DETECTION OF FAST PHASE OF NYSTAGMUS

USING DIGITAL FILTERING

by

MOHAMMAD-ALI MASSOUMNIA

B.S., University of California, Irvine
December 1981

SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
MAY 1983

© Massachusetts Institute of Technology

Signature of Author... Department of Aeronautics and Astronautics

Certified by...

Dr. Charles M. Oman
Thesis Supervisor

Accepted by.................................................................

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

Professor Harold Y. Wachman
Chairman, Departmental Graduate Committee

MAY 25 1983
Archives
Detection of fast phase of nystagmus using digital filtering

by

Mohammad-Ali Massoumnia

Submitted to the Department of Aeronautics and Astronautics
on May 6, 1983 in partial fulfillment of the
requirements of the degree of Master of Science in
Aeronautics and Astronautics

ABSTRACT

An algorithm for the detection of fast eye movements is developed. The algorithm detects the fast movements on the basis of velocity and pseudo acceleration of the eye. The detection strategy builds on the approach taken by Michaels [1]. The velocity and pseudo acceleration are calculated by using digital finite impulse response filters. The procedures for designing the filters are also discussed in detail.

Once the events are detected, they are identified and an estimate of slow phase velocity is obtained. Also available as output are one point per beat slow phase velocity, cumulative slow phase position, event indicator flag, and pseudo acceleration of the eye.

A complete analysis of the performance of the program is also given. The machine-detected fast movements are compared with the ones detected and identified by a human operator. On the basis of these comparisons the performance of algorithm is evaluated.

Thesis Supervisor: Dr. Charles M. Oman
Title: Senior Research Engineer, Department of Aeronautics and Astronautics
ACKNOWLEDGMENTS

The author like to thank his thesis supervisor for his help and guidance throughout the research effort. Dr. Oman made helpful criticisms as the work progressed, and his help in producing this thesis is appreciated.

I also like to thank all the members of Man Vehicle Laboratory, specially my friends W. Cook, A. Arrott, and P. Wetzel for helping me in recording some of the eye movement data used in this work. Helpful comments of Dr. J. Tole and Dr. A. Weiss are also appreciated.

Last but not least, I thank my family whose encouragement and support made this thesis possible.

This research was supported by the NASA (Washington) contract NAS W-3651 and NASA (Houston) contract NAS 9-15343.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>7</td>
</tr>
<tr>
<td>Chapter I  Introduction</td>
<td>8</td>
</tr>
<tr>
<td>Chapter II  Background</td>
<td></td>
</tr>
<tr>
<td>2.1 Nystagmus characteristics</td>
<td>10</td>
</tr>
<tr>
<td>2.2 An overview of nystagmus processing techniques</td>
<td>13</td>
</tr>
<tr>
<td>Chapter III  Detection of fast eye movements</td>
<td></td>
</tr>
<tr>
<td>3.1 Frequency content of nystagmus and its velocity</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Detection and identification criteria</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Design of pseudo acceleration filter</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Design of velocity filter</td>
<td>36</td>
</tr>
<tr>
<td>3.5 Setting the thresholds</td>
<td>40</td>
</tr>
<tr>
<td>3.6 Organization of fast phase detector program</td>
<td>42</td>
</tr>
<tr>
<td>Chapter IV  Results</td>
<td></td>
</tr>
<tr>
<td>4.1 Outputs available from the current program</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Some sample outputs of the program</td>
<td>49</td>
</tr>
<tr>
<td>4.3 Performance analysis of the algorithm</td>
<td>49</td>
</tr>
<tr>
<td>Chapter V  Conclusion</td>
<td>65</td>
</tr>
<tr>
<td>Appendix I</td>
<td></td>
</tr>
<tr>
<td>Suggestions for an alternative approach</td>
<td>67</td>
</tr>
<tr>
<td>Appendix II</td>
<td></td>
</tr>
<tr>
<td>Description and program listing of fast phase detector and associated subprograms</td>
<td>69</td>
</tr>
<tr>
<td>References</td>
<td>118</td>
</tr>
<tr>
<td>Fig-2.1.1</td>
<td>Different types of nystagmus</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Fig-3.1.1</td>
<td>Discrete Fourier transform of nystagmus</td>
</tr>
<tr>
<td>Fig-3.1.2</td>
<td>Discrete Fourier transform of velocity of nystagmus</td>
</tr>
<tr>
<td>Fig-3.1.3</td>
<td>A model for raw eye velocity during PRN</td>
</tr>
<tr>
<td>Fig-3.1.4</td>
<td>Plot of $S_a(t)$ and $S_b(t)$</td>
</tr>
<tr>
<td>Fig-3.1.5</td>
<td>A model for fast phase velocity and its Fourier transform</td>
</tr>
<tr>
<td>Fig-3.1.6</td>
<td>A model for fast phase velocity and its Fourier transform</td>
</tr>
<tr>
<td>Fig-3.1.7</td>
<td>Overlap of frequency content of fast phase and slow phase</td>
</tr>
<tr>
<td>Fig-3.2.1</td>
<td>Velocity and acceleration of eye during post-rotational nystagmus</td>
</tr>
<tr>
<td>Fig-3.2.2</td>
<td>A slowed down fast phase</td>
</tr>
<tr>
<td>Fig-3.3.1</td>
<td>Magnitude of H for a three point Stirling double differentiator</td>
</tr>
<tr>
<td>Fig-3.3.2</td>
<td>Magnitude of H for a five point Stirling double differentiator</td>
</tr>
<tr>
<td>Fig-3.3.3</td>
<td>Magnitude of H for a seven point Stirling double differentiator</td>
</tr>
<tr>
<td>Fig-3.3.4</td>
<td>Magnitude of z transform of pseudo acceleration filter for 60 Hz. sampling</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Fig-3.3.5</td>
<td>Magnitude of z transform of pseudo acceleration filter for 100 Hz. sampling</td>
</tr>
<tr>
<td>Fig-3.3.6</td>
<td>Magnitude of z transform of an all integer USR pseudo acceleration filter</td>
</tr>
<tr>
<td>Fig-3.3.7</td>
<td>Magnitude of z transform of an optimized pseudo acceleration filter</td>
</tr>
<tr>
<td>Fig-3.4.1</td>
<td>Magnitude of z transform of velocity filter</td>
</tr>
<tr>
<td>Fig-3.4.2</td>
<td>Magnitude of z transform of velocity filter</td>
</tr>
<tr>
<td>Fig-4.2.1</td>
<td>Output of fast phase detector program for a sinusoidal nystagmus</td>
</tr>
<tr>
<td>Fig-4.2.2</td>
<td>Output of fast phase detector program for a post rotational nystagmus</td>
</tr>
<tr>
<td>Fig-4.2.3</td>
<td>Output of fast phase detector program for an optokinetic nystagmus</td>
</tr>
<tr>
<td>Fig-I.1.1</td>
<td>Orthonormal detection</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table-4.3.1  Type of records used for evaluation  62
Table-4.3.2  Detection and identification probabilities  63
Table-4.3.3  Probability of detecting and identifying a fast phase  63
Chapter I
Introduction

An accurate analysis of eye movement data is critical in studies of vestibular and oculomotor system function. A typical application is the analysis of vestibular nystagmus induced by angular or linear acceleration or caloric stimulation (see section 2.1). With these tests the velocity of eye during the slow phase is the major parameter of interest and provides the basis for study of input/output relationships of the vestibular system. In addition to slow phase velocity other parameters of nystagmus like fast phase frequency, duration of response, and cumulative slow phase position can also be correlated with the function of the vestibular apparatus (see section 2.1) [13].

In order to obtain such information from the raw eye position one should have the ability to accurately detect the saccadic jumps and fast phase of nystagmus. For example, to find the cumulative slow phase position one needs to remove the fast phase movements and piece together the slow phase segments. To find the slow phase velocity one wants to remove the spikes associated with differentiating the fast movements from the raw eye velocity.

Currently there are some algorithms available which attempt to detect the fast phases and obtain the necessary information from the raw eye position. Although these programs may be adjusted to yield good performance on a particular record, performance may be much poorer on a new record. This suggests that a certain level of user knowledge of both eye movements and algorithm is required for successful operation.
In this thesis an algorithm is described that works well without user interaction, and can handle a wide variety of records without assuming any a priori knowledge of the response to the stimulus. The fundamental approach of my algorithm is detection of fast eye movements based on velocity and acceleration characteristics of the eye. Although saccadic eye movements made during changes in direction of gaze may involve different mechanisms than those of the fast phase of nystagmus, they appear sufficiently similar that detection methods for one type of eye movement may be equally suitable for the other. However, after the detection, the algorithm identifies the event as either fast phase, saccade, or other fast movements. The algorithm is developed such that near on-line processing can also be achieved.

In section 2.1 and 3.1 the characteristics and frequency content of nystagmus is discussed which will be used in the later sections for the design of the acceleration and velocity filters. In section 3.2 detection criteria are established. In section 3.3 a procedure for the design of a bandlimited double differentiator is given. Output of this filter will be used to locate the starting and the ending point of a fast movement. In chapter 4, performance of the program is evaluated by comparing the machine identified fast phases and saccades with the ones detected and identified by a human operator.
Chapter II

Background

2.1 Nystagmus characteristics

Nystagmus is an alternating saw-tooth like displacement of the eyes. The velocity in one direction is greater than the velocity in the other direction (see Fig-2.1.1). The component of movement with higher velocity are commonly referred to as "fast phases" if they occur as part of the nystagmic pattern. The lower velocity movements in the opposite direction are called "slow phases."

The designated directions are anatomically oriented: e.g. in right beating nystagmus, the cornea of the eye moves toward the right side of the head during the fast phase, in down beating nystagmus, the cornea moves toward the lower eye lid.

Nystagmus may be evoked using different procedures. Depending on the type of stimulus used, nystagmus can be, for example, classified as either vestibular pre- or post- rotational nystagmus (PRN), optokinetic nystagmus (OKN), or caloric nystagmus (CAN) yet other types of nystagmus have been defined. Nystagmus is considered usually to be pathologic when present spontaneously in the head erect position or when it occurs after the head has been placed in any of several standard positions (positional nystagmus).

Vestibular nystagmus (see Fig-2.1.1.a) is primarily caused by stimulation of the semicircular canals during the rotation of the head with respect to inertial space. A head rotation about a vertical axis leads to deformation of the cupulae of the horizontal semicircular
Fig-2.1.1 Different types of nystagmus
canals, which induces image-stabilizing slow phase eye movements in a compensatory direction. As head motion continues, the eyes jump back (eyes do not exactly jump back to the center, but the eye position usually deviates in the direction of the fast phase, see Fig-2.1.1.a) rapidly to pick up another position and repeat the saw-tooth pattern. The maximum frequency of fast phase is approximately 5 Hz., and its amplitude varies from 1 to 10 degrees [3].

Optokinetic nystagmus (see Fig-2.1.1.b) also known as train nystagmus is a characteristic saw-tooth pattern of eye motion evoked by movement of a visual field. Optokinetic nystagmus consists of a slow phase in which the eye fixates on a portion of the moving field and follows it with pursuit motion and a fast phase or return saccadic jump in which the eye fixates on a new position of the field. The minimum time between fast phases is approximately .2 sec. resulting in a maximum frequency of 5 Hz., although the nystagmus frequency may be considerably less for slow field motions. The amplitude of optokinetic nystagmus is variable, generally from 1 to 10 degrees [3].

Caloric nystagmus (see Fig-2.1.1.c) is induced upon introduction of water above or below body temperature into the external auditory canal thus producing convection stimuli to the semicircular canals. The duration and strength of nystagmus for standard temperatures constitute a set of clinical measurements useful in the diagnosis of vestibular disease [4].

The most common techniques of recording nystagmus are electrooculography (EOG), nystagmography (ENG), and photoelectric nystagmography (PENG) [3]. For EOG and ENG, recording electrodes are affixed to the skin at both sides of and above and below the eyeball. The presence of
an electric potential between the cornea and the retina functions as an
electric dipole, permitting any eyeball rotation other than roll to be
easily recorded electrically. However, EOG uses DC recording, but ENG
uses AC recording. The PENG measures these eye movements by the change
in the light reflected by the scleral limbus from a fixed light source to
two fixed photoelectric cells. EOG recording is used for most of the eye
movements shown in this work.

In the next section a review of the existing nystagmus processing
algorithms is given.

2.2 An overview of nystagmus processing techniques

There are three classes of methods used in nystagmus analysis,
manual, semi-automatic, and automated. Manual techniques, which are
usually used by E.N.T clinics, involve detecting the fast phases and
estimating the velocity from chart strip recording of eye movement. A
simple analog differentiator followed by a peak detector and a counter
can be used to semi-automate the analysis of nystagmus. Since the fast
phase velocity is typically greater than the slow phase velocity,
appropriate selection of threshold on the peak detector allows fast
phases occurring in one direction to be counted [4]. This process can be
expanded by integrating the slow phase velocity to obtain the slow phase
position [5]. Honrubia [6] digitized eye movement data and performed
computer analysis to yield parameters of nystagmus which he correlated to
the vestibular stimulus. He too used an analog differentiator to obtain
the velocity of the raw eye position in order to distinguish slow phase
segments.
In the same year (1971) Tole and Young [7] developed a hybrid computer program called MITNYS; this system took the further step of estimating slow phase velocity during a fast phase based on the velocity prior to fast phase. Here again fast phases were distinguished by their relative velocities. MITNYS-II [2] used a digital differentiator and incorporated an involved scheme for noise suppression. Eye movements were sampled and digitized and on the basis of velocity and direction, each sample point was classified as either slow or fast phase as soon as they were obtained. The algorithm was adaptive in that velocity thresholds for fast phase determination were dependent on the current slow phase velocity. Using the latter program Tole reported a correlation coefficient of .932 when comparing his results to hand analysis of peak slow phase velocity for caloric records. The correlation coefficient was calculated by plotting his estimate of peak slow phase velocity versus the program's estimate for several different runs.

Herberts [8] described a system with which clinical tests are recorded on analog tape and then digitally converted and stored for later analysis. Digital analysis includes a difference calculation and appropriate threshold for fast phase detection. Outputs of the algorithm include slow phase velocity, cumulative position, saccade frequency, and duration of response. The velocity of slow phase is determined by the difference between start and end position of slow phase divided by its duration; thus, inflection points in slow phase can not be detected.

Gentiles [9] described a system similar to Herberts' but with several improvements. A difference operation followed by threshold detection was used for saccade determination, then a second difference operation was performed to yield an acceleration estimate; the maximum
magnitudes of this estimate were used to indicate start and finish of the fast phase. Although this technique is inherently sensitive to noise, most problems were avoided by lowpassing the input with a fourth order Butterworth filter followed by 50 Hz. sampling. The algorithm includes several ad-hoc adaptive features. In periods of strong response when slow and fast phase velocities are nearly equal in magnitude, direction of slow phase was determined by the average of slow phase velocity. When the fast phase frequency was large, the velocity threshold for fast phase detection was increased for greater error immunity. Likewise, if the frequency was too small, the threshold was decreased to adjust for a possible low gain problem. Gentiles also compared his program outputs to the hand analysis of several nystagmus waveforms. He reported a correlation coefficient of .984 between the program and the average of results obtained from eight vestibular test laboratories in manually estimating the peak slow phase velocity for caloric records. When individually comparing his results with those of the laboratories, the correlation was lower, but still above .9.

Using a new approach, Tole and Michaels [1,10] treated a fast phase as an event and used what they claim to be matched filtering for detecting it. The incoming raw eye position was sampled and kept in a one second long temporary buffer then it was convolved with a 9 point finite impulse response (FIR) filter, and the result was compared with a threshold. If the output of filter was larger than a threshold, it was called a fast phase, and the algorithm would start to search the temporary buffer for the end of fast phase. The algorithm was on-line with one second of delay. Using this scheme Michaels reported a correlation coefficient of .94 between his results and an experienced
physician estimate of peak slow phase velocity. As it is pointed out by Michaels the estimate of peak slow phase velocity is not a good measure of performance because peak point is usually associated with high velocity fast phases and it is quite simple to obtain a good estimate of peak slow phase velocity; however, one should look at the probability of detection or probability of miss of a fast phase for evaluating the algorithm's performance. Tole [11] later enhanced the performance of algorithm by using an adaptive threshold and changing the conditions that the acceleration should satisfy in order to be classified as a fast phase.

Barnes [12] developed a computer analysis algorithm which uses the pattern recognition capabilities of the operator to distinguish between fast and slow phases of nystagmus. The basis of the procedure is that the operator can set up threshold limits around an expected slow phase velocity waveform through the use of cursor controls. Points lying outside the threshold limits are recognised as fast phase eye movements and are discarded. Then least squares curve fitting procedure is used to establish the relationship between the oculomotor response and the stimulus waveform. The procedure inherently requires an assumption concerning oculomotor function response which is sometimes unknown and thus the method is not applicable to many different classes of eye movements.

Wall and Black [13] described an ad-hoc program which processes large volumes of highly variable data off-line. First alternating relative maximum and minimum points of nystagmus are connected using straight line segments so that further tests can be applied to detect the fast phase. The slopes of the two line segments connecting each maximum
to its neighboring minima are calculated. The segment having the larger slope magnitude is selected as a possible fast phase. The candidate line segment is chosen to represent a fast phase if both preset slope and displacement magnitude thresholds are exceeded. After the fast phase is detected then the slow phase velocity and other useful information are calculated.

As we can see most of the existing algorithms use an ad-hoc approach to the problem of detecting the fast phase. In this thesis I will use Tole and Michaels' [1,10] idea to develop a very reliable and still simple algorithm for processing a wide variety of eye movements; however, my approach will be a formal one based on digital signal processing techniques.
Chapter III

Detection of fast eye movements

3.1 Frequency content of nystagmus and its velocity

For processing of eye signal and its velocity we need some understanding of their frequency content. As it is clear from Fig-2.1.1, nystagmus and especially its slow phase are low frequency content signals. To have some quantitative measure the magnitude of discrete Fourier transform of two different nystagmus pattern was found which are shown in Fig-3.1.1. (These are 1024 point DFT with a 3 point Hanning smoothing window.) More than 95% of displacement spectrum is below 10 Hz.; however velocity spectrum may have higher frequency content (e.g. Fig-3.1.2). Therefore, for the purpose of finding the slow phase velocity and suppressing the noise one is mostly interested in the first 10 Hz. of nystagmus. This fact is used later on to find the unit sample response (USR) of digital filters used for detection.

To get a better understanding of frequency content of eye velocity the following approach is used. The eye velocity during a post rotational nystagmus can be approximated as in Fig-3.1.3. Signal S(t) can be thought of as the sum of the following two signals namely S_a(t) and S_b(t) where S_a(t) is the slow phase velocity (SPV) which we are interested in and S_b(t) is the fast phase velocity (FPV). Note that each of the spikes in S_b(t) can be approximated as sin(at)/t neglecting the side lobes (Fig-3.1.5.b). (We can even approximate each spike as a triangular signal shown in Fig-3.1.6.b and still reach the same conclusion.) The width of the velocity spikes correspond to the duration
Fig-3.1.1 Discrete Fourier transform of nystagmus

Fig-3.1.2 Discrete Fourier transform of velocity of nystagmus
Fig-3.1.3 A model for raw eye velocity during PRN

slow phase velocity $S_a(t)$

fast phase velocity $S_b(t)$

$S(t) = S_a(t) + S_b(t)$

Fig-3.1.4 Plot of $S_a(t)$ and $S_b(t)$
Fig-3.1.5 A model for fast phase velocity and its Fourier transform

Fig-3.1.6 A model for fast phase velocity and its Fourier transform
of fast phases which are usually somewhere between 50 msec. to 150 msec. which correspond to a frequency range of 20 to 7 Hz.

$S_a(t)$ as we know is the sum of two exponentials with time constants $\tau_1$ and $\tau_2$; therefore, $S(t)$ can be approximated as in equation 3.1.1.

$$S(t) = A_1 e^{-t/\tau_1} - A_1 e^{-t/\tau_2} + \sum_{n=0}^{N} \frac{B_n}{(t-\tau_n)} \sin(a(t-\tau_n))$$

for $t \geq 0$ \hspace{1cm} 3.1.1

given $\int_0^\infty S(t)dt = 0$ because the eye position at steady state is zero (subject is looking forward). The Fourier transform of $S(t)$ is

$$S(\omega) = \int_0^\infty S(t) e^{-j\omega t} dt$$

$S(t) = 0$ for $t < 0$ \hspace{1cm} 3.1.2

$$S(\omega) = \frac{A_1}{1/\tau_1 + j\omega} - \frac{A_1}{1/\tau_2 + j\omega} + \sum_{n=0}^{N} B_n \frac{u_{-1}(\omega+a) - u_{-1}(\omega-a)}{\pi} e^{-j\omega n}$$ \hspace{1cm} 3.1.3

$$S(\omega) = \frac{A_1/\tau_1}{1/\tau_1^2 + \omega^2} - \frac{A_1/\tau_2}{1/\tau_2^2 + \omega^2} + \pi (u_{-1}(\omega+a) - u_{-1}(\omega-a)) \sum_{n=0}^{N} B_n \cos\omega n$$

$$S_1'(\omega) \hspace{1cm} S_1(\omega)$$

$$-j \left\{ \frac{A_1}{1/\tau_1^2 + \omega^2} - \frac{A_1}{1/\tau_2^2 + \omega^2} + \pi (u_{-1}(\omega+a) - u_{-1}(\omega-a)) \sum_{n=0}^{N} B_n \sin\omega n \right\}$$ \hspace{1cm} 3.1.4

$$S_2'(\omega) \hspace{1cm} S_2(\omega)$$

$$|S(\omega)|^2 = \frac{A_1^2(1/\tau_2^2 - 1/\tau_1^2)}{(1/\tau_1^2 + \omega^2)(1/\tau_2^2 + \omega^2)} + \pi (u_{-1}(\omega+a) - u_{-1}(\omega-a)) x$$

$$(\pi S_1^2(\omega) + \pi S_2^2(\omega) + 2S_1'(\omega)S_1(\omega) + 2S_2'(\omega)S_2(\omega)).$$ \hspace{1cm} 3.1.5
As we can see in Fig-3.1.7, spectrum of the slow phase velocity

\[
\frac{A_1^2(1/\tau_2-1/\tau_1)^2}{(1/\tau_1^2+\omega^2)(1/\tau_2^2+\omega^2)}
\]

is completely overlapped by the spectrum of fast phase (I assumed \(a=27\pi\times10\) \(\tau_1=1\) sec. & \(\tau_2=10\) sec.). Therefore, a distinction between the SPV and FPV in the frequency domain is not possible, and a time domain processing scheme seems more appropriate.

3.2 Detection and identification criteria

Accepting a constraint of time domain processing, a good starting point is to look at the velocity and acceleration of the eye, and detect the events on the basis of these two signals satisfying certain conditions. It will be shown that the same strategy can be used to detect a saccade, a fast phase, or any other fast eye movements. After detection, they can be identified using another test procedure.

The first and second derivatives of eye position during a post rotational nystagmus are shown in Fig-3.2.1. Each fast phase is associated with a velocity peak and two acceleration peaks with opposite signs. Note that in these figures, the curves shown are not the actual acceleration and velocity of the eye, but rather lowpassed versions of them; the reason for doing so and the procedure for designing the filters are explained in details in section 3.3 & 3.4.

Therefore, the beginning of a fast phase, point A, is arbitrarily chosen as the point where the magnitude of acceleration is larger than a starting threshold, \(T_s\). The reason that velocity is not compared with a
$u_1(\omega+a) - u_1(\omega-a) \quad \frac{A_1(1/\tau_2 - 1/\tau_1)}{\sqrt{(1/\tau_2^2 + \omega^2)(1/\tau_1^2 + \omega^2)}}$

Fig-3.1.7 Overlap of frequency content of fast phase and slow phase
Fig-3.2.1 Velocity and acceleration of eye during post-rotational nystagmus
threshold is because the mean value of velocity is not zero, but the mean of acceleration is. However, to reduce false detection we need to simultanously look at the output of both velocity and acceleration filters. A careful look at Fig-3.2.1 shows that velocity and acceleration have the same sign during the onset. This criteria can be used to ensure that point C (Fig-3.2.1) is not chosen as the beginning of a fast phase. Another consideration is that a starting point occurs only when the magnitude of acceleration is increasing. This fact can be used to eliminate points B & D as choices for the starting point of a fast phase.

The ending point E can be chosen as the point where the magnitude of acceleration goes below a certain threshold $T_e$, and the sign of velocity and acceleration during the end of fast phase are different from their respective signs during the onset. This procedure guarantees that even the fast phases shown in Fig-3.2.2 can be completely detected. Also additional minor tests are implemented to enhance the performance of the program.

One important point to mention is that the current algorithm is implemented to increase the probability of event detection as much as possible and at the same time achieve an acceptably low probability of false detection. As it is well known increasing the probability of detection also increases the probability of false alarm, and there is a trade off between these conflicting quantities.

After the event is detected, it will be classified as either a saccade from near zero velocity, a fast phase, or other type of fast movement. A saccade is called when the velocity of slow phase preceding the event is less than 2 deg./sec.. A fast phase is called when the fast movement preceding the current event has the same direction as the
Fig-3.2.2 A slowed down fast phase
current event, and the event is identified as other if it is neither a saccade nor a fast phase.

3.3 Design of pseudo acceleration filter

In the last section we saw that two opposite peaks of the acceleration signal were used for the detection of event. Because we only need these two opposite peaks it is a good idea to lowpass filter the output of noise sensitive double differentiator to suppress the additional spectral peaks which are associated with wideband differentiation, the EMG noise from facial muscles, and 60 Hz. power line noise. Therefore, a decision was made to first design an optimal double differentiator, and then convolve its USR with a lowpass filter (LPF) with an appropriate corner frequency. A relatively similar approach is taken by Wolfe [14] to derive the unit sample response of complex filters using convolution of simple filters; however, this work was done independently and in some respects, such as the order of the filter, is superior to his.

Now, we are left with the task of choosing type and order of the double differentiator. Remembering that we want to actually bandlimit the output of the filter a double differentiator based on the Stirling interpolation formula seems most appropriate [15]. Note that Stirling differentiators are very accurate at low frequencies but they have limited bandwidth.

For finding the order of the filter I will drive the unit sample response corresponding to several different orders of Stirling double differentiators and then by comparison I will choose the one appropriate to our application. Based on Stirling interpolation formula [15]:
\[ X(nT+pT) = (1 + \frac{p^2}{2!} \delta^2 + \frac{p^2(p^2-1)}{4!} \delta^4 + ...) X(nT) \]

\[ + \frac{p}{2} (\delta X(nT-\frac{T}{2}) + \delta X(nT+\frac{T}{2})) \]

\[ + \frac{p(p^2-1)}{2 \times 3!} (\delta^3 X(nT-\frac{T}{2}) + \delta^3 X(nT+\frac{T}{2})) + ... \quad (3.3.1) \]

where \(0<p<1\) and

\[ \delta X(nT+\frac{T}{2}) = X(nT+T) - X(nT) \quad (3.3.2) \]

We know by chain rule

\[ \frac{dX(t)}{dt} \bigg|_{t=nT+pT} = \frac{dX(nT+pT)}{dp} \times \frac{dp}{dt} = \frac{1}{T} \frac{dX(nT+pT)}{dp} \quad (3.3.3) \]

\[ \frac{d^2X(t)}{dt^2} \bigg|_{t=nT+pT} = \frac{d}{dt} \left( \frac{dX(t)}{dt} \right) \bigg|_{t=nT+pT} = \frac{1}{T^2} \frac{d^2X(nT+pT)}{dp^2} \quad (3.3.4) \]

\[ \frac{d^2X(t)}{dt^2} \bigg|_{t=nT} = \frac{1}{T^2} \frac{d^2X(nT)}{dp^2} = \left(2 \frac{\delta^2}{2!} - 2 \frac{\delta^4}{4!} + ... \right) X(nT) \quad (3.3.5) \]

and using 3.3.2

\[ \delta^2 X(nT) = \delta X(nT+\frac{T}{2}) - \delta X(nT-\frac{T}{2}) \]

\[ = X(nT+T) - 2X(nT) + X(nT-T). \quad (3.3.6) \]

Therefore, by keeping only the first term in 3.3.5 the difference equation for a 3 point double differentiator becomes

\[ \frac{d^2X(t)}{dt^2} = \frac{X(nT+T) - 2X(nT) + X(nT-T)}{T^2} \quad (3.3.7) \]
A causal transfer function for this filter is

\[ H(z) = \frac{1}{T^2} \frac{z^{-1} - 2 + z}{z} \]  

and its magnitude response is given in Fig-3.3.1. Using the same procedure the transfer function for a 5 point filter is

\[ H(z) = \frac{-z^{-2} + 16z^{-1} - 30 + 16z - z^2}{12T^2 z^2} \]  

(See Fig-3.3.2 for frequency response), and for a 7 point filter the transfer function is

\[ H(z) = \frac{2z^{-3} - 27z^{-2} + 270z^{-1} - 490 + 270z^1 - 27z^2 + 2z^3}{180T^2 z^3} \]  

(See Fig-3.3.3 for frequency response.)

By looking at the magnitude of \( H(e^{j\omega}) \) in Figures 3.3.1, 2&3 and assuming that our sampling frequency will be somewhere between 40 to 200 Hz., even a 3 point double differentiator seems quite adequate because we are interested only on the first 10 Hz. frequency response of acceleration filter. Remember that more than 95% of nystagmus power spectrum is below 10 Hz. (see section 3.1).

After finding the USR of the 3 point double differentiator \( h_{3p}(nT) \), we need to convolve it with the USR of an appropriate lowpass filter. Keeping in mind the processing time and possible on-line application we have a constraint on the order of overall filter which we can use. On
Fig-3.3.1 Magnitude of $H$ for a three point Stirling double differentiator

Fig-3.3.2 Magnitude of $H$ for a five point Stirling double differentiator
Fig-3.3.3 Magnitude of $H$ for a seven point Stirling double differentiator.
the basis of the past work done by Tole and Michaels [1,10] I chose a 9 point over all filter; however, this selection is arbitrary. Therefore, setting the order of overall filter to 9 results in a 7 point lowpass filter.

For designing a 7 point lowpass filter we proceed as follow [15]

\[ H(e^{j\omega}) = 1 \quad \text{for} \quad |\omega| \leq \omega_c \]
\[ = 0 \quad \text{for} \quad \omega_c < |\omega| \leq \omega_s / 2 \]  \hspace{1cm} 3.3.11

where \( \omega_c \) is the cut-off angular frequency and \( \omega_s \) is the sampling angular frequency. In general

\[ H(e^{j\omega T}) = \sum_{n=0}^{\infty} h(nT)e^{-j\omega nT} \]  \hspace{1cm} 3.3.12

\[ h(nT) = \frac{1}{\omega_s} \int_{-\omega_s/2}^{\omega_s/2} H(e^{j\omega T})e^{-j\omega nT}d\omega. \]  \hspace{1cm} 3.3.13

For a lowpass filter

\[ h(nT) = \frac{1}{\omega_s} \int_{-\omega_c}^{\omega_c} e^{j\omega nT}d\omega = \frac{1}{n\pi} \sin\omega_c nT. \]  \hspace{1cm} 3.3.14

To truncate the USR of filter I use a Kaiser window given by

\[ W_k(nT) = \frac{I_0(\beta)}{I_0(\alpha)} \quad \text{for} \quad |n| \leq (N-1)/2 \]
\[ = 0 \quad \text{otherwise} \]  \hspace{1cm} 3.3.15

where \( \alpha \) is an independent parameter. \( \beta \) is given by

\[ \beta = \alpha \sqrt{1 - \left(\frac{2n}{N-1}\right)^2} \]  \hspace{1cm} 3.3.16
where $N$ is the order of the filter ($N=7$ for our case.) $I_0(x)$ is the zeroth order Bessel function

$$I_0(x) = 1 + \sum_{k=1}^{\infty} \left( \frac{1}{k!} \left( \frac{x}{2} \right)^k \right)^2$$

and the truncated USR is

$$h_w(nT) = w_k(nT)h(nT) \quad |n| \leq \frac{N-1}{2}$$

$$h_w(nT) = \frac{1}{\pi} \sum_{k=1}^{\infty} w_k(nT) \sin(nw_cT) \quad |n| \leq \frac{N-1}{2}.$$  \hspace{1cm} 3.3.17

Finally, $h(nT)$, the unit sample response of pseudo acceleration filter (PAF) is found by convolving $h_w(nT)$ and $h_{3p}(nT)$

$$h(nT) = h_w(nT) * h_{3p}(nT).$$  \hspace{1cm} 3.3.18

Note that $h_w(0) = h(0) = 2f_cT_s$, where $T_s$ is the sampling period and $f_c$ is the cutoff frequency. A sample run of the program for two different sampling frequencies results in the transfer functions

$$H(z) = \frac{\cdot 0145z^{-4} + \cdot 0705z^{-3} + \cdot 0498z^{-2} - \cdot 0695z^{-1} - \cdot 1306}{\cdot 0695z^1 + \cdot 0498z^2 + \cdot 0705z^3 + \cdot 0145z^4} T^2 \frac{z^4}{z^4}$$

for 60 Hz. sampling frequency (see Fig-3.3.4 for frequency response) and

$$H(z) = \frac{\cdot 0281z^{-4} + \cdot 0577z^{-3} + \cdot 0221z^{-2} - \cdot 0576z^{-1} - \cdot 1008}{\cdot 0576z^1 + \cdot 0221z^2 + \cdot 0577z^3 + \cdot 0281z^4} T^2 \frac{z^4}{z^4}$$

for 100 Hz. sampling frequency (see Fig-3.3.5 for frequency response).
Fig-3.3.4 Magnitude of z transform of pseudo acceleration filter for 60 Hz. sampling

Fig-3.3.5 Magnitude of z transform of pseudo acceleration filter for 100 Hz. sampling
For $f_s = 60$ Hz, a very good all integer unit sample response filter is given by (see Fig-3.3.6 for frequency response)

$$H(z) = \frac{z^{-4} + 2z^{-3} + z^{-2} - 2z^{-1} + 2z^{-1} + 2z^{-2} + z^{-4}}{32 z^2} \cdot 3.3.23$$

Note that the all integer USR filter is very useful for real time application where speed is important. This filter has a smaller high frequency lobes and a better inband accuracy than the one used by Michaels [1].

If we are interested in the exact value of the acceleration of the signal we can further improve our filter by multiplying the unit sample response by a constant $A$, where $A$ is found by minimizing $J$.

$$J = \int_0^{2\pi f_c} (H(e^{j\omega T}) - A\omega^2)^2 d\omega \quad f_c = 10 \text{ Hz.} \quad 3.3.22$$

Therefore, set $\delta J/\delta A = 0$

$$A = \frac{\int_0^{2\pi f_c} \omega^2 H(e^{j\omega T}) d\omega}{\int_0^{2\pi f_c} \omega^4 d\omega} \quad f_c = 10 \text{ Hz.} \quad 3.3.23$$

If we carry out the calculation for the case $1/T = f_s = 100$ Hz, we get $A = 1.2$. A plot of $1.2H(e^{j\omega})$ is shown in Fig-3.3.7. It is clear that this is a better fit to $\omega^2$ in the range 0 to 10 Hz.

### 3.4 Design of velocity filter

To find the derivative of the eye position I first used an optimal 9 point differentiator. A 9 point filter was chosen because it is more
Fig-3.3.6 Magnitude of z transform of an all integer USR pseudo acceleration filter

Fig-3.3.7 Magnitude of z transform of an optimized pseudo acceleration filter
efficient to implement two filters of the same size (Note that we can use the same counter for performing the convolution, and the other filter is the pseudo acceleration filter driven in 3.3). However, as was expected I found that an optimal velocity filter has a noisy output, and it causes considerable ringing in the output. The ringing is because of the butterworth antialiasing filter used at the sampling time. Remember that the cutoff of antialiasing filter is $f_c \leq f_s / 2$ where $f_s$ is the sampling frequency, and the frequency of damped oscillation, $f_d$, created by this filter is $f_d = 0.707f_c$; therefore, an oscillation with frequency $f_d$ appears when we differentiate the eye signal. To eliminate this ringing I could use either an RC lowpass filter or to bandlimit the velocity filter to a frequency lower than $f_d$. I chose the latter solution; therefore, the velocity filter does not need to be a variable filter; only $f_{cv}$, bandwidth of velocity filter, should be at a frequency lower than $f_d$. Therefore, I used the REMEZ exchange algorithm [16] for finding the USR of velocity filter. I chose a stopband of .2 to .5 and a differentiating band of 0 to .07 (these values are normalized frequency $f_s = 1$ Hz.). The resulting 9 point FIR filter is

$$H(z) = \frac{-0.0332z^{-4} - 0.0715z^{-3} - 0.0678z^{-2} - 0.0522z^{-1} + 0.0522z^{-1} + 0.0678z^{-2} + 0.0715z^{-3} + 0.0332z^{-4}}{T z^{4}}$$

(see Fig-3.4.1 for magnitude response). Another filter with different specifications (see Fig-3.4.2) is
Fig-3.4.1 Magnitude of z transform of velocity filter

Fig-3.4.2 Magnitude of z transform of velocity filter
\[ H(z) = \frac{-0.0777z^{-4} - 0.0714z^{-3} - 0.1078z^{-2} - 0.0870z^{-1}}{+0.0870z^1 + 0.1078z^2 + 0.0714z^3 + 0.0077z^4}. \]

3.4.2

3.5 Setting the thresholds

As it was explained in section 3.2 we need to choose two thresholds for the purpose of detecting the beginning and the end of a fast movement. The easiest threshold to choose is just a constant value. As a matter of fact constant threshold works quite well on low noise data where velocity of slow phase is neither very low nor very high. Unfortunately, the general eye movement recording is not noise free. Thus low frequency noise may cause false event detection as may very high velocity slow phases. Also events with low velocity slow phase preceding them can be easily missed. These difficulties point out that the methods by which thresholds for event detector are set are at least as important as the actual detection method chosen.

In the current program the thresholds are calculated through the following scheme. First the running sum of output of pseudo acceleration filter (PAF) is calculated during the future N sample points using the formula (in current program \( N = \frac{f_s}{4} \)):

\[ Y(nT) = X(nT) + X((n-1)T) + \ldots + X((n+1-N)T). \]

However, this nonrecursive filter is same as the following recursive filter:
\[ Y(nT) = Y((n-1)T) + X(nT) - X((n-N)T). \]  \hspace{1cm} 3.5.2

Next the running sum of output of PAF squared is calculated using the recursive formula:

\[ Z(nT) = Z((n-1)T) + X^2(nT) - X^2((n-N)T). \]  \hspace{1cm} 3.5.3

Then the RMS of pseudo acceleration is calculated through

\[ \text{RMS} = \left( \frac{Z(nT)}{N} - \frac{Y^2(nT)}{N^2} \right)^{1/2}. \]  \hspace{1cm} 3.5.4

This RMS is a measure of activity in the acceleration signal. Therefore, when there are too many spikes in the acceleration signal the RMS value will increase. However, this RMS can be mixed with a constant value minimum acceleration threshold (MAT) to reduce the sensitivity of the adaptive threshold. Using this procedure the starting threshold, \( T_s \), is found by taking a linear combination of RMS and MAT (the weighting in

\[ T_s = \text{RMS} + 0.5 \times \text{MAT} \]  \hspace{1cm} 3.5.5

this equation is found by monitoring the performance of the program for several different values.) If \( T_s \) ever goes below MAT, it is set equal to MAT. This effectively eliminates the detection of very low acceleration event, and reduces the false detection probability.

The choice of MAT depends on the quality of the signal; however, if the signal to noise ratio is reasonably high, a value of 500. to 1200. deg./sec.\(^2\) can be used. I usually set MAT=1000. If the incoming sample signal is highly quantized MAT should be increased to yield a reasonable
performance.

The ending threshold, $T_e$, is found by multiplying $T_s$ by a constant which has a magnitude less than unity because the acceleration during the end of a fast phase or a saccade is usually smaller than the acceleration during the beginning.

$$T_e = .7T_s$$

3.6 Organization of fast phase detector program

The ideas in the past sections are implemented in a program which I call it fast phase detector. As it was mentioned in the introduction same strategy is used to detect all different types of fast eye movement; however, the events are later classified as either fast phases, saccades, or other type of fast eye movements by event indicator flag. The main tasks of the detection section which are also available in a subroutine, are discussed in the following paragraphs.

First task is computation of the USR of the 9 point PAF by calling the subprogram FILTER. Subprogram FILTER uses the design procedure outlined in section 3.3 to derive the USR of the PAF for a user specified sampling frequency. This subprogram also derives the USR of a 7 point LPF which is used to smooth out the slow phase velocity. Subprogram FILTER calls subprogram BESSEL for computing the zeroth order Bessel function anytime it is needed.

After deriving the USR of the filter and reading the USR of velocity filter from the data statement, the program initializes the necessary constants and variables. Then the data points are read from logical unit 4 (see appendix II) to a one second long ring buffer. As the data are
read, the output of the PAF and velocity filter are calculated and kept in separate ring buffers. Note that computing the output of filters requires a delay of $4 = (N-1)/2$ sample points. After the buffers are initially filled up, the program starts to look for an event.

If a fast phase (or saccade) is not detected the program branches to the slow phase processing section where the event indicator flag is set to zero, and the slow phase velocity and other outputs are written to their respective files (see appendix II). Note that in the existing program the slow phase velocity is further smoothed by convolving it with the 7 point LPF derived by subprogram FILTER.

If an event is detected by satisfying the conditions mentioned in section 3.2, the program starts to search forward for an specified maximum number of sample points (MFRS1) for the end of event. If an end point is not detected, the event will be assigned the maximum length. After the end point is assigned, the velocity during the event is computed by taking the average of velocity before and after the fast movement. Then the event is classified as either fast phase, saccade, or other types of fast movement by using the criteria outlined in section 3.2 and assigning an appropriate value to the event indicator flag (see section 4.1). Then the data processed indicator is advanced for the length of the event, and the necessary outputs are written to their corresponding logical units (see appendix II). Note that at this point the ring buffer may have a large gap inside it because all the points during a detected event are processed without reading any new data point; therefore, the program executes a loop to fill up the ring buffer. When the buffer is full again the program starts its detection routine as before (see flowchart for fast phase detector in next page). Note that
the algorithm is implemented with possible future on-line application in mind. For this reason a relatively complicated ring buffering is used.

Documentation for the fast phase detector and also for other routines used for user interaction, plotting, and processing of the data under different file structures are given in the program listing of each specific program in appendix II. There is also a user manual on how to use the library and associated programs in this appendix.

FLOWCHART FOR

FAST PHASE DETECTOR
Start

Define Modula & sign function

Function ISGN

Enter the ending & starting point, input file specification

Set up the I/O channels

Enter sampling frequency, minimum acceleration threshold and scale factor.

Find the USRs of acceleration filter, 7 point lowpass filter, and velocity filter

SUB FILTER SUB BESSEL

Read raw eye position

Check for the end of input file

End

Exit
Scale raw eye position, compute output of PAF. A velocity filter also computes the moving average & RMS of acceleration over past 1/4 sec.

Check if temporary buffer has enough data

Yes
Use moving RMS of Acc. to set up the thresholds.

Check if output of PAF exceeds the $T_s$

Yes

Check if acceleration and velocity have the same sign

Yes

Check if acceleration is increasing in magnitude during the beginning of event

No Event

No

No Event

Yes

Event
4.1 Outputs available from current program

The outputs available from the current program are: slow phase velocity, pseudo acceleration, event indicator flag, cumulative slow phase position, and one point per beat slow phase velocity. The first two have been already discussed in detail. The event indicator flag (see Fig-4.2.1.b) indicates the occurrence of a jump in the eye movement, and it is set equal to 100, or -100 depending on the direction of event, if the event is a saccadic jump from a velocity of less than 2 deg./sec. The indicator is set to 200 (-200) if the event is identified as a nystagmus. If the event is neither a fast phase nor a saccade, indicator flag is set to 300 (-300) indicating a different type of fast eye movement. If no event is detected, the flag is set to zero indicating a slow or stationary phase.

The cumulative slow phase position (see Fig-4.2.1.g) is the position that the eyeball would reach if it could rotate unrestrained in the socket, and is obtained by removing the fast phase segments from the nystagmus record and piecing together the slow phase segments.

The one point per beat slow phase velocity (see Fig-4.2.1.e) is a plot of time versus the average of slow phase velocity per slow phase beat. Therefore, it is not an equal interval sample signal but the interval between the points depends on the duration of slow and fast phases. Note that in current program if a slow phase is longer than half of a second the program finds the average for every half of a second and
the average for the remainder of the slow phase. The one point per beat slow phase velocity is sometimes more useful to work with because it is a much shorter file than the slow phase velocity file.

4.2 Some sample outputs of the program

In the following pages (Fig-4.2.1 to Fig-4.2.3) a series of records analysed by the fast phase detector program are shown. The resulting slow phase velocity, one point per beat slow phase velocity, pseudo acceleration, event indicator flag, cumulative slow phase position, and raw eye position for several different type of eye movements are shown. The figures themselves are self explanatory.

4.3 Performance analysis of the algorithm

As in every detection-identification problem, the best way to evaluate the performance of an algorithm is to compute the detection and false alarm probabilities of detecting an event, and also the probability of identifying each particular event. This is different from the ad-hoc approach taken by many others [eg: 9,7] which does not take a probabilistic approach to event detection and is based on evaluating the performance only for the estimate of the peak slow phase velocity. Although it is mentioned in [1] that the evaluation should be based on detecting every fast movement, Michaels fails to quantify his results as it is done here. In addition, previous workers have not tried to classify the events detected by their program.

However for this problem, it is expected that the performance probabilities are a function of slow phase velocity because in eye movement processing a better performance is expected when the velocity of
Fig-4.2.1 Output of fast phase detector program for a sinusoidal nystagmus
d) Raw velocity of the eye

e) One point per beat slow phase velocity

Fig-4.2.1 Continued
d) Raw velocity of the eye

f) Slow phase velocity

Fig-4.2.1 Continued
f) Slow phase velocity

g) Cumulative slow phase position

Fig-4.2.1 Continued
a) Raw eye position

b) Event indicator flag

c) Pseudo acceleration of the eye

Fig-4.2.2  Output of fast phase detector program for a post rotational nystagmus
d) Raw velocity of the eye

e) One point per beat slow phase velocity

Fig-4.2.2 Continued
d) Raw velocity of the eye

f) Slow phase velocity

Fig-4.2.2 Continued
Fig-4.2.3 Output of fast phase detector program for an optokinetic nystagmus
d) Raw velocity of the eye

e) One point per beat slow phase velocity

Fig-4.2.3 Continued
d) Raw velocity of the eye

f) Slow phase velocity

Fig-4.2.3 Continued
f) Slow phase velocity

g) Cumulative slow phase position

Fig-4.2.3 Continued
slow phase preceding an event is high, but not too high. To take into account this dependence, the performance probabilities for several different ranges of slow phase velocity were found.

Define the detection and false alarm probabilities as follows [17]:

\[ P_{D_1} = P\{ \text{detecting an event} \mid \text{an event has occurred} \& V_i < |SPV| < V_{i+1} \} \]

(\( P(A|B) \) means conditional probability of A given B )

and

\[ P_{FA_i} = P\{ \text{detecting an event} \mid \text{no event occurred} \& V_i < |SPV| < V_{i+1} \} \]

and the identification probabilities

\[ P_{I_i}^A = P\{ \text{identifying as A} \mid \text{it is A} \} \]
\[ P_{I_i}^B = P\{ \text{identifying as B} \mid \text{it is B} \& V_i < |SPV| < V_{i+1} \} \]
\[ P_{I_i}^C = P\{ \text{identifying as C} \mid \text{it is C} \& V_i < |SPV| < V_{i+1} \} \]

where event A represents a saccade from near zero velocity, event B represents a fast phase, and event C represents other types of fast movements. Note that probability of identifying an event as a saccade does not depend on the SPV because the SPV preceding a saccade is by definition always close to zero. The following values for \( V_i \) was chosen:

\[ V_0 = 0 \text{ deg./sec.} \quad V_1 = 10 \text{ deg./sec.} \quad V_2 = 20 \text{ deg./sec.} \quad V_3 = 30 \text{ deg./sec.} \quad V_4 = \infty \text{ deg./sec.} \]
Next the fast phase detector program was used to process a set of 10 different files containing a wide variety of eye movements (e.g. pre-rotational nystagmus (PRRN), post-rotational nystagmus without dumping (PRN) with dumping (PRNW)), continuos optokinetic nystagmus (OKNC), sinusoidal optokinetic nystagmus (OKN), caloric nystagmus (CAN), reading nystagmus (REN), and saccadic fixations (SAC)) recorded with different amplifiers and sampling rates because it is important that the algorithm works well with different type of eye movements, sampling frequencies, and different recording environment (see Table-4.3.1).

<table>
<thead>
<tr>
<th>Type</th>
<th>Sampling rate</th>
<th>Duration analysed</th>
<th>Recording technique</th>
<th>Total # of events in the duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>OKN</td>
<td>120 (Hz.)</td>
<td>51 (sec.)</td>
<td>A</td>
<td>167</td>
</tr>
<tr>
<td>PRN</td>
<td>100</td>
<td>21</td>
<td>B</td>
<td>71</td>
</tr>
<tr>
<td>PRNW</td>
<td>120</td>
<td>51</td>
<td>C</td>
<td>175</td>
</tr>
<tr>
<td>PRNW</td>
<td>120</td>
<td>32</td>
<td>C</td>
<td>73</td>
</tr>
<tr>
<td>REN</td>
<td>100</td>
<td>41</td>
<td>A</td>
<td>97</td>
</tr>
<tr>
<td>OKNC</td>
<td>100</td>
<td>33</td>
<td>A</td>
<td>45</td>
</tr>
<tr>
<td>OKNC</td>
<td>100</td>
<td>28</td>
<td>A</td>
<td>29</td>
</tr>
<tr>
<td>SAC</td>
<td>175</td>
<td>38</td>
<td>D</td>
<td>65</td>
</tr>
<tr>
<td>PRRN</td>
<td>120</td>
<td>51</td>
<td>C</td>
<td>146</td>
</tr>
<tr>
<td>CAN</td>
<td>120</td>
<td>38</td>
<td>C</td>
<td>85</td>
</tr>
</tbody>
</table>

* A=EOG recording with antialiasing LPF (Butterworth) cutoff at 37 Hz.
B=EOG recording with antialiasing LPF (Butterworth) cutoff at 10 Hz.
C=EOG recording with antialiasing LPF (RC) cutoff at 40 Hz.
D=PEENG recording

Table-4.3.1
The performance probabilities were calculated, and the results are shown in Table-4.3.2.

<table>
<thead>
<tr>
<th>i=0</th>
<th>( P_{Di} = 0.973 = 614/631 )</th>
<th>( P_{FAi} = 0.073 = 46/631 )</th>
<th>( P_{Ii}^A = 0.824 = 155/188 )</th>
<th>( P_{Ii}^B = 0.822 = 323/393 )</th>
<th>( P_{Ii}^C = 0.697 = 23/33 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td>( P_{Di} = 0.995 = 200/201 )</td>
<td>( P_{FAi} = 0.010 = 2/201 )</td>
<td>-</td>
<td>( P_{Ii}^B = 0.975 = 195/200 )</td>
<td>-</td>
</tr>
<tr>
<td>i=2</td>
<td>( P_{Di} = 1.00 = 95/95 )</td>
<td>( P_{FAi} = 0.000 )</td>
<td>-</td>
<td>( P_{Ii}^B = 1.00 = 95/95 )</td>
<td>-</td>
</tr>
<tr>
<td>i=3</td>
<td>( P_{Di} = 1.00 = 38/38 )</td>
<td>( P_{FAi} = 0.000 )</td>
<td>-</td>
<td>( P_{Ii}^B = 1.00 = 38/38 )</td>
<td>-</td>
</tr>
</tbody>
</table>

(A/B means: B is the number of events detected by the operator, A is the number of events detected by the program.)

Table-4.3.2

Therefore, probability of detecting a fast phase and identifying it correctly is given by \( P = P_{Di} \times P_{Ii}^B \). The calculated values of \( P \) are given in Table-4.3.2.

<table>
<thead>
<tr>
<th>i=0</th>
<th>i=1</th>
<th>i=2</th>
<th>i=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>0.800</td>
<td>0.970</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table-4.3.3

Similarly, probability of detecting a saccade from near zero velocity is found to be equal to 0.802.

As it was expected the probability of detection increases as the slow phase velocity increases; however, for very high velocities it was expected that the performance degrades, but the program did an excellent job even on events with very high SPV preceding them.
The interesting point to note is that the probability of false alarm decreases as the SPV preceding an event increases. This itself accounts for relatively low probability of identification for events with low SPV preceding them because the current identification criteria is based on the idea that no false event detection is occurring. Another point to note is that in pre-rotational or optokinetic nystagmus the first fast movement after a change of direction which is usually preceded by a low SPV is always identified incorrectly, and this itself biases the identification probability for low SPV range.

If we compare the detection probabilities with the results obtained by Michaels [1], it is clear that the program is performing a better job in detecting the events starting from both very low and high slow phase velocities because Michaels found a probability of .85 to .94 for detecting the fast movement and for this algorithm the probability of detection is higher than .973.

If we are interested only in the slow phase velocity, detecting and removing a fast movement is the only problem in hand; however, if the identification is important for a given application then the probability of identification is of major interest. Unfortunately others have not calculated a performance value for classifying fast eye movements; therefore, the results obtained here can not be compared with the numerical results in [1,7,9]. However, depending on the application a probability of .8 in detecting and correctly identifying a fast phase of nystagmus which is preceded by a low velocity slow phase seems quite acceptable.
Chapter V
Conclusion

One of the early design goals was to derive a relatively simple algorithm which is clearly understood and to avoid ad-hoc procedures as much as possible. The author thinks this goal is achieved because the detection scheme which uses digital signal processing techniques is relatively simple and accurate, and ad-hoc solutions are avoided. The current program is transportable (beside the input output unit assignment) to the other machines because it is written in a high level language. Also the algorithm is designed to be insensitive to changes in sampling rate for a range of 40 to 200 Hz.

Although the program works quite well in detecting a fast eye movement, it makes some mistakes in identifying an event; however, some of the mistakes like the first event after the change of direction in the fast movement is because of the way the current identification criteria is defined.

In the other hand, the performance probabilities evaluated in chapter IV can be improved by choosing a particular set of eye movement records, but this was avoided by analysing a relatively large variety of different types of eye movement records. In addition it was tried to explore the effect of slow phase velocity preceding an event on the detection and classification probabilities.

Although I think that the current algorithm can not be improved unless other detection strategies are used, the next step can be to
implement the current algorithm on-line because some of the tests done by the author indicate that on-line processing can be achieved even on a micro computer if integer arithmetic is used.
Appendix I

Suggestions for an alternative approach

As it is clear from section 4.3 the current program works quite well even on noisy data. However, another promising detection scheme is orthonormal filtering. In this detection scheme the incoming signal is convolved with a set of filters which have a USR orthogonal to each other. Therefore, if we have two such filters, by convolving the incoming signal by these two filters and plotting the output against each other on an orthogonal axis, we can develop an event detector. One way of doing such detection is to plot acceleration and jerk of the signal against each other (see Fig-I.1.1) then by monitoring the growth of the vector R we can detect a fast phase.

However, the jerk filter is quite sensitive to noise and it should be heavily low passed. The reason that velocity is not plotted against acceleration is because velocities before and after each fast phase are not the same. However, for saccade detection we can use the plot of velocity against acceleration.

The above algorithm was implemented, and it performed equivalently as the current program in detecting the saccadic jumps, but the performance was lower than the current algorithm for detecting fast phases starting from low slow phase velocities.
Fig. I.1.1 Orthanormal detection
Appendix II

Program listing of fast phase detector and associated subprograms

Several different versions of fast phase detector is available now. Currently there are two programs "M2MI86" and "M2DETE" which can be used to process the eye movement data and scan through the results interactively. There is also a batch version of fast phase detector "M2DEBA" which can be used to process a large collection of data without requiring an operator in attendance. In addition the subroutine "M2FFD1" can be used by a calling program to perform the detection and derivation of different outputs. The current programs are all written in FORTRAN language. A PDP-11/34 using an RT-11 version 4 operating system is used to implement all the programs. Graphic routines use the DEC VT-11 display unit. All the main programs are in a file with the same name as program's name and an extension FOR. All the subroutines are in file named M2DELI.SUB. The object code for these subroutines can be found in a library named M2LIBR.

Main Programs M2DETE and M2MI86

The only difference between the two programs M2MI86 and M2DETE is that the first one reads and writes the data word by word, but the latter one reads and writes 256 words at a time; thus, it is faster. Both programs first ask for the location of the starting and the ending point of the segment of the file to be processed (e.g. the starting point=1015 and the ending point=3012). If the starting point of the segment is not an integral multiple of 256 plus one, M2DETE lowers the user specified
starting point to the closest multiple of 256 plus one, and similarly the ending point is raised to the closest multiple of 256. Next the user is asked to enter the file specification.

Then both programs copy the desired segment of the input file to a scratch file which has the first point to be processed as its first point. In M2DETE this function is performed by calling the subroutine COPY (see page 75). Next the user is asked to specify the following parameters sequentially:

1) (I) sampling rate
2) (R) minimum acceleration threshold in deg./sec.²
3) (R) scale factor for input file in degrees per unit of A/D converter
   (I = Integer & R = Real).

Then both programs derive the USR of PAF and start the detection. After the detection, the user can scan through the results (SPV, output of PAF, event indicator flag, cumulative slow phase position) by entering the appropriate commands. Next the user can ask for the one point per beat slow phase velocity and its corresponding plot. At last the user is inquired about the files he wants to keep. Note that the output files are assigned as scratch files in these routines; therefore, if each is not copied to a permanent file at this stage it will be lost.

Main program M2DEBA

M2DEBA is basically the same program as M2DETE but all the input parameters are read from logical unit 5, which is assigned to a file named M2INPU.DAT. It also calls none of the graphic subroutines. There is an existing command procedure file M2D.COM which runs the batch version of fast phase detector. When you execute M2D by typing @M2D,
editor program is run first. This enables you to edit the file M2INPU.DAT used by M2DEBA to read the input parameters. The order in which the parameters should be written to M2INPU.DAT file are as follow:

1) (I) Ending point
2) (I) Starting point
3) (A) Input file specification (note: you need to enter a space before the file name, and the file name should use all 10 letters)
4) (I) Sampling rate
5) (R) Minimum acceleration threshold
6) (R) Scale factor
7) (I) 0=You do not want one point per beat slow phase velocity
     1=You want one point per beat slow phase velocity
8) (I) 0=Copy no more file
     1=Keep SPV
     2=Keep output of PAF
     3=Keep event indicator flag
     4=Keep cumulative slow phase position
     (e.g. if you want (1) & (2) enter 1<cr> 2<cr> 0<cr>)
9) (I) 0=Terminate the processing
     1=Continue to the next file

(I = Integer, R = Real, A = ASCII character)

Then you enter the parameters for the next file you want to process and repeat, and for the last file you enter (C) for step 9. After you finished editing M2INPU.DAT, exit by typing <EX$$>. Next M2D executes the M2DEBA program. The program terminates whenever it encounters a <0> at step 9. The output files will have the same name as the input file
with different extensions. The extension for slow phase velocity is SPV, for one point per beat slow phase velocity is SPL, for pseudo acceleration is ACC, for event indicator flag is SAC, and for cumulative slow phase position is CSP.

**SUBROUTINE M2FFD1**

To use the M2FFD1 subroutine the user first should transfer the sampling rate, scale factor, minimum acceleration threshold, and length of the segment of the file to be processed to the subroutine by using a common block named "DETECT".

```
COMMON DETECT/ISAMPL,GAIN,CNSTRM,ILENGT/
```

- **ISAMPL (I)** sampling rate
- **GAIN (R)** scale factor degree per unit of A/D converter
- **CNSTRM (R)** minimum acceleration threshold deg./sec\(^2\) (MAT)
- **ILENGT (I)** length of the segment of the file to be processed

Next the calling program should assign the logical unit 4 to the input file. This channel should be assigned as an unformatted binary file with 256 words per record. The first point to be processed should be the first point of the file. If this is not so, use the small program shown in M2DETE and COPY subprogram to rewrite the input file on a scratch file which has the first point at its beginning (see program listing of M2DETE on page 75). The outputs slow phase velocity, pseudo acceleration, fast phase indicator, and cumulative slow phase position are written respectively to units 11 to 14 by the subroutine. These unit should be assigned by the calling program as unformatted binary file with 256 words per record. These units can be assigned as scratch files and later copied to a permanent file via COPY subprogram (see program
listing of M2DETE on page 75).

SUBROUTINE M2FFD3

This subroutine computes the one point per beat slow phase velocity. The calling program should assign an output file to a specific logical unit, and the logical unit number should be transferred to the subroutine via its argument.

CALL M2FFD3(I)

I (I) output logical unit to be used

Note that the inputs are read from units 11 (event indicator flag) and unit 13 (SPV) in 256 words per record. The results are written to an unformatted binary file with 2 words per record. The first word is the one point per beat slow phase velocity, and the second word is the time tag. The last 2 words in the file are always -1 and -1 (end of file indicator).

Main program M2PSP1

This program can be used to plot the one point per beat slow phase velocity. The user only needs to specify the name of the input file (obviously the input file should be the output of either M2FFD3, M2MI86, M2DETE, or M2DEBA.) The program also plots the result on a logarithmic scale under user request.

Main program M2PLOT

This program plots the raw eye position and event indicator flag simultaneously on the same graph. It can be used for performance evaluation of the program. The user only needs to enter name of the eye
position file. The file name should use all 10 letters.

**Main program M2EVAL**

This program plots raw eye position, event indicator flag, and one point per beat slow phase velocity. This program was used in computing the performance probabilities listed in section 4.3. The user needs to enter the name of the eye position file, a bias value (if the A/D was producing only positive numbers), and the number of blocks (256 words) to be shown.

The description of other subroutines are given in their program listing in the following pages. The object code of all the subroutines are available in library M2LIBR; therefore, the user can link his program with this library whenever needed.
PROGRAM M2DETE
C THIS PROGRAM IMPLEMENTS THE USER I/O AND CALLING OF DIFFERENT
C SUBPROGRAMS TO DETECT FAST PHASE OF NYSTAGMUS AND DERIVE THE
C NECESSARY OUTPUTS
C BY MOHAMMAD-ALI MASSOUNMIA
C AUG-21-82
C
COMMON/DETECT/ ISAMPL,GAIN, CNSTRM, ILENGT, IL
C
C FORMAT STATEMENTS
C
0005 30 FORMAT(’,’ ENTER THE ENDING POINT FOR YOUR INPUT FILE: ’$’)
0006 40 FORMAT(’ ENTER THE STARTING POINT FOR YOUR INPUT FILE: ’$’)
0007 50 FORMAT(’,’ ENTER INPUT FILE SPECIFICATION’,/)
0008 60 FORMAT(’,’ ENTER THE SAMPLING RATE: ’$’)
0009 70 FORMAT(’ ENTER THE MIN. ACC. THRESHOLD(500. - 15000. D/S^2): ’$’)
0010 80 FORMAT(’,’ ENTER THE SCALE FACTOR DEG/UNIT: ’$’)
0011 90 FORMAT(’,’ 1) NO ADDITIONAL OUTPUT’,/,
*’ 2) AVE, SLW, PHASE VEL. (1 PNT. PER NEAT),/,
*’ ENTER YOUR CHOICE: ’$’)
0012 100 FORMAT(’,’ DO YOU WANT 1PNT/BEAT PLOT NO(0),YES(1=OS,2=LS)?’$’)
0013 110 FORMAT(’,’ 0) KEEP NO MORE FILE’,/,
*’ 1) KEEP SLW, PHASE VELOCITY’,/,
*’ 2) KEEP ACC’,/,
*’ 3) KEEP SAC, IND’,/,
*’ 4) KEEP CUMALITIVE SLOW PHASE POSITION’,/,
*’ ENTER YOUR CHOICE: ’$’)
0014 120 FORMAT(’,’ FILE IS SAVED; ENTER YOUR CHOICE: ’$’)
0015 130 FORMAT(’,’ THE STARTING POINT IS CHANGED TO ’,IS,
*’ AND THE ENDING POINT TO ’,IE)
0016 140 FORMAT(’,’ ENTER OUTPUT FILE SPECIFICATION’,/)
C
C I/O ASSIGNMENT
C
0017 150 TYPE 30
0018 READ(5,10) IL
C
0019 TYPE 40
0020 READ(5,10) ILEN
C
0021 IF(IL.LE.ILEN.OR.ILE.0.OR.ILEN.LE.0)GO TO 150
C NOW THE STARTING AND THE ENDING POINTS ARE CHANGED TO
C MULTIPLES OF 256
0023 IL1=IL/256
0024 IF(IL1*256.EQ.IL) GO TO 160
0026 IL1=IL+1
0027 160 IL2=IL1*256
0028 IL3=ILEN/256
0029 IL4=IL3*256+1
0030 IL3=IL3+1
0031 ILENGT=IL2-IL4+1
0032 IL5=IL1-IL3+1
C
0033 WRITE(6,130)IL4,IL2
0034 TYPE 50
0035 CALL ASSIGN(1,'LY02FP,EOG',-1,'RDO')
0036 DEFINE FILE 1 (IL1,256,U,NREC)
0037 CALL ASSIGN (4,'LY02FP,EOG',9,'SCR')
0038 DEFINE FILE 4 (IL5,256,U,NREC)
0039 CALL M2COPY(1,4,IL3,IL5)
0040 CLOSE(UNIT=1)
0041 DO 210 IUNIT=11,14
0042 IU=IUNIT
0043 CALL ASSIGN(IU,'LY02FP,EXT',IUNIT-5,'SCR')
0044 DEFINE FILE IU (IL5,256,U,NREC1)
0045 210 CONTINUE
0046 220 TYPE 60
0047 READ(5,10)ISAMPL
0048 IF(ISAMPL.LT.40.OR.ISAMPL.GT.200)GO TO 220
0050 230 TYPE 70
0051 READ(5,20)CNSTRM
0052 IF(CNSTRM.LT.500.OR.CNSTRM.GT.15000.)GO TO 230
0054 TYPE 80
0055 READ(5,20)GAIN
0056 CALL THE FAST PHASE DETECTOR SUBROUTINE
0057 CALL M2FDI1
0058 CALL M2FDI2(IL5)
0059 CLOSE(UNIT=4)
0060 240 TYPE 90
0061 READ(5,10)IDUM
0062 IF(IDUM.LT.1.OR.IDUM.GT.2)GO TO 240
0063 GOTO (270,250)IDUM
0064 250 TYPE 140
0065 CALL ASSIGN(4,'LY02FP,EOG',-1,'NEW')
0066 DEFINE FILE 4(ILNGT,2,U,NREC)
0067 CALL M2FDI3(4)
0068 260 TYPE 100
0069 READ(5,10)IDUM
0070 IF(IDUM.LT.0.OR.IDUM.GT.2)GO TO 260
0071 IF(IDUM.EQ.0)GO TO 270
0074 CALL P3(4,IDUM)
0075 CLOSE(UNIT=4)
0076 270 TYPE 110
0077 275 READ(5,10)IDUM
0078 IF(IDUM.EQ.0)GO TO 1000
FORTRAN IV V02.5-2 Sun 31-Apr-83 01:52:59 PAGE 003

0080  IF(IDUM,GT.4,OR,IDUM,LT.0) GO TO 270
0082  TYPE 140
0083  CALL ASSIGN(1,'LY02FP,EXT','-1,'NEW')
0084  DEFINE FILE 1(IL5,256,U,NREC)
0085  IU=10+IDUM
0086  CALL M2COPY(IU,1,1,IL5)
0087  CLOSE(UNIT=IU)
0088  CLOSE(UNIT=1)
0089  TYPE 120
0090  GO TO 275
0091  1000  STOP
0092  END

FORTRAN IV Storage Map for Program Unit M2DET

Local Variables, PSECT $DATA, Size = 000030 (12 words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDUM</td>
<td>I*2</td>
<td>000024</td>
<td>ILEN</td>
<td>I*2</td>
<td>000000</td>
<td>IL1</td>
<td>I*2</td>
<td>000002</td>
</tr>
<tr>
<td>IL2</td>
<td>I*2</td>
<td>000004</td>
<td>IL3</td>
<td>I*2</td>
<td>000006</td>
<td>IL4</td>
<td>I*2</td>
<td>000010</td>
</tr>
<tr>
<td>IL5</td>
<td>I*2</td>
<td>000012</td>
<td>IU</td>
<td>I*2</td>
<td>000020</td>
<td>IUNIT</td>
<td>I*2</td>
<td>000016</td>
</tr>
<tr>
<td>NREC</td>
<td>I*2</td>
<td>000014</td>
<td>Eav</td>
<td>NREC1</td>
<td>I*2</td>
<td>000022</td>
<td>Eav</td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /DETECT/, Size = 000016 (7 words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAMPL</td>
<td>I*2</td>
<td>000000</td>
<td>GAIN</td>
<td>R*4</td>
<td>000002</td>
<td>CNSTRM</td>
<td>R*4</td>
<td>000006</td>
</tr>
<tr>
<td>ILN</td>
<td>I*2</td>
<td>000012</td>
<td>IL</td>
<td>I*2</td>
<td>000014</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSIGN</td>
<td>R*4</td>
<td>M2COPY</td>
<td>I*2</td>
<td>M2FD1</td>
<td>I*2</td>
<td>M2FD2</td>
<td>I*2</td>
</tr>
<tr>
<td>P3</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
0001 PROGRAM M2DEBA
0002 C FAST PHASE DETECTOR USING BATCH PROCESSING
0003 C MOHAMMAD-ALI MASSONNIA
0004 C AUG-21-82
0005 COMMON/Detect, Isampl, Gain, Cnstrm, Ilen, IL
0006 LOGICAL*1 Name(10), Name1(15)
0007 DATA Name1/'S', 'P', 'V', 'A', 'C', 'C', 'S', 'A', 'C',
0008 * 'C', 'S', 'P', 'S', 'P', '1'/
0009 C
0010 FORMAT STATEMENTS
0011 10 FORMAT(I7)
0012 20 FORMAT(E12.5)
0013 30 FORMAT(//' ENTER THE ENDING POINT FOR YOUR INPUT FILE: ')'')
0014 40 FORMAT(//' ENTER THE STARTING POINT FOR YOUR INPUT FILE: ')'')
0015 50 FORMAT(//' ENTER INPUT FILE SPECIFICATION: ')'')
0016 60 FORMAT(//' ENTER THE SAMPLING RATE: ')'')
0017 70 FORMAT(//' ENTER THE MIN. ACC. THRESHOLD(500.-15000.D/S^2): ')'')
0018 80 FORMAT(//' ENTER THE SCALE FACTOR DEG/UNIT: ')'')
0019 90 FORMAT(//' 1)NO ADDITIONAL OUTPUT',/,
0020 * 2)AVE. SLW. PHASE VELO. (1 PNT. PER NEAT)',/,
0021 * 3) ENTER YOUR CHOICE: ')'')
0022 110 FORMAT(//' 0) KEEP NO MORE FILE',/,
0023 * 1) KEEP SLW. PHASE VELOCITY',/,
0024 * 2) KEEP ACC.',/,
0025 * 3) KEEP SAC. IND.',/,
0026 * 4) KEEP CUMULATIVE SLOW PHASE POSITION!',/,
0027 * ENTER YOUR CHOICE: ')'')
0028 120 FORMAT(//' FILE IS SAVED : ENTER YOUR CHOICE: ')'')
0029 130 FORMAT(//' THE STARTING POINT IS CHANGED TO ',I5,
0030 * AND THE ENDING POINT TO ',I5)
0031 140 FORMAT(//' ENTER OUTPUT FILE SPECIFICATION: ')'')
0032 145 FORMAT(X,11A1)
0033 147 FORMAT(//' ENTER <0> TO EXIT THE PROGRAM: ')'')
0034 C
0035 I/O ASSIGNMENT
0036 C
0037 OPEN(UNIT=5, NAME='M2INPU.DAT', TYPE='OLD')
0038 150 TYPE 30
0039 READ(5,10)IL
0040 160 WRITE(6,10)IL
0041 C
0042 TYPE 40
0043 170 READ(5,10)ILEN
0044 180 WRITE(6,10)ILEN
0045 C
0046 IF(IL.LE.ILEN.OR.IL.LE.0.OR.ILEN.LE.0) GO TO 150
0047 190 IL1=IL/256
0048 200 IF(IL1*256.EQ.IL) GO TO 160
0049 210 IL1=IL1+1
0050 220 IL2=IL1*256
0051 230 IL3=ILEN/256
0052 240 IL4=IL3*256+1
0053 250 IL3=IL3+1
FORTTRAN IV V02.5-2 Sat 30-Apr-83 09:48:02 PAGE 002

0037 ILENGT=IL2-IL4+1
0038 ILS=IL1-IL3+1

C
0039 WRITE(6,130)IL4,IL2
0040 TYPE 50
0041 READ(5,145)NAME
0042 WRITE(6,145)NAME
0043 CALL ASSIGN(1,NAME,10,'RDD')
0044 DEFINE FILE 1 (IL1,256,U,NREC)

C
0045 CALL ASSIGN (4,'LY02FP.EOG',9,'SCR')
0046 DEFINE FILE 4 (IL5,256,U,NREC)

C
0047 CALL M2COPY(1,4,IL3,IL5)

C
0048 CLOSE(UNIT=1)

C
0049 DO 210 IUNIT=11,14
0050 IU=IUNIT
0051 CALL ASSIGN(IU,'LY02FP.EXT',IUNIT-5,'SCR')
0052 DEFINE FILE IU (IL5,256,U,NREC1)
0053 210 CONTINUE

C
0054 220 TYPE 60
0055 READ(5,10)ISAMPL
0056 WRITE(6,10)ISAMPL
0057 IF(ISAMPL.LT.40.OR.ISAMPL.GT.200)GO TO 220

C
0059 230 TYPE 70
0060 READ(5,20)CNSTRM
0061 WRITE(6,20)CNSTRM
0062 IF(CNSTRM.LT.500.OR.CNSTRM.GT.15000.)GO TO 230

C
0064 TYPE 80
0065 READ(5,20)GAIN
0066 WRITE(6,20)GAIN

C CALL THE FAST PHASE DETECTOR SUBROUTINE
0067 CALL M2FFD1
0068 CLOSE(UNIT=4)

C
0069 240 TYPE 90
0070 READ(5,10)IDUM
0071 WRITE(6,10)IDUM
0072 IF(IDUM.LT.1.OR.IDUM.GT.2)GO TO 240
0074 GOTO (270,250)IDUM
0075 250 TYPE 140
0076 DO 260 ID=1,3
0077 ID1=12+ID
0078 260 NAME(7+ID)=NAME1(ID1)
0079 WRITE(6,145)NAME
0080 CALL ASSIGN(4,NAME,10,'NEW')
0081 DEFINE FILE 4(ILENGT,2,U,NREC)
0082 CALL M2FFD3(4)
C
0083       CLOSE(UNIT=4)
  C
0084      270       TYPE 110
0085      275       READ(5,10)IDUM
0086      280       WRITE(6,10)IDUM
0087      290       IF(IDUM.GE.0)GO TO 290
0088      290       IF(IDUM.GT.4.OR.IDUM.LT.0) GO TO 270
0091      290       TYPE 140
0092      290       DO 280 ID=1,3
0093      290       ID1=(IDUM-1)*3+ID
0094      280       NAME(7+ID)=NAME1(ID1)
0095      280       WRITE(6,145)NAME
0096      290       CALL ASSIGN(1,NAME,10,'NEW')
0097      290       DEFINE FILE 1(IL5,256,U,NREC)
0098      290       IU=10+IDUM
0099      290       CALL M2COPY(IU,1,1,IL5)
0100      290       CLOSE(UNIT=1)
0101      290       TYPE 120
0102      290       GO TO 275
0103      290       DO 300 ID=11,14
0104      300       IU=ID
0105      300       CLOSE(UNIT=IU)?
  ***
0106      300       TYPE 147
0107      300       READ(5,10)IDUM
0108      300       WRITE(6,10)IDUM
0109      300       IF(IDUM.NE.0)GO TO 150
0111      300       STOP
0112      300       END
FORTRAN IV  
Storage Map for Program Unit M2DEBA

Local Variables, .PSECT $DATA, Size = 000066 ( 27. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>I*2</td>
<td>000060</td>
<td>IDUM</td>
<td>I*2</td>
<td>000056</td>
<td>ID1</td>
<td>I*2</td>
<td>000062</td>
</tr>
<tr>
<td>ILEN</td>
<td>I*2</td>
<td>000032</td>
<td>IL1</td>
<td>I*2</td>
<td>000034</td>
<td>IL2</td>
<td>I*2</td>
<td>000036</td>
</tr>
<tr>
<td>IL3</td>
<td>I*2</td>
<td>000040</td>
<td>IL4</td>
<td>I*2</td>
<td>000042</td>
<td>IL5</td>
<td>I*2</td>
<td>000044</td>
</tr>
<tr>
<td>IU</td>
<td>I*2</td>
<td>000052</td>
<td>IUNIT</td>
<td>I*2</td>
<td>000050</td>
<td>NREC</td>
<td>I*2</td>
<td>000046</td>
</tr>
<tr>
<td>NREC1</td>
<td>I*2</td>
<td>000054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /DETECT/, Size = 000016 ( 7. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAMPL</td>
<td>I*2</td>
<td>000000</td>
<td>GAIN</td>
<td>R*4</td>
<td>000002</td>
<td>CNSTRM</td>
<td>R*4</td>
<td>000006</td>
</tr>
<tr>
<td>ILENGT</td>
<td>I*2</td>
<td>000012</td>
<td>IL</td>
<td>I*2</td>
<td>000014</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>L*1</td>
<td>$DATA</td>
<td>000000</td>
<td>000012</td>
<td>( 5.) (10)</td>
</tr>
<tr>
<td>NAME1</td>
<td>L*1</td>
<td>$DATA</td>
<td>000012</td>
<td>000017</td>
<td>( 8.) (15)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSIGN</td>
<td>R*4</td>
<td>M2COPY</td>
<td>I*2</td>
<td>M2FD1</td>
<td>I*2</td>
<td>M2FD3</td>
<td>I*2</td>
</tr>
</tbody>
</table>
TYPE M2MI86
FORTRAN IV V02.5-2 Mon 09-May-83 02:57:27

C FAST PHASE DETECTOR
C MOHAMMAD-ALI MAASSOMNIA
C AUG-21-82
C
0001 COMMON/DETECT/ ISAMPL
0002 DIMENSION POS(200),ACC(200),IS(200),VEL1(200),VEL(200),
* CP0S(200),H(9),V(9),Q(7),AC1(3),ISAS(7)
C
C MODULE FUNCTION
0003 MOD(I)=MOD(I,ISAMPL)+1
C
C EVENT INDICATOR ASSIGNMENT
0004 DATA ISAS/-300,-200,-100,0,100,200,300/
C
C I/O ASSIGNMENT
C
0005 TYPE 109
0006 109 FORMAT(’/,’ HOW LONG (# OF SMPL PNTS) IS YOUR INPUT FILE? ’$)
0007 READ(5,102)I LENG T
C
0008 TYPE 107
0009 107 FORMAT(’ WHAT IS THE STARTING POINT? ’$)
0010 READ(5,102)ILENGT
C
0011 TYPE 110
0012 110 FORMAT(’ ENTER INPUT FILESPEC’/)
0013 CALL ASSIGN(1,’LYO2FP,EOG’,-1,’RDO’)
0014 DEFINE FILE 1 (ILENGT,1,U,NREC)
C
0015 ILENGT=ILENGT-ILEN
C
0016 CALL ASSIGN (4,’LYO2FP,EOG’,9,’SCR’)
0017 DEFINE FILE 4 (ILENGT,1,U,NREC)
C
0018 DO 108 IDUM=1,ILEN GT
0019 READ(1,’IDUM+ILEN)IDUMMY
0020 WRITE(4,’IDUM)IDUMMY
0021 108 CONTINUE
C
0022 CLOSE(UNIT=1)
C
0023 DO 112 IUNIT=1,3
0024 CALL ASSIGN(IUNIT,’LYO2FP,EXT’,IUNIT+3,’SCR’)
0025 DEFINE FILE IUNIT (ILENGT,1,U,NREC1)
0026 112 CONTINUE
0027 CALL ASSIGN(10,’LYO2FP,EXT’,10,’SCR’)
0028 DEFINE FILE 10(ILENGT,1,U,NREC1)
C
0029 111 TYPE 101
0030 101 FORMAT(’ WHAT IS THE SAMPLING FREQUENCY ’$)
0031 READ(5,102)ISAMPL
0032 IF(ISAMPL.LT.40.OR.ISAMPL.GT.200)GO TO 111
0034 102 FORMAT(I7)
C
FORTRAN IV

0035  113  TYPE 103
0036   103  FORMAT(' WHAT IS MIN. ACC. THRESHOLD (500-10000DEG/SEC2) ?'*)
0037    READ(5,106)CNSTRM
0038        IF(CNSTRM.LT.500.0R.CNSTRM.GT.10000.)GO TO 113

C
0040   TYPE 105
0041  105  FORMAT(' WHAT IS SCALE FACTOR ?'*)
0042    READ(5,106)GAIN
0043  106  FORMAT(E12.5)

C
C MAIN ALGORITHM STARTS HERE
C INITIALIZE VARIABLES AND CONSTANTS

0044     NS=ISAMPL/2
0045    FSAMPL=FLOAT(ISAMPL)
0046   DELTAT=1./FSAMPL
0047  FSAMP2=FLOAT(ISAMPL)**2
0048    IC11=0
0049    A2=0.
0050   SUM3=0.
0051    J12=1
0052    SUMA=0.
0053   SUMASQ=0.
0054    ND=0
0055    J=5
0056   ISLWLNL=5
0057     NS1=NS/2
0058    MFSR1=NS/3
0059    MTEMLN=NS/2

C
C FIND THE UNIT SAMPLE RESPONSE OF THE PAF FILTER
C AND VELOCITY FILTER

0060    DATA V/-0.0322,-.0716,-.0678,-.0522,0.,0.0522,0.0678,0.0716,0.0322/
0061    DATA V/-0.0077,-.0714,-.1078,-.0870,0.0870,1078,0.0714,0.0077/
0062    DATA V/,0.065,-.026,-.215,-.249,0.,.249,.215,.026,-.065/
0063    DATA AC1/1.1,-2.,1.1/
0064  CALL FILTER(AC1,H,0)

C
C INITIALIZE ALL ARRAYS TO ZERO

0065    DO 19 IC12=1,1200
0066     POS(IC12)=0.
0067    CPOS(IC12)=0.
0068    ACC(IC12)=0.
0069    VEL(IC12)=0.
0070       19 CONTINUE

C
C ACTUAL PROCESSING STARTS HERE

0071    READ(4,ND+1)IPOS
0072  IF(NREC.GE.ILENGT) GO TO 900
0073     K1=MD(ND)
0074    POS(K1)=GAIN*FLOAT(IPOS)
0075    ND=ND+1
0076    IF(ND.LT.9) GO TO 10
FORTRAN IV
V02.5-2  Mon 09-May-83 02:57:27

0079 N=ND-5

C
C COMPUTE THE OUTPUT OF FILTERS
0080 SUM1=0.
0081 SUM=0.
0082 DO 2 IC2=-4,4
0083 POS1=POS(MD(N+IC2))
0084 SUM=SUM+H(IC2+5)*POS1
0085 SUM1=SUM1+V(IC2+5)*POS1
0086 2 CONTINUE
0087 K3=MD(N)
0088 SUM=SUM*FSAMP2
0089 ACC(K3)=SUM
0090 VEL(K3)=SUM1*FSAMPL
0091 IF(N.LT.NS1) GO TO 17

C
C FIND THE MOVING AVERAGE AND VARIANCE OF ACC OVER PAST .25 SEC
0093 A2=ACC(MD(N-NS1))
0094 17 SUMA=SUMA+SUM-A2
0095 SUMASQ=SUMASQ+SUM**2-A2**2
0096 IF(N.LT.NS1) GO TO 23
0098 FNS=FLOAT(NS1)
0099 ACCRM=(SUMASQ-SUMA**2/FNS)/FNS
0100 ACCRMS=SQR(ABS(ACCRM))

C
C CHECK IF THE CIRCULAR BUFFER IS FULL
C IF NOT LOOP BACK TO FILL IT UP
0101 23 IF(N-J.LT.MTEMLE) GO TO 10

C
C SET UP THE THRESHOLDS AND
0103 K4=MD(J)
0104 TRESHA=ACCRMS+.5*CNSTRM
0105 IF(TRESHA.LT.CNSTRM)TRESHA=CNSTRM
0107 TRESHB=TRESHA*.7

C
C CHECK FOR THE START OF A FAST MOVEMENT EVENT
C IF SLOW PHASE GO TO 20
0108 CI1=ACC(K4)
0109 IF(ABS(CI1).LT.TRESHA) GO TO 20
0111 IF(ABS(CI1).LT.ABS(ACC(MD(J-1)))) GO TO 20
0113 ISG1=ISGN(CI1)
0114 IF(ISGN(VEL(MD(J+1))).NE.ISG1) GO TO 20

C
C FIND THE SLOW PHASE VELOCITY DURING THE PRECEEDING
C SLOW PHASE
0116 IF(ISLWLN.NE.0)AVV3=SUM3/ISLWLN

C
C SEARCH FORWARD FOR THE END OF EVENT
0118 DO 12 IC4=1,MFSR1
0119 MDJIC4=MD(J+IC4)
0120 A1=ACC(MDJIC4)
0121 A3=VEL(MDJIC4)
0122 IF(ABS(A1).GE.TRESHB.AND.
* .ISGN(A1).NE.ISG1) IC5=IC4
IF(ISGN(A3).NE.ISG1.AND.ABS(A1).LT.TRESHB) GOTO 11
12 CONTINUE
C
C FIND THE AVERAGE OF VELOCITY BEFORE AND AFTER EVENT
11 AVV=(VEL(MD(J-3))+VEL(MD(J+IC5)))/2.
C
C HOLD THE VELOCITY DURING THE FAST PHASE LOW PASS THE SLOW
C PHASE VELOCITY AND SET THE EVENT INDICATOR FLAG
C THE EVENT DETECTOR IS SET ACCORDING TO THE DIRECTION
C OF FAST PHASE AND ACCORDING TO THE SLOW PHASE VELOCITY
C PRECEEDING IT
J11=3*ISG1
IF(ISG1.EQ.J12)J11=2*ISG1
IF(ABS(AVV3).LT.4.)J11=ISG1
J11=J11+4
C
K10=MD(J-3)
DO 7 IC6=-2,IC5+2
K9=MD(J+IC6)
CPOS(K9)=CPOS(K10)+FLOAT(IC6+3)*AVV3*DELTAT
IS(K9)=ISAS(J11)
VEL(K9)=AVV3
VEL1(K9)=0.
DO 7 IC8=1,7
VEL1(K9)=VEL1(K9)+O(IC8)*VEL(MD(J+IC6-IC8+1))
7 CONTINUE
C
J12=ISG1
IC7=J-2
C ADVANCE THE DATA USED INDICATOR
J=J+IC6
C SET THE SLOW PHASE DURATION INDICATOR TO ZERO
ISLWLN=0
SUM3=0.
C
C WRITE THE RESULTS
DO 8 IC8=IC7,J
K7=MD(IC8)
VE=VEL1(K7)
IF(ABS(VE).LT.30000.)GO TO 135
VE=30000.*ISGN(VE)
I90=INT(VE)
AC12=ACC(K7)
IF(ABS(AC12).LT.30000.)GO TO 136
AC12=30000.*ISGN(AC12)
I91=INT(AC12)
I92=INT(CPOS(K7))
WRITE(1,'(IC8)')I90
WRITE(2,'(IC8)')I91
WRITE(3,'(IC8)')IS(K7)
WRITE(10,'(IC8)')I92
8 CONTINUE
C
C GO BACK AND READ MORE DATA
0167 J=J+1
0168 GO TO 10

C
C ADVANCE THE SLOW PHASE DURATION INDICATOR
0169 20 ISWLNL=ISWLNL+1
C SET THE SACCADIC INDICATOR TO ZERO
0170 IS(K4)=0
C LOW PASS THE SLOW PHASE VELOCITY
C AND WRITE THE RESULTS
0171 VEL1(K4)=0.
0172 KS=M1D(J-1)
0173 CP0S(K4)=CP0S(K5)+POS(K4)-POS(K5)
0174 DO 22 IC8=1,7
0175 22 VEL1(K4)=VEL1(K4)+O(IC8)*VEL(M1D(1-IC8+J))
0176 SUM3=SUM3+VEL1(K4)
0177 IF(ISWLNL.LT.3) GO TO 21
0179 J1=J-2
0180 K4=M1D(J1)
0181 VE=VEL1(K4)
0182 IF(ABS(VE).LT.30000.)GO TO 138
0184 VE=30000.*ISGN(VE)
0185 138 I90=INT(VE)
0186 AC12=ACC(K4)
0187 IF(ABS(AC12).LT.30000.)GO TO 139
0189 AC12=30000.*ISGN(AC12)
0190 139 I91=INT(AC12)
0191 I92=INT(CP0S(K4))
0192 WRITE(1,'J1)I90
0193 WRITE(2,'J1)I91
0194 WRITE(3,'J1)IS(K4)
0195 WRITE(10,'J1)I92

C
C ADVANCE DATA USED INDICATOR
0196 21 J=J+1
C GO BACK AND READ MORE DATA
0197 GO TO 10

C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0198 900 TYPE 802
0199 802 FORMAT(/,' AT THIS POINT I CAN PLOT THE FOLLOWING:','
   * 1) NO GRAPHS AT ALL,'/,'*
   * 2) RAW EYE POS. & SACCADIC IND. ON THE SAME GRAPH','/,'*
   * 3) SACCADIC IND. & PSUEDO ACC. ON THE SAME GRAPH','/,'*
   * 4) SLOW PHASE VELOCITY & RAW EYE POS. ON THE SAME GRAPH','/,'*
   * 5) RAW EYE POSITION & PSUEDO ACC. ON THE SAME GRAPH','/,'*
   * 6) RAW EYE POS. AND CUMM. SLOW PHASE POS. ON THE SAME GRAPH'
   * TYPE YOUR SELECTION: '*)
0200 READ(5,102)IDUM
0201 GO TO(803,804,805,806,807,808)IDUM
0202 804 CALL P2(ILENGLT,3,4,1,400,5,200)
0203 GO TO 900
0204 805 CALL P2(ILENGLT,3,2,1,400,40,500)
0205 GO TO 900
0206 806 CALL P2(ILENGLT,1,4,1,400,5,300)
GO TO 900
CALL P2(ILENGTH,2,4,40,300,5,200)
GO TO 900
CALL P2(ILENGTH,10,4,1,500,5,200)
GO TO 900
CLOSE(UNIT=4)

C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

0213 901 TYPE 902
0214 902 FORMAT(//,' THESE ADDITIONAL OUTPUTS ARE AVAILABLE:',//,
* 1) NO ADDITIONAL OUTPUT',//,
* 2) AVE. SLW. PHASE VEL.(1 PNT. PER BEAT)',//,
* 3) ',//,
* 4) ',//,
* TYPE YOUR SELECTION: ')$

READ(5,102)IDUM
GOTO (905,301)IDUM

C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C MOHAMMAD-ALI MAASSOUNIA
C REVISION 8-SEP-82
C THIS PROGRAM FINDS THE ONE POINT PER BEAT SLOW PHASE VELOCITY
C BY AVERAGING THE VELOCITY DURING THE SLOW PHASE AND ASSIGNING IT TO
C A TIME IN THE MIDDLE OF SLOW PHASE. (NOTE THAT THE POINTS
C WILL NOT BE EQUAL DISTANCE A PART.)
C THE PROGRAM ALSO PLOTS THE 1 PNT PER BEAT SLW. PHASE VELOCITY
C UNDER REQUEST
C

0217 301 TYPE 904
0218 904 FORMAT(//,' ENTER OUTPUT FILESPEC'/)
0219 CALL ASSIGN (4,'LY02FP.EXT',-1,'NEW')
0220 DEFINE FILE 4(ILENGTH,2,U,NREC)
0221 IC1=1
0222 IC2=0
0223 IC3=0
0224 IC4=0
0225 IC5=1
0226 ISUM=0
0227 320 READ(1'IC1)IVEL1
0228 READ(3'IC1)IS1
0229 IF(IC1.GE.ILENGTH)GO TO 324
0231 IF(IC3.GE.ISAMPL/2)IS1=100
0233 IF(IS1.NE.0)GOTO 322
0235 ISUM=ISUM+IVEL1
0236 IC1=IC1+1
0237 IC3=IC3+1
0238 IC2=1
0239 GO TO 320

C
0240 322 IF(IC2.NE.1) GO TO 321
0242 ISUM=ISUM/IC3
0243 IC4=IC1-(IC3/2)
0244 WRITE(4'IC4)IC4,ISUM
0245 IC5=IC5+1
ISUM=0
IC3=0
IC2=0
IC1=IC1+1
GO TO 320
IC4=-1
ISUM=-1
WRITE(4,'IC5.IC4.ISUM

TYPE 332
FORMAT('DO YOU WANT THE 1PNT/BAT PLOT',/,
* 0) NO, 1) YES(ORDINARY SCALE), 2) YES(LOG SCALE)? '$)
READ(5,102)IDUM
IF(IDUM.EQ.0)GO TO 901
CALL F3(4,IDUM)
CLOSE(UNIT=4)
GO TO 901

+++

TYPE 910
FORMAT('TYPE YOUR CHOICE!',/,
* 0) KEEP NO MORE FILE',/,
* 1) KEEP SLW. PHASE VELOCITY',/,
* 2) KEEP ACC.',/,
* 3) KEEP SAC. IND.',/,
* 4) KEEP CUMM. SLOW PHASE POS.',/,
* ?*')

READ(5,102)IDUM
IF(IDUM.EQ.0)GO TO 1000
IF(IDUM.GT.4.OR.IDUM.LT.0)GO TO 912
IF(IDUM.EQ.4)IDUM=10
TYPE 904
CALL ASSIGN(4,'LY02FP.EXT','-1',NEW')
DEFINE FILE 4(I4,1,1,U,NREC)
DO 914 IC1=1,I4
READ(IDUM',IC1)IC2
WRITE(4,'IC1.IC2
CONTINUE
CLOSE(UNIT=IDUM)
CLOSE(UNIT=4)
TYPE 915
FORMAT('FILE SAVED WHAT ELSE?',/,' ?*')
GO TO 912
STOP
END
FORTRAN IV     Diagnostics for Program Unit .MAIN.

In line 0018, Warning: Possible modification of index "IDUM"
In line 0023, Warning: Possible modification of index "IUNIT"
In line 0149, Warning: Possible modification of index "IC8"
In line 0274, Warning: Possible modification of index "IC1"

FORTRAN IV     Storage Map for Program Unit .MAIN.

Local Variables, .PSECT $DATA, Size = 011254 ( 2390. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCRM</td>
<td>R*4</td>
<td>011046</td>
<td>ACCRMS</td>
<td>R*4</td>
<td>011052</td>
<td>AC12</td>
<td>R*4</td>
<td>011150</td>
</tr>
<tr>
<td>AVV</td>
<td>R*4</td>
<td>011120</td>
<td>AVV3</td>
<td>R*4</td>
<td>011076</td>
<td>A1</td>
<td>R*4</td>
<td>011106</td>
</tr>
<tr>
<td>A2</td>
<td>R*4</td>
<td>010754</td>
<td>A3</td>
<td>R*4</td>
<td>011112</td>
<td>CI1</td>
<td>R*4</td>
<td>011070</td>
</tr>
<tr>
<td>CNSTRM</td>
<td>R*4</td>
<td>010724</td>
<td>DELTAT</td>
<td>R*4</td>
<td>010742</td>
<td>FNS</td>
<td>R*4</td>
<td>011042</td>
</tr>
<tr>
<td>FSAMPL</td>
<td>R*4</td>
<td>010736</td>
<td>FSAMP2</td>
<td>R*4</td>
<td>010746</td>
<td>GAIN</td>
<td>R*4</td>
<td>010730</td>
</tr>
<tr>
<td>I</td>
<td>I*2</td>
<td>010704</td>
<td>IC1</td>
<td>I*2</td>
<td>011164</td>
<td>IC11</td>
<td>I*2</td>
<td>010752</td>
</tr>
<tr>
<td>IC12</td>
<td>I*2</td>
<td>011012</td>
<td>IC2</td>
<td>I*2</td>
<td>011032</td>
<td>IC3</td>
<td>I*2</td>
<td>011166</td>
</tr>
<tr>
<td>IC4</td>
<td>I*2</td>
<td>011102</td>
<td>IC5</td>
<td>I*2</td>
<td>011116</td>
<td>IC6</td>
<td>I*2</td>
<td>011130</td>
</tr>
<tr>
<td>IC7</td>
<td>I*2</td>
<td>011136</td>
<td>IC8</td>
<td>I*2</td>
<td>011134</td>
<td>IDUM</td>
<td>I*2</td>
<td>010714</td>
</tr>
<tr>
<td>IDUMMY</td>
<td>I*2</td>
<td>010716</td>
<td>ILEN</td>
<td>I*2</td>
<td>010710</td>
<td>ILENGT</td>
<td>I*2</td>
<td>010706</td>
</tr>
<tr>
<td>IP0S</td>
<td>I*2</td>
<td>011014</td>
<td>ISG1</td>
<td>I*2</td>
<td>011074</td>
<td>ISLWLN</td>
<td>I*2</td>
<td>011002</td>
</tr>
<tr>
<td>ISUM</td>
<td>I*2</td>
<td>011170</td>
<td>IS1</td>
<td>I*2</td>
<td>011174</td>
<td>IUNIT</td>
<td>I*2</td>
<td>010720</td>
</tr>
<tr>
<td>IVEL1</td>
<td>I*2</td>
<td>011172</td>
<td>I90</td>
<td>I*2</td>
<td>011146</td>
<td>I91</td>
<td>I*2</td>
<td>011154</td>
</tr>
<tr>
<td>I92</td>
<td>I*2</td>
<td>011156</td>
<td>J</td>
<td>I*2</td>
<td>011000</td>
<td>J1</td>
<td>I*2</td>
<td>011162</td>
</tr>
<tr>
<td>J11</td>
<td>I*2</td>
<td>011124</td>
<td>J12</td>
<td>I*2</td>
<td>010764</td>
<td>K1</td>
<td>I*2</td>
<td>011016</td>
</tr>
<tr>
<td>K10</td>
<td>I*2</td>
<td>011126</td>
<td>K3</td>
<td>I*2</td>
<td>011040</td>
<td>K4</td>
<td>I*2</td>
<td>011056</td>
</tr>
<tr>
<td>K5</td>
<td>I*2</td>
<td>011160</td>
<td>K7</td>
<td>I*2</td>
<td>011140</td>
<td>K9</td>
<td>I*2</td>
<td>011132</td>
</tr>
<tr>
<td>MDIC4</td>
<td>I*2</td>
<td>011104</td>
<td>MFSR1</td>
<td>I*2</td>
<td>011006</td>
<td>MTEMLE</td>
<td>I*2</td>
<td>011010</td>
</tr>
</tbody>
</table>
| N          | I*2  | 011020 | ND         | I*2  | 010776 | NREC       | I*2  | 010712 Eav
| NREC1      | I*2  | 010722 Eav | NS     | I*2  | 010734 | NS1        | I*2  | 011004 |
| POS1       | R*4  | 011034 | SUM        | R*4  | 011026 | SUMA       | R*4  | 010766 |
| SUMASQ     | R*4  | 010772 | SUM1       | R*4  | 011022 | SUM3       | R*4  | 010760 |
| TRESHA     | R*4  | 011060 | TRESHB     | R*4  | 011064 | VE         | R*4  | 011142 |

COMMON Block /DETECT/, Size = 000002 ( 1. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAMPL</td>
<td>I*2</td>
<td>000000</td>
</tr>
</tbody>
</table>
Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>Size</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>R*4</td>
<td>$DATA</td>
<td>001440</td>
<td>001440</td>
<td>(400,) (200)</td>
</tr>
<tr>
<td>AC1</td>
<td>R*4</td>
<td>$DATA</td>
<td>010624</td>
<td>000014</td>
<td>(6,) (3)</td>
</tr>
<tr>
<td>CPOS</td>
<td>R*4</td>
<td>$DATA</td>
<td>007020</td>
<td>001440</td>
<td>(400,) (200)</td>
</tr>
<tr>
<td>H</td>
<td>R*4</td>
<td>$DATA</td>
<td>010460</td>
<td>000044</td>
<td>(18,) (9)</td>
</tr>
<tr>
<td>IS</td>
<td>I*2</td>
<td>$DATA</td>
<td>003100</td>
<td>000620</td>
<td>(200,) (200)</td>
</tr>
<tr>
<td>ISAS</td>
<td>I*2</td>
<td>$DATA</td>
<td>010640</td>
<td>000016</td>
<td>(7,) (7)</td>
</tr>
<tr>
<td>O</td>
<td>R*4</td>
<td>$DATA</td>
<td>010570</td>
<td>000034</td>
<td>(14,) (7)</td>
</tr>
<tr>
<td>POS</td>
<td>R*4</td>
<td>$DATA</td>
<td>000000</td>
<td>001440</td>
<td>(400,) (200)</td>
</tr>
<tr>
<td>V</td>
<td>R*4</td>
<td>$DATA</td>
<td>010524</td>
<td>000044</td>
<td>(18,) (9)</td>
</tr>
<tr>
<td>VEL</td>
<td>R*4</td>
<td>$DATA</td>
<td>005360</td>
<td>001440</td>
<td>(400,) (200)</td>
</tr>
<tr>
<td>VEL1</td>
<td>R*4</td>
<td>$DATA</td>
<td>003720</td>
<td>001440</td>
<td>(400,) (200)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>R*4</td>
<td>ASSIGN</td>
<td>R*4</td>
<td>FILTER</td>
<td>R*4</td>
<td>FLOAT</td>
<td>R*4</td>
<td>INT</td>
<td>I*2</td>
</tr>
<tr>
<td>ISGN</td>
<td>I*2</td>
<td>M</td>
<td>I*2</td>
<td>MOD</td>
<td>I*2</td>
<td>P</td>
<td>R*4</td>
<td>P3</td>
<td>R*4</td>
</tr>
<tr>
<td>SORT</td>
<td>R*4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE M2FFD1
C FAST PHASE DETECTOR
C BY MOHAMMAD-ALI MASSOUINIA
C AUG-21-82
C
0002 COMMON/DETECT/ISAMPL,GAIN,CNSTRM,ILENGT
0003 DIMENSION POS(200),ACC(200),IS(200),VEL1(200),VEL(200),*
C POS(200),H(9),V(9),O(7),AC1(3),ISAS(7)
C ACC RING BUFFER FOR STORING PAF
C IS RING BUFFER FOR STORING SACCACDE INDICATOR
C VEL RING BUFFER FOR STORING SMOOTHED SPR
C VEL RING BUFFER FOR STORING SPR
C CP0S RING BUFFER FOR STORING CUMMULATIVE POSITION
C H USR OF PAF
C V USR OF VELOCITY FILTER
C O USR OF 7 POINT LOW PASS FILTER
C AC1 USR OF 3 POINT STIRLING DOUBLE DIFFERENTIATOR
C ISAS INDICATOR FLAG ASSIGNMENT
C ISAMPL SAMPLING FREQUENCY
C GAIN SACLE FACTOR DEG./UNIT OF A/D CONVERTOR
C CNSTRM MINIMUM ACCELERATION THRESHOLD (MAT)
C ILENGT LENGTH OF THE SEGMENT OF INPUT FILE TO BE PROCESSED
C MFSR1 MAXIMUM FORWARD SEARCH FOR THE END OF AN EVENT
C MTEMLE MAXIMUM LENGTH OF RING BUFFER
C ISLWLN DURATION OF A SLOW PHASE
C NS1 WINDOW LENGTH FOR THE RUNNING RMS OF ACCELERATION
C AVV3 AVERAGE VELOCITY DURING A SLOW PHASE
C
0004 FUNCTION MD(I)=MOD(I,ISAMPL)+1
C
0005 C INDICATOR FLAG ASSIGNMENT
0006 DATA ISAS/-300,-200,-100,0,100,200,300/
C
0007 C INITIALIZE VARIABLES AND CONSTANTS
0008 NS=ISAMPL/2
0009 FSAMPL=FLOAT(ISAMPL)
0010 DELTAT=1./FSAMPL
0011 FSAMP2=FLOAT(ISAMPL)**2
0012 IC11=0.
0013 A2=0.
0014 SUM3=0.
0015 J12=1
0016 SUMA=0.
0017 SUMASQ=0.
0018 ND=0
0019 J=5
0020 ISLWLN=5
0021 MFSR1=NS/3
0022 MTEMLE=NS/2
C
0023 C FIND THE UNIT SAMPLE RESPONSE OF THE PAF FILTER
C AND VELOCITY FILTER
0022      DATA V/-.0332,-.0716,-.0678,-.0522,0.,-.0522,.0678,.0716,.0332/
         C    DATA V/-.0077,-.0714,-.1078,-.0870.,-.0870,.1078,.0714,.0077/
         C    DATA V/.065,-.026,.215,.249,0.,.249,.215,.026,-.065/
         C
0023      DATA AC1/1.,-2.,1.,/
0024      CALL FILTER(AC1,H,0)
         C
0025      C INITIALIZE ALL ARRAYS TO ZERO
         DO 300 IC12=1,200
0026      POS(IC12)=0.
0027      CPOS(IC12)=0.
0028      ACC(IC12)=0.
0029      VEL(IC12)=0.
0030      VEL1(IC12)=0.
0031      IS(IC12)=0
0032      300      CONTINUE
         C
0033      C ACTUAL PROCESSING STARTS HERE
0034      FORMAT(/','SAMP. RATE=','15,'MAT=','F9.1,'SCALE FACTOR=','F9.5,/')
0035      WRITE(*,305)ISAMPL,CNSTRM,GAIN
0036      310      ND1=ND+1
0037      IF(ND1.GT.ILEN1)GO TO 460
0038      CALL M2RD1(4,ND1,IPOS)
0039      K1=MD(N)
0040      POS(K1)=GAIN*FLOAT(IPOS)
0041      ND=ND+1
0042      IF(ND.LT.9) GO TO 310
0043      N=ND-5
         C
0045      C COMPUTE THE OUTPUT OF FILTERS
0046      SUM1=0.
0047      SUM=0.
0048      DO 320 IC2=-4,4
          POS1=POS(MD(N+IC2))
0049      SUM=SUM+H(IC2+5)*POS1
0050      SUM1=SUM1+V(IC2+5)*POS1
0051      320      CONTINUE
         C
0052      K3=MD(N)
0053      SUM=SUM*FSAMP2
0054      ACC(K3)=SUM
0055      VEL(K3)=SUM1*FSAMPL
0056      IF(N.LT.NS1) GO TO 330
         C
0058      C FIND THE MOVING AVERAGE AND VARIANCE OF ACC OVER PAST .25 SEC
0059      A2=ACC(MD(N-NS1))
0060      SUMA=SUMA+SUM-A2
0061      SUMASQ=SUMASQ+SUM**2-A2**2
0062      IF(N.LT.NS1) GO TO 340
0063      FNS=FLOAT(NS1)
0064      ACCRM=(SUMASQ-SUM**2/FNS)/FNS
0065      ACCRMS=SQRT(ABS(ACCRM))
         C
0066      C CHECK IF THE CIRCULAR BUFFER IS FULL
C SET UP THE_THRESHOLDS AND
0068   K4=MD(J)
0069   TRESHA=ACCRMS+CNSTRM*.5
0070   IF(TRESHA.LT.CNSTRM)TRESHA=CNSTRM
0072   TRESHB=TRESHA*.7

C CHECK FOR THE START OF FAST MOVEMENT EVENT
C IF SLOW PHASE GO TO 410
0073   C15=ABS(ACC(K4))
0074   IF(C15.LT.TRESHA) GO TO 410
0076   C16=ABS(ACC(MD(J-1)))
0077   IF(C15.LT.C16)GO TO 410
0079   ISG1=ISGN(ACC(K4))
0080   IF(ISGN(VEL(MD(J+1))).NE.ISG1) GO TO 410

C FIND THE SLOW PHASE VELOCITY DURING THE PRECEEDING
C SLOW PHASE
0082   IF(ISLWL,N.E.0)AVV3=SUM3/ISLWN

C SEARCH FORWARD FOR THE END OF EVENT
0084   DO 350 IC4=1,MFSR1
0085   MDJIC4=MD(J+IC4)
0086   A1=ACC(MDJIC4)
0087   A3=VEL(MDJIC4)
0088   IF(ABS(A1).GE.TRESHB.AND*
*   .ISGN(A1).NE.ISG1) IC5=IC4
0090   IF(ISGN(A3).NE.ISG1.AND.ABS(A1).LT.TRESHB) GOTO 360
0092   350 CONTINUE

C FIND THE AVERAGE OF VELOCITY BEFORE AND AFTER EVENT
0093   360   AVV=(VEL(MD(J-3))+VEL(MD(J+3+IC5)))/2.
C
C HOLD THE VELOCITY DURING THE EVENT LOW PASS THE SLOW
C PHASE VELOCITY AND SET THE EVENT INDICATOR FLAG
C THE EVENT INDICATOR IS SET ACCORDING TO THE DIRECTION
C OF FAST MOVEMENT AND ACCORDING TO THE SLOW PHASE VELOCITY
C PRECEEDING IT
0094   J11=3*ISG1
0095   IF(ISG1.EQ.J12)J11=2*ISG1
0097   IF(ABS(AVV3).LT.2.)J11=ISG1
0099   J11=J11+4

C
0100   K10=MD(J-3)
0101   DO 370 IC6=-2,IC5+2
0102   K9=MD(J+IC6)
0103   CPPOS(K9)=CPPOS(K10)+FLOAT(IC6+3)*AVV3*DELTAT
0104   IS(K9)=IAS6(J11)
0105   VEL(K9)=AVV
0106   VEL1(K9)=0.
0107   DO 370 IC8=1,7
0108   VEL1(K9)=VEL1(K9)+O(IC8)*VEL(MD(J+IC6-IC8+1))
0109   370 CONTINUE
C
0110  J12=ISG1
0111  IC7=J-2
C ADVANCE THE DATA USED INDICATOR
0112  J=J+IC6
C
C SET THE SLOW PHASE DURATION INDICATOR TO ZERO
0113  ISLWLN=0
0114  SUM3=0.
C
C WRITE THE RESULTS
0115  DO 400 IC8=IC7,J
0116     IC9=IC8
0117     K7=MD(IC9)
0118     VE=VEL1(K7)
0119     IF(ABS(VE),LT,30000.)GO TO 380
0120     VE=30000.*ISGN(VE)
0122  380  I90=INT(VE)
0123     AC12=ACC(K7)
0124     IF(ABS(AC12),LT,30000.)GO TO 390
0126     AC12=30000.*ISGN(AC12)
0127  390  I91=INT(AC12)
0128     I92=INT(CPOS(K7))
0129    CALL M2WR4(IC9,190,191,IS(K7),I92)
0130  400   CONTINUE
C
C GO BACK AND READ MORE DATA
C
0131  J=J+1
0132  GO TO 310
C
C ADVANCE THE SLOW PHASE DURATION INDICATOR
0133   410   ISLWLN=ISLWLN+1
C SET THE EVENT INDICATOR TO ZERO
0134   IS(K4)=0
C LOW PASS THE SLOW PHASE VELOCITY
C AND WRITE THE RESULTS
C
0135   K5=MD(J-1)
0136   CPOS(K4)=CPOS(K5)+POS(K4)-POS(K5)
0137   VEL1(K4)=0.
0138    DO 420 IC8=1,J7
0139    VEL1(K4)=VEL1(K4)+O(IC8)*VEL(MD(1-IC8+J))
0140   420   CONTINUE
0141    SUM3=SUM3+VEL1(K4)
0142   IF(ISLWLN,LT,3)GO TO 450
0144    J1=J-2
0145   K4=MD(J1)
0146   VE=VEL1(K4)
0147   IF(ABS(VE),LT,30000.)GO TO 430
0149    VE=30000.*ISGN(VE)
0150   430   I90=INT(VE)
0151    AC12=ACC(K4)
0152   IF(ABS(AC12),LT,30000.)GO TO 440
0154       AC12=30000.*ISGN(AC12)
0155  440       I91=INT(AC12)
0156       I92=INT(CPOS(K4))
0157       CALL M2WR4(J1,I90,I91,IS(K4),I92)

C
C ADVANCE DATA USED INDICATOR ; GO BACK AND READ MORE DATA

0158  450       J=J+1
0159       GO TO 310

C
0160  460       DO 470 I1=J1,ILENGT
0161       I2=I1
0162  470       CALL M2WR4(I2,0,0,0,0)
0163       RETURN
0164       END
FORTRAN IV Storage Map for Program Unit M2FFD1

Local Variables, .PSECT $DATA, Size = 011222 ( 2377, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCRM</td>
<td>R*4</td>
<td>011022</td>
<td>ACCRMS</td>
<td>R*4</td>
<td>011026</td>
<td>AC12</td>
<td>R*4</td>
<td>011132</td>
</tr>
<tr>
<td>AVV</td>
<td>R*4</td>
<td>011100</td>
<td>AVV3</td>
<td>R*4</td>
<td>011056</td>
<td>A1</td>
<td>R*4</td>
<td>011066</td>
</tr>
<tr>
<td>A2</td>
<td>R*4</td>
<td>010726</td>
<td>A3</td>
<td>R*4</td>
<td>011072</td>
<td>C15</td>
<td>R*4</td>
<td>011044</td>
</tr>
<tr>
<td>C16</td>
<td>R*4</td>
<td>011050</td>
<td>DELTAT</td>
<td>R*4</td>
<td>010714</td>
<td>FNS</td>
<td>R*4</td>
<td>011016</td>
</tr>
<tr>
<td>FSAMFL</td>
<td>R*4</td>
<td>010710</td>
<td>FSAMP2</td>
<td>R*4</td>
<td>010720</td>
<td>I</td>
<td>I*2</td>
<td>010704</td>
</tr>
<tr>
<td>IC11</td>
<td>I*2</td>
<td>010724</td>
<td>IC12</td>
<td>I*2</td>
<td>010764</td>
<td>IC2</td>
<td>I*2</td>
<td>011006</td>
</tr>
<tr>
<td>IC4</td>
<td>I*2</td>
<td>011062</td>
<td>IC5</td>
<td>I*2</td>
<td>011076</td>
<td>IC6</td>
<td>I*2</td>
<td>011110</td>
</tr>
<tr>
<td>IC7</td>
<td>I*2</td>
<td>011116</td>
<td>IC8</td>
<td>I*2</td>
<td>011114</td>
<td>IC9</td>
<td>I*2</td>
<td>011120</td>
</tr>
<tr>
<td>IFOS</td>
<td>I*2</td>
<td>010770</td>
<td>ISG1</td>
<td>I*2</td>
<td>011054</td>
<td>ISLWLN</td>
<td>I*2</td>
<td>010754</td>
</tr>
<tr>
<td>I1</td>
<td>I*2</td>
<td>011146</td>
<td>I2</td>
<td>I*2</td>
<td>011150</td>
<td>I90</td>
<td>I*2</td>
<td>011130</td>
</tr>
<tr>
<td>I91</td>
<td>I*2</td>
<td>011136</td>
<td>I92</td>
<td>I*2</td>
<td>011140</td>
<td>J</td>
<td>I*2</td>
<td>010752</td>
</tr>
<tr>
<td>J1</td>
<td>I*2</td>
<td>011144</td>
<td>J11</td>
<td>I*2</td>
<td>011104</td>
<td>J12</td>
<td>I*2</td>
<td>010736</td>
</tr>
<tr>
<td>K1</td>
<td>I*2</td>
<td>010772</td>
<td>K10</td>
<td>I*2</td>
<td>011106</td>
<td>K3</td>
<td>I*2</td>
<td>011014</td>
</tr>
<tr>
<td>K4</td>
<td>I*2</td>
<td>011032</td>
<td>K5</td>
<td>I*2</td>
<td>011142</td>
<td>K7</td>
<td>I*2</td>
<td>011222</td>
</tr>
<tr>
<td>K9</td>
<td>I*2</td>
<td>011112</td>
<td>MDJIC4</td>
<td>I*2</td>
<td>011064</td>
<td>MFSR1</td>
<td>I*2</td>
<td>010760</td>
</tr>
<tr>
<td>MTEMLE</td>
<td>I*2</td>
<td>010762</td>
<td>N</td>
<td>I*2</td>
<td>010774</td>
<td>ND</td>
<td>I*2</td>
<td>010750</td>
</tr>
<tr>
<td>ND1</td>
<td>I*2</td>
<td>010766</td>
<td>NS</td>
<td>I*2</td>
<td>010706</td>
<td>NS1</td>
<td>I*2</td>
<td>010756</td>
</tr>
<tr>
<td>POS1</td>
<td>R*4</td>
<td>011010</td>
<td>SUM</td>
<td>R*4</td>
<td>011002</td>
<td>SUMA</td>
<td>R*4</td>
<td>010740</td>
</tr>
<tr>
<td>SUMASQ</td>
<td>R*4</td>
<td>010744</td>
<td>SUM1</td>
<td>R*4</td>
<td>010776</td>
<td>SUM3</td>
<td>R*4</td>
<td>010732</td>
</tr>
<tr>
<td>TRESHA</td>
<td>R*4</td>
<td>011034</td>
<td>TRESHB</td>
<td>R*4</td>
<td>011040</td>
<td>VE</td>
<td>R*4</td>
<td>011124</td>
</tr>
</tbody>
</table>

COMMON Block /JDET/ , Size = 000014 ( 6, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAMPL</td>
<td>I*2</td>
<td>000000</td>
<td>GAIN</td>
<td>R*4</td>
<td>000002</td>
<td>CNSTRM</td>
<td>R*4</td>
<td>000006</td>
</tr>
<tr>
<td>ILENGT</td>
<td>I*2</td>
<td>000012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>R*4</td>
<td>$DATA</td>
<td>001440</td>
<td>001440 ( 400)</td>
<td>(200)</td>
</tr>
<tr>
<td>AC1</td>
<td>R*4</td>
<td>$DATA</td>
<td>010624</td>
<td>000014 ( 6)</td>
<td>(3)</td>
</tr>
<tr>
<td>CPOS</td>
<td>R*4</td>
<td>$DATA</td>
<td>007020</td>
<td>001440 ( 400)</td>
<td>(200)</td>
</tr>
<tr>
<td>H</td>
<td>R*4</td>
<td>$DATA</td>
<td>010460</td>
<td>000044 ( 18)</td>
<td>(9)</td>
</tr>
<tr>
<td>IS</td>
<td>I*2</td>
<td>$DATA</td>
<td>003100</td>
<td>000620 ( 200)</td>
<td>(200)</td>
</tr>
<tr>
<td>ISAS</td>
<td>I*2</td>
<td>$DATA</td>
<td>010640</td>
<td>000016 ( 7)</td>
<td>(7)</td>
</tr>
<tr>
<td>O</td>
<td>R*4</td>
<td>$DATA</td>
<td>010570</td>
<td>000034 ( 14)</td>
<td>(7)</td>
</tr>
<tr>
<td>POS</td>
<td>R*4</td>
<td>$DATA</td>
<td>000000</td>
<td>001440 ( 400)</td>
<td>(200)</td>
</tr>
<tr>
<td>V</td>
<td>R*4</td>
<td>$DATA</td>
<td>010524</td>
<td>000044 ( 18)</td>
<td>(9)</td>
</tr>
<tr>
<td>VEL</td>
<td>R*4</td>
<td>$DATA</td>
<td>005360</td>
<td>001440 ( 400)</td>
<td>(200)</td>
</tr>
<tr>
<td>VEL1</td>
<td>R*4</td>
<td>$DATA</td>
<td>003720</td>
<td>001440 ( 400)</td>
<td>(200)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>R*4</td>
<td>FILTER</td>
<td>R*4</td>
<td>FLOAT</td>
<td>R*4</td>
<td>INT</td>
<td>I*2</td>
<td>ISGN</td>
<td>I*2</td>
</tr>
<tr>
<td>MD</td>
<td>I*2</td>
<td>MOD</td>
<td>I*2</td>
<td>M2RD1</td>
<td>I*2</td>
<td>M2WR4</td>
<td>I*2</td>
<td>SQRRT</td>
<td>R*4</td>
</tr>
</tbody>
</table>
SUBROUTINE M2FFD2(I1)
C THIS SUBROUTINE CALLS THE PLOTTING ROUTINES
C BY MOHAMMAD-ALI MASSOUMNIA
C 28-MAR-83
C
0002 500 FORMAT(//' AT THIS POINT I CAN PLOT THE FOLLOWING:'//,
  * 1) NO GRAPHS AT ALL'//,
  * 2) RAW EYE POS. & SACCADIE IND. ON THE SAME GRAPH'//,
  * 3) SACCADIE IND. & PSUEDO ACC. ON THE SAME GRAPH'//,
  * 4) SLOW PHASE VELOCITY & RAW EYE POS. ON THE SAME GRAPH'//,
  * 5) RAW EYE POSITION & PSEUOD ACC. ON THE SAME GRAPH'//,
  * 6) RAW EYE POS. AND CUMM. SLOW PHASE POS. ON THE SAME GRAPH',
  * ENTER YOUR CHOICE: '$')
C
0003 505 FORMAT(I5)
C
0004 510 TYPE 500
0005 READ(5,505)IDUM
0006 IF(IDUM.LT.1.OR.IDUM.GT.6) GO TO 510
0008 GOTO(570,520,530,540,550,560)IDUM
0009 520 CALL M2P2(I1,13,4,1,400,5,200)
0010 GO TO 510
0011 530 CALL M2P2(I1,13,12,1,400,40,500)
0012 GO TO 510
0013 540 CALL M2P2(I1,11,4,1,400,5,200)
0014 GO TO 510
0015 550 CALL M2P2(I1,12,4,40,300,5,200)
0016 GO TO 510
0017 560 CALL M2P2(I1,14,4,1,500,5,200)
0018 GO TO 510
C
0019 570 RETURN
0020 END

FORTRAN IV Storage Map for Program Unit M2FFD2

Local Variables, $DATA, Size = 000004 ( 2, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDUM</td>
<td>I*2</td>
<td>000002</td>
<td>I1</td>
<td>I*2</td>
<td>@ 000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2P2</td>
<td>I*2</td>
</tr>
</tbody>
</table>
SUBROUTINE M2FID3(IUNIT)
C MOHAMMAD-ALI MASSOUMNIA
C REVISION 8-SEP-82
C THIS PROGRAM FINDS THE ONE POINT PER BEAT SLOW PHASE VELOCITY
C BY AVERAGING THE VELOCITY DURING THE SLOW PHASE AND ASSIGNING IT TO
C A TIME IN THE MIDDLE OF SLOW PHASE. (NOTE THAT THE POINTS
C WILL NOT BE EQUAL DISTANCE A PART.)
C
COMMON/DETECT/ ISAMPL,SW1,SW2,ILENGT
0003 IC1=1
0004 IC2=0
0005 IC3=0
0006 IC4=0
0007 IC5=1
0008 ISUM=0
0009 20 CALL M2RD2(11,13,IC1,IVEL1,IS1)
0010 IF(IC1.GE.ILENGT)GO TO 50
C IF NO FAST PHASE IN HALF SECOND INSERT AN ARTIFICIAL
C FAST PHASE
0012 IF(IC3.GE.ISAMPL/2)IS1=100
0014 IF(IS1.NE.0)GOTO 30
0016 ISUM=ISUM+IVEL1
0017 IC1=IC1+1
0018 IC3=IC3+1
0019 IC2=1
0020 GO TO 20
C
0021 30 IF(IC2.NE.1) GO TO 40
0023 ISUM=ISUM/IC3
0024 IC4=IC1-((IC3/2)
0025 WRITE(IUNIT,'IC4,ISUM
0026 IC5=IC5+1
0027 ISUM=0
0028 IC3=0
0029 IC2=0
0030 40 IC1=IC1+1
0031 GO TO 20
C
C WRITE AN END OF FILE INDICATOR
0032 50 IC4=-1
0033 ISUM=-1
0034 WRITE(IUNIT,'IC4,ISUM
C
0035 RETURN
0036 END
FORTRAN IV  Storage Map for Program Unit M2FFD3

Local Variables, .PSECT $DATA:, Size = 000022 ( 9. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
<td>I*2</td>
<td>000002</td>
<td>IC2</td>
<td>I*2</td>
<td>000004</td>
<td>IC3</td>
<td>I*2</td>
<td>000006</td>
</tr>
<tr>
<td>IC4</td>
<td>I*2</td>
<td>000010</td>
<td>IC5</td>
<td>I*2</td>
<td>000012</td>
<td>ISUM</td>
<td>I*2</td>
<td>000014</td>
</tr>
<tr>
<td>IS1</td>
<td>I*2</td>
<td>000020</td>
<td>IUNIT</td>
<td>I*2</td>
<td>000000</td>
<td>IVEL1</td>
<td>I*2</td>
<td>000016</td>
</tr>
</tbody>
</table>

COMMON Block /DETECT/, Size = 000014 ( 6. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAMPL</td>
<td>I*2</td>
<td>000000</td>
<td>SW1</td>
<td>R*4</td>
<td>000002</td>
<td>SW2</td>
<td>R*4</td>
<td>000006</td>
</tr>
<tr>
<td>ILENGT</td>
<td>I*2</td>
<td>000012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2RD2</td>
<td>I*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


SUBROUTINE FILTER(H1,Y,H)
C MOHAMMAD-ALI MASSOUMNIA
C JUN-15-82
C THIS SUBROUTINE FINDS A 7-PNT FIR LOW PASS FILTER
C AND CONVOLVES IT WITH THE H1(3-PNT FIR) AND RETURNS THE
C RESULTING CONVOLUTION AND LPF.
C
C H1 IS THE USR OF 3 POINT STIRLING DOUBLE DIFFERENTIATOR
C Y IS THE USR OF 9 POINT PAF
C H IS THE USR OF 7 POINT LOW PASS FILTER
C F2 IS THE CORNER FREQUENCY OF LOW PASS FILTER
C
COMMON/DETECT/ ISAMPL
DIMENSION H(7),H1(3),WK(7),Y(9)
ALPHA=3.
T=1./FLOAT(ISAMPL)
CALL BESSEL(ALPHA,OALPHA)
PI=3.141592
F2=10.
WC=2.*PI*F2

C FIND THE 47 POINT LONG KAISER WINDOW
C
DO 10 N=1,3
BETA=ALPHA*SQRT(1.-FLOAT(N)**2/9.)
CALL BESSEL(BETA,OBETA)
WK(N+4)=OBETA/OALPHA
WK(-N+4)=WK(N+4)
10 CONTINUE

C TRUNCATE THE USR OF LPF USING THE WINDOW
C
DO 20 I=1,7
N3=I-4
IF(N3.EQ.0)GO TO 20
H(I)=WK(I)*SIN(WC*N3*T)/(FLOAT(N3)*PI)
20 CONTINUE

WK(4)=1.
H(4)=2.*F2*T
SUM=0.
SUM1=0.

C NORMALIZE THE WINDOW AND THE LPF
C
DO 25 I=1,7
SUM1=SUM1+WK(I)
25 CONTINUE
SUM=SUM+H(I)

28 CONTINUE

C CONVOLVE THE LPF WITH THE THREE POINT DOUBLE DIFFERENTIATOR
C
0034  DO 30 N=1,9
0035    Y(N)=0.
0036  DO 40 K=1,N
0037     IF(K.GT.3) GO TO 30
0039     IF(N-K+1.GT.7) GO TO 40
0041    Y(N)=Y(N)+H1(K)*H(N-K+1)
0042  40  CONTINUE
0043  30  CONTINUE
0044    RETURN
0045   END

FORTRAN IV            Storage Map for Program Unit FILTER

Local Variables, .PSECT $DATA, Size = 000162 (  57. words)

Name  Type  Offset   Name  Type  Offset   Name  Type  Offset
ALPHA R*4  000044   BETA R*4  000116   F2   R*4  000104
I     I*2  000126   K     I*2  000142   N     I*2  000114
N3    I*2  000130   OALPHA R*4  000074   OBETA R*4  000122
PI    R*4  000100   SUM   R*4  000132   SUM1 R*4  000136
T     R*4  000070   WC    R*4  000110

COMMON Block /DETECT/, Size = 000002 (  1. words)

Name  Type  Offset   Name  Type  Offset   Name  Type  Offset
ISAMPL I*2  000000

Local and COMMON Arrays:

Name  Type  Section Offset --------Size-------- Dimensions
H     R*4  @ $DATA  000004  000034 ( 14,) (7)
H1    R*4  @ $DATA  000000  000014 (  6,) (3)
WK    R*4  @ $DATA  000006  000034 ( 14,) (7)
Y     R*4  @ $DATA  000002  000044 ( 18,) (9)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name  Type  Name  Type  Name  Type  Name  Type
BESSEL R*4  FLOAT R*4  SIN R*4  SQR T R*4
SUBROUTINE BESSEL(X1,Y1)

C MOHAMMAD-ALI MASSOUMNIA
C JUNE-12-1982
C THIS PROGRAM COMPUTES THE ZERO ORDER BESSEL FUNCTION
C
C X1 IS THE INPUT
C Y1 IS THE COMPUTED OUTPUT

N=1
Y1=1
DO 10 K=1,15
N=K*N
Y=((X1/2.)**K/N)**2
IF(Y.LT.1.E-5)GO TO 20
10 Y1=Y1+Y
20 RETURN
END
FUNCTION ISGN(X)
C THIS PROGRAM COMPUTES THE MODIFIED SIGN OF ITS ARGUMENT
C MOHAMMAD-ALI MASSOUMNIA
C SEP-2-1982
C
0002      ISGN=0
0003      IF(X.LT.-3.)ISGN=-1
0005      IF(X.GT.3.)ISGN=1
0007      RETURN
0008      END

FORTRAN IV    Storage Map for Program Unit ISGN

Local Variables, .PSECT $DATA, Size = 000004 ( 2, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISGN</td>
<td>I*2</td>
<td>000002</td>
<td>Eov X</td>
<td>R*4</td>
<td>@ 000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE M2RD1(IP,N1,N2)
C THIS SUBROUTINE READS A 256 WORDS LONG ARRAY
C BY MOHAMMAD-ALI MASOUMNIA
C APR-7-83
C
DIMENSION I(256)
N3=N1/256
N4=N1-N3*256
N5=N3+1
IF(N4.EQ.0)N4=256
IF(N4.EQ.1)READ(IP,N5,ERR=10)I
N2=I(N4)
10 RETURN
END

FORTRAN IV Storage Map for Program Unit M2RD1

Local Variables, .PSECT $DATA, Size = 001014 ( 262. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>I*2</td>
<td>000000</td>
<td>N1</td>
<td>I*2</td>
<td>000002</td>
<td>N2</td>
<td>I*2</td>
<td>000004</td>
</tr>
<tr>
<td>N3</td>
<td>I*2</td>
<td>001006</td>
<td>N4</td>
<td>I*2</td>
<td>001010</td>
<td>N5</td>
<td>I*2</td>
<td>001012</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>$DATA</td>
<td>000006</td>
<td>001000 (256)</td>
<td>(256)</td>
</tr>
</tbody>
</table>
SUBROUTINE M2RD2(IP1, IP2, N1, N2, N6)
C THIS SUBROUTINE READS TWO VARIABLES IN 256 WORDS LONG BLOCK
C BY MOHAMMAD-ALI MASSOUMNIA
C APR-7-83
C
COMMON/BUFFER/I1(256), I2(256)
C
DIMENSION I1(256), I2(256)
N3 = N1/256
N4 = N1 - N3*256
N5 = N3 + 1
IF(N4.EQ.0) N4 = 256
IF(N4.EQ.1) GO TO 10
N2 = I1(N4)
N6 = I2(N4)
RETURN
READ(IP1,'N5,ERR=20) I1
READ(IP2,'N5,ERR=20) I2
N2 = I1(N4)
N6 = I2(N4)
RETURN
END

FORTRAN IV Storage Map for Program Unit M2RD2
Local Variables, .PSECT $DATA, Size = 000020 ( 8. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP1</td>
<td>I*2</td>
<td>@ 000000</td>
<td>IP2</td>
<td>I*2</td>
<td>@ 000002</td>
<td>N1</td>
<td>I*2</td>
<td>@ 000004</td>
</tr>
<tr>
<td>N2</td>
<td>I*2</td>
<td>@ 000006</td>
<td>N3</td>
<td>I*2</td>
<td>000012</td>
<td>N4</td>
<td>I*2</td>
<td>000014</td>
</tr>
<tr>
<td>N5</td>
<td>I*2</td>
<td>000016</td>
<td>N6</td>
<td>I*2</td>
<td>@ 000010</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /BUFFER/, Size = 002000 ( 512. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I*2</td>
<td>000000</td>
<td>I2</td>
<td>I*2</td>
<td>001000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size----- Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I*2</td>
<td>BUFFER</td>
<td>000000</td>
<td>001000 ( 256.) (256)</td>
</tr>
<tr>
<td>I2</td>
<td>I*2</td>
<td>BUFFER</td>
<td>001000</td>
<td>001000 ( 256.) (256)</td>
</tr>
</tbody>
</table>
SUBROUTINE M2COPY(N1,N2,N3,N4)
C? THIS SUBROUTINE COPIES A FILE OVER ANOTHER FILE
C BY MOHAMMAD-ALI MASSOUMNIA
C APR-7-83
C
C N1 THE UNIT THAT IT SHOULD READ FROM
C N2 THE UNIT IT SHOULD WRITE TO
C N3 THE STARTING BLOCK
C N4 THE NUMBER OF BLOCKS TO COPY
C
COMMON/BUFFER/I(256)
DIMENSION I(256)
DO 20 J=1,N4
J3=J
N5=N3+J3-1
READ(N1,N5,ERR=30)I
WRITE(N2,J3,ERR=30)I
20 CONTINUE
30 RETURN
END
SUBROUTINE M2WR1(N1,I1,I2)
C THIS SUBROUTINE WRITES 1 VARIABLE IN 256 WORDS PER BLOCK
C BY MOHAMMAD-ALI MASSOUMNIA
C REVISED APR-7-83
C
0002    DIMENSION I3(256)
0003    I4=I1/256
0004    I5=I1-I4*256
0005    IF(I5.NE.0) GO TO 10
0007    I3(256)=I2
0008    WRITE(N1,'I4,ERR=20')I3
0009    RETURN
0010    10 I3(I5)=I2
0011    20 RETURN
0012    END

FORTRAN IV Storage Map for Program Unit M2WR1

Local Variables, .PSECT $DATA, Size = 001012 ( 261. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I*2</td>
<td>@ 000002</td>
<td>I2</td>
<td>I*2</td>
<td>@ 000004</td>
<td>I4</td>
<td>I*2</td>
<td>001006</td>
</tr>
<tr>
<td>I5</td>
<td>I*2</td>
<td>001010</td>
<td>N1</td>
<td>I*2</td>
<td>@ 000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>-----Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>I*2</td>
<td>$DATA</td>
<td>000006</td>
<td>001000 ( 256.)</td>
<td>(256)</td>
</tr>
</tbody>
</table>
SUBROUTINE M2WR4(J1,L1,L2,L3,L4)
  C THIS SUBROUTINE WRITE 4 VARIABLES IN 256 WORDS PER BLOCK
  C CONFIGURATION
  C BY MOHAMMAD-ALI MASSOUMNIA
  C REVISED APR-7-83

COMMON/BUFFER/J7(256),J8(256),J9(256),J10(256)
  C DIMENSION J7(256),J8(256),J9(256),J10(256)

J2=J1/256
J3=J1-J2*256
IF(J3,NE.0) G0 TO 10
J7(256)=L1
J8(256)=L2
J9(256)=L3
J10(256)=L4
WRITE(11',J2,ERR=20)J7
WRITE(12',J2,ERR=20)J8
WRITE(13',J2,ERR=20)J9
WRITE(14',J2,ERR=20)J10
RETURN
J7(J3)=L1
J8(J3)=L2
J9(J3)=L3
J10(J3)=L4
RETURN
END

---

**FORTRAN IV Storage Map for Program Unit M2WR4**

Local Variables, .PSECT $DATA, Size = 000016 ( 7. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>I*2</td>
<td>000000</td>
<td>J2</td>
<td>I*2</td>
<td>000012</td>
<td>J3</td>
<td>I*2</td>
<td>000014</td>
</tr>
<tr>
<td>L1</td>
<td>I*2</td>
<td>000002</td>
<td>L2</td>
<td>I*2</td>
<td>000004</td>
<td>L3</td>
<td>I*2</td>
<td>000006</td>
</tr>
<tr>
<td>L4</td>
<td>I*2</td>
<td>000010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COMMON Block /BUFFER/, Size = 004000 ( 1024. words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>J7</td>
<td>I*2</td>
<td>000000</td>
<td>J8</td>
<td>I*2</td>
<td>001000</td>
<td>J9</td>
<td>I*2</td>
<td>002000</td>
</tr>
<tr>
<td>J10</td>
<td>I*2</td>
<td>003000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>J10</td>
<td>I*2</td>
<td>BUFFER</td>
<td>003000</td>
<td>001000 ( 256.)</td>
<td>(256)</td>
</tr>
<tr>
<td>J7</td>
<td>I*2</td>
<td>BUFFER</td>
<td>000000</td>
<td>001000 ( 256.)</td>
<td>(256)</td>
</tr>
<tr>
<td>J8</td>
<td>I*2</td>
<td>BUFFER</td>
<td>001000</td>
<td>001000 ( 256.)</td>
<td>(256)</td>
</tr>
<tr>
<td>J9</td>
<td>I*2</td>
<td>BUFFER</td>
<td>002000</td>
<td>001000 ( 256.)</td>
<td>(256)</td>
</tr>
</tbody>
</table>
SUBROUTINE P3(I9,ICH0)
C THIS SUBROUTINE PLOTS THE ONE POINT PER BEAT C
C SLOW PHASE VELOCITY C
C BY MOHAMMAD-ALI MASSOUMIA C
C MAR-20-82 C
D
DIMENSION IBUF(1000)
CALL INIT(IBUF,1000)
CALL APNT(100.,100.)
CALL LVECT(0.,800.)
CALL APNT(100.,500.)
CALL LVECT(800.,0.)
DO 5 I1=1,9
CALL APNT(90.,I1*100.)
5 CONTINUE
DO 7 I1=1,9
CALL APNT(I1*100.,490.)
CALL LVECT(0.,20.)
7 CONTINUE
I=1
READ(I9'I,ERR=30)I3,I2
IF(I3.EQ.-1.AND.I2.EQ.-1) GO TO 20
IF(ICH0.EQ.2) GO TO 15
R1=FLOAT(I2*5+500)
GO TO 17
R1=ALOG(ABS(.001+I2))*50.+500.
R2=FLOAT(I3)/10.+100.5
CALL APNT(R2,R1)
I=I+1
GO TO 10
PAUSE
RETURN
END
### FORTRAN IV Storage Map for Program Unit P3

Local Variables, .PSECT $DATA, Size = 003766 (1019 words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I*2</td>
<td>003726</td>
<td>ICHO</td>
<td>I*2</td>
<td>@ 000002</td>
<td>I1</td>
<td>I*2</td>
<td>003724</td>
</tr>
<tr>
<td>I2</td>
<td>I*2</td>
<td>003732</td>
<td>I3</td>
<td>I*2</td>
<td>003730</td>
<td>I9</td>
<td>I*2</td>
<td>@ 000000</td>
</tr>
<tr>
<td>R1</td>
<td>R*4</td>
<td>003734</td>
<td>R2</td>
<td>R*4</td>
<td>003740</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF</td>
<td>I*2</td>
<td>$DATA</td>
<td>000004</td>
<td>003720 (1000,)</td>
<td>(1000)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>R*4</td>
<td>ALOG</td>
<td>R*4</td>
<td>APNT</td>
<td>R*4</td>
<td>FLOAT</td>
<td>R*4</td>
<td>INIT</td>
<td>I*2</td>
</tr>
<tr>
<td>LVECT</td>
<td>I*2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE M2P2(I4, I6, I7, I11, I12, I13, I14)
C THIS SUBROUTINE PlOTS TWO GRAPHS AT THE SAME TIME
C THE INPUT FILE IS READ IN 256 WORDS BLOCK
C BY MOHAMMAD-ALI MASSOUMnia
C APR-2-83
C
COMMON/BUFFER/I(256)
DIMENSION R(256), Q(256), IBUF(512)
I6=256
DO 20 I1=1, I4
I3=I1
READ(I6, I3, ERR=90) I
DO 10 I2=1, 256
R(I2)=FLOAT(I(I2)/I11+I12)
READ(I7, I3, ERR=90) I
DO 11 I2=1, 256
Q(I2)=FLOAT(I(I2)/I13+I14)
CALL INIT(IBUF, 512)
CALL YGRA(4, , R, I16)
CALL APNT(0, , 0, )
CALL YGRA(4, , Q, I16)
PAUSE
CALL INIT(IBUF, 512)
20 CONTINUE
90 RETURN
END
SUBROUTINE P2(I4, I6, I7, I11, I12, I13, I14)
C THIS PROGRAM PLOTS TWO GRAPH AT THE SAME TIME
C INPUT FILE IS READ WORD BY WORD
C BY MOHAMMAD-ALI MASSOUMIA
C MAR-10-82
C
DIMENSION R(256), Q(256), IBUF(512)
I16=256
I3=I4/256
DO 20 I1=0, I3
DO 10 I2=1, 256
I15=11*256+I2
IF(I15.GE.I4)GO TO 11
READ(I6, I15, ERR=90)I5
R(I2)=FLOAT(I5/I11+I12)
READ(I7, I15, ERR=90)I9
Q(I2)=FLOAT(I9/I13+I14)
CONTINUE
GO TO 12
I16=I2
CALL INIT(IBUF, 512)
CALL YGRA(4., R, I16)
CALL APNT(0., 0.,)
CALL YGRA(4., Q, I16)
PAUSE
CALL INIT(IBUF, 512)
CONTINUE
RETURN
END

FORTRAN IV Storage Map for Program Unit P2

Local Variables, .PSECT $DATA, Size = 006046 ( 1555, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>I*2</td>
<td>006026</td>
<td>I11</td>
<td>I*2</td>
<td>@ 000006</td>
<td>I12</td>
<td>I*2</td>
<td>@ 000010</td>
</tr>
<tr>
<td>I13</td>
<td>I*2</td>
<td>@ 000012</td>
<td>I14</td>
<td>I*2</td>
<td>@ 000014</td>
<td>I15</td>
<td>I*2</td>
<td>006032</td>
</tr>
<tr>
<td>I16</td>
<td>I*2</td>
<td>006022</td>
<td>I2</td>
<td>I*2</td>
<td>006030</td>
<td>I3</td>
<td>I*2</td>
<td>006024</td>
</tr>
<tr>
<td>I4</td>
<td>I*2</td>
<td>@ 000000</td>
<td>I5</td>
<td>I*2</td>
<td>006034</td>
<td>I6</td>
<td>I*2</td>
<td>@ 000002</td>
</tr>
<tr>
<td>I7</td>
<td>I*2</td>
<td>@ 000004</td>
<td>I9</td>
<td>I*2</td>
<td>006036</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size-----</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF</td>
<td>I*2</td>
<td>$DATA</td>
<td>004016</td>
<td>002000</td>
<td>(512,) (512)</td>
</tr>
<tr>
<td>Q</td>
<td>R*4</td>
<td>$DATA</td>
<td>002016</td>
<td>002000</td>
<td>(512,) (256)</td>
</tr>
<tr>
<td>R</td>
<td>R*4</td>
<td>$DATA</td>
<td>000016</td>
<td>002000</td>
<td>(512,) (256)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>APNT</td>
<td>R*4</td>
<td>FLOAT</td>
<td>R*4</td>
<td>INIT</td>
<td>I*2</td>
<td>YGRA</td>
<td>R*4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
C THIS PROGRAM PLOTS THE ONE POINT PER BEAT SLOW PHASE VELOCITY
C BY MOHAMMAD-ALI MASSOUMNIA
C JUN-18-82
0001 10 FORMAT(' ', 'THIS PROGRAM PLOTS THE SLOW PHASE VELOCITY ')
0002 15 FORMAT(' ', 'ENTER INPUT FILESPEC ')
0003 20 FORMAT(' ', 'DO YOU WANT MORE PLOT (0)NO,(1)YES: ')
0004 25 FORMAT(I3)
0005 27 FORMAT(' ', 'ORDINARY SCALE (1), LOG SCALE (2): ')
0006 30 TYPE 10
0007 30 TYPE 15
0008 CALL ASSIGN ('1', 'LYO2FP.EXT', '-1', 'RDO')
0009 DEFINE FILE 1 (5000, 2, U, NREC)
0010 TYPE 27
0011 READ(5, 25) IDUM
0012 CALL P3(1, IDUM)
0013 CLOSE(UNIT=1)
0014 TYPE 20
0015 READ(5, 25) IDUM
0016 IF(IDUM, EQ, 0) GO TO 40
0018 GO TO 30
0019 40 STOP
0020 END

FORTRAN IV Storage Map for Program Unit .MAIN.

Local Variables, .FSECT $DATA, Size = 000004 ( 2, words)

Name Type Offset Name Type Offset Name Type Offset
IDUM I*2 000002 NREC I*2 000000 Eav

Subroutines, Functions, Statement and Processor-Defined Functions:

Name Type Name Type Name Type Name Type
ASSIGN R*4 P3 R*4
0001 PROGRAM M2PLOT
C THIS PROGRAM PLOTS EYEMOVEMENT AND FAST PHASE INDICATOR
C ON THE SAME GRAPH. IT IS USED FOR PERFORMANCE EVALUATION
C BY MOHAMMAD-ALI MASSOUMNIA
C 1-MAY-83
0002 COMMON/BUFFER/J(256)
0003 LOGICAL*1 NAME(10),NAME1(3)
0004 DATA NAME1/'S','A','C'/
0005 10 FORMAT(1I4)
0006 20 FORMAT(' ENTER FILE SPECIFICATION: ')$
0007 25 FORMAT(I5)
0008 5 FORMAT(' ENTER START BLOCK & ENDING BLOCK: ',2I7)
0009 27 TYPE 20
0010 READ(5,10)NAME
0011 READ(5,5)IS,IE
0012 CALL ASSIGN(1,NAME,10,'RDO')
0013 DEFINE FILE 1 (IE,256,U,NREC)
0014 I3=IE-IS+1
0015 CALL ASSIGN (3,'LY,E0G',5,'SCR')
0016 DEFINE FILE 3 (I3,256,U,NREC)
0017 DO 35 I=IS,IE
0018 I1=I
0019 I2=I-IS+1
0020 READ(1*I1)J
0021 WRITE(3*I2)J
0022 35 CONTINUE
0023 CLOSE(UNIT=1)
C
0024 DO 30 I=8,10
0025 30 NAME(I)=NAME1(I-7)
0026 CALL ASSIGN (2,NAME,10,'RDO')
0027 DEFINE FILE 2 (60,256,U,NREC)
0028 CALL M2P2(I3,2,3,1,400,4,200)
0029 CLOSE(UNIT=3)
0030 CLOSE(UNIT=2)
0031 TYPE 40
0032 40 FORMAT(' DO YOU WANT MORE PLOT NO(0),YES(1)? '$)
0033 READ(5,25)IDUM
0034 IF(IDUM.EQ.0)GO TO 50
0036 GO TO 27
0037 50 STOP
0038 END
.TYPE RKO:M2EVA1
FORTRAN IV V02.5-2 Fri 06-May-83 13:16:14

0001 DIMENSION Q(256),R(256),JN(256),IBUF(10000)
0002 LOGICAL*1 NAME(10),NAME1(6)
0003 DATA NAME1/’S’,’A’,’C’,’S’,’P’,’1’/
0004 I1=1
0005 100 FORMAT(’ENTER FILE SPECIFICATION: ’)$
0006 110 FORMAT(11A1)
0007 112 FORMAT(’ENTER BIAS VALUE: ’)$
0008 114 FORMAT(I7)
0009 115 FORMAT(X,11A1)
0010 116 FORMAT(’ENTER THE # OF BLOCKS TO BE SHOWN: ’)$
0011 117 FORMAT(’BLOCK #:’)$

C
0012 TYPE 100
0013 READ(5,110)NAME
0014 CALL ASSIGN(1,NAME,10,’RDO’)
0015 DEFINE FILE 1 (60,256,U,NREC)

C
0016 TYPE 112
0017 READ(5,114)I12

C
0018 TYPE 116
0019 READ(5,114)I91

C
0020 DO 120 I=1,3
0021 NAME(I+7)=NAME1(I)
0022 120 CONTINUE
0023 CALL ASSIGN(2,NAME,10,’RDO’)
0024 DEFINE FILE 2 (60,256,U,NREC)

C
0025 DO 130 I=4,6
0026 NAME(I+4)=NAME1(I)
0027 130 CONTINUE
0028 CALL ASSIGN(3,NAME,10,’RDO’)
0029 DEFINE FILE 3 (1000,2,U,NREC)

C
0030 DO 20 I1=1,I91
0031 CALL INIT(IBUF,10000)
0032 WRITE(6,117)I1
0033 DO 1 I=0,8
0034 CALL APNT(100.+86.*I,100.)
0035 CALL LVECT(0.,800.)
0036 1 CONTINUE

C
0037 DO 2 I=1,9
0038 C=100.
0039 IF(I.EQ.5)C=0.
0040 CALL APNT(C,100.*I)
0041 CALL LVECT(868.-C,0.)
0042 2 CONTINUE

C
0043 I6=(I1-1)*256
0045 3 READ(3,’J1,ERR=90)I4,I2
0046 IF(I4,GT,I1*256) GO TO 5
0047 IF(I4,LE,-1.AND,I2,LE,-1) GO TO 5
FORTRAN IV

V02.5-2  Fri 06-May-83 13:16:14

0050  R1=FLOAT(I2*5+500)
0051    I5=I4-I6
0052  R2=FLOAT(I5)*3.1+100.
0053    CALL APNT(R2-5.,R1)
0054    CALL LVECT(10.,0.)
0055    CALL APNT(R2,R1-5.)
0056    CALL LVECT(0.,10.)
0057    J1=J1+1
0058    GO TO 3
0059  5    I3=I1
0060    CALL APNT(100.,0.)
0061    I16=256
0062    READ(1'I3,ERR=90)JN
0063    DO 10 I2=1,256
0064  10   R(I2)=FLOAT(JN(I2)-I12)*.25+500.
0065    CALL YGRA(3.,R,I16)
0066    CALL APNT(100.,0.)
0067    READ(2'I3,ERR=90)JN
0068    DO 11 I2=1,256
0069  11   Q(I2)=JN(I2)*.8+500.
0070    CALL YGRA(3.,Q,I16)
0071    PAUSE
0072  20   CONTINUE
0073  90    STOP
0074    END

FORTRAN IV

Storage Map for Program Unit .MAIN.

Local Variables, $PSECT $DATA, Size = 054176 (11327, words)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>R*4</td>
<td>054114</td>
<td>I</td>
<td>I*2</td>
<td>054110</td>
<td>I1</td>
<td>I*2</td>
<td>054112</td>
</tr>
<tr>
<td>I12</td>
<td>I*2</td>
<td>054104</td>
<td>I16</td>
<td>I*2'</td>
<td>054142</td>
<td>I2</td>
<td>I*2</td>
<td>054124</td>
</tr>
<tr>
<td>I3</td>
<td>I*2</td>
<td>054140</td>
<td>I4</td>
<td>I*2</td>
<td>054122</td>
<td>I5</td>
<td>I*2</td>
<td>054132</td>
</tr>
<tr>
<td>I6</td>
<td>I*2</td>
<td>054120</td>
<td>I91</td>
<td>I*2</td>
<td>054106</td>
<td>J1</td>
<td>I*2</td>
<td>054100</td>
</tr>
<tr>
<td>NREC</td>
<td>I*2</td>
<td>054102</td>
<td>Eqv</td>
<td>R1</td>
<td>054126</td>
<td>R2</td>
<td>R*4</td>
<td>054134</td>
</tr>
</tbody>
</table>

Local and COMMON Arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Section</th>
<th>Offset</th>
<th>------Size------</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF</td>
<td>I*2</td>
<td>$DATA</td>
<td>005000</td>
<td>047040 (10000.)</td>
<td>(10000)</td>
</tr>
<tr>
<td>JN</td>
<td>I*2</td>
<td>$DATA</td>
<td>004000</td>
<td>001000 ( 256.)</td>
<td>( 256)</td>
</tr>
<tr>
<td>NAME</td>
<td>L*1</td>
<td>$DATA</td>
<td>054040</td>
<td>000012 (   5.)</td>
<td>(   5)</td>
</tr>
<tr>
<td>NAME1</td>
<td>L*1</td>
<td>$DATA</td>
<td>054052</td>
<td>000006 (    3.)</td>
<td>(    3)</td>
</tr>
<tr>
<td>Q</td>
<td>R*4</td>
<td>$DATA</td>
<td>000000</td>
<td>002000 ( 512.)</td>
<td>( 512)</td>
</tr>
<tr>
<td>R</td>
<td>R*4</td>
<td>$DATA</td>
<td>002000</td>
<td>002000 ( 512.)</td>
<td>( 512)</td>
</tr>
</tbody>
</table>

Subroutines, Functions, Statement and Processor-Defined Functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>APNT</td>
<td>R*4</td>
<td>ASSIGN</td>
<td>R*4</td>
<td>FLOAT</td>
<td>R*4</td>
<td>INIT</td>
<td>I*2</td>
<td>LVECT</td>
<td>I*2</td>
</tr>
</tbody>
</table>
REFERENCES


[9] Gentiles W.; Application of automated techniques to the study of vestibular function in Man, Ph.D. thesis Department of Electrical Engineering, University of Toronto, 1974


[12] Barnes G.R.; A procedure for the analysis of nystagmus and other eye movements; Aviation, space, and enviromental Medicine July 1982


[16] Rabiner l. & Gold B.; Theory and application of digital signal processing Prentice-Hall

[17] Van trees, H; Detection, Estimation, and Demodulation, Wiley