VAR w:ivec; VAR cost:real);
TYPE
  precord=RECORD
            pnt:ivec;
          END;

VAR
  pay1,last,this:rvec;
  enter,tmp,count,i,size:integer;
  payoff,t2:real;
  n0inarow,pointers:ivec;
  pointfile: FILE OF precord;
```

```
{***************************************************************************
 *                                                                         *
 *                                                                         *
 *                               MAX                                       *
 *                                                                         *
 *       This function returns the maximum  between i and stop of          *
 *       the vector t and its position p.                                  *
 *                                                                         *
 ***************************************************************************}

FUNCTION max(VAR p:integer;i,stop:integer; t:rvec):real;
  {this function returns the maximum between i and stop of the vector t and its position p}
VAR
  j:integer;
  tj,tmp:real;

 BEGIN
   tmp:=-9.99999E20;
   FOR j:= i TO stop DO
    BEGIN
      tj:=t[j];
      IF tj>tmp
       THEN
        BEGIN
          p:=j;
          tmp:=tj;
        END;
    END;
   max:=tmp;
 END; {max}
```

```
{***************************************************************************
 *                                                                          *
 *                              FINDTHEMAX                                  *
 *                                                                          *
 *        This  procedure finds  the maximum  payoff possible from          *
 *        position  [count,i] and returns themax and p the pointer          *
 *        to its location.   if the maximum payoff is negative and          *
 *        i is  an edge  position then p and themax are  zeroed to          *
 *        indicate a path outside the array.                                *
 *                                                                          *
 *                                                                          *
 ***************************************************************************}

PROCEDURE findthemax(bcount:real;i:integer;a,last:rvec;
                     VAR n0inarow:integer; VAR themax:real;VAR p:integer);


VAR
  tmp,iplus4,j,stop:integer;
  sum:real;
  t:rvec;


 BEGIN
   {The payoff function is symmetrical between a and b.

    *
    *        If w(t)<>w(t+1)
    *        then
    *        payoff=sum i=w(t) to w(t+1) b[t]*a[i]
    *                   -alpha * (w(t)-w(t+1)-1) * (w(t)-w(t+1)-1)
    *        else
    *        payoff=b[t]*a[w(t)]-alpha*(2*number of zeros in a row + 1)**2
    *
    *        This makes the penalty for k zeros in a row the same as for a jump of k+1
    *       which preserves the desired symmetry.}

   n0inarow:=succ(n0inarow);
   sum:=bcount*a[i];

   t[i]:=sum-alpha*(2*n0inarow-1)+last[i];
   {2n0+1=n0*n0-(n0-1)*(n0-1)}
```

```
{look only at i and above as monotonic ^ path}
{look only 4 up for speed}
iplus4:=i+4;
IF iplus4<nsamples
 THEN stop:=iplus4
 ELSE stop:=nsamples;
tmp:=0;

FOR j:=succ(i) TO stop DO
 BEGIN
   t[j]:=sum-(tmp*tmp)*alpha+last[j];
   tmp:=succ(tmp);
   sum:=sum+bcount*a[j];
 END;

themax:=max(p,i,stop,t);

{check if going out the top is better}
IF (iplus4>nsamples)  AND (themax<0.0)
 THEN
  BEGIN
    themax:=0.0;
    p:=0;{a 0 pointer means out the top}
  END;

IF p<>i
 THEN n0inarow:=0;

END;
{findthemax}
```

```
BEGIN
  FOR i:=1 TO nsamples DO
   BEGIN
     last[i]:=0.0;
     n0inarow[i]:=0;
   END;
  {zeroing loop}

  size:=nsamples+1;
  rewrite(pointfile,pfilename,defaultname,size);
  IF size<1
   THEN writeln('*** pointfile rewrite error ***',chr(7))
   ELSE

    BEGIN
      writeln('Warping begins.   Columns done:  ');

      FOR count:=nsamples DOWNTO 1 DO
       BEGIN
         IF count MOD 10 = 0
          THEN writeln;
         write(count:5); { display count to show activity }
         FOR i:=1 TO nsamples DO
         findthemax(b[count],i,a,last,n0inarow[i],this[i],pointers[i]);

         last:=this;
         pointfile^.pnt:=pointers;
         put(pointfile);
         pay1[count]:=this[1];
       END;
      {for count}

      writeln;
```

```
{Find the best payoff in the column [1,*] }

payoff:=max(w[1],1,nsamples,last);

{find the best payoff in the row [*,1] if this is better then the
 optimal path starts below 1 and comes up partway}

t2:=max(enter,1,nsamples,pay1);

writeln('Column 1 payoff=',payoff:12:6,'  t2 =',t2:12:6);

IF payoff<t2
 THEN
  BEGIN
    payoff:=t2;
    { a pointer of -1 indicates a path below the array}
    FOR count:=1 TO enter-1 DO w[count]:=-1;
    {Enter the array at the optimal position}
    w[enter]:=1;
  END

 ELSE enter:=1;

 {start from the column that the path enters the array and track the path
  to the far side}

count:=enter;
writeln('seeking pointer number: ');

WHILE count<nsamples DO
 BEGIN
   seek(pointfile,nsamples-count+1);
   tmp:=pointfile^.pnt[w[count]];
   { a 0 pointer means that the optimal path went out the top}
   IF tmp>0
    THEN w[count+1]:=tmp
    ELSE
    WHILE count<nsamples DO
     BEGIN
       w[count+1]:=tmp;
       count:=succ(count);
     END;
   count:=succ(count);
 END;
{while}
```

```
        IF payoff<=0.0
         THEN cost:=1.0E20
         ELSE cost:=1.0/payoff;

        close(pointfile);
      END;
    {rewrite file ok}
 END;   {warp}

{A+}
{T+} { Compiler Switches to turn on array bounds and stack checking}

{*****************************************************************************
 *                                                                           *
 *                              OUTPUTF                                      *
 *                                                                           *
 *       Writes  out the warping function and cost  function into            *
 *       the outfile  WARPED.DAT.                                            *
 *                                                                           *
 *****************************************************************************}


PROCEDURE   outputf(aemgnumber:c8;wtime:ivec;VAR cost:real);


VAR
  i,diff,n0inarow,count: integer;
  outfile:text;

 BEGIN
   write(aemgnumber,' compared to the Standard wave: Cost Factor = ',cost:7:3);

   rewrite(outfile,outfilename);
   writeln(outfile);
   write(outfile,'  ALPHA = ',alpha:12:8,'nsamples =',nsamples:5);
   writeln(outfile,' start = ',start:5,' every = ',every:5);
   write(outfile,aemgnumber,' compared to  the Standard. Cost = ',cost:7:3);
   writeln(outfile);
```

```
    FOR count:=1 TO nsamples DO
     BEGIN
       write(outfile,wtime[count]:5);
       write(wtime[count]:5);
       IF count MOD 10 = 0
        THEN
         BEGIN
           writeln(outfile);
           writeln;
         END;
     END;
    writeln(outfile);
    close(outfile);
 END;
 {outf}

 {**********************************************************************
  *                                                                    *
  *                          MAIN BLOCK                                *
  *                                                                    *
  *                                                                    *
  *       Asks  the user  for  the emgnumber  of  the wave  to  be     *
  *       warped.   Reads in that wave and the standard wave using     *
  *       INPUTF.   Warps these  waves using WARP  and outputs the     *
  *       results using OUTPUTF.                                       *
  *                                                                    *
  **********************************************************************}

BEGIN
  writeln('  DYNAMIC TIME WARPING  TO STANDARD WAVE ');
  writeln;
  IF   fetchstandardwave(bdata)
   THEN
    BEGIN
      writeln;
      writeln('Enter the emgnumber of the wave that you wish to warp.');
      readln(aemgnumber);
      IF inputf(aemgnumber,adata)
       THEN
        BEGIN
          warp(adata,bdata,wtime,cost);
          outputf(aemgnumber,wtime,cost);
        END;   {input a}
    END;    {fetch}
END.
```

APPENDIX 4) THE PROGRAM WEFT

```
CONST
  maxnsamples=300;
  warpfilename='DL:warp.dat';
  sydefaultname='SY:JUNKK.EMG';
  defaultname = 'DY:TEST.EMG';
  dilabel='Dispersion Index';
  ident='V06.01';

TYPE
  c3=ARRAY[1..3] OF char;
  c5 = ARRAY[1..5] OF char;
  c6=ARRAY[1..6] OF char;
  c8=ARRAY[1..8] OF char;
  c9=ARRAY[1..9] OF char;
  c10=ARRAY[1..10] OF char;
  c11=ARRAY[1..11] OF char;
  c13=ARRAY[1..13] OF char;
  c16=ARRAY[1..16] OF char;
  c17=ARRAY[1..17] OF char;
  c19=ARRAY[1..19] OF char;
  c20=ARRAY[1..20] OF char;
  c28=ARRAY[1..28] OF char;
  c29=ARRAY[1..29] OF char;
  c30=ARRAY[1..30] OF char;
  c32=ARRAY[1..32] OF char;
  c33=ARRAY[1..33] OF char;
  c75 = ARRAY[1..75] OF char;
  c80=ARRAY[1..80] OF char;
  waveformtype=ARRAY[1..1024] OF integer;
  warptype=ARRAY[1..maxnsamples] OF integer;
  i1000=ARRAY[1..1000] OF integer;
VAR
  s1,s2,s3,s4:real;
  stim1,stim2:real;
  delay,latency,sweepspeed:real;
  lok,st1ok,st2ok,dok,s1ok,s2ok,s3ok,s4ok,swok:char;
  wavetype : ARRAY[1..80] OF char;
  channelnumber : integer;
  expansionfactor : integer;
  name : c30;
  age : integer;
  sex : c6;
  emgnumber : c10;
  message,patienttype : c80;
```

```
date : c9;
di : real;
error:boolean;
asciifile : text;
asciifilename : c6;
nsamples,start,every,size,i:integer;
data:i1000;
warp:warptype;

{****************************************************************************
 *                                                                          *
 *                               OKLETTER                                   *
 *                                                                          *
 *        Checks to see  if the letter passed it is a valid letter          *
 *        for a filename.  If not an 'A' is returned.                       *
 *                                                                          *
 ****************************************************************************}

FUNCTION okletter(a:char):char;
 BEGIN
   IF ((a<='Z') AND (a>='A')) OR
    ((a<='z') AND (a>='a')) OR
    ((a<='9') AND (a>='0'))

    THEN okletter:=a
    ELSE okletter:='A';

 END;
```

```
{*********************************************************************
 *                                                                   *
 *                         CREATEASCIIFILENAME                       *
 *                                                                   *
 *         This  short procedure constructs the asciifilename from the  *
 *    EMG number.                                                    *
 *                                                                   *
 *********************************************************************}


PROCEDURE createasciifilename;

 BEGIN {createasciifilename}
   asciifilename[1] := okletter(emgnumber[2]);
   asciifilename[2] := okletter(emgnumber[4]);
   asciifilename[3] := okletter(emgnumber[5]);
   asciifilename[4] := okletter(emgnumber[6]);
   asciifilename[5] := okletter(emgnumber[7]);
   asciifilename[6] := okletter(emgnumber[8]);
 END;
{createasciifilename}


{*********************************************************************
 *                                                                   *
 *                          INCREMENTNAME                            *
 *                                                                   *
 *        Increments the file name in alphabetical order  starting   *
 *        at  the sixth  letter.   Does this  by  incrementing the   *
 *        emgnumber.                                                 *
 *                                                                   *
 *********************************************************************}

PROCEDURE incrementname;
VAR
  i:integer;
  done:boolean;
 BEGIN
   i:=4;
   emgnumber[i]:=okletter(succ(emgnumber[i]));
   createasciifilename;
 END;
```

```
{***********************************************************************
 *                                                                     *
 *                             READFLOPPY                              *
 *                                                                     *
 *        Reads in  the specified emg  data  file from  the floppy     *
 *        disk.  All of the emg variables are global.                  *
 *                                                                     *
 ***********************************************************************}

PROCEDURE readfloppy(VAR error:boolean;VAR data:i1000);

VAR
  asciifile:text;


  {*********************************************************************
   *                                                                   *
   *                             RDATA                                 *
   *                                                                   *
   *       This procedure is one large read statement that reads in an *
   *    emg waveform from an asciifile.  There are many different styles *
   *    of  asciifiles  produced  by different versions of the software. *
   *    All of the styles  since  May  1981  are  compatable  with  this *
   *    procedure  as  it  uses  default  values if the newer fields are *
   *    empty.                                                         *
   *                                                                   *
   *********************************************************************}

  PROCEDURE rdata;

  VAR
    i:integer;
    skip9:c9;
    skip10:c10;
    skip13:c13;
    skip19:c19;
    skip16:c16;
    skip20:c20;
    skip28:c28;
    skip32:c32;
    id:c6;
    tmp:integer;
```

```
BEGIN {rdata}

  FOR i:= 1 TO 8 DO readln(asciifile);

  readln(asciifile,skip19,name);
  readln(asciifile,patienttype);
  readln(asciifile,id);

  FOR i:= 1 TO 3 DO readln(asciifile);
  readln(asciifile,skip9,emgnumber,age,skip10,sex,skip10,date);

  FOR i:= 1 TO 5 DO readln(asciifile);

  readln(asciifile,latency,lok,stim1,st1ok,stim2,st2ok,delay,dok);
  FOR i:= 1 TO 5 DO readln(asciifile);
  readln(asciifile,s1,s1ok,s2,s2ok,s3,s3ok,s4,s4ok);
  readln(asciifile);
  readln(asciifile);
  readln(asciifile,skip13,sweepspeed);
  read(asciifile,skip32,channelnumber);
  read(asciifile,skip32);
  IF (skip32='        Digital expansion factor=')
   THEN readln(asciifile,expansionfactor)
   ELSE
    BEGIN
      readln(asciifile);
      expansionfactor:=1
    END;

  readln(asciifile,wavetype);

  readln(asciifile);

  IF (id<'V06.00') OR (id>'V99.99')
   THEN
    BEGIN
      FOR i:= 1 TO 1000 DO
       BEGIN
         read(asciifile,tmp);
         data[i]:=64*tmp;
       END
    END {idok}
   ELSE
    BEGIN
      FOR i:= 1 TO 1000 DO
      read(asciifile,data[i]);
    END;
```

```
    IF NOT eof(asciifile)
     THEN readln(asciifile,skip16);
    IF (skip16=dilabel)
     THEN read(asciifile,di)
     ELSE di:=999.9;
  END;
 {rdata}

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

BEGIN
  createasciifilename;
  reset(asciifile,asciifilename,defaultname,size);

  error:= (size=-1);

  IF  error
   THEN
    BEGIN
      writeln('The file ',asciifilename,'.EMG does not exist on the floppy disk');
      writeln(chr(007B));
    END
   ELSE
   rdata;
END;
```

```
{*********************************************************************
 *                                                                   *
 *                      CONVERTRECORDTOASCII                         *
 *                                                                   *
 *        This   long  boring  procedure  does  the  actual  work  of *
 *    converting the patient data into the final form.   This  routine *
 *    is one large write statement.                                  *
 *                                                                   *
 *********************************************************************}


PROCEDURE convertrecordtoascii;

CONST
  b5='     ';

VAR
  dataitem:integer;
  tmp:real;

 BEGIN  {convertrecordtoascii}
   writeln(asciifile,'*                                                            *');
   writeln(asciifile,'*                                                            *');
   writeln(asciifile,'*               E M G   P A T I E N T   D A T A              *');
   writeln(asciifile,'*                                                            *');
   writeln(asciifile,'*                                                            *');
   writeln(asciifile,'*                        ',asciifilename,'.EMG                          *');
   writeln(asciifile,'*                                                            *');
   writeln(asciifile);
   writeln(asciifile,'Name of patient:    ',name);
   writeln(asciifile,patienttype);
   writeln(asciifile,ident);
   writeln(asciifile);
   writeln(asciifile);
   writeln(asciifile,'         EMGNUMBER         AGE           SEX             DATE');
   write(asciifile,'          ',emgnumber,'     ',age:7,'           ',sex);
   writeln(asciifile,b5,b5,date);

   writeln(asciifile);
   writeln(asciifile);
   writeln(asciifile,'         LATENCY       STIMULUS1      STIMULUS2          DELAYII');
   writeln(asciifile,'         seconds         amps           amps             seconds');
```

```
writeln(asciifile);
write(asciifile,latency:15:6,lok,stim1:15:6,st1ok);
writeln(asciifile,stim2:15:6,st2ok,delay:16:6,dok);
writeln(asciifile);
writeln(asciifile);
writeln(asciifile,'      SENSITIVITY 1   SENSITIVITY 2   SENSITIVITY 3   SENSITIVITY 4');
writeln(asciifile,'     volts/division        v/d               v/d               v/d');
writeln(asciifile);
writeln(asciifile,s1:15:10,s1ok,s2:15:10,s2ok,s3:15:10,s3ok,s4:15:10,s4ok);
writeln(asciifile);
writeln(asciifile);
writeln(asciifile,'sweepspeed = ',sweepspeed:9:6,swok,' seconds/division');
write(asciifile,'the data is from channel number ',channelnumber:3);
writeln(asciifile,'       Digital expansion factor=',expansionfactor:3);

writeln(asciifile,wavetype);
writeln(asciifile);


FOR dataitem:=1 TO 1000 DO
 BEGIN
   tmp:=data[dataitem];
   IF abs(tmp)>32750.0
    THEN tmp:=32750.0*abs(tmp)/tmp;
   write(asciifile,round(tmp):7);
   IF dataitem MOD 10 = 0
    THEN
    writeln(asciifile);

 END;
writeln(asciifile);
IF abs(di)>10
 THEN
 writeln(asciifile,' The dispersion index is not defined')
 ELSE writeln(asciifile,'Dispersion index ',di:5:1);


 writeln(asciifile)
END;
/* convertrecordtoascii */
```

```
{**************************************************************************
 *                                                                        *
 *                              TOFLOPPY                                  *
 *                                                                        *
 *        Writes out the new data file to the floppy disk from the        *
 *        global variables.                                               *
 *                                                                        *
 **************************************************************************}

PROCEDURE tofloppy;

 BEGIN {flop}
   emgnumber[1]:='T';
   emgnumber[2]:='W';
   emgnumber[4]:='1';
   createasciifilename;
   reset(asciifile,asciifilename,defaultname,size);
   writeln;

   WHILE (size<>-1)
   DO
    BEGIN
      write(' EMGNUMBER ',emgnumber);
      writeln(' has already been used');
      incrementname;
      reset(asciifile,asciifilename,defaultname,size);
    END;

   size := 20;
   rewrite (asciifile,asciifilename,defaultname,size);
   WHILE size = -1 DO
    BEGIN
      writeln('The floppy disk is full or not present.   ',chr(007B));
      writeln('Place a new data disk in the drive, then press <RETURN>. ');
      readln;
      rewrite (asciifile,asciifilename,defaultname,size);
    END;

   convertrecordtoascii;
   close (asciifile);
   writeln;
   writeln(' So ',emgnumber,' will be used');
   writeln;
   writeln('DY:',asciifilename,'.EMG has been created');
 END;  {flop}
```

```
{***************************************************************************
 *                                                                         *
 *                              READWARP                                   *
 *                                                                         *
 *        Reads the file WARP.DAT that contains  the result of the         *
 *        last warping.                                                    *
 *                                                                         *
 ***************************************************************************}

PROCEDURE readwarp(VAR error:boolean;VAR emgnumber:c10;VAR message:c80;
                   VAR nsamples,start,every:integer; VAR warp:warptype);

VAR
  warpfile:text;
  size,i,line:integer;
  found:boolean;
  skip9:c9;
  skip10:c10;
  skip33:c33;
  buffer:c80;

 BEGIN
   reset(warpfile,warpfilename,defaultname,size);
   error:=size<0;
   IF NOT error
    THEN
     BEGIN
        REPEAT
         read(warpfile,skip33);
         found:= (skip33[3]='A') AND (skip33[4]='L');
         IF not found
          THEN readln(warpfile)
        UNTIL found;

       error:=eof(warpfile);

       IF error
        THEN writeln('RECORD NOT FOUND',chr(7))
        ELSE
         BEGIN
           readln(warpfile,nsamples,skip9,start,skip9,every);
           readln(warpfile,emgnumber,skip10,buffer);
           message:=
           'WARPED TO                                                            ';
           FOR i:= 12 TO 80 DO message[i]:=buffer[i-11];
```

```
            writeln(emgnumber,skip10,buffer);
            FOR i:=1 TO nsamples DO
             BEGIN
               read(warpfile,warp[i]);
               IF warp[i]>nsamples
                THEN warp[i]:=0;
             END;
            FOR i:=nsamples+1 TO maxnsamples DO warp[i]:=0;
            close(warpfile);
          END;
         {record found}
     END;  {file ok}
 END;  {readwarp}

{*****************************************************************************
 *                                                                           *
 *                                   WEFT                                    *
 *                                                                           *
 *         This  procedure is  the core of the whole  program.   It          *
 *         linearly interpolates  the warping  function  to produce          *
 *         1000 values and then applies it to the EMG waveform.              *
 *                                                                           *
 *****************************************************************************}

PROCEDURE weft(VAR error:boolean;nsamples,start,every:integer;
               VAR warp:warptype;VAR data:i1000);


VAR
  data2,warp2:i1000;
  base,count,stop,lag,i,j:integer;


 BEGIN
   i:=1;
   WHILE  (i<=maxnsamples) AND (warp[i]<1) DO   i:=succ(i);
   lag:=i-1;

   WHILE (i<maxnsamples) AND (warp[i]>0) AND (warp[i]<=maxnsamples)
   DO i:=succ(i);

   stop:=i-1;

   FOR i:= 1 TO 1000 DO warp2[i]:=-1;
```

```
  count:=start+lag*every+1;

  FOR i:= lag+1 TO stop-1 DO
   BEGIN
     {linear interpolation of warp function}
     base:=(warp[i]-1)*every+start;

     FOR j:=1 TO every DO
      BEGIN
        warp2[count]:=base+(warp[i+1]-warp[i])*(j-1);
        count:=succ(count);
      END;{j}
   END;{i}
  warp2[count]:=warp[stop]*every+start;
  FOR i:=1 TO 1000 DO
   BEGIN
     IF (warp2[i]<1) OR (warp2[i]>1000)
      THEN data2[i]:=0
      ELSE data2[i]:=data[warp2[i]];
   END;
  data:=data2;
END;  {warp}
```

```
{*************************************************************************
 *                                                                       *
 *                              MAIN BLOCK                               *
 *                                                                       *
 *      This program reads the warpfile  using READWARP and then         *
 *      the emgdata file using READFLOPPY.  The warping function         *
 *      is  applied  by  WEFT  and  the result  written  out  by         *
 *      TOFLOPPY.   The resulting file can be graphed or plotted         *
 *      by the program EMG.                                              *
 *                                                                       *
 *************************************************************************}


BEGIN {main}
  writeln('         EXECUTION   BEGINS');

  readwarp(error,emgnumber,message,nsamples,start,every,warp);
  IF NOT error
   THEN
    BEGIN
      writeln('Warpfile read ok',nsamples,start,every);
      readfloppy(error,data);
      writeln('Floppy read ok');

      patienttype:=message;
      IF NOT error
       THEN
        BEGIN
          weft(error,nsamples,start,every,warp,data);
          IF NOT error
           THEN tofloppy;
        END;
    END;
END.
```

APPENDIX 5) DESIGN OF RESEARCH FACILITIES

Introduction

New research facilities for the Vancouver General Hospital's Neuromuscular Diseases Unit were designed, implemented, and evaluated.

The Neuromuscular Diseases Unit (EMG Laboratory) is a diagnostic laboratory for the the testing of patients, the training of neurology residents, and research in Evoked Potentials, Nerve Action Potentials, and Electromyography.

Research Functions Required for the Neuromuscular Diseases Unit

-Store waveforms from the two DISA Digital Systems with relevant patient data.

-Graph the stored waves on a CRT, and produce high-quality hardcopy.

-Expand sections of a stored wave.

-Subtract and average waves.

-Index, numerically and alphabetically, the stored waves.

-Calculate very involved formulas quickly.

-Prevent accidental loss of information.

-Word processing.

-Print large program and data files.

-Efficient programming manpower utilization.

-Accommodate possible future requirements.

Assessment of Requirements

Computer and Other Hardware Requirements

-Minimum 16 bit word length.

-Floating point arithimetic hardware.

-Six Megabytes of on-line system storage.

-Five Megabytes per month of archival storage.

-Fast parallel I/O and multiline serial I/O.

-Console terminals

-Graphic CRT and Plotter.

-Two printers
- Quick printer.
- Letter quality printer.

-flexibility to expand or adapt.

The computer system was to be used for research. It was not to form an integral part of the patient-care provided by the Neuromuscular Diseases Unit. This meant that the primary emphasis in selection of the system was to be on the services offered, rather than on reliability through redundancy and expensive "instant" service contracts. The system was to be maintained by the Unit's engineer. A week's down-time would delay research, but would not paralyze the whole Neuromuscular Diseases Unit.

Computer Software Requirements

-A general purpose operating system.

-Structured high level language compilers
- . Enabling maintainable code to be efficiently written.

-General purpose software
- . To allow the nontechnical laboratory personnel to work with waveforms.

-A reliable method of backing up files.

-Device drivers and interfaces.

-Word processing.

-Software tools
- . To support the writing of special purpose code.

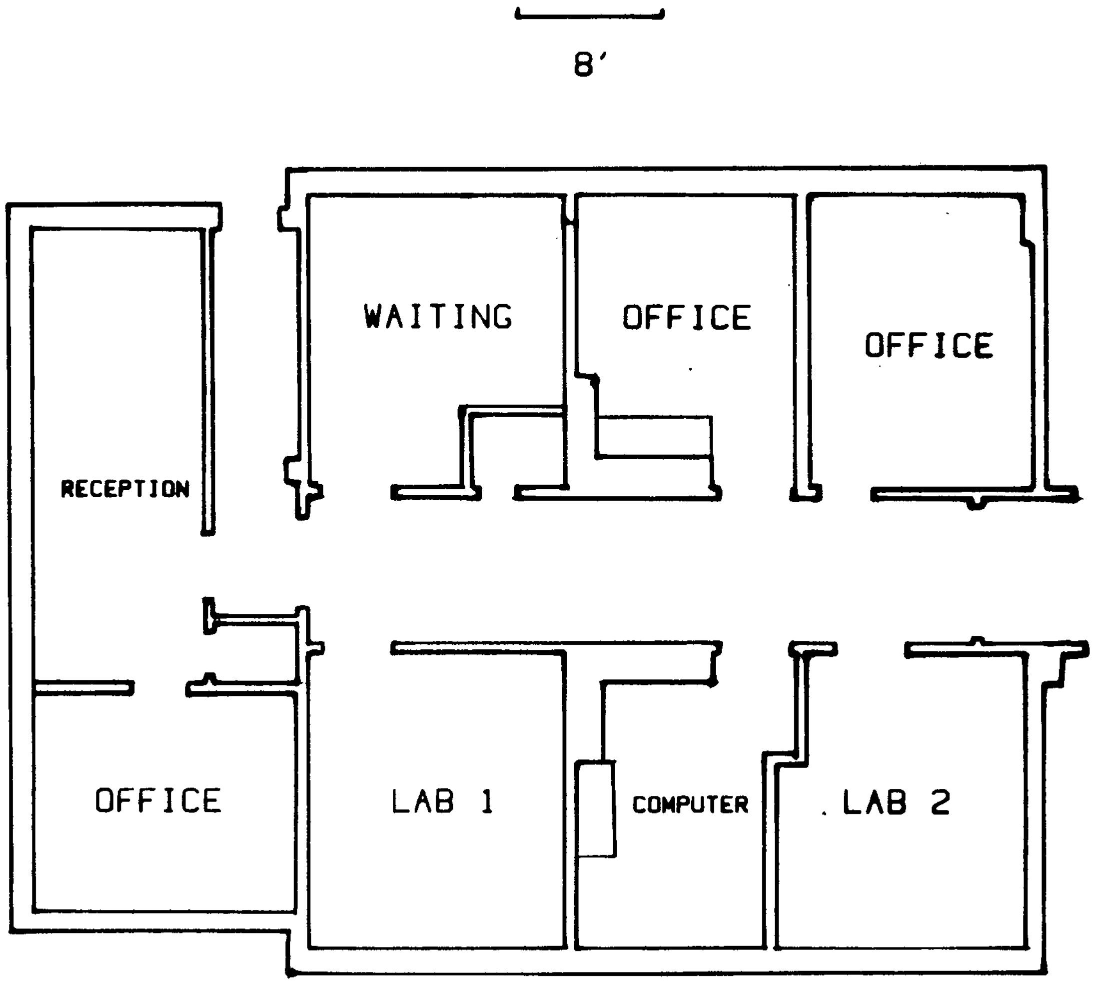
-Documentation standards
- . For efficient maintenance of code written.

Building Services

-Individual temperature controls for each room.

-The labs needed an electrically and acoustically quiet enviroment.
- . Acoustic noise comes from interior echoes or is transmitted from outside the labs.

-Fast parallel I/O between the computer and each lab.


8'
WAITING
OFFICE
OFFICE
RECEPTION
OFFICE
LAB 1
COMPUTER
LAB 2

-Access to terminals, printers and plotters in each room.

-Flexible communications wiring system
  . Specific access needs could change every six months.

-External communications would be needed in the future.

Design

Computer and Other Hardware Selected

-LSI-11/23 CPU, with memory management and floating point hardware. The PDP-11 family was chosen, despite its 16 bit address limitation, because all of the lab's existing software was written for a PDP-11/20, and because of the abundance of optional boards available for the Q-bus.

-MSV11LK board provided 256 kilobytes of parity memory.

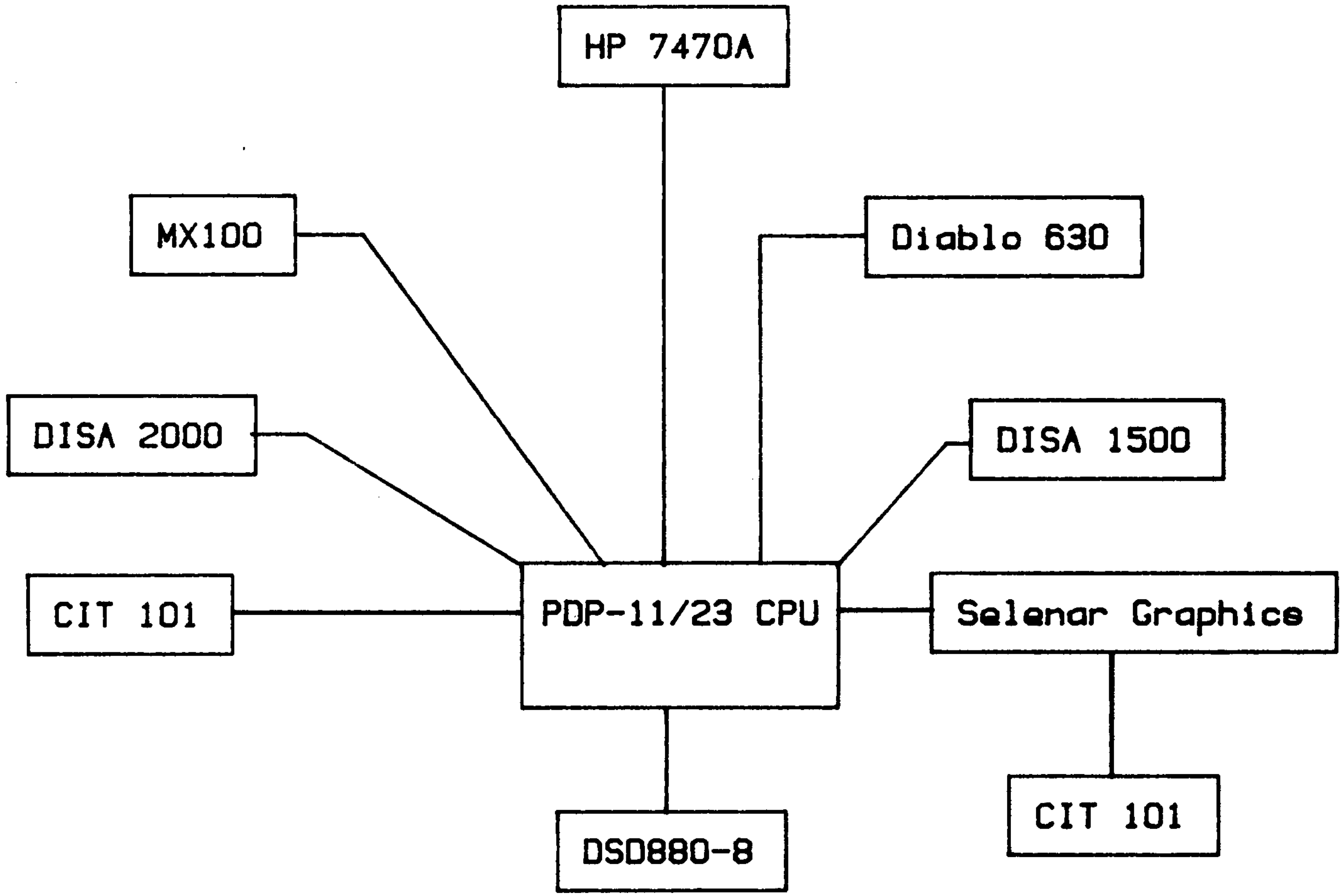
-Data Systems Design 880-8 containing a 7.8 Megabyte Winchester disk and a double-density double-sided 8" floppy disk drive. The DSD880-8 was chosen because of its price advantage over an RL02-RX02 combination.

-DRV11 16 bit parallel I/O board linked the computer system to the DISA 1500 EMG unit.

-IBV11 IEEE 488 bus controller to connect the DISA Neuromatic 2000 EMG system to the computer system.

-Two DLV11-Js provided eight serial RS232 lines.

-Two CIT101 terminals, one with a Selenar 201 graphics card.


HP 7470A
MX100
Diablo 630
DISA 2000
DISA 1500
CIT 101
PDP-11/23 CPU
Selenar Graphics
DSD880-8
CIT 101

-Diablo 630 daisy-wheel printer.

-Epson MX100 dot-matrix printer.

-HP 7470 two pen plotter.

Software Implemented

The software used in the Neuromuscular Diseases Unit consisted of three basic groups:

System Software

-RT11FB operating system with multiple terminal and system job support.

-Fortran, Pascal and C compilers

-A backup utility was written
- To incrementally duplicate the contents of the Winchester disk and of the data floppy disks.
- Done weekly and augmented by biannual total filesave

-Pascal to Hewlett-Packard plotter interface routines
- Written to use the minimum of memory while generating and queueing the HP-GL instructions for the plotter.

-DISA Interpack library
- Communicated with the DISA 1500.
- Modified to produce smaller overlays of the assembler code.

-Console control utility

- Written to move the control of the console terminal between four different users.

-QUEUE

- Run as a system job to allow files to be queued to the printer.

User Software

The user software that was designed to be fail-safe and user-friendly. The user software consisted of three major systems and some miscellaneous programs.

-Control files were written to support the QTEXT word processing program. QTEXT allowed the user to create, edit and print letter-quality documents.

-The waveform manipulation system contained seven programs. The programs EMG and OVER allowed the user to index and store, plot, and manipulate patient waveforms from the DISA 1500. Duplicate copies of the waves and associated patient data were archived separately. The waveforms could be subtracted, averaged, or expanded, and displayed on the graphics screen or plotted on paper. The programs FILTER and FILT2 digitally filtered the waveforms, while WARP and WEFT provided a dynamic time warping capability. These programs are documented in the appendices.

-The Entrap system composed of ENTRAP, ENREAD, and ENPLOT was written as a graphics based aid for the diagnosis of entrapment neuropathies.

-The miscellaneous user software that was written included DRAFT; a drafting program for the HP plotter, and DRAW; a program that produced very quick hardcopy graphs of the waveforms from the DISA 1500. RANDOM was a very small, very fast (<40 microseconds) interrupt driven program that produced randomly timed triggering pulses for the DISA 1500 Stimulator.

Research Software

The research programs were for the use of the engineer rather than laboratory personnel, and therefore user-friendly features were sacrificed for flexibility. Also included in this category were the test programs that were written to test the operation and reliability of the hardware and to assist in its repair. The research programs evolved through many versions. When a research program proved useful to the work being done in the laboratory, it was rewritten as a user program and fully documented.

Building Services Designed

Communications Wiring

-Star-topology serial RS232 communication

- Centered at a large wall-mounted patch panel in the computer room.
- Twelve individually shielded twisted-pairs ran in conduit from the patch panel to three D25 sockets mounted on a wall panel in every room.
- Allowed fast and flexible wiring changes.

-Parallel communication

- Forty shielded twisted-pairs to each laboratory
- Terminated at in-wall 2' telephone boxes.

-Telephone connections

- In each lab and office by desks.
- In computer room to be used with a modem.

-Extra pair of wires to each peripheral

- To allow a central security alarm system to be built in the future.

Grounding

-One common room ground for each lab

- For patient safety.

-Isolated ground for the computer

- To prevent ground current problems.

-Star-connections of all ground wires and shielding

- To prevent ground loops.

Lighting

-The two labs and the engineer's office were provided with two separately switched lighting circuits.

- Standard fluorescent light fixtures were used to produce bright and uniform lighting.
- Dimmer-controled incandescent track lights were installed to allow soft and nonuniform lighting.
- Low level incandescent lighting improved the visibility of cathode ray tube screens.

2.6) Evaluation and Discussion

Computer and Other Equipment

The computer system fulfills its function adequately.

Had all of the money for hardware been available sooner, a 30 Megabyte Winchester should have been purchased instead of the 7.8 Megabyte one. For an extra two thousand dollars four times the disk space would have been available. This extra disk space will soon be needed.

Two further items should be purchased.

- The DSD 880-30 30 Megabyte Winchester disk and streaming 1/4" tape drive combination would greatly expand the storage capacity and simplify backup procedures. Having a second disk drive would improve the system reliabliity by allowing one disk to be removed for repair without paralysing the entire system.

- A second graphics terminal should be aquired to be used by the research engineer. This would allow the engineer to work more efficiently.

Software

The QTEXT word-processing system is unreliable. It needs to be fixed or replaced.

The single user operating RT11 system causes problems as often several users desire concurrent access to the computer.

There is 256 kilobytes of memory in the computer, but RT11 allows easy access to only one fifth of it.

The operating system should be changed to UNIX. This would allow multiple users to share the resources and would ease the memory management bottleneck. Minor modifications will have to be made to all of the programs to reflect the change in file systems.

Interfaces should be written so that the user programs can be used with the Neuromatic 2000.

It would be useful to have the Michegan Communications Protocol implemented to connect the EMG system to UBC NET.

Building Services

The extra ten feet of in-wall wire proved too much for the new DISA parallel Interpack. A hole was punched through the wall between the computer room and Lab #1 and the wires run directly. This resulted in a loss of long term flexibility and reliability, but solved the short term problem.

-The serial lines have worked well.

-There has been no grounding problems.

-The lighting options greatly improved the comfort of the people operating the Laboratory's equipment.