

19994
P-191

CHARACTERIZATION OF SLOW AND FAST PHASE NYSTAGMUS

(NASA Research Grant NAG 9-303)

Charles S. Lessard, Ph.D.

Carlos A. Rodriguez-Garcia, M.Eng.

Wing Chan Wong, Ph.D.

Jae J. Im, M.S.

Glenn F. Schmidt, M.S.

1 January 1990 through 28 June 1991

Bioengineering Program, Industrial Engineering Department,
Rm. 233, Zachry Engineering Center, Texas A&M University
College Station, TX 77843-3120

(NARS-CO-199400) CHARACTERIZATION OF SLOW
AND FAST PHASE NYSTAGMUS Final Technical
Report, 1 Jan. 1990 - 28 Jun. 1991 (Texas
A&M Univ.) 201 p. CSCL 06P

N91-25589

Unclass
0019994

03/92

ABSTRACT

A current literature review of the analog and digital processes of vestibular and optical kinetic nystagmus reveals little agreement in the methods used by various laboratories. The strategies for detection of saccade (fast phase velocity component of nystagmus) vary between laboratories, and most of the processes have not been evaluated and validated with a standard database. The purpose of this study is to survey major vestibular laboratories in the U.S. that perform computer analyses of vestibular and optokinetic reflexes to stimuli, and to establish a baseline from which to standardize data acquisition and analysis programs. The vestibular laboratories that cooperated in the evaluation were:

1. UCLA, Los Angeles, CA
2. MIT, Space Laboratory, Cambridge, MA
3. Pittsburgh, PA - Harvard, Cambridge, MA
4. USAF/SAM, Brooks AFB, San Antonio, TX
5. SBRI, JSC (NASA), Houston, TX

The concept of an Error Index was employed as the criterium for evaluating the performance of the vestibular analysis software programs. The performance criterium is based on the detection of saccades and is the average of the percentages of missed detections and false detections. Evaluation of the programs produced results for lateral gaze with saccadic amplitude of one (1), two (2), three (3), five (5) and ten (10) degrees with various signal-to-noise ratios. In addition, results were obtained for sinusoidal pursuit of 0.05 Hz, 0.10 Hz and 0.50 Hz with saccades from one (1) to ten (10) degrees at various signal-to-noise ratios. Selection of the best program was made from the performance in the lateral gaze with three (3) degrees of saccadic amplitude and in the 0.10 Hz sinusoid with three (3) degrees of saccadic amplitude. The FPID program written for SBRI, NASA and the UCLA SINUXEC were the best overall. The FPID program produced errors less than 5% with signal-to-noise ratios of 2.5 or more, and the SINUXEC program produced errors less than 5% error at signal-to-noise ratios of 4.0 or more.

TABLE OF CONTENTS

	<u>page</u>
Abstract	i
Table of Contents	ii
List of Tables	iii
List of Figures	v
Chapter	
I Introduction	1
II Background	4
III Evaluation Criteria	6
IV Electro-oculographic Database	8
Synthetic Data Generation	8
Human Subject Data	10
V Description of Programs and Results	12
US Air Force Nystagmus Analysis Program	12
UCLA Nystagmus Analysis Programs	20
SINUXEC	21
CALORXEC	26
MIT Nystagmus Analysis Program (M2MI86)	29
Harvard Nystagmus Analysis Program	36
NASA Nystagmus Analysis Program (FPID)	43
VI Discussion - Comparison of Results	52
VII Conclusions	56
VIII Recommendations	57
IX References	59
Appendix A: User's Manual for NASDAT.FOR - Synthetic Data Generation Program	
Appendix B: User's Manual for FPID.FOR and FPID.C - Fast Phase Identification and Slow Phase Reconstruction Program	
Appendix C: Program Listing: NASDAT.FOR	
Appendix D: Program Listing: FPID.C	
Appendix E: Program Listing: FPID.FOR	

List of Tables

<u>Table</u>		<u>Page</u>
I	Velocity of slow and fast phases as a function of nystagmus amplitude for simulated data 1/10 data point ratio	10
II	Results from the analysis of human optokinetic data by USAF/SAM Program ..	19
III	Results from the analysis of human sinusoidal pursuit data by USAF/SAM Program	19
IV	Results from the analysis of human pseudorandom data by USAF/SAM Program	20
V	Results from the analysis of human optokinetic data by SINUXEC Program ...	25
VI	Results from the analysis of human sinusoidal pursuit data by SINUXEC Program	25
VII	Results from the analysis of human pseudorandom data by SINUXEC Program	26
VIII	List of the presetting parameters for the Harvard Program	41
IX	Results from the analysis of human optokinetic data by Harvard Program	41
X	Results from analysis of human sinusoidal pursuit data by Harvard Program ...	42
XI	Results from analysis of human pseudorandom data by Harvard Program	42
XII	Amplitude of saccades in the optokinetic data set calculated manually and by FPID Program	44
XIII	Noise amplitude and SNR for artificial data with one (1) degree saccades	47
XIV	Results from analysis of human optokinetic data by FPID Program	49
XV	Results from the analysis of human sinusoidal pursuit data by FPID Program ..	50
XVI	Results from the analysis of human pseudorandom data by FPID Program	50
XVII	Comparison of the Program Performance with Subject Data	55
A1	Maximum amplitude of noise and corresponding data files	
A2	Prompts and responses for generating midposition gaze data with one (1) degree nystagmus without noise.	
A3	Prompts and responses required for generating midposition gaze data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise.	

Table

- A4 Prompts and responses required for generating sinusoidal pursuit data with one (1) degree nystagmus without noise.
- A5 Prompts and responses required for generating sinusoidal pursuit data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise.

List of Figures

Figure

- 1A Flowchart of NASDAT Program.
- 1B Flowchart of NASDAT Program (continued).
- 1C Flowchart of NASDAT Program (continued).
- 2 Sample synthetic Midposition Gaze data with five (5) degrees of saccadic jump (without noise).
- 3 Sample synthetic 0.5 Hz sinusoidal pursuit data with five (5) degrees of saccadic jump (without noise).
- 4A Plot of optokinetic stimulus.
- 4B Plot of human optokinetic EOG data set.
- 5A Plot of 0.10 Hz sinusoidal pursuit stimulus.
- 5B Plot of human sinusoidal pursuit EOG data set.
- 6A Plot of pseudorandom stimulus.
- 6B Plot of human pseudorandom EOG data set.
- 7 Overall flowchart of US Air Force Program.
- 8A Flowchart of calibration section of US Air Force Program.
- 8B Flowchart of calibration section of US Air Force Program (continued).
- 9A Flowchart of analysis section of US Air Force Program.
- 9B Flowchart of analysis section of US Air Force Program (continued).
- 10 Plot of individual errors by USAF Program for midposition gaze data (maximum SNR of 25).
- 11 Plot of individual errors by USAF Program in midposition gaze data (maximum SNR of 200).
- 12 Plot of Error Index by USAF Program for midposition gaze data.
- 13 Plot of individual errors by USAF Program for sinusoidal pursuit data with two (2) degrees of saccadic jump.
- 14 Plot of individual errors by USAF Program for sinusoidal pursuit data with five (5) degrees of saccadic jump.

Figure

- 15 Plot of Error Index by USAF Program for 0.05 Hz sinusoidal pursuit data.
- 16 Plot of Error Index by USAF Program for 0.10 Hz sinusoidal pursuit data.
- 17 Plot of Error Index by USAF Program for 0.50 Hz sinusoidal pursuit data.
- 18 Plot of Error Index by USAF Program for 0.80 Hz sinusoidal pursuit data.
- 19A Flowchart of UCLA SINUXEC Program.
- 19B Flowchart of UCLA SINUXEC Program (continued).
- 20 Plot of individual errors and Error Index by SINUXEC Program for midposition gaze data with one (1) degree of saccadic jump.
- 21 Plot of Error Index by SINUXEC Program for midposition gaze data.
- 22 Plot of individual errors by SINUXEC Program for midposition gaze data.
- 23 Plot of Error Index by SINUXEC Program for 0.50 Hz sinusoidal pursuit data.
- 24 Plot of Error Index by SINUXEC Program for 0.05 Hz sinusoidal pursuit data.
- 25 Plot of Error Index by SINUXEC Program for 0.10 Hz sinusoidal pursuit data.
- 26 Plot of individual errors by SINUXEC Program for 0.10 Hz sinusoidal pursuit data.
- 27A Flowchart of UCLA CALORXEC Program.
- 27B Flowchart of UCLA CALORXEC Program (continued).
- 28 Plot of Error Index by CALORXEC Program for midposition gaze data.
- 29 Plot of individual errors by CALORXEC Program for midposition gaze data.
- 30 Plot of individual errors by CALORXEC Program for sinusoidal pursuit data with 1:5 and 1:15 data point ratios.
- 31 Plot of Error Index by CALORXEC Program for sinusoidal pursuit data with 1:5 and 1:15 data point ratios.
- 32A Flowchart of MIT M2MI86 Program.
- 32B Flowchart of MIT M2MI86 Program (continued).
- 33 Plot of individual errors and Error Index by MIT M2MI86 Program for midposition gaze data with one (1) degree of saccadic jump.
- 34 Plot of individual errors by MIT M2MI86 Program for midposition gaze data.

Figure

- 35 Plot of Error Index by MIT M2MI86 Program for midposition gaze data.
- 36 Plot of individual errors by MIT M2MI86 Program for 0.50 Hz sinusoidal pursuit data.
- 37 Plot of Error Index by MIT M2MI86 Program for 0.05 Hz sinusoidal pursuit data.
- 38 Plot of Error Index by MIT M2MI86 Program for 0.10 Hz sinusoidal pursuit data.
- 39 Plot of Error Index by MIT M2MI86 Program for 0.50 Hz sinusoidal pursuit data.
- 40 Plot of output for optokinetic data analysis by MIT M2MI86 Program.
- 41 Plot of individual errors by MIT M2MI86 Program for 0.10 Hz sinusoidal pursuit data.
- 42A Flowchart of Harvard Program.
- 42B Flowchart of Harvard Program (continued).
- 43 Plot of Error Index by Harvard Program for midposition gaze data.
- 44 Plot of individual errors by Harvard Program for midposition gaze data.
- 45 Plot of Error Index by Harvard Program for 0.05 Hz sinusoidal pursuit data.
- 46 Plot of Error Index by Harvard Program for 0.10 Hz sinusoidal pursuit data.
- 47 Plot of Error Index by Harvard Program for 0.50 Hz sinusoidal pursuit data.
- 48 Plot of individual errors by Harvard Program for 0.50 Hz sinusoidal pursuit data.
- 49A Flowchart of FPID Program.
- 49B Flowchart of FPID Program (continued).
- 50 Plot of individual errors and Error Index by FPID Program for midposition gaze data with one (1) degree of saccadic jump.
- 51 Plot of Error Index by FPID Program for midposition gaze data.
- 52 Plot of individual errors by FPID Program for midposition gaze data.
- 53 Plot of Error Index by FPID Program for 0.50 Hz sinusoidal pursuit data.
- 54 Plot of Error Index by FPID Program for 0.05 Hz sinusoidal pursuit data.
- 55 Plot of Error Index by FPID Program for 0.10 Hz sinusoidal pursuit data.

Figure

- 56 Plot of noise on the human sinusoidal pursuit data set.
- 57 Plot of Error Index by five (5) of the programs (USAF/SAM, UCLA SINUXEC, MIT M2MI86, Harvard and NASA FPID) for midposition gaze data with three (3) degrees of saccadic jump.
- 58 Plot of Error Index by five (5) of the programs (USAF/SAM, UCLA SINUXEC, MIT M2MI86, Harvard and NASA FPID) for 0.10 Hz sinusoidal pursuit data with three (3) degrees of saccadic jump.
- A1 Plot of simulated midposition gaze data with five (5) degrees of saccadic jump without noise.
- A2 Plot of simulated midposition gaze data with five (5) degrees of saccadic jump with 0.5 degree (maximum amplitude) noise.
- A3 Plot of simulated sinusoidal pursuit data with five (5) degrees of saccadic jump without noise.
- A4 Plot of simulated sinusoidal pursuit data with five (5) degrees of saccadic jump with 0.5 degree (maximum amplitude) noise.

CHAPTER I

INTRODUCTION

Although vestibular function tests have been employed for over eighty years, accepted formal standards for manual and computerized clinical evaluation of vestibular dysfunction have yet to be established in the United States. Without standards, vestibular clinics and laboratories perform similar tests with different data processing techniques [1]; therefore, test results from the various vestibular research laboratories are not directly comparable, thereby, hindering the overall process of vestibular dysfunction research. Detailed measurements of nystagmic movements are essential for the evaluation of vestibular and visual oculomotor reflexes [2], and standardization of vestibular function testing and data processing is necessary in order to unify clinical and laboratory vestibular function investigations in the United States.

With the increased usage of computers and digital signal processing techniques, many clinics and research laboratories record and analyze eye movement data with individually tailored software. The objective of this part of the study, entitled "Characterization of Slow and Fast Phase Nystagmus" (NASA Grant NAG 9-303, sponsored by the Space Biomedical Research Institute (SBRI), Space and Life Science Division, Johnson Space Center (JSC), National Aeronautics and Space Administration), is twofold:

1. to evaluate and compare computer programs used in the United States (programs which quantify, characterize, and analyze nystagmus components, *i.e.*, slow phase velocity and saccades); and
2. to develop an automated nystagmus analysis program for NASA.

The scope of this report involves the evaluation of various computer programs used in the U.S. and their ability to analyze simulated and real electro-oculogram (EOG) data and completes Tasks 03 and 04 of the statement of work. Part I, "Survey of Vestibular Laboratories: Baseline for Standardization," which completed Task 01 and Part II, "Vestibular Function Tests

Standardization Proposal," which completed Task 02, were submitted in 1988 and 1989, respectively. Copies of the Part II report were sent to the CHABA, Office of the National Research Council (NRC) and used in the final NRC Standard for Vestibular Dysfunction.

Evaluation of the various computer algorithms (Part III of the study) was divided into four efforts:

1. obtaining listings of programs currently used by vestibular laboratories in the United States,
2. creating a representative "data base" of ocular movement responses to various visual stimuli,
3. rewriting the various computer programs in FORTRAN 77 for evaluation on a VAX system, and
4. developing the criteria for evaluation and, subsequently, evaluating the various programs.

Vestibular laboratories in the United States, willing to cooperate and participate in the software evaluation phase, are: School of Aerospace Medicine (USAF/SAM), Brooks AFB, San Antonio, TX; UCLA Medical School, Los Angeles, CA; Space Research Laboratory, MIT, Cambridge, MA; Dr. C. Wall's Vestibular Laboratory, Harvard University, Cambridge, MA (C. Wall's program was obtained while he was at the Vestibular Laboratory, Pittsburgh, PA); and SBRI Vestibular Laboratory, JSC (NASA), Houston, TX. For the study conducted at Texas A&M University, these laboratories provided listings of the nystagmus analysis algorithms, along with permission to evaluate their algorithms.

A simulated (synthesized) nystagmus data base was developed and an experimental human subject nystagmus data base was acquired at the physiological signal processing facility, Texas A&M University, College Station, TX. The various nystagmus analysis algorithms, which perform nystagmus identification and characterization, were rewritten in FORTRAN 77 for use with the VAX computer system at Texas A&M and were evaluated on their accuracy in correctly detecting the occurrence of saccadic movements with the simulated and human EOG data sets. Each program was tested with the same sets of artificially generated EOG simulated

data.

The synthetic data sets simulate different types of eye movement (*e.g.*, midposition gaze or sinusoidal) with and without superimposed nystagmus and random magnitude noise. The data sets generated with superimposed nystagmus have adjustable nystagmus amplitude and fast and slow phase velocities. Random noise is included with some data sets to test the accuracy of the programs in the presence of extraneous events. The intent is to simulate real data acquisition conditions, under which the signal-to-noise ratio (SNR) varies with recording conditions [3]. An Error Index is calculated for each data set tested to evaluate and compare the accuracy of the various programs under different recording conditions. Results from this evaluation logically lead to the second aspect of this study: to develop an automated or interactive ocular nystagmus analysis program for NASA.

In the second half of the study, the best program method or best parts of programs from the participating vestibular laboratories will be selected and modified for application to the SBRI. The Fast Phase Identification (FPID) nystagmus analysis program, which was originally written in FORTRAN by Dr. Charles S. Lessard [4] for use with the SBRI's LSI-11 system, automatically identifies fast phase nystagmus, reconstructs slow phase position and velocity waveforms and computes magnitude and phase of the response to sinusoidal stimulus.

The program FPID described in this report is a modified version of an original FORTRAN program developed for the DEC LSI-11 system. The code was modified to extend its use for personal computers, and the modified FPID is composed of two versions to run on different systems:

1. FPID.FOR: written in FORTRAN 77, for use with VAX, IBM mainframes.
2. FPID.C: written in C (Microsoft v.5.1), for use with IBM personal computers (PC's) or compatible personal computers.

These two different versions of the FPID program should meet the data analysis needs of most clinics or laboratories regardless of the data acquisition and computer system in use.

CHAPTER II

BACKGROUND

Evaluation of vestibular dysfunction is performed by analyzing a patient's eye movements during various test conditions. At the present, electronic methods are most commonly used in vestibular clinics and laboratories for measuring eye movements.

An eye movement measuring system consists of four subsystems: stimulation, detection, signal conditioning, and recording. To elicit eye movements, the stimulation system presents one or more stimuli provided by an array of LED's placed in front of the patient in both the horizontal and vertical directions. The array may be used to present step changes in position or smooth pursuit (sinusoidal) stimuli.

The resulting eye movement is detected by an eye movement detection system, of which the most popular is electro-oculography. The electro-oculography method for acquiring ocular movements relies on the corneo-retinal potential oriented along the long axis of the eye. Consequently, an electrode placed in the vicinity of the eye becomes more positive as the eye rotates toward the electrode and less positive when the eye rotates in the opposite direction [5]. The eye movement signals detected by EOG are small magnitude signals of about $20 \mu\text{V}/\text{degree}$ that require amplification gains of 1000 and lowpass filtering below 60 Hz before recording or displaying. Lowpass filters are used to eliminate high frequency noise due to the high gain of the system while highpass filters eliminate the effects of baseline drift caused by electrode capacitance changes.

Specialized software is often used to analyze eye position data in clinics and research laboratories; therefore, following the signal conditioning circuitry, analog to digital (A/D) conversions are necessary to digitally analyze the eye position data. The accuracy of the data analysis software is affected by the signal conditioning circuitry.

Vestibular dysfunction is characterized by abnormal eye movements commonly referred to

as nystagmus. Nystagmus is separated into two components or phases: fast and slow. The following terms are important in the clinical description of nystagmus [6]:

1. Beat mode
2. Conjugate or disconjugate eye movements
3. Plane of eye movement
4. Direction of fast and slow phases
5. Amplitude
6. Beat frequency
7. Velocity of the fast and slow phases
8. Changes in direction or amplitude
9. Influences of head position or changes of head position.

The parameters most commonly measured and analyzed by software are: direction, duration, peak velocity, average velocity, and beat frequency. Several programs exist for analyzing these parameters; however, each program uses its own method of feature extraction and parameter measurement. Each algorithm uses different criteria for fast and slow phase detection, velocity and duration estimation. Additionally, each algorithm is affected differently by extraneous events such as eye blinks and noise. The quantification of the nystagmus parameters are commonly used to locate and diagnose the cause of vestibular dysfunction. Identification of significant patterns in the eye movement could provide insight about the mechanism involved in the production of nystagmus, as well as the formulation of comprehensive models for nystagmus induced by a variety of stimuli [2]. The consistency and symmetry of saccade accuracy in normal subjects suggests that saccade accuracy measurements can be a sensitive clinical test of the oculomotor control system [7]. It is necessary, therefore, that the parameters be measured accurately and that these parameters be common for comparison. This approach should help standardize vestibular dysfunction testing analysis and facilitate direct comparison between laboratories.

CHAPTER III

EVALUATION CRITERIA

Performance in the detection of saccades formed the basis for development of a criteria by which to quantitatively evaluate the various nystagmus programs. The concept of an error index is employed as the criteria for evaluating the best performing vestibular analysis software program. Two error measures were used to evaluate the detection of saccades. The performance criterium consists of a weighted average of two classes of errors: the error of missed detection of a true saccade, and the error of false detection (detection of an event when a true saccade did not occur).

The method of evaluating the error in detection of a saccade consists of running the test data set through one of the analysis programs, *i.e.*, UCLA, USAF, FPID, etc., and comparing when the program detects a fast phase component (during which samples) to when the fast phase (true saccade) really occurs in the test data. If the sample indices of the detected event were within the sample index interval (± 2 samples or ± 10 msec at 200 Hz sampling rate) of true saccade, the program was credited with correct detection of the event.

Missed-detection error (E1) is calculated by subtracting the number of correctly detected fast phase components (saccades) from the total number of known true saccades in the record (Equation 1).

$$E1 = \text{Total number of true saccades} - \text{Number of correctly detected saccades} \quad (1)$$

The percent missed-detection error is obtained by dividing the missed-detection error (E1) by the total number of known true saccades in the test data and multiplying the quotient by 100 (Equation 2).

$$E1\% = \left(\frac{E1}{\text{total number of known true saccades}} \right) (100) \quad (2)$$

The false-detection error (E2) is calculated by subtracting the number of correctly detected saccades from the total number of events detected by the program (Equation 3).

$$E2 = \text{Number of events detected by program} - \text{Number of correctly detected saccades} \quad (3)$$

The percent false-detection error is obtained by dividing the false-detection error (E2) by the total number of events detected by the program and multiplying the quotient by 100 (Equation 4).

$$E2\% = \left(\frac{E2}{\text{total number of detected saccades}} \right) (100) \quad (4)$$

The Error Index (in percentage) is the average value of the missed-detection error and false-detection error percentages (Equation 5).

$$\text{Error Index \%} = \frac{E1\% + E2\%}{2} \quad (5)$$

CHAPTER IV

ELECTRO-OCULOGRAPHIC DATABASE

Simulated Data Generation

The program, NASDAT, was developed to generate simulated midposition gaze and sinusoidal pursuit EOG data with which to evaluate the various nystagmus analysis computer programs. Fifteen seconds of artificial EOG data were generated for each test run and were stored in files to be read by each nystagmus analysis program. Figure 1 is a flowchart of the test data generation program, NASDAT.

At the beginning of the NASDAT program (Fig. 1), the operator has the option of generating either a simulated midposition gaze EOG data set or a simulated sinusoidal pursuit EOG data set. The program is initialized by entering values for several nystagmus parameters, (*i.e.*, sampling rate, magnitude of saccades, direction of saccades, velocities, thresholds). The sampling rate (SR) is necessary to determine the time interval between data points. The velocity of the nystagmus components may be entered or calculated by dividing the product of the sampling rate and the magnitude of the saccadic jump by the number of points in the respective nystagmus component. For example, in calculation of the fast phase velocity (FPV) component, the denominator (#PTS) is the number of data points in the fast phase component of nystagmus; to determine the slow phase velocity (SPV), the denominator (#PTS) denotes the number of data points in the slow phase component of nystagmus.

Additionally, the data point ratio (DPR) may be entered or obtained by calculating the ratio between the number of fast phase data points and the number of slow phase data points. Simulated EOG data were generated with data point ratios of 1:5, 1:8, 1:10, 1:12, and 1:15.

The initialization subroutine requires data-specific parameters before simulated nystagmus data is generated (e.g., slow phase velocity). An example of the NASDAT program output is presented in Figure 2, where the simulated midposition gaze EOG data without noise resembles

a sawtooth waveform. Note that all fast phase events (saccades) are of equal magnitude and in the same direction. For the simulated sinusoidal pursuit EOG data, initial direction, frequency, and maximum amplitude of the base sinusoid waveform must be entered, as well as the data point ratio (DPR) and the slope difference threshold. The data point ratio is not entered for the gaze data because the parameter is calculated by the program from the fast phase and slow phase velocities entered by the user. The slope difference threshold is entered for simulated sinusoidal pursuit EOG data such that saccadic events do not occur at the maxima or minima of the base sinusoid, a necessary condition if the simulated EOG is to be consistent with human ocular responses during sinusoidal tracking tasks. The value of the slope difference threshold parameter was determined experimentally and set at 0.001 for the frequencies between 0.10 and 0.80 Hz.

After the program is initialized, the operator must decide if the addition of saccades is desired. If the operator desires to superimpose saccades (nystagmus beats) on the simulated base signal (midposition gaze or sinusoid), the operator must enter the desired magnitude of nystagmus beats (AMP) (one, two, three, five or ten degrees); then the desired direction of the nystagmus beat (upward beating or downward beating) is entered.

Normal ocular pursuit responses to sinusoidal stimulus consist of a base sinusoidal-waveform with corrective saccades opposite to the direction of ocular movement. This convention assumes that, during normal ocular pursuit, prediction errors occur as subjects overshoot the stimulus. The simulated sinusoidal pursuit EOG data with superimposed saccades contain fast events opposite to the direction of the base sinusoid; therefore, fast events in both directions (upward and downward) occur within one cycle of data as shown in the NASDAT output (Fig. 3). The slow phase velocity is not entered for the simulated sinusoidal EOG data because the velocity can be determined from the knowledge of the frequency of the sinusoid.

Once the base waveform has been generated, the beginning and ending times of each fast phase are determined and the base waveform root-mean square value (BRMS) is calculated

according to Equation (6):

$$\text{BRMS} = \left\{ \left[\sum_{i=1}^N (Y_i - \bar{Y})^2 \right] / N \right\}^{0.5} \quad (6)$$

where Y_i is the i^{th} data point and \bar{Y} is the mean value of the eye position data. The signal-to-noise ratio of the signal is calculated from the ratio of the base waveform RMS value (BRMS) to the RMS value of the added noise (NRMS). If noise is not added to the signal, SNR is arbitrarily set to infinity. After the data has been generated, the data and timing files are created. The data files contain one column of eye position data as well as data parameters, if so desired, at the end. The timing files contain beginning and ending times of each fast phase.

The velocities of the fast and slow phases of the simulated data were constant for each nystagmus amplitude at a given data point ratio. For example, Table I is a list of the slow and fast phase velocities for the 1/10 data point ratio. For more information on the execution of the NASDAT program consult the **NASDAT Software User's Manual** (Appendix A) before using the program.

Table I. Velocity of slow and fast phases as a function of nystagmus amplitude for simulated data with 1/10 data point ratio.

<u>Nystagmus Amplitude</u> (°)	<u>Slow Phase Velocity</u> (°/sec)	<u>Fast Phase Velocity</u> (°/sec)
1	6	66
2	13	133
3	20	200
5	33	333
10	60	666

Human Subject Data

Real EOG data were used to verify results obtained with the simulated EOG data. Human EOG data collected at the Texas A&M University Physiologic Signal Acquisition and Processing Laboratory were analyzed with the different programs to evaluate their performance

with human EOG data. Three types of data were analyzed: optokinetic pursuit, sinusoidal pursuit and pseudorandom pursuit. Figures 4A and 4B illustrate a five (5) second segment of the stimulus and optokinetic EOG response. The stimulus was presented at a constant velocity of 29.2 degrees/sec from -15 degrees to +15 degrees. The pattern of saccades followed by smooth pursuit was generated as the subject tried to fixate on the repeatable, unidirectional stimulus appearing in the field of vision at -15 degrees (*i.e.*, entering the field of vision) and disappearing at +15 degrees (*i.e.*, leaving the field of vision). Although the saccades in this data set are not true fast phases, the readily identifiable events make this a suitable data set for analysis. The sinusoidal stimulus had an amplitude of 20 degrees and a frequency of 0.10 Hz. Figure 5 illustrates the sinusoidal pursuit stimulus and response collected during a 10-second period. The pseudorandom stimulus consisted of the sum of eight (8) sinusoidal frequencies ($f_1 = 0.015$ Hz; $f_2 = 0.026$ Hz; $f_3 = 0.050$ Hz; $f_4 = 0.096$ Hz; $f_5 = 0.190$ Hz; $f_6 = 0.378$ Hz; $f_7 = 0.751$ Hz and $f_8 = 1.500$ Hz). Pseudorandom data were collected during a four minute period [8]; however, only six (6) seconds from a pseudorandom data set were analyzed as part of this study. Figures 6A and 6B illustrate the pseudorandom stimulus and EOG response used for program evaluation. The human subject data were low-pass filtered at 45 Hz before sampling at 200 Hz.

CHAPTER V

DESCRIPTION OF PROGRAMS AND RESULTS

Presentation of material in this section is arranged by laboratory. The nystagmus analysis program of each laboratory is briefly described before performance results from synthetically generated data then human subject data, respectively. The order of presentation by laboratory are USAF/SAM, UCLA, MIT, Harvard, and NASA/SBRI laboratories. In all cases, input parameters were set according to instructions from or after discussion with the respective laboratory. Once set, the input parameters were not reset. The rationale for not fine tuning threshold parameters for each run is based on the desire to obtain the dynamic range of operation for each program in a minimum hands-on, automated mode.

U. S. Air Force Nystagmus Analysis Program

The eye movement analysis program obtained from the U. S. Air Force was developed by the Data Science and Clinical Science Division, United States Air Force School of Aerospace Medicine in Brooks Air Force Base, San Antonio, Texas. The flowchart of the overall program, shown in Figure 7, depicts the overall structure of the data analysis program. The program, written to analyze data on-line, consists of a main program, written in FORTRAN, and a number of sub-programs, written in assembly language. The program requires initialization of three parameters, CALSUM, KLIMIT and ILIMIT. The Air Force program consists of a calibration section, a data collection and analysis section, and an exit section. The sub-programs are called by the main program to perform initialization of hardware, clear memory spaces, control the motorized rotational chair, control data collection, and perform fast phase detection. Only the two fast phase detection algorithms (HLOGIC and MLOGIC), pertinent to this study, were converted from assembly language to FORTRAN 77 for evaluation. Other on-line processing algorithms were not re-coded and are not discussed in

this report. A minor modification was made to read test data from a file rather than from an analog-to-digital (A/D) converter.

Description of the Algorithm

Calibration Subroutine: The calibration section was written to: 1.) control the presentation of a constant velocity stimulus to the subject; 2.) calculate the eye velocity to the eye position conversion factor; and 3.) determine the velocity threshold for fast phase detection. Thus, in this program the user need not enter any parameter. The calibration section, which is depicted by the flowchart shown in Figure 8, controls an optokinetic stimulus to turn clockwise and then counterclockwise at a constant velocity of 30 degree/sec for 10.48 seconds in each direction. When used on-line, the EOG data were filtered by a precision (3-second time constant), one (1) pole analog RC high-pass filter before digitization [9].

For this study, all simulated data are filtered with an equivalent digital highpass filter which simulated the one-pole analog RC highpass filter. The filtered and digitized EOG data are read and processed point by point by two specially designed 31-point, finite impulse response (FIR), velocity and acceleration digital filters. The output of the velocity digital filter is a measure of eye velocity, whereas, the output of the acceleration filter is a measure of eye acceleration. Accordingly, these filters remove the effects of the RC filter and high frequency noise [9]. The instantaneous eye velocity and eye acceleration are compared to predetermined corresponding thresholds KLIMIT and ILIMIT, respectively. When either threshold is exceeded, the eye velocity value is set to zero because it is assumed that a fast phase has been detected. Then, the processed eye velocity data are separated into the left and the right-going parts and are further processed to remove overshoots and undershoots caused by the digital velocity filter. Subsequently, left- and right-going eye velocity calibration data are averaged individually to obtain the calibrated factor for the person's left- and right-eye movements. An overall calibration factor, CALSUM, is calculated from the left calibration factor, the right

calibration factor, and an empirically determined factor. The velocity threshold, KLIMIT, is updated to 2.5 times the value of the right calibration factor. Both the CALSUM and the KLIMIT are used in the analysis subroutine.

Analysis Subroutine: In the analysis section, which is illustrated by the flowchart in Figure 9, the high-pass filtered input data passed through the same digital velocity and acceleration filters used in the calibration section. The instantaneous eye acceleration is compared to the preset acceleration threshold (same as in the calibration section). If the instantaneous eye acceleration is larger than the preset threshold, an event is said to have occurred. The algorithm HLOGIC sets a fast phase flag ISHIFT and maintains ISHIFT set during the period in which the eye acceleration exceeds the threshold. When the eye acceleration drops below the threshold, the ISHIFT is kept set by MLOGIC for the next five additional input data points. The instantaneous eye velocity is set to zero when the ISHIFT flag is set or when the eye velocity exceeded the threshold KLIMIT (calculated in the calibration section). For example, a probable fast phase is indicated by a series of zeroes in the processed eye velocity data. The locations of zeroes correspond to the non-zero locations of ISHIFT. Thus the values of ISHIFT are extracted and output to a file for later fast phase identification performance evaluation. The latter part of the analysis program performs the following tasks:

1. Remove fast phases
2. Reconstruct the slow phase waveform
3. Average the reconstructed slow phase
4. Filter the averaged slow phase, and
5. Multiply the final filtered slow phase by the scale correction factor CALSUM to scale in degrees.

In addition, the program displays, on CRT, graphs of filtered, scaled slow phase eye position,

and input stimulus. Based upon the eye position data given in the graph, the operator may decide to save the data or to collect another segment of data. Since the functions performed by the latter part of the analysis program, *i.e.*, reconstructing, filtering and scaling, are pertinent to this research, these functions were not evaluated.

Because the U. S. Air Force program does not provide a fast phase indication file, fast phase identification information is extracted and output to a file. The fast phase identification (FPI) data and the nystagmus timing (NI) data are read by a performance evaluation program called "PERFORM." PERFORM aligns timing points of FPI data with the NI data timing by shifting the FPI data 15 data points to correct for the time delay produced by the 31-point digital velocity filter. One additional offset point was allowed between the FPI and NI data. If the FPI data indicates a fast phase event when the NI data does not contain a saccadic jump, then a false detection is said to have occurred; conversely, if the FPI data does not indicate a fast phase event, yet the NI data contains a saccadic jump, then a missed detection is said to have occurred. The Air Force nystagmus analysis program was not evaluated for its accuracy in the timing of fast phases because the program was not written to detect ending points of saccades.

Simulated Data Parameters

Simulated EOG data used to evaluate the US Air Force nystagmus program had a sampling period of approximately 16 msec. (*i.e.*, 62 Hz sampling frequency). Each data file was 400 seconds in length. The simulated midposition gaze and sinusoidal pursuit data were generated with simulated saccadic jumps of one, two, three, five, and ten degrees of magnitude. The simulated sinusoidal pursuit data contained saccades of one two, three, and five degrees superimposed on sinusoids of 0.05, 0.10, 0.50 Hz, and 0.80 Hz. The amplitude of the sinusoidal pursuit data sets was 20 degrees. The data were not filtered through an anti-alias filter.

Results with Simulated Midposition Gaze Data

The results of the simulated midposition gaze test at each of five saccadic amplitudes (one, two, three, five, and ten degrees) with a 1/10 data point ratio, show a decrease in performance of the program with an increase in the magnitude of the nystagmus saccadic jump. As shown in Figure 10, the Air Force program does not produce any errors with simulated data containing saccadic jumps of two (2) degrees until the signal-to-noise ratio (SNR) is less than six. For the simulated data containing saccadic jumps of three (3) and five (5) degrees, the program does not produce detection errors until the SNR is less than 10 and 17, respectively. For saccades of ten (10) degrees, the Error Index remained about 11%, even at a SNR of 100, as shown in Figure 11. Generally, the false-detection errors are larger than the missed-detection errors except for data sets containing saccadic jumps of five (5) and ten (10) degrees at SNR above six (6) and 12, respectively. The reversal of missed-detection and false-detection error magnitudes for five (5) degrees and ten (10) degrees is attributed to the use of fixed thresholds to detect fast phases. When random noise is superimposed on a large magnitude saccade, the period during which the velocity and acceleration thresholds are exceeded becomes increasingly longer as larger magnitude noise is added. HLOGIC, which is used to keep track of the period during which the eye acceleration exceeds the acceleration threshold, maintains the fast phase flag ILIMIT in the valid state for an extended period. At the end of this period, only one count of fast phase detection is credited toward the program; whereas, several fast phases may have occurred during this period, and are considered being "missed detected." Thus, the false detection counts decrease; however, the number of missed detections increases as the magnitude of random noise increases. The combined error curves for the various magnitudes of saccade are shown in Figure 12. The general trend of degradation in performance as the saccadic magnitude increased from two (2) to ten (10) degrees is also the consequence of fixed velocity and acceleration thresholds. Analysis of the one (1) degree data indicates a high percentage of missed-detection error and is not illustrated in

the graphs.

Results with Simulated Sinusoidal Pursuit Data

The U. S. Air Force program was evaluated with simulated sinusoidal pursuit EOG data at five magnitudes (one, two, three, five, and ten degrees), four frequencies (0.05, 0.10, 0.50 and 0.80 Hz), and a data point ratio of 1/10. Results of 0.8 Hz sinusoid with two (2) degree saccades (Fig. 13) indicate that the program was not designed for sinusoids at 0.8 Hz since the maximum velocity of the slow phase at 0.8 Hz is larger than the velocity threshold set by the program. The fixed velocity threshold causes the HLOGIC to set the fast phase flag at wrong locations and/or for extended periods of time; thus, results at 0.8 Hz are considered to be erroneous and questionable.

The percentage of missed-detection and false-detection errors by the USAF program is illustrated in Figure 13. This figure, typical of the error profile for all the nystagmus amplitudes tested, indicates that at SNR greater than 20 the performance of the USAF/SAM nystagmus analysis program is good, *i.e.*, detection errors of 5% or less. The performance decreases (greater detection errors) as the magnitude of the saccadic jumps is increased from two (2) to ten (10) degrees for the same value of SNR (Fig. 15). The false-detection errors are larger than the missed-detection errors as the amplitude of the noise is increased (decrease in SNR). A decrease in the false-detection error, shown in Figure 14, occurs at SNR less than five (5) when the HLOGIC maintains the fast phase flag for an extended period of time.

The Error Index for the frequencies tested (Figs. 15 - 18) indicates that the program is consistent and independent of sinusoidal frequencies between 0.05 Hz and 0.50 Hz. The performance degrades about 10% at 0.50 Hz when compared to either 0.05 or 0.10 Hz for the larger magnitude saccades (five and ten degrees). The "zigzagging" nature of errors when the SNR is less than nine (9) is an indication that the program performance is inconsistent or unreliable at low SNR. The Error Index profiles shown in Figures 15 through 18 indicate that

the program performance is better with small amplitude saccades (two degrees) and large amplitudes of noise (*i.e.*, small SNR) than with large amplitude saccades (ten degrees). The best overall performance of the program occurs with the 0.10 Hz sinusoidal waveform with three degree saccades. The results for the one (1) degree of saccadic jump at all frequencies are erratic and unreliable. The frequency limitation of the US Air Force program at 0.80 Hz is illustrated in Figure 18.

Results with Human Subject Data

Optokinetic, sinusoidal pursuit, and pseudorandom pursuit data collected from human subjects were analyzed with the USAF/SAM nystagmus program. The human subject data were collected at 200 Hz; however, as the Air Force program uses a sampling rate of 61.445 Hz [10], a sampling rate conversion was performed on the data to bring the effective sampling rate to 62 Hz. The time marks identified by hand-scoring subject data with a sampling rate of 200 Hz were adjusted to the corrected sampling rate of 62 Hz before the program was evaluated. Fast phase timing information is not included in the output as the Air Force program does not provide timing information.

Table II summarizes the fast phase detections of the program with corresponding magnitude of saccade. For the optokinetic data set, five (5) saccades were missed, resulting in 38.46% missed-detection error; and 13 saccades were falsely identified, resulting in 61.9% false-detection error. The Error Index for the optokinetic data set is 51.18%.

Results with the sinusoidal pursuit data set (0.10 Hz) indicate very high false-detection errors and high missed-detection errors. The results are summarized in Table III. The USAF/SAM program identified 36 fast phases; two of the five true fast phases were detected, resulting in 60.0% missed-detection error, and 34 fast phases were falsely detected, resulting in 94.44% false-detection error. The Error Index for the human sinusoidal pursuit data is 77.22%.

Table II. Results from the analysis of human optokinetic data by USAF/SAM Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	8.0	detected
2	15.6	detected
3	18.0	missed
4	5.6	detected
5	18.0	detected
6	14.2	detected
7	12.4	detected
8	14.6	detected
9	13.6	detected
10	16.2	missed
11	19.2	missed
12	20.2	missed
13	5.4	missed

Table III. Results from the analysis of human sinusoidal pursuit data by USAF/SAM Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	3.0	missed
2	3.0	detected
3	3.0	detected
4	3.0	missed
5	4.0	missed

Results with the pseudorandom data indicate equal percentages of missed-detection and false-detection errors. Table IV summarizes the results of the pseudorandom data analysis. A total of 14 fast phases were identified by the program; six (6) were false detections, resulting in a 42.85% false-detection error, and eight (8) were correct identifications (*i.e.*, six (6) fast phases were missed), again resulting in 42.85% missed-detection error. The Error Index for the pseudorandom data set is 42.85%. Examination of Table IV for the saccadic amplitude corresponding to missed detection of a fast phase reveals that a greater number of large magnitude fast phases were missed than were small magnitude fast phases, results which are in agreement with those obtained from simulated nystagmus data.

Table IV. Results from the analysis of human pseudorandom data by USAF/SAM Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	3.2	detected
2	2.7	detected
3	4.0	detected
4	4.5	missed
5	3.7	detected
6	4.6	detected
7	5.2	missed
8	3.8	missed
9	5.0	missed
10	4.8	missed
11	5.3	detected
12	5.8	missed
13	5.0	detected
14	3.0	detected

UCLA Nystagmus Analysis Programs

The University of California at Los Angeles (UCLA) vestibular research laboratory developed two programs (SINUXEC and CALORXEC) to analyze eye movements. The programs identify saccadic eye movements on the basis of minimum velocity and duration criteria, and measure saccadic amplitude, duration, and maximum velocity [11]. The first program analyzes sinusoidal pursuit or rotational test eye position data, and the second program analyzes gaze test eye position data. Because both programs contain several parameter input and plotting subroutines, the fast component identification (FCID) portions of the programs were extracted and analyzed separately.

The UCLA programs were designed to analyze data on-line; therefore, the fast component identification is performed at the assembly language level as the data is collected. The assembly language code was developed with the PDP-11 computer (Digital Equipment Corporation) [12]. To test the FCID portions of the programs, the algorithms were converted to FORTRAN so that the VAX system at Texas A&M University could be used to evaluate the analysis algorithms. The assembly language commands were converted on a one-to-one basis

to equivalent fortran commands. Parameters, necessary for the programs to run, are entered by the user before each program run. Both programs analyze one data point at a time from an input eye position data file. The necessary parameters are checked, updated, and written to an output data file.

SINUXEC Program

Description of the Algorithm

The sinusoidal pursuit execution (SINUXEC) program is used with eye position data created during the pursuit or rotational test. The program initializes the following fast component (FC) parameters:

1. FC direction (IDIREC)
2. Minimum FC duration (MINDUR)
3. Maximum FC duration (MAXDUR)
4. Minimum FC amplitude (MINAMP)
5. Minimum peak velocity (MINPKV)
6. Minimum FC velocity (MINVEL).

Because the program analyzes each data point during data collection, the FC parameters are scaled by the sampling rate (SR) to obtain the parameters in terms of the number of samples rather than the timing of each sample.

The flowchart for the SINUXEC program is shown in Figures 19A and 19B. The SINUXEC program begins execution by calculating the difference between the n^{th} and $n^{\text{th}}+2$ sample points (derivative function with a two-point central difference equation). The absolute value of this difference is stored, and the direction flag set to 1 if the difference is positive (right) or to -1 if the difference is negative (left). Then the program determines if a FC has started, and, if not, the start time, start amplitude, and direction are stored. The FC duration counter is incremented, and the current difference compared to the maximum difference for that

FC. If the current difference is greater than the maximum difference, the current difference is stored; otherwise, the program processes the next data point.

The program checks the direction of the fast component (FC) difference with the previous difference. If the directions are not the same, the end of a FC may have occurred; then the program checks the current FC parameter values with the limit values entered by the user. If the directions are the same and the pursuit test performed, the current difference is compared to MINVEL. If the current difference is less than MINVEL, the program branches and checks the FC parameters against the limit values entered by the user. If it is not a pursuit test or the current difference is greater than MINVEL - the SINUXEC program increments the FC duration counter, stores the value of the maximum difference, and returns to process the next data point.

If an end to a FC is identified, the FC duration counter is compared to MINDUR and MAXDUR. If the duration counter is greater than MINDUR and less than MAXDUR, the maximum difference is compared to MINPKV. If the difference is greater than MINPKV, the starting amplitude is subtracted from the ending amplitude and the absolute difference is compared to MINAMP. If the difference is greater than MINAMP and the direction is the same as IDIREC, the program stores the following parameters [11]:

1. Starting time
2. Starting position
3. Ending time
4. Ending position
5. Maximum difference (maximum velocity).

The maximum difference is converted back to maximum velocity and stored in an output file with the other parameters; then the program clears the parameters. If any of the comparisons failed, the program clears the FC duration counter, the maximum difference register and returns to process the next data point.

Simulated Data Parameters

The simulated data analyzed by the SINUXEC program had a sampling period of 5 msec. (*i.e.*, 200 Hz sampling frequency) and the length of each data file was 15 seconds. The simulated midposition gaze data were generated with saccades of one, two, three, five, and ten degrees. The simulated sinusoidal pursuit data contained saccades of one, two, three, and five degrees superimposed on sinusoids of 0.05, 0.10, and 0.50 Hz. The amplitude of the sinusoidal pursuit data sets was 20 degrees. The data were filtered with a simulated anti-alias digital filter with cutoff at 40 Hz.

Results with Simulated Midposition Gaze Data

The SINUXEC program is very consistent in analyzing the simulated midposition gaze data. Figure 20 illustrates the missed- and false-detection errors and the Error Index for the midposition gaze data with one (1) degree nystagmus. The missed-detection error ratio is low (less than 5%) for all SNR, but the false-detection error ratio increases as the SNR is decreased below four (4). The false detection-error ratio is responsible for the Error Index profiles illustrated in Figure 21. Figures 21 and 22 illustrate the consistency of the SINUXEC program in its detection capability as a function of SNR and amplitude of saccades. Figure 22 shows that the majority of errors are false detections of saccades in the analysis of midposition gaze data.

Results with Simulated Sinusoidal Pursuit Data

The results from the analysis of the simulated sinusoidal pursuit data are not as clear as those from the analysis of midposition gaze data. Apparently, the accuracy of the analysis is dependent on the frequency of the sinusoid and the amplitude of nystagmus. Figure 23 illustrates the results of the analysis of sinusoidal pursuit data at a frequency of 0.50 Hz. The plot shows the high Error Index for the data set with one (1) degree nystagmus with a decrease

in the Error Index as nystagmus amplitude is increased. High incidence of missed and false detections is observed at all frequencies for the one (1) degree nystagmus data set. For all other amplitudes as previously observed with the midposition gaze data; however, the results show a high number of false detections with a low number of missed detections. Figures 24 and 25 summarize the Error Index performance of SINUXEC for sinusoidal pursuit data at 0.05 Hz and 0.10 Hz, respectively. In summary, the performance of the SINUXEC program in analyzing sinusoidal pursuit data, as with midposition gaze data, is consistent in producing low missed-detection errors but high false-detection errors (Fig. 26).

Results with Human Subject Data

The SINUXEC program requires the user to input several nystagmus parameters prior to analyzing the data. With results from the simulated data analysis and *a priori* knowledge of the stimulus parameters used in collecting the human data, the input parameters for the SINUXEC are:

1. Minimum saccadic amplitude- 3 degrees
2. Minimum saccadic duration- 0.015 sec.
3. Maximum saccadic duration- 0.050 sec.
4. Minimum peak saccadic velocity- 150 degrees/sec.
5. Minimum slow component velocity- optokinetic: 30 degrees/sec., sinusoidal: 8 degrees/sec., pseudorandom: 6 degrees/sec.

The optokinetic data set generates high missed-detection and false-detection errors. Table V summarizes the results of the analysis of the human optokinetic data with the SINUXEC program. Nine (9) saccades out of 13 were missed, resulting in a 69.23% missed-detection error; and eight (8) events were falsely identified as saccades, resulting in 66.67% false-detection error. The combined Error Index for the optokinetic data set is 67.95%.

Table V. Results from the analysis of human optokinetic data by SINUXEC Program.

<u>Fast Phase #</u>	<u>Hand-scored</u>			<u>Computer</u>	
	<u>Start</u>	<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>
1	0.350	0.380	8.0	0.360	0.395
2	0.850	0.900	15.6	-	-
3	1.735	1.790	18.0	1.750	1.800
4	2.155	2.185	5.6	2.165	2.195
5	2.710	2.775	18.0	-	-
6	3.580	3.650	14.2	-	-
7	4.495	4.550	12.4	-	-
8	5.635	5.705	14.6	-	-
9	6.580	6.640	13.6	-	-
10	7.790	7.865	16.2	-	-
11	8.975	9.035	19.2	-	-
12	10.125	10.190	20.2	-	-
13	10.665	10.695	5.4	10.675	10.710

Table VI summarizes the results from the analysis of the sinusoidal data from human subjects which indicate low false-detection error and high missed-detection error. The SINUXEC program did not detect any fast phases in the sinusoidal pursuit data set, resulting in 100% missed-detection error, 0% false-detection error, and an overall 50% Error Index for the human sinusoidal pursuit data. The results from the subject data analysis differ from the results of the analysis of 0.10 Hz simulated sinusoidal data, where the false-detection error is higher than the missed-detection error (Fig. 26).

Table VI. Results from the analysis of human sinusoidal pursuit data by SINUXEC Program.

<u>Fast Phase #</u>	<u>Hand-scored</u>			<u>Computer</u>		<u>SNR</u>
	<u>Start</u>	<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>	
1	2.855	2.870	3.0	-	-	1.67
2	5.125	5.145	3.0	-	-	1.93
3	7.100	7.120	3.0	-	-	1.50
4	9.425	9.445	3.0	-	-	1.61
5	9.545	9.570	4.0	-	-	2.03

The pseudorandom EOG data set has higher SNR than does the sinusoidal EOG data set; therefore, the performance of the SINUXEC program improved. Table VII summarizes the results of the pseudorandom data analysis. Only two (2) saccades were missed, and no false

detections recorded, resulting in a 14.28% missed-detection error, a 0% false-detection error, and an overall 7.14% Error Index for the human pseudorandom data.

Table VII. Results from the analysis of human pseudorandom data by SINUXEC Program.

<u>Fast Phase #</u>	<u>Hand-scored</u>		<u>Computer</u>		
	<u>Start</u>	<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>
1	0.603	0.650	3.2	0.640	0.665
2	0.865	0.885	2.7	-	-
3	1.150	1.175	4.0	1.160	1.190
4	1.465	1.490	4.5	1.475	1.500
5	1.710	1.730	3.7	1.720	1.745
6	2.505	2.525	4.6	2.515	2.540
7	2.710	2.735	5.2	2.725	2.750
8	3.005	3.030	3.8	3.015	3.045
9	3.725	3.755	5.0	3.740	3.765
10	4.065	4.090	4.8	4.075	4.100
11	4.295	4.325	5.3	4.305	4.340
12	4.975	5.005	5.8	4.985	5.020
13	5.205	5.235	5.0	5.220	5.250
14	5.750	5.775	3.0	-	-

CALORXEC Program

Description of the Algorithm

The gaze test execution program (CALORXEC) analyzes eye position data produced during the midposition gaze test. The flowchart for the CALORXEC program is shown in Figures 27A and 27B. The program begins by initializing the following fast component identification (FCID) parameters used by the FCID portion of the CALORXEC program:

1. Minimum duration (DURMIN), expressed in number of samples
2. Minimum amplitude (MINAMP)
3. Fast Component direction (DIRFC).

Similar to the SINUXEC program, the CALORXEC program is an on-line analysis program that analyzes the eye position data, point-by-point as the data is sampled. The program stores the absolute difference of the n^{th} and $(n^{\text{th}} + 1)$ data points and sets the direction flag to one (1) if the difference is positive (right) and to minus one (-1) if the difference is negative (left). Subsequently, the program checks for the start of a fast component (FC) and,

if a fast component has started, the program branches to compare the previous direction with the current direction. If a fast component has not started, the program stores the start time, the current amplitude, and the current direction before comparing the previous direction to the current direction. If the previous direction is the same as the current direction, the FC duration counter is incremented and the current difference is compared to the maximum difference for the current FC. If the current difference is greater than the maximum difference, the current difference is stored as the new maximum difference. The program then returns to process the next data point.

If the current direction differs from the previous direction, a possible end of the current FC may have occurred; then, the program branches and compares the FC duration counter to DURMIN. If the FC duration is not greater than DURMIN, the program clears the FC duration counter and maximum difference register before returning to process the next data point. Otherwise, the program branches and calculates the absolute FC amplitude from the difference between the starting amplitude and the current amplitude. Before comparing the current FC amplitude to MINAMP, the program determines if the current FC is consistent with DIRFC. If the current amplitude is not greater than MINAMP or if the directions are not consistent, the program clears the FC duration counter and maximum difference register before returning to process the next data point. Otherwise, the program increments the variable NSAC (the number of saccades) and then compares the value in NSAC to the maximum number of saccades allowed per data record. If NSAC is less than the maximum, the program stores the following FC parameters to a data file:

1. Starting time
2. Starting amplitude
3. Ending time
4. Ending amplitude
5. Maximum difference.

After storing the current FC parameters or if NSAC is greater than the maximum, the program clears the FC duration counter and returns to the next data point.

Simulated Data Parameters

The simulated data analyzed by the CALORXEC program had a sampling period of 5 msec. (*i.e.*, 200 Hz sampling frequency), and the length of each data file was 15 seconds. The simulated midposition gaze data were generated with saccades of one, two, three, five, and ten degrees in magnitude. The simulated sinusoidal pursuit data, which was filtered with an anti-alias filter ($f_c = 40$ Hz), contained saccades of one, two, three, five, and ten degrees superimposed on sinusoids of 0.05, 0.10, and 0.50 Hz. The amplitude of the sinusoidal pursuit data sets was 20 degrees.

Results with Simulated Mid-position Gaze Data

The CALORXEC nystagmus analysis program was initially evaluated with simulated midposition gaze data. No errors, *i.e.*, 0% Error Index, are observed for the gaze data without noise at the five saccadic amplitudes (one, two, three, five, and 10degrees) and different data point ratios, *i.e.*, number of fast phase data points to slow phase data points (1:5, 1:8, 1:10, 1:12, and 1:15).

Figure 28 shows the effect of adding noise to midposition gaze data containing the five amplitudes of saccade. The best results are obtained for 10-degree saccades for SNR's less than 20. The program produces significant errors for one (1) degree saccades. Acceptable results (Error Index of 5% or less) are obtained for saccadic amplitudes greater than one degree with signal-to-noise ratios greater than 33. Figure 29 shows that the reason for the high Error Index is the high value of missed-detection error.

Results with Simulated Sinusoidal Pursuit Data

The next test series evaluated the performance of the UCLA CALORXEC Program with simulated sinusoidal pursuit data without noise. Data with three different ratios of fast phase to slow phase data points (1:5, 1:10, and 1:15) were analyzed.

For the sinusoidal pursuit data with 3 degrees nystagmus at 0.05 Hz and 0.10 Hz, the CALORXEC program produces 100% missed-detection errors, which are caused by the inability of the program to differentiate between the slow and the fast phases at these frequencies (Fig. 30). Figure 31 illustrates the results of the analysis of sinusoidal pursuit data with 3 degrees nystagmus at the three sinusoidal frequencies. The high Error Indexes at 0.05 Hz and 0.10 Hz are caused by the high frequency of missed-detection errors shown in Figure 30.

Results with Human Subject Data

Comparison of the two UCLA programs, SINUXEC and CALORXEC, by Schmidt [13] showed that for every test condition the SINUXEC program always produces significantly better results than the CALORXEC program. Therefore, only the UCLA SINUXEC program was evaluated with actual human data.

MIT Program (M2MI86)

The Massachusetts Institute of Technology nystagmus analysis programs M2MI86 and M2DETE were developed by Mohammad-Ali Massoumnia under contract NAS-W-3651, NASA (Washington) and contract NAS-9-15343, NASA (Houston). The only difference between M2MI86 and M2DETE is that the M2MI86 program reads and writes the data word by word, whereas the latter program reads and writes 256 words at a time; thus, the M2DETE program is faster [14]. Because speed is not of primary concern for the performance

evaluation, the M2MI86 program was chosen to be evaluated. The M2MI86 program is written in FORTRAN. Changes to the program have been minor input/output modifications allowing the program to read from and write to data files.

Description of the Algorithm

Before execution of the program, several parameters must be entered by the user. The parameters include:

1. Starting point of the data file
2. Ending point of the data file
3. Sampling rate
4. Minimum acceleration threshold (MAT)
5. A/D scale factor.

The data is copied to a scratch file for processing.

The M2MI86 algorithm detects fast movements of the eye on the basis of the eye's velocity and pseudo acceleration. The flowchart for the M2MI86 program is shown in Figures 32A and 32B. The algorithm uses digital finite impulse response (FIR) filters to calculate the velocity and pseudo acceleration of the eye. Once an event is detected, it is classified as either a saccade, a fast phase of nystagmus, or "other" type of fast movement. Simultaneously, the program produces the following outputs:

1. One point per beat slow phase velocity
2. Slow phase velocity
3. Pseudo-acceleration
4. Event indicator flag
5. Cumulative slow phase position.

The detection of fast components is based on the sign and magnitude of the instantaneous velocity and acceleration of the eye. The beginning of a fast phase is the point where the

absolute magnitude of acceleration is larger than a starting threshold (T_s), the sign of velocity and acceleration is the same, and the absolute magnitude of acceleration is increasing. The starting threshold (T_s) is based on the root-mean-square (RMS) value of the acceleration signal. Velocity is not compared with T_s because the mean value of the velocity signal is not zero; whereas, the mean value of the acceleration signal is zero [14]. The ending point of a fast phase is the point where the absolute magnitude of acceleration drops below an ending threshold (T_e), and the signs of velocity and acceleration are different from their respective signs during the onset of the fast phase.

Following the detection of an event, the event is classified as a saccade if the velocity of the slow phase preceding the event is less than 2 degrees/sec. The event is classified as a fast phase if the fast movement preceding the current event has the same direction [14]. If the preceding slow phase does not meet any of the criteria, it is classified as "other" type of fast movement.

The M2MI86 program calculates the acceleration of the eye using a 3-point double differentiator. Because the two opposite peaks of the acceleration waveform are the points of interest, the output of the noise sensitive double differentiator is lowpass filtered to attenuate the spectral peaks often associated with wideband differentiation. The combination of filters produce a 9-point pseudo-acceleration filter (PAF). The 3-point double differentiator, based on the Stirling interpolation formula, is accurate at low frequencies but has limited bandwidth. The filter is a 7-point lowpass filter with the cutoff frequency (ω_c) and filter coefficients based on the sampling frequency. The cutoff frequency and sampling frequency (ω_s) are related by Equation 7.

$$\omega_c \leq \omega_s + 2. \quad (7)$$

For each run of the program, the unit sample responses (USR) of the 9-point PAF and the

7-point lowpass filter are calculated. The unit sample response of the PAF is used to calculate the acceleration output.

An optimal 9-point differentiator is used to find the derivative of the eye position signal. The USR of this filter is fixed to reduce ringing in the output caused by the frequency of damped oscillation (f_d) created by the Butterworth antialiasing filter used during sampling [14]. The velocity filter is band limited to a frequency lower than f_d , as described by Equations 8 and 9.

$$f_d = 0.707 * f_c \quad (8)$$

$$f_c \leq f_s / 2 \quad (9)$$

where f_c is the cutoff frequency below 20 Hz and f_s the sampling frequency.

The acceleration thresholds determining the beginning and the ending of a fast movement are calculated as follows. The root-mean-square (RMS) value of the pseudo-acceleration filter is calculated during the next N sample points (where $N = f_s$). The RMS value is a measure of activity in the acceleration signal; therefore, when there are too many spikes in the acceleration signal, the RMS value increases [14]. Combining the RMS value with a constant minimum acceleration threshold (MAT) entered by the user reduces the sensitivity of the adaptive threshold. The minimum acceleration threshold varies according to the SNR of the eye position data. The linear combination of RMS and MAT determines the starting threshold (T_s), which is calculated using Equation 10.

$$T_s = \text{RMS} + 0.5 * \text{MAT} \quad (10)$$

Multiplying T_s by a constant with magnitude less than unity, as described by Equation 11, produces the ending threshold (T_e).

$$T_e = 0.7 * T_s \quad (11)$$

Convolving the eye position data in the scratch file with the USR's of the PAF and velocity filters produces the acceleration and velocity outputs. Outputs from the filters are stored in separate ring buffers. After the buffers are filled, the program searches 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 [14]. Before the slow phase velocity is stored, it is further smoothed with the 7-point lowpass filter used for the PAF.

Following the detection of an event, the program searches forward for the end of the event. After finding the end point, the velocity during the event is computed by averaging the velocity before and after the event. The detected event is classified and the saccade indicator set to an appropriate magnitude. The classifications and corresponding magnitudes are:

0 = a data point not belonging to a fast event

± 100 = a data point from a fast component which is in the direction of the previous slow component

± 200 = a data point from a fast component which is in the opposite direction of the previous slow component

± 300 = a data point belonging to an unidentified fast event.

The sign identifies the direction of the component, positive for right-going fast components, negative for left-going fast components. The data process indicator advances the length of the event while the necessary outputs are written to their files.

The output available from the program includes one point per beat slow phase velocity, slow phase velocity, acceleration, saccade indicator, and cumulative slow phase velocity. The cumulative slow phase velocity is the position that the eyeball would reach if it could rotate without restraint in the socket, and it 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 is a plot of time versus the average of slow phase velocity per slow phase beat. [14].

Simulated Data Parameters

The simulated data analyzed by the M2MI86 program had a sampling period of 5 msec.

(i.e., 200 Hz sampling frequency), and the length of each data file was 15 seconds. The simulated midposition gaze data were generated with saccades of one, two, three, five, and ten degrees. The simulated sinusoidal pursuit data, which was filtered with an anti-alias filter ($f_c = 35$ Hz), contained saccades of one two, three, and five degrees superimposed on sinusoids of 0.05, 0.10, and 0.50 Hz. The amplitude of the sinusoidal pursuit data sets was 20 degrees. The type of saccade determined by the program was not considered in the analysis of the data. The only criteria for detection were the approximate timing and the direction of the fast event.

Results with Simulated Mid-position Gaze Data

The performance of the M2MI86 program in analyzing the simulated midposition data is contradictory. Figure 33 illustrates the results of the analysis of midposition gaze data with 1 degree nystagmus. This figure illustrates the typical profile of the error ratios and Error Index for midposition gaze data with all the nystagmus amplitudes tested (one, two, three, five and ten degrees). The missed-detection error ratio is less than 5% for SNR greater than seven (7) and remains close to 5% for SNR less than six (6). However, the false-detection error ratio increases exponentially at SNR less than ten (10) and is as high as 30% at SNR of five (5). The error profiles illustrated in Figure 33 are also observed for the other nystagmus amplitudes; false-detection errors increase at a high rate as the SNR falls below five (5) while the missed-detection errors remain relatively constant (Fig. 34). The interesting feature of the results from the analysis of midposition gaze data is the increase in the number of false detections as nystagmus amplitude increases. For the three degree nystagmus data the false-detection error ratio is 38% at a SNR of five (5), but for the data with five (5) degree nystagmus the false-detection error ratio increases to 43% at a SNR of five (5). The increase in false-detection errors as a function of SNR and nystagmus amplitude is responsible for the Error Index profiles illustrated in Figure 35. This figure shows how the performance of the M2MI86 program deteriorates as the signal-to-noise ratio (SNR) decreases.

Results with Simulated Sinusoidal Pursuit Data

The most common errors in the analysis of sinusoidal pursuit data with the M2MI86 program are false detections (as observed in the analysis of the midposition gaze data). Figure 36 illustrates the consistently low missed-detection errors contrasted by exponentially increasing false-detection errors for sinusoidal data at 0.50 Hz. As nystagmus amplitude increases, more false detections are recorded, but missed detections remain relatively constant. Figures 37 through 39 illustrate the Error Index of the simulated sinusoidal pursuit data at 0.05, 0.10, and 0.50 Hz, respectively. These figures indicate that the program performs better at 0.50 Hz with smaller magnitudes of saccadic jump, *i.e.*, one and two degrees, than at the lower frequencies, 0.05 and 0.10 Hz. At all the frequencies tested, the false-detection errors cause the increase in the Error Index as SNR decreases. It should be noted that program parameters were not reset or optimized for any particular run. The best performance of the program, 5% Error Index with SNR of seven (7), appears to be with two (2) degrees of saccadic amplitude and a sinusoidal frequency of 0.50 Hz. For the 0.05 and 0.10 Hz sinusoidal data, the errors become significant at a SNR of approximately 12.

Results with Human Subject Data

The form of the output of the M2MI86 program makes it impossible to accurately determine the timing information of the saccades in the human EOG data sets. Figure 40 is a graph of the output from M2MI86 for the optokinetic EOG data set. This output, as well as the output from the sinusoidal and pseudorandom data sets, was evaluated by expanding sections of the graph, determining the approximate timing of the saccadic events from the index on the graph, and correlating this information with the timing of the true saccades in each data set determined by manually. The results in tabular format are not presented.

The analysis of the optokinetic data set indicates few missed-detection errors; only two (2) fast events are missed, resulting in a 15.38% missed-detection error. However, false

detections are numerous; 58 events are falsely identified, resulting in a 84.06% false-detection error. The Error Index for the human optokinetic data set is 49.72%.

As with the other programs evaluated, the signal-to-noise ratio of the sinusoidal EOG data set caused problems. The results of the analysis of simulated data show that for the 0.10 Hz sinusoidal data with SNR lower than 10, false detections recorded by M2MI86 increase exponentially and are as high as 50% (Fig. 41). For the human sinusoidal pursuit data set, no fast events are missed (0% missed-detection error); however, 119 events are falsely identified, resulting in a 95.97% false-detection error. The Error Index for the human sinusoidal pursuit data is 47.98%.

The pseudorandom EOG generated fewer false detections. Only one (1) fast event is missed, resulting in a 7.14% missed-detection error; and 22 events are falsely identified, resulting in a 62.86% false-detection error. The Error Index for the pseudorandom data set is 35.0%.

Harvard Program

The nystagmus algorithms were developed originally for the Raymond E. Jordan Human Vestibular Clinical Laboratory of the Eye and Ear Hospital, Pittsburgh, PA, by Conrad Wall III, Ph.D., who subsequently moved his vestibular laboratory projects to the Massachusetts Eye and Ear Infirmary Vestibular Laboratory, Harvard Medical School, Boston, MA. These algorithms were developed for off-line analysis of nystagmus. The program consists of two major components: 1.) a saccade/slow component identification, and 2.) modeling. Only the fast phase detection algorithm portion of the first component of the program was used in this study. The program was recoded into FORTRAN 77 for use in the VAX system at Texas A&M University. Minor modifications of the input/output algorithms of the program were made in order to read from and write to data files on hard disk rather than magnetic tape.

Description of the Algorithm

The algorithm used by the Harvard laboratory [15] performs an identification of extrema by scanning the data for occurrences of amplitude maxima and minima [11][12][16]. A synthetic nystagmus representation is obtained by straight line connections of extrema. The synthetic nystagmus is used to identify the fast phase segments. A preset parameter called "peak detection" is used to identify extreme points. The program identifies an event as a fast phase component of nystagmus if a segment between a maxima and a minima has a slope greater than a preset minimum slope threshold and a time duration greater than a preset minimum time limit.

A major disadvantage of the program is that these three parameters are in counts of A/D units not in physical units. Thus, the parameters depend upon the sampling interval, the A/D resolution (12-bit, etc.), and the amplification factor. This requires an operator with more than nominal knowledge. In general, this approach is by trial-and-error.

Once a fast phase is identified, the fast phase portion is removed from slow phase eye position and replaced with a first order (straight line) extrapolation. Then, the vertical displacements of the extrapolation and the fast phase segment are added to provide an overall positional offset in reconstructing the slow phase nystagmus, which is referred to as the "estimated cumulative slow phase" [15]. A simple first difference (central difference equation) is calculated from the cumulative slow phase position to obtain the slow phase velocity. The slow phase velocity is fitted with a sinusoidal least squares fit rather than a Fast Fourier Transform.

A flowchart of the fast phase identification portion of the program is shown in Figure 42. Before the program can be executed, the operator must initialize the program by presetting four parameters: window width (IDEL), minimum slope threshold (SLIM), minimum amplitude displacement threshold (TLIM), and maximum amplitude displacement threshold (DISMAX). These parameters are important because they affect the accuracy of fast phase detection and slow phase reconstruction. The parameters are set by an operator with either *a priori*

knowledge of the data or by trial-and-error. *A priori* knowledge must take into account: the type of data set, the sampling rate, and the amplification factor. The program uses a moving window search based on the preset value IDEL to scan the EOG data and detects peaks in the data (*i.e.*, maxima and minima) as points which are higher or lower in amplitude than the adjacent points. A flag is set to one (1) if the extremum is a maximum or set to minus one (-1) if the extremum is a minimum. Information on the points of extrema is stored in two arrays. The flags, which specify each extremum as a maximum or minimum, are stored in an array, MXMNI. A second array, MXMN, stores the coordinates that represent time and magnitude of each extrema. Values of slope (or average velocity) are calculated from each maximum peak and the next neighboring minimum peak value. When the program finds a positive value of one (maximum) in the array MXMNI, the absolute value of the slopes of the segment to the left and to the right of the present maximum is computed. The line segment with the largest absolute value between right slope and left slope is selected as a fast phase candidate. Subsequently, slope and displacement magnitudes of the segment are compared with the preset threshold values (*i.e.*, the slope magnitude is compared with the minimum slope threshold (SLIM)), and the amplitude displacement is compared to the displacement value between the minimum amplitude displacement threshold (TLIM) and the maximum amplitude displacement threshold (DISMAX). If the selected segment meets the fast phase detection criteria, the segment is marked as a fast phase and the actual magnitude values are stored in the array CON. The elements that do not belong to a fast phase are stored as zeros in the array CON. Finally, the program outputs starting times of the detected fast phases.

Other algorithms of the program include: rejection of fast component, extrapolation of the slow component position, computation of slow component velocity, and calculation of gain and phase from the relationship between slow component velocity response and the stimulus.

Proper adjustments of the input parameters to the Harvard program were made before analyzing the data. Window width, IDEL, is set to 1 so that the extrema search would go

through all eye position data points to detect maximum and minimum amplitude. The slope limit (SLIM), minimum amplitude (TLIM) and maximum amplitude (DISMAX) are set empirically after determining the initial value that provided the best fast phase detection. Because the Harvard program provides only the starting time of the identified fast phase, the accuracy of the ending point timing of the fast phase was not evaluated. Starting times of the fast phases produced by the program are compared to the timing information of the artificial data (nystagmus timing data) being tested. If the timing data from the program indicates a fast phase when the data does not contain a fast phase, it is counted as a false detection. If the data indicates a fast phase, but the program does not, then the event is counted as a missed detection.

Simulated Data Parameters

The simulated data analyzed by the Harvard program had a sampling period of 10 msec. (*i.e.*, 100 Hz sampling frequency), and the length of each data file was 15 seconds. The simulated midposition gaze data were generated with saccades of one, two, three, five, and ten degrees. The simulated sinusoidal pursuit data contained saccades of one, two, three, five, and ten degrees superimposed on sinusoids of 0.05, 0.10, and 0.50 Hz. The amplitude of the sinusoidal pursuit data sets was 20 degrees. The simulated data were not filtered with an anti-alias filter.

Results with Simulated Midposition Gaze Data

The Harvard program produces no errors in the analysis of the artificial midposition gaze data without noise. The effects of adding noise to the midposition gaze data at the five nystagmus amplitudes are shown in Figure 43. As indicated in Figure 43, when the SNR is less than 5, the Error Index is greater than 50% for any amplitude of nystagmus. The Error Index is about 30% with ten degrees of nystagmus amplitude at a SNR of seven (7).

Significant errors above 70% result at any amplitude of nystagmus when the SNR is less than four (4). In general, the missed-detection errors are larger than the false-detection errors (Fig. 44).

Results with Simulated Sinusoidal Pursuit Data

The analysis of simulated sinusoidal pursuit data without noise with the Harvard program produces no errors, *i.e.*, 0% Error Index. Figures 45 through 48 show the results of adding noise at three sinusoidal frequencies, 0.05, 0.10 and 0.50 Hz, for the five different nystagmus amplitudes. Figure 45 shows the results of adding noise to the 0.05 Hz sinusoidal nystagmus pursuit waveform. The results indicate that the Error Index is greater than 50% when the SNR is less than six (6). Figures 45 through 47 show that the program appears to be more efficient (*i.e.*, small Error Index) at the smaller saccadic displacements of one (1) and two (2) degrees. The worst performance at the 0.05 and 0.10 Hz sinusoids occurs with nystagmus amplitudes of five (5) degrees (Fig. 45 and 46). As the frequency of the sinusoidal base waveform is increased to 0.50 Hz, the performance of the program improves (Fig. 47). In general, the ability of the program to detect fast phases decreases as the nystagmus amplitude increases. Figure 47 shows that the best results (10% Error Index) at the 0.50 Hz sinusoid occur with two (2) degrees of nystagmus amplitude and a SNR of eight (8) or more. For the 0.50 Hz sinusoid, the Error Index exceeds 50% at a nystagmus amplitude of ten (10) degrees and a SNR less than five (5).

In summary, missed-detection errors are generally greater than false-detection errors (Fig. 48). The program produces better results at the higher frequency (0.50 Hz) and smaller amplitude of nystagmus. At SNR less than five (5), significant errors result at all frequencies and nystagmus amplitudes.

Results with Human Data

The Harvard program, as written, uses data collected at a sampling rate of 100 Hz. Sampling rate conversion was performed on the human data to modify the sampling rate from 200 Hz to 100 Hz. The Harvard program threshold parameters were set on the basis of the hand-scored data. The values of the threshold parameters are listed in Table VIII.

Table VIII. List of the presetting parameters for the Harvard program.

<u>Type of the test data</u>	<u>SLIM</u>	<u>TLIM</u>	<u>DISMAX</u>
Optokinetic	0.18	5.4	20.2
Sinusoidal Pursuit	0.14	3.0	4.0
Pseudorandom Pursuit	0.12	2.7	5.8

For the optokinetic data set, 13 events are manually identified as saccades; but, the Harvard program identifies 20 events as fast phases. Table IX summarizes the fast phase detection results with corresponding magnitude. Out of the twenty detected events, ten are false detections, resulting in a 50% false-detection error. Three (3) true events are missed detections, resulting in a 23% missed-detection error. The Error Index for the optokinetic data set is 37%.

Table IX. Results from the analysis of human optokinetic data by Harvard Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	8.0	detected
2	15.6	detected
3	18.0	missed
4	5.6	detected
5	18.0	detected
6	14.2	detected
7	12.4	detected
8	14.6	missed
9	13.6	detected
10	16.2	detected
11	19.2	detected
12	20.2	detected
13	5.4	missed

Table X summarizes the results of the evaluation of the human sinusoidal data set. The sinusoidal pursuit data set contains only five (5) fast phases. The Harvard program identified seven (7) events as fast phases of which six (6) are false detections and 4 are missed detections, resulting in 86% false-detection error and 80% missed-detection error. The Error Index for the sinusoidal data set is 83%, which is very high when compared to other types of data sets, *i.e.*, optokinetic and pseudorandom.

Table X. Results from analysis of human sinusoidal pursuit data by Harvard Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	3.0	detected
2	3.0	missed
3	3.0	missed
4	3.0	missed
5	4.0	missed

Table XI summarizes the results of the pseudorandom data set analysis. For the pseudorandom pursuit data set, 14 events are identified as true fast phases based on the

Table XI. Results from analysis of human pseudorandom data by Harvard Program.

<u>Fast Phase #</u>	<u>Amplitude (°)</u>	<u>Detection Result</u>
1	3.2	detected
2	2.7	missed
3	4.0	missed
4	4.5	detected
5	3.7	missed
6	4.6	detected
7	5.2	detected
8	3.8	detected
9	5.0	missed
10	4.8	detected
11	5.3	detected
12	5.8	detected
13	5.0	missed
14	3.0	detected

manually scored data. Four (4) events of the thirteen events detected by the Harvard program are false detections, resulting in a 30% false-detection error. Five (5) of the true fast phases are

missed detections, for a missed-detection error of 36%. The resulting Error Index for the pseudo-random pursuit data is 33%.

Both the simulated data and human data analyses show the inconsistency and high error percentages in the Harvard fast phase detection algorithm. One may argue about initial threshold settings not being optimized, but this implies that the program is not user friendly or automatic.

NASA Nystagmus Analysis Program (FPID)

The FPID program was created to automatically detect and eliminate fast phases from the EOG so that the slow phase characteristics of the EOG could be analyzed independently. For this study, modifications to the NASA FPID program included: 1.) output of timing information, and 2.) calculation of fast phase amplitude. Two versions of FPID were developed in order to meet the computing demands of research and clinical vestibular laboratories. The first version, **FPID.FOR**, is written in FORTRAN 77; the second version, **FPID.C**, is written in C for use on IBM or compatible personal computers with the Microsoft C Compiler version 5.1 (Microsoft Corp., Redmond, WA). The algorithm for EOG analysis is the same for the two versions; however, the format of the program output varies from system to system such that the code is not fully transportable. The FORTRAN version was developed with a VAX mainframe computer but could be compiled on any other FORTRAN compiler adhering to the VAX (DEC) standards. The C version was developed on an IBM-AT compatible computer. Several commands in **FPID.C** are not ANSI standard; therefore, users of compilers other than Microsoft C must consult the **FPID Software User's Manual** (Appendix B) before using the program. In both versions, **FPID.FOR** and **FPID.C**, fast phase timing information is saved on a formatted file. For additional information on the operation of the programs, consult the **FPID Software User's Manual** in the appendix.

Description of the Algorithm

Figure 49 is a flowchart of the algorithm for the analysis of the EOG. The data analysis algorithm, which is the same for all versions, has three main components: data processing, fast phase identification, and slow phase reconstruction. The data processing component consists of:

1. DC removal
2. Moving average smoothing, and
3. Digital filtering.

Initially, the user may choose to eliminate the DC component from the data by subtracting the mean of the data from each data point. Immediately following the removal of the DC offset, the data is smoothed with a five (5) point weighted moving average. Smoothing serves to low-pass filter the data and reduce incorrect identification of fast phases [4]. As illustrated in Table XII, the attenuation of the signal by the weighted moving average is negligible; the difference

Table XII. Amplitude of saccades in the optokinetic data set calculated manually and by FPID.

<u>Hand-Calculation</u> (°)	<u>Computer-Calculation</u> (°)	<u>Difference</u> (°)
8.0	7.15	0.85
15.6	15.34	0.26
18.0	17.15	0.85
5.6	5.95	0.35
18.0	17.56	0.44
14.2	13.76	0.44
12.4	12.23	0.17
14.6	14.51	0.09
13.6	12.90	0.70
16.2	15.70	0.50
19.2	18.56	0.64
20.2	19.27	0.93
5.4	5.03	0.37

between the manual and computer calculations of saccadic amplitude is less than one (1) degree. After the five-point smoother, a 15-point, FIR low-pass filter is applied to the signal. The filter coefficients in the original version of FPID were designed for a sampling rate of 120

Hz ($\Theta_c = \pi/2$) [4]. Even though these coefficients were not recalculated for the 200 Hz sampling rate, the higher filter cutoff did not affect the results significantly.

Following the digital processing of the EOG data, the fast phase identification process is performed. The algorithm for fast phase identification includes:

1. First difference
2. Velocity threshold-crossing detection, and
3. Zero-crossing detection.

The filtered EOG is differentiated with a fourth-order, first difference algorithm to get the velocity profile of the EOG data. Spikes in the velocity signal suggest the presence of a fast phase. A velocity threshold for detection of fast phases is determined from the root-mean-square (RMS) value of the velocity signal. The value of the threshold is twice the RMS value of the first derivative. The algorithm searches for threshold crossing data points as a means of detecting the fast events in the data. Once the threshold-crossing point is determined, the previous zero-crossing data point is searched to determine the beginning of a fast phase. The zero-crossing data point following the threshold-crossing data point is then searched to determine the end of the fast phase. The algorithm also determines the direction of the fast phase, up- or down-beating. By convention, right-beating saccades in the horizontal EOG are considered to be positive, and left-beating saccades are considered to be negative. The fast phase amplitude is also calculated at this stage. Previous studies have employed the same algorithm to accurately detect fast phases from human EOG data [4].

The third, and last, portion of the FPID program is the slow phase reconstruction. Slow phase reconstruction consists of:

1. Least-squares estimation
2. Height correction, and
3. Moving average smoothing.

Once the starting and ending times of each phase are determined, least-squares estimates of the slope and y-intercept are calculated from the ten points immediately preceding the fast phase. The estimates are then used to re-calculate the previously eliminated fast phase data points by linear fit. After completing the reconstruction, but before continuing the search for subsequent fast phases, a height correction factor is implemented to concatenate the remaining part of the signal to the reconstructed portion of the EOG at the correct amplitude. Once all the points in the signal have been analyzed, the reconstructed data may be further smoothed with the same five (5) point moving average previously employed. Fast phase timing information and amplitude are stored on file in a tabular format after the data set has been completely analyzed. Also stored on file is the reconstructed EOG.

Although FPID is an automatic EOG analysis program, it is dependent on data acquisition parameters such as sampling rate. The software code must provide flexibility and/or be easily modified; therefore, it may be used for data analysis in any laboratory. Flexibility has been achieved by implementation of subroutines and functions and portability of code. Flexibility allows users to implement the program without modifying the code. Unfortunately, some signal processing algorithms (*e.g.*, digital filtering) depend on parameters set during data collection. In this case, a code that is easily modified is absolutely necessary. In addition to the documentation of the program, the modular code in FPID, implemented through subroutines and functions, is easy to follow and simplifies the process of modifying the code.

Simulated Data Parameters

The simulated data analyzed by the FPID program had a sampling period of 5 msec. (*i.e.*, 200 Hz sampling frequency), and the length of each data file was 15 seconds. The simulated midposition gaze data were generated with saccades of one, two, three, five, and ten degrees. The simulated sinusoidal pursuit data also contained saccades of one, two, three, five, and ten degrees superimposed on sinusoids of 0.05, 0.10, and 0.50 Hz. The amplitude of the

sinusoidal pursuit data sets was 20 degrees. The data were not filtered with an anti-alias filter.

Results with Simulated Mid-position Gaze Data

The analysis with simulated EOG data was a comprehensive study of the performance of the program (*i.e.*, fast phase detection capability) as a function of SNR. Simulated mid-position gaze and sinusoidal pursuit data were generated using the NASDAT program developed for the study. The highest signal-to-noise ratio for the data was approximately 14; the lowest SNR was approximately 1.5. Table XIII illustrates the noise levels (in degrees) with the corresponding SNR for data sets with one (1) degree nystagmus. Initially, data sets with SNR as high as 80 were analyzed; however, the results did not show an improvement in the fast phase detection capability of FPID.

TABLE XIII. Noise amplitude and SNR for artificial data with one (1) degree saccades.

<u>Noise</u> (°)	<u>SNR</u>
0.0	∞
0.1	13.43
0.2	6.75
0.3	4.52
0.4	3.38
0.5	2.72
1.0	1.34

The typical fast phase detection capability of FPID for midposition gaze data with all the nystagmus amplitudes tested is illustrated in Figure 50. As shown, the Error Index is zero for SNR greater than four (4) and increases exponentially as the SNR falls below four. As illustrated in Figure 51, saccadic amplitude does not affect the detection capability of FPID. Individual missed-detection and false-detection errors, shown in Figure 52, further illustrate the independence of the fast phase identification algorithm from nystagmus magnitude. The results from the midposition gaze analysis show the remarkable consistency in fast phase detection by the FPID program.

Results with Simulated Sinusoidal Pursuit Data

In order to compare the results from simulated midposition gaze and simulated sinusoidal pursuit trials, the amplitude of the base sinusoid was not included in calculation of the SNR. The signal-to-noise ratio of the simulated sinusoidal pursuit data was calculated in the same manner as for the simulated midposition gaze data. Simulated midposition gaze and sinusoidal pursuit data were generated with the same saccadic amplitude and noise levels (*i.e.*, same SNR) to allow for comparison of results.

The results from the analysis of sinusoidal pursuit data are slightly different from those of gaze data. The analysis of sinusoidal data illustrates the effect of frequency and saccadic amplitude on the fast phase detection capability of FPID (Figs. 53 - 55). Missed and false-detection errors (individual errors) in the 0.50 Hz data are high when the saccadic amplitude is less than three (3) degrees (Figure 53). Although not depicted in Figure 53, individual errors in the 0.50 Hz data without noise (one and two degree nystagmus) are also very high. The individual errors decrease sharply for saccadic amplitudes greater than two (2) degrees.

The apparent inability of FPID to detect fast phases of small amplitude in the simulated 0.50 Hz sinusoidal data stems from a discrepancy between the velocity of the fast phases and the average velocity of the sinusoid. The FPID program uses a velocity threshold to detect fast phases; therefore, the program's capability to detect fast phases is dependent on the magnitude of the velocity of nystagmus and the RMS value of the velocity signal, not on nystagmus amplitude or sinusoid frequency. Results from the 0.05 Hz and 0.10 Hz data show that the individual errors and the Error Index profile are consistent. Figures 54 and 55 show exponentially increasing Error Index as the SNR falls below four (4). The average velocities of the slow phases in the 0.05 Hz and 0.10 Hz sinusoidal data were much slower than the fast phase velocities; therefore, the velocity threshold algorithm was capable of detecting fast phases of fast phase velocity of the nystagmus beats.

The analysis of simulated EOG data shows that FPID is very accurate and consistent in

detecting fast phases in midposition gaze data. The results from the sinusoidal data analysis suggest a discrepancy between simulated sinusoidal data with one (1) and two (2) degrees nystagmus beats and real EOG data. The simulated sinusoidal pursuit data with one (1) and two (2) degree nystagmus at 0.50 Hz indicates that the fast phase velocity was not much higher than the slow phase velocity, giving rise to high missed-detection errors.

Results with Human Subject Data

The three types of EOG data were analyzed using FPID: optokinetic, sinusoidal and pseudo-random pursuit. The summary of the results of the analysis of optokinetic data is presented in Table XIV. The results show that FPID does not miss any saccades, a 0% missed-detection error ratio, but falsely detects one (1) saccade, resulting in a 7.69% false-detection error ratio. The Error Index for this data set is 3.85%. The event falsely identified as a saccade by FPID is relatively large in amplitude, 5.2 degrees, with a low average velocity of 115 degrees/sec.

Table XIV. Results from the analysis of human optokinetic data by FPID Program.

<u>Fast Phase #</u>	<u>Hand-scored</u>			<u>Computer</u>	
	<u>Start</u>	<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>
1	0.350	0.380	8.0	0.360	0.395
2	0.850	0.900	15.6	0.860	0.910
3	1.735	1.790	18.0	1.745	1.795
4	2.155	2.185	5.6	2.135	2.200
5	2.710	2.775	18.0	2.725	2.780
6	3.580	3.650	14.2	3.590	3.655
7	4.495	4.550	12.4	4.505	4.565
8	5.635	5.705	14.6	5.645	5.715
9	6.580	6.640	13.6	6.595	6.650
10	7.790	7.865	16.2	7.800	7.865
11	8.975	9.035	19.2	8.985	9.045
12	10.125	10.190	20.2	10.140	10.190
13	10.665	10.695	5.4	10.675	10.705

Results of the sinusoidal pursuit analysis are summarized in Table XV. The program performs poorly; three saccades are missed, resulting in a 60% missed-detection error, and 56 saccades are falsely identified, resulting in a 95.24% false-detection error. The Error Index for

the human sinusoidal data set is 77.62%.

Table XV. Results from the analysis of human sinusoidal pursuit data by FPID Program.

<u>Fast Phase #</u>	<u>Hand-scored</u>			<u>Computer</u>		<u>SNR</u>
	<u>Start</u>	<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>	
1	2.855	2.870	3.0	-	-	1.67
2	5.125	5.145	3.0	5.140	5.160	1.93
3	7.100	7.120	3.0	-	-	1.50
4	9.425	9.445	3.0	-	-	1.61
5	9.545	9.570	4.0	9.550	9.585	2.03

A likely explanation for the high Error Index for the human sinusoidal pursuit EOG is the low SNR of the signal with respect to the amplitude of the saccades. The noise on the sinusoidal EOG was extracted by subtracting the 0.10 Hz base sinusoid from the signal, and the average RMS value of the remaining noise, shown in Figure 56, was calculated. The amplitude of the sinusoid subtracted from the EOG was determined according to the amplitude of the EOG after the DC shift on the EOG was eliminated. The root-mean-square value of each saccade was calculated to determine the SNR of each saccade using Equation 12:

$$SNR_{\text{saccade}} = \frac{RMS_{\text{saccade}}}{RMS_{\text{noise}}} \quad (12)$$

In the analysis of simulated sinusoidal pursuit data at 0.10 Hz, a signal-to-noise ratio of four (4) or less generates an Error Index higher than 30% (Fig. 55). The highest SNR of the fast phases in the sinusoidal data set is 2.03 (Table XV); therefore, the high noise level with respect to the nystagmus beats was responsible for the high Error Index observed in the subject data sinusoidal EOG.

The pseudorandom data test set contained less noise than the sinusoidal pursuit test data set; and the results were much better than those from the sinusoidal data. The results summarized in Table XVI reflect much lower error ratios; one (1) saccade is missed, resulting in a 7.14% missed-detection error, and no false identifications occur (0% false-detection error). The Error

Index for the pseudorandom data set is 3.57%.

The results from simulated data analysis prove the consistency of FPID in detecting fast phases under controlled conditions; however, the results from the subject data analysis illustrate the complexities of automatic EOG analysis using computers. The human data contains nystagmus of varying amplitude and velocity, which, when coupled with artifact, generate higher than expected errors. It seems that the sensitivity of the automatic velocity threshold technique for fast phase detection depends on the type of data being analyzed.

Table XVI. Results from the analysis of human pseudorandom data by FPID Program.

<u>Fast Phase #</u>	<u>Start</u>	<u>Hand-scored</u>		<u>Computer</u>	
		<u>End</u>	<u>Amplitude</u>	<u>Start</u>	<u>End</u>
1	0.603	0.650	3.2	0.635	0.660
2	0.865	0.885	2.7	0.865	0.885
3	1.150	1.175	4.0	1.155	1.185
4	1.465	1.490	4.5	1.475	1.495
5	1.710	1.730	3.7	-	-
6	2.505	2.525	4.6	2.515	2.535
7	2.710	2.735	5.2	2.720	2.745
8	3.005	3.030	3.8	3.015	3.035
9	3.725	3.755	5.0	3.735	3.760
10	4.065	4.090	4.8	4.070	4.095
11	4.295	4.325	5.3	4.305	4.330
12	4.975	5.005	5.8	4.985	5.010
13	5.205	5.235	5.0	5.215	5.245
14	5.750	5.775	3.0	5.760	5.835

CHAPTER VI

DISCUSSION - COMPARISON OF RESULTS

Evaluation of the various software used in detection of saccades (fast phase component of nystagmus) included recoding nystagmus analysis programs from six vestibular laboratories in the United States to FORTRAN computer language, generating a simulated EOG data base, and evaluating each program with an Error Index criteria as a function of SNR. The concept of an Error Index was employed as the criterium for evaluating the vestibular analysis software program with the best performance, *i.e.*, the smallest percentage of Error Index at the lowest SNR.

Figures 21 and 51 indicate that the UCLA SINUXEC AND NASA FPID programs are extremely consistent in detecting midposition gaze saccades and that the percent Error Index is independent of the magnitude of saccadic jump. Both programs appear to have comparable performances, *i.e.*, less than a 5% Error Index for a SNR of four (4), although the UCLA SINUXEC program (Fig. 22) has a low missed-detection error (less than 2% missed detection at a SNR of 3.5) but a high false-detection error (8% false detection at a SNR of 3.5) while the NASA FPID program (Fig. 52) has equal detection errors (missed and false detections) of 4.0% error at a SNR of 2.5.

For the simulated midposition gaze test, Figure 12 shows that the USAF/SAM program performs best with two (2) and three (3) degree of saccadic jumps, but the performance deteriorates (larger Error Index) as the saccadic jump increases from five (5) to ten (10) degrees. Similar to the NASA program, the missed-detection error equals the false-detection error for the same value of saccadic jump. The midposition gaze test results from the MIT program (Fig. 35) show a consistent percent Error Index of about 5% at a SNR of eight (8) at saccadic jumps of one, two, three, five, and ten degrees. Similar to the UCLA program, the MIT program (Fig. 34) performance had less than a 2% missed-detection error with SNR

greater than four at all saccadic jump magnitudes but had a high false-detection error rate at SNR less than ten (10). The low missed-detection error coupled with a high false-detection error in both the UCLA SINUXEC and MIT M2MI86 programs may be attributed to low settings of threshold values. As a threshold is decreased more events are detected, which results in less missed detections of true events, but at the expense of an increase in detection of false events.

In contrast to the UCLA and MIT programs, the Harvard program midposition gaze test results (Fig. 44) show a lower false-detection error than missed-detection error. This indicates that perhaps threshold settings may be decreased.

Figure 57 shows the Error Index for the midposition gaze test at three (3) degrees of saccadic jump for five of the programs evaluated in this study. As illustrated by this figure, the NASA FPID and the UCLA SINUXEC programs have comparable performance; whereas, the USAF/SAM and the MIT programs perform comparably but not as good as the UCLA and NASA programs.

The results of the nystagmus analysis programs with simulated sinusoidal tests performed at 0.05, 0.10 and 0.50 Hz for the various saccadic jump/magnitudes (one, two, three, five, and ten degrees) show patterns similar to results obtained with the simulated gaze data. The USAF/SAM (Figs. 15 - 17), UCLA SINUXEC (Figs. 23 - 25), and NASA FPID (Figs. 53 - 55) programs show better performance (lower Error Index as a function of SNR) at the lower frequencies (0.05 and 0.10 Hz) than at the higher frequency (0.50 Hz). In contrast, the MIT M2MI86 (Figs. 37 - 39) and the Harvard programs (Figs. 45 - 47) perform better at the higher frequency (0.50 Hz) than at the lower frequencies (0.05 and 0.10 Hz). The variation in performance as the sinusoidal frequency varies is probably due to the digital or optimal filter designs used in the various programs and the parameters of the simulated sinusoidal data. The USAF/SAM and the Harvard programs perform better with small saccadic jump magnitudes of two (2) and three (3) degrees than at larger saccadic magnitudes of five (5) and ten (10)

degrees. The performance of the UCLA SINUXEC, the MIT M2MI86 and NASA FPID programs are consistent and independent of the saccadic jump magnitude, except at 0.50 Hz, when the fast phase and slow phase velocities are approximately the same value and less than 100 deg./sec. (i.e., one (1) degree saccadic jumps have a maximum average velocity of 66 deg./sec. and the slow phase component of the 0.50 Hz sinusoid has maximum average velocity of approximately 46 deg./sec.). The NASA/FPID program (Fig. 53) shows poor performance with simulated sinusoidal pursuit data at 0.50 Hz and saccadic jumps of one (1) and two (2) degrees. The decreased performance may be the result of the average fast phase velocity of the simulated nystagmus being approximately the same as the average velocity of the slow component. The fast phase velocity of the one (1) and two (2) degree nystagmus beats in the simulated data may not be high enough for adequate detection by programs that employ velocity filters or any other type of fast phase detection based on the velocity of the EOG. This situation is not likely to happen with real EOG data because the fast phase of nystagmus beats during sinusoidal pursuit will generally have much greater velocity than the slow phase. The fast phase velocities of the three, five and ten degree nystagmus beats are considerably higher than the average velocity of the slow component of the 0.05, 0.10, and 0.50 Hz sinusoids (Table I) and did not generate as many errors as the data sets with lower magnitude saccadic jumps.

The performance of the six programs may be compared for the 0.10 Hz sinusoid with saccadic amplitude of three (3) degrees (Fig. 58). The NASA FPID, UCLA SINUXEC and USAF/SAM programs have comparable performances.

The results of program performance with three types of data acquired from subject EOG responses to optokinetic, sinusoidal pursuit, and pseudorandom test stimuli are summarized in Table XVII. As indicated in the table, the NASA FPID program had the best performance, or lowest Error Index, 3.8% for the optokinetic test and 3.6% for the pseudorandom test. The UCLA SINUXEC had a 7.1% Error Index for the pseudorandom test. All programs

performed poorly with the sinusoidal pursuit test data. An examination of the signal-to-noise ratio indicated that the SNR was about two (2) when the RMS value of the saccade was compared to the RMS value of the noise. Results with the simulated sinusoidal data indicated that a high percentage of Error Index can be anticipated when the signal-to-noise ratio is less than four (4).

Table XVII. Comparison of the Program Performance with Subject Data

PROGRAM \ DATA TYPE	OPTOKINETIC			SINUSOIDAL			PSEUDORANDOM		
	MDE	FDE	EI	MDE	FDE	EI	MDE	FDE	EI
USAF/SAM	38.5	61.9	51.2	60.0	94.4	72.2	42.9	42.9	42.9
UCLA (SINUXEC)	69.2	66.7	68.0	100	0	50.0	14.3	0	7.1
MIT (M2MI86)	15.4	84.1	49.7	0	96.0	48.0	7.4	62.9	35.0
HARVARD	23.0	50.0	37.0	80.0	86.0	83.0	36.0	30.0	33.0
NASA (FPID)	0	7.69	3.8	60.0	95.2	77.6	7.1	0	3.6

MDE: Missed Detection Error; FDE: False Detection Error; EI: Error Index. All values are in percentage.

CHAPTER VII

CONCLUSIONS

The objective of this study was to evaluate the performance of six nystagmus analysis programs. The programs were evaluated on their ability to detect the occurrence of saccadic (fast phase) movements. An Error Index was used to evaluate and compare the accuracy of the nystagmus analysis programs. The conclusions of this study are:

1. The NASA-FPID and the UCLA-SINUXEC programs have the most consistent performance.
2. The performance (smallest Error Index at the lowest signal-to-noise ratio) of the NASA-FPID and UCLA-SINUXEC programs is less than 5% error in index for signal-to-noise ratios greater than four (4).
3. Both programs perform better with sinusoidal pursuit frequencies less than 0.50 Hz than at frequencies greater than or equal to 0.50 Hz.
4. The NASA-FPID and UCLA-SINUXEC programs had the best performance of the six (6) programs when tested with pseudorandom EOG data from human subjects.

CHAPTER VIII

RECOMMENDATIONS

The traditional approach used in the analysis of a system is to determine the system's transfer function by observing the system response to a predetermined deterministic excitation. Knowledge of the system's transfer function permits prediction of the system response to other deterministic excitations. Although linear systems analysis is the most popular and useful analytical method used to evaluate explicit mathematical models, *i.e.*, the transfer function of a biological system, the method applies only to systems that are linear.

The most commonly used excitation signals in systems analyses are single frequency sinusoid, step, and impulse functions. The use of other than deterministic input signals did not fully develop until computers with a large number handling capacity became available. The Wiener method, nonlinear system identification by observation of a system's response to zero-mean Gaussian white noise, enables one to predict the response (linear and nonlinear) of the system to any input; however, the method is not without problems.

A pseudorandom input stimulus lies between the simple single frequency sinusoid and the complex random Gaussian white noise; but, according to Victor and Shapley [17] can produce results similar to the Wiener method. An excellent description of the sum of sinusoids method capability for both linear and nonlinear analyses in the frequency domain was presented by Victor and Shapley. The crux of the method is the use of a modulation signal that is a sum-of-sine waves and the measurement of nonlinear response of the system as cross-talk between the input frequencies. Thus, the sum-of-sinusoids method uses a deterministic rather than random Gaussian noise input stimulus, is not restricted to a small signal regime like the harmonic input method, and is different from the use of a single sinusoid method. The difference is based on the response to a sum-of-sinusoids stimulus which may contain the nonlinear interaction terms, *i.e.*, the sum and difference frequency terms which are the intermodulation of cross-talk

frequencies.

Expansion of the automated analysis of optokinetic and vestibular responses from linear system analysis to nonlinear analysis using the sum-of-sinusoids technique is recommended.

CHAPTER IX

REFERENCES

- [1] Lessard, C.S. Survey of vestibular laboratories: Baseline for standardization. NASA Research Grant NAG 9-303, Phase I Report, June 1988.
- [2] Baloh, R.W., C.G.Y. Lau, A.W. Sills and V. Honrubia. The patterns of eye movements during physiologic vestibular nystagmus in man. Trans. Am Acad. Oph. Otol. **34**(2):339-347, 1977.
- [3] Baloh, R.W., A.W. Sills, W.E. Kumley and V. Honrubia. Quantitative measurement of saccade amplitude, duration and velocity. Neurology **25**(11):1065-1070, 1975.
- [4] Lessard, Charles S. General purpose algorithms for characterization of slow and fast phase nystagmus. NASA/ASEE Summer Faculty Fellowship Report. August, 1986.
- [5] Coats, A.C. Electronystagmography. In: Physiological measures of the audio-vestibular system, L.J. Bradford, *Ed.*, Academic Press, New York, 1975.
- [6] Uemura, T., J. Suzuki, J. Hozawa and S.M. Highstein. Neuro-otological examination. University Park Press, Baltimore, MD, 1977.
- [7] Honrubia, V. and R.W. Baloh. Reaction time and accuracy of the saccadic eye movements of normal subjects in a moving-target task. Aviat. Space Environ. Med. **47**(11):1165-1167, 1976.
- [8] Wong, Wing Chan. Nonlinear analysis of human optokinetic (smooth pursuit tracking) responses. Ph.D. Thesis, Texas A&M University, College Station, Texas, Dec. 1990.
- [9] E. J. Engelken, K. W. Stevens, and J. W. Wolfe. Application of digital filters in the processing of eye movement data. Behav. Res. Meth. & Instrum. **14**:314-319, 1982.
- [10] K. W. Stevens. Personal communication, July 1989.
- [11] Baloh, R.W., W.E. Kumley and V. Honrubia. Algorithm for analyses of saccadic eye movements using a digital computer. Aviat. Space Environ. Med. **47**(5):523-527, 1976.
- [12] A.W. Sills, V. Honrubia and W.E. Kumley. Algorithm for the multi-parameter analysis of nystagmus using a digital computer. Aviat. Space Environ. Med. **46**(7):934-942, 1975.
- [13] Schmidt, Glenn F. Vestibular function test program analysis. Masters Thesis, Texas A&M University, College Station, Texas, May 1990.
- [14] Massoumnia, M.A. Detection of fast phase of nystagmus using digital filtering. Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1983.

- [15] Wall, C., III and F.O. Black. Algorithms for the clinical analysis of nystagmus eye movements. IEEE/BME Trans. Biomed. Eng. 28(9):638-646, 1981.
- [16] Baloh, R.W., L. Langhoffer, V. Honrubia and R.D. Yee. Microcomputer analysis of eye movements. Diagnosis 2(4):52-57, 1984.
- [17] Victor, J.D., R. Shapley, and B. W. Knight. Nonlinear analysis of cat retinal ganglion cells in the frequency domain. Proc. Nat. Acad. Sci. USA 74:3068-3072, 1977.



APPENDIX A
SOFTWARE USER'S MANUAL:
SYNTHETIC DATA GENERATION PROGRAM (NASDAT.FOR)

I. INTRODUCTION

The NASDAT.FOR program has been developed to generate 15 seconds of simulated EOG data. The source code was written in FORTRAN 77 for a VAX computer system (8650), but the code is transportable to other systems with adequate memory capacity. Two types of EOG data are generated, midposition gaze and sinusoidal pursuit. The program is designed to:

1. generate data containing fast phases of variable amplitude;
2. generate data with variable slow phase to fast phase data point ratio; and
3. add white noise with fixed maximum amplitude.

The data point ratio is defined as the ratio of the number fast phase points to the number slow phase points in each nystagmus beat, which determines the length of the nystagmus beats in the simulated data. The time interval between sample points in the simulated EOG data is determined by the sampling rate selected by the user.

The simulated EOG data is generated in a three step process:

1. generation of the base wave;
2. superposition of nystagmus beats; and
3. addition of noise.

The first step is always completed because data may be generated without superimposed nystagmus. It is important to note that midposition gaze data without nystagmus is a straight line at zero (0) degrees. The second step is completed if nystagmus beats are to be superimposed on the base wave. In this case the amplitude of nystagmus and the average fast phase velocity must be entered by the user. The third step is completed only if noise is to be superimposed on the base wave. If noise is added to the data, the user must enter the maximum amplitude of noise. If noise is added to the data, an option is provided to select a simulated anti-alias filter is provided. The three lowpass filters with cutoff frequencies at 33 Hz, 45 Hz and 80 Hz are implemented with recursive digital filters designed for a 200 Hz sampling rate. If the user selects a different sampling rate (f_s), the cutoff frequency of these filters will be changed by a factor (ef) described in Equation (1)

$$ef = 200 + f_s \quad (1)$$

If a sampling rate other than 200 Hz is selected the new cutoff frequency (f_{cn}) of the simulated anti-alias filter will change according to Equation 2.

$$f_{cn} = f_c + ef \quad (2)$$

II. SOFTWARE REQUIREMENTS

The software and data files required for generation of simulated electro-oculogram (EOG) data are listed below:

- 1.- NASDAT.FOR: Source code for generation of simulated data
- 2.- FREQ.DAT: Random frequency data file
- 3.- MGxx.DAT: Noise amplitude data files (xx=maximum amplitude of noise)

The source code, NASDAT.FOR, is listed in this report. The random frequency data file, FREQ.DAT, is required to ensure the whiteness of the noise added to the signal. The noise amplitude data files contain noise data with maximum amplitude determined by the xx suffix. Table A1 lists the available noise amplitudes and their corresponding file names.

Table A1. Maximum amplitude of noise and corresponding data files.

<u>Noise Amplitude</u> (°)	<u>Filename</u>	<u>Noise Amplitude</u> (°)	<u>Filename</u>
0.1	MG0_2.DAT	3.0	MG3.DAT
0.2	MG0_1.DAT	4.0	MG4.DAT
0.3	MG0_3.DAT	5.0	MG5.DAT
0.4	MG0_4.DAT	6.0	MG6.DAT
0.5	MG0_5.DAT	7.0	MG7.DAT
1.0	MG1.DAT	8.0	MG8.DAT
1.5	MG1_5.DAT	9.0	MG9.DAT
2.0	MG2.DAT	10.0	MG10.DAT
2.5	MG2_5.DAT		

III. INPUT REQUIREMENTS

Data parameters common to both types of data (midposition gaze and sinusoidal) are entered by the user at the beginning of the program. This input determines:

1. type of data to generate;
2. sampling rate of data collection (used to determine the time interval between data points);
3. if nystagmus is to be added to the data, the amplitude of nystagmus,
4. if noise is to be added to the data, the maximum amplitude of noise; and
5. if anti-alias filtering is to be simulated, the cutoff of the lowpass digital filter.

Before the data is generated the user must enter the following information:

1. the name of the data file and fast phase timing file,
2. the format of the data file
 - a. formatted or unformatted
 - b. sequential or direct access
 - c. time index or EOG data only
3. if input data parameters are to be included with the EOG data file.

The user can select the format of the EOG data files such that the data can be analyzed by any data analysis program; however, timing files are formatted with sequential access. The input parameters are included at the end of the EOG data files for user reference. A detailed description of the operation of the program appears in the text this report (Chapter IV). The input parameters specific to the type of data being generated (*i.e.*, midposition gaze or sinusoidal pursuit) are discussed in the following.

A. Midposition Gaze

Before midposition gaze data can be generated the user must input the average slow phase velocity and the direction of the fast phases. The slow phase velocity (in deg./sec.) is necessary to determine the slope of the slow phase. The user also selects the direction of the fast phases (upward or downward). The user can select upward saccades or downward saccades. The direction of the fast phases is constant in each midposition gaze data file; therefore, the direction of the slow phase is always opposite to the direction of the fast phase. Tables A2 and A3 are examples of the prompts and responses required for generating

midposition gaze data.

Table A2. Prompts and Responses for generating midposition gaze with one (1) degree nystagmus without noise (Fig. A1).

1- MIDPOSITION GAZE

2- SINUSOIDAL PURSUIT

3- EXIT

(ALWAYS USE CAPITAL LETTERS)

ENTER TEST NUMBER FOR DATA TO BE GENERATED: 1
ENTER SAMPLING RATE (SAMPLES/SECOND): 200
ENTER 1 FOR SUPERIMPOSED NYSTAGMUS: 1
ENTER NYSTAGMUS AMPLITUDE (DEGREES): 1
ENTER FAST PHASE VELOCITY (DEGREES/SECOND): 400
ENTER SLOW PHASE VELOCITY (DEGREES/SECOND): 40
ENTER 1 IF RIGHT -(UP)- OR 0 IF LEFT- (DOWN)-BEATING: 1
ENTER HORIZONTAL MIDPOSITION DATA FILE NAME
(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): 'NYS50.DAT'
ENTER FORM OF OUTPUT FILES (FORMATTED OR UNFORMATTED)
(IN APOSTROPHES): 'FORMATTED'
ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)
(IN APOSTROPHES): 'SEQUENTIAL'
DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'Y'
DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'N'
DO YOU WANT SUPERIMPOSED RANDOM NOISE
(Y OR N IN APOSTROPHES): 'N'
ENTER OUTPUT NYSTAGMUS TIME FILENAME
(IN APOSTROPHES): 'NYS10.TIM'

B. Sinusoidal Pursuit

The input parameters necessary before generating sinusoidal pursuit data are:

1. amplitude of the sinusoid;
2. frequency of the sinusoid;
3. initial direction of the sinusoid (upward or downward);
4. slope difference threshold; and
5. the data point ratio.

The amplitude of the sinusoid, in degrees, is determined by the user. The frequency of the sinusoid, in Hz, is important because it allows the user to evaluate the frequency dependency of the EOG analysis program. The program is not limited to any specific frequency range such that the user may select any frequency. However, the user needs to generate at least one full

Table A3. Prompts and responses required for generating midposition gaze data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise (Fig A2).

1- MIDPOSITION GAZE

2- SINUSOIDAL PURSUIT

3- EXIT

(ALWAYS USE CAPITAL LETTERS)

ENTER TEST NUMBER FOR DATA TO BE GENERATED: 1
ENTER SAMPLING RATE (SAMPLES/SECOND): 200
ENTER 1 FOR SUPERIMPOSED NYSTAGMUS: 1
ENTER NYSTAGMUS AMPLITUDE (DEGREES): 5
ENTER FAST PHASE VELOCITY (DEGREES/SECOND): 400
ENTER SLOW PHASE VELOCITY (DEGREES/SECOND): 40
ENTER 1 IF RIGHT (UP) OR 0 IF LEFT (DOWN) BEATING: 1
ENTER HORIZONTAL MIDPOSITION DATA FILE NAME
(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): 'NYS505.DAT'
ENTER FORM OF OUTPUT FILES (FORMATTED OR UNFORMATTED)
(IN APOSTROPHES): 'FORMATTED'
ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)
(IN APOSTROPHES): 'SEQUENTIAL'
DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'Y'
DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'N'
DO YOU WANT SUPERIMPOSED RANDOM NOISE
(Y OR N IN APOSTROPHES): 'Y'
ENTER OUTPUT NYSTAGMUS TIME FILENAME
(IN APOSTROPHES): 'NYS10.TIM'
DO YOU WANT THE LOWPASS NOISE FILTER
(Y OR N IN APOSTROPHES)
(ONLY GOOD FOR 200 HZ SAMPLING RATE): 'N'
ENTER UPPER MAGNITUDE BOUND: 0.5
ENTER RANDOM NOISE FREQUENCY FILENAME
(IN APOSTROPHES): 'FREQ.DAT'
ENTER RANDOM NOISE MAGNITUDE FILENAME
(IN APOSTROPHES): 'MAG05.DAT'

cycle of data and keep in mind that the data files are 15 seconds in length when selecting the frequency of the sinusoid. The user may select the initial direction of the sinusoid (upward or downward), but, in this case, the direction of the fast phases cannot be selected by the user because fast phases are automatically generated by the program in a direction opposite to the direction of the slow phase of the sinusoid. The slope difference threshold is necessary to avoid generating nystagmus beats at the maxima and minima of the sinusoid (an adequate value for the slope difference threshold is 0.001). An expanded description of the slope difference threshold is presented in the description of the operation of the program in Chapter IV. The

data point ratio must be entered so that the appropriate location of the fast phases can be determined. Since the slow phase in a sinusoidal pursuit task can occur in either direction (upward and downward), the data point ratio is used to determine where and in which direction a fast phase will occur. As with midposition data, sinusoidal data are generated in tenths of degrees. Tables A4 and A5 are examples of the prompts and responses required for generating sinusoidal pursuit data.

Table A4. Prompts and responses required for generating sinusoidal pursuit data with one (1) degree nystagmus without noise (Fig. A3).

1- MIDPOSITION GAZE

2- SINUSOIDAL PURSUIT

3- EXIT

(ALWAYS USE CAPITAL LETTERS)

ENTER TEST NUMBER FOR DATA TO BE GENERATED: 2
ENTER SAMPLING RATE (SAMPLES/SECOND): 200
ENTER 1 FOR SUPERIMPOSED NYSTAGMUS: 1
ENTER NYSTAGMUS AMPLITUDE (DEGREES): 5
ENTER FAST PHASE VELOCITY (DEGREES/SECOND): 400
ENTER SINUSOID MAGNITUDE (DEGREES): 20
ENTER FREQUENCY (HERTZ): .5
ENTER HORIZONTAL MIDPOSITION DATA FILE NAME
(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): 'SIN505.DAT'
ENTER FORM OF OUTPUT FILES (FORMATTED OR UNFORMATTED)
(IN APOSTROPHES): 'FORMATTED'
ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)
(IN APOSTROPHES): 'SEQUENTIAL'
DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'Y'
DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'N'
DO YOU WANT SUPERIMPOSED RANDOM NOISE
(Y OR N IN APOSTROPHES): 'N'
ENTER DATA POINT RATIO (INT >1): 10
ENTER THE SLOPE DIFFERENCE THRESHOLD: .001
ENTER OUTPUT NYSTAGMUS TIME FILENAME
(IN APOSTROPHES): 'SIN101.TIM'

Table A5. Prompts and responses required for generating sinusoidal pursuit data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise (Fig. A4).

1- MIDPOSITION GAZE
2- SINUSOIDAL PURSUIT
3- EXIT
(ALWAYS USE CAPITAL LETTERS)

ENTER TEST NUMBER FOR DATA TO BE GENERATED: 2
ENTER SAMPLING RATE (SAMPLES/SECOND): 200
ENTER 1 FOR SUPERIMPOSED NYSTAGMUS: 1
ENTER NYSTAGMUS AMPLITUDE (DEGREES): 5
ENTER FAST PHASE VELOCITY (DEGREES/SECOND): 400
ENTER SINUSOID MAGNITUDE (DEGREES): 20
ENTER FREQUENCY (HERTZ): .5
ENTER HORIZONTAL MIDPOSITION DATA FILE NAME
(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): 'SIN505.DAT'
ENTER FORM OF OUTPUT FILES (FORMATTED OR UNFORMATTED)
(IN APOSTROPHES): 'FORMATTED'
ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)
(IN APOSTROPHES): 'SEQUENTIAL'
DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'Y'
DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE
(Y OR N IN APOSTROPHES): 'N'
DO YOU WANT SUPERIMPOSED RANDOM NOISE
(Y OR N IN APOSTROPHES): 'Y'
ENTER DATA POINT RATIO (INT >1): 10
ENTER THE SLOPE DIFFERENCE THRESHOLD: .001
ENTER OUTPUT NYSTAGMUS TIME FILENAME
(IN APOSTROPHES): 'SIN101.TIM'
DO YOU WANT THE LOWPASS NOISE FILTER
(Y OR N IN APOSTROPHES): 'N'
DO YOU WANT THE LOWPASS NOISE FILTER
(Y OR N IN APOSTROPHES)
(ONLY GOOD FOR 200 HZ SAMPLING RATE): 'N'
ENTER UPPER MAGNITUDE BOUND: 0.5
ENTER RANDOM NOISE FREQUENCY FILENAME
(IN APOSTROPHES): 'FREQ.DAT'
ENTER RANDOM NOISE MAGNITUDE FILENAME
(IN APOSTROPHES): 'MAG05.DAT'

IV. PROGRAM OUTPUT

A. Midposition Gaze

The program automatically creates two data files, a timing file and a simulated EOG data file. The timing file contains the timing information (in seconds) of the fast events in the EOG file. The output is in tabular form with the number of the fast event, the slow phase starting times listed in one column and the fast phase starting times listed in a second column. The

timing information is in seconds, and the magnitude of the midposition data is in tenths of degrees. If the user selected the inclusion of the data parameters with output file, these parameters are appended to the EOG data file. Figure A1 is a sample output of simulated midposition gaze data with one (1) degree nystagmus without noise. Figure A2 is a sample output of simulated midposition gaze data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise.

B. Sinusoidal Pursuit

As with the midposition data, timing and simulated sinusoidal pursuit data files are automatically generated by the NASDAT program. The timing file contains a column with the slow phase ending time and another column with the fast phase ending time. The occurrence of nystagmus beats in the simulated sinusoidal pursuit data is different from that in the midposition gaze data, since fast phases in the sinusoidal data are opposite in direction to the slow phases, and fast phases occur in either upward or downward directions. Figure A3 is a sample of simulated sinusoidal pursuit data with (1) degree nystagmus without noise, and Figure A4 is a sample of simulated sinusoidal pursuit data with one (1) degree nystagmus with 0.5 degrees (maximum amplitude) noise.

APPENDIX B
SOFTWARE USER'S MANUAL:
FAST PHASE IDENTIFICATION PROGRAM (FPID)

I. INTRODUCTION

The following instructions will guide users of the Fast Phase Identification Program, version FPID.FOR or FPID.C. The FPID.FOR program is written in FORTRAN 77 for a VAX computer system (8650); whereas, the FPID.C program is written in C for an IBM-AT compatible machine (Packard Bell ISVT-286) with the Microsoft C Compiler (version 5.1). Although the hardware requirements are different for each version, the algorithm for fast phase identification and slow phase reconstruction is the same. A complete description of the operation of the program appears in the text of this report.

II. SOFTWARE REQUIREMENTS

Compilation of the source code is required before implementation. FPID.FOR is written in FORTRAN 77 such that any FORTRAN compiler adhering to this standard can be used to compile and run the program. The code in FPID.C is not fully transportable and, as written, requires compilation with the Microsoft C Compiler (version 5.1 or higher) (Microsoft Corp., Redmond, WA). Other compilers may be used after minor modifications to the source code of FPID.C.

One software requirement common to both versions of the FPID program is the format of the input EOG data files. Data files must contain only EOG data in a single column arrays. Also, the data to be analyzed by FPID must be floating point numbers (*i.e.*, real). Timing data or data point index should not be included in the data file. If a data file does not meet these criteria, the data must be transformed for the FPID program to operate properly. Two data acquisition parameters are necessary so that the data can be read properly. The user must enter the sampling rate and the length of the sample record to be analyzed (in seconds). The sampling rate is used to determine the time interval between data points and, in conjunction with the length of the sample record, to determine how many data points from the EOG data file are to be analyzed. The total number of points to be analyzed must be less than or equal to

the total number of points in the file. The EOG data files must be in the same directory where the FPID is stored.

A. FPID.FOR

The FORTRAN version of FPID was developed on VAX computer; therefore, the size of the data arrays is not limited as they are in FPID.C. The data arrays used in the program and their description are listed in Table B1. The size of these arrays has been arbitrarily set at 10000 and may be changed according to the user requirements. The FPID.FOR code is standard FORTRAN 77; therefore, the program should be fully transportable.

Table B1. Description of data arrays in FPID.FOR

<u>NAME</u>	<u>DESCRIPTION</u>
HEOG	Original EOG data
YHEOG	DC shifted and smoothed EOG
FDEOG	First derivative of EOG
SDEOG	Second derivative of EOG
SPEOG	Reconstructed EOG
SPYHEOG	Smoothed reconstructed EOG
TIME	Timing for each data point

B. FPID.C

The code for FPID.C was translated from the FORTRAN version and modified to take advantage of the modular structure of the C language. The program employs dynamic memory allocation by calling `calloc()` (ANSI standard) from the `get_memory()` function. Dynamic memory allocation is used to increase the size of the data arrays beyond the 32 Kbytes data stack limit. However, the size of the arrays is limited by the memory capability of the system. For a system with 640 Kbytes of memory data arrays are limited to approximately 3000 elements each. For systems with more memory, the size of the EOG data arrays (HEOG, YHEOG, FDEOG, TIMES) may be changed by altering the value of the variable **SIZEB** in the

#define statement at the beginning of the source file. The timing information arrays (RTBEG, RTEND, LTBEG, LTEND) are initialized in the variable **SIZEA** in the **#define** statement at the beginning of the source file. Each of the timing information arrays is initialized to 100 elements, but the value of **SIZEA** may be changed if an unusual occurrence of saccadic events is observed in the data. Table B2 lists the names and description of the arrays in FPID.C. A complete description of variables used in the program is included at the end of the source code.

Table B2. Description of data arrays in FPID.FOR

<u>NAME</u>	<u>DESCRIPTION</u>
HEOG	Original and smoothed EOG data
YHEOG	DC shifted and filtered EOG
FDEOG	First derivative of EOG
TIMES	Timing for data
RTBEG	Beginning of right-going fast phase
RTEND	End of right-going fast phase
LTBEG	Beginning of left-going fast phase
LTEND	End of left-going fast phase

Screen graphics is available with the FPID.C version. Plotting of data is accomplished by calling graphics functions supplied with the Microsoft Compiler (v.5.1) from the **plot()** function in FPID.C. Plotting by FPID.C is possible on IBM AT compatible computers with: 1.) an Enhanced Graphics Adapter (EGA) card, and 2.) the Microsoft C Compiler (version 5.1), otherwise the calls to the **plot()** function must be deleted from the code before compiling.

III. PROGRAM OUTPUT

A. FPID.FOR

The program automatically creates two output data files, a timing file and a reconstructed EOG data file. The timing file contains the timing information (in seconds) of the saccadic

events in the EOG file. The output is in tabular form with the starting and ending times, and the direction of each event listed. The timing file has a .TIM extension (*e.g.*, *filename.TIM*). The reconstructed EOG file contains the EOG slow phases after removal of the fast phases, and the name of the reconstructed data file is the same as that of the original data file with '-R' concatenated at the end, *e.g.*, *filename-R.DAT*. The output data file is a single column array of EOG data, neither time nor index information is included in the output EOG file. The format of the timing and reconstructed data files is the default FORTRAN I/O file standard. The timing and reconstructed EOG data files are stored in the directory where FPID.FOR is stored.

B. FPID.C

The program automatically creates two output data files, a timing file and a reconstructed EOG data file. The timing file contains the timing information (in seconds) of the saccadic events in the EOG file. The output is in tabular form with the starting and ending times, and the direction of each event listed. The timing file has a .TIM extension (*e.g.*, *filename.TIM*). The reconstructed EOG file contains the EOG slow phases after removal of the fast phases, and the name of the reconstructed EOG output data file is the same as that of the original data file with '_R' concatenated at the end, *e.g.*, *filename_R.DAT*. The reconstructed data file is a single column array of EOG data, neither time or nor index information is included in the output EOG file. The timing and reconstructed EOG data files are saved in the directory where FPID.C is stored.

A limitation of the DOS system in which FPID.C was developed is in the length of the filenames. The name of a file must not be longer than eight (8) characters plus a three (3) character extension, *e.g.*, *filename.dat*. Therefore, the name of the original data file should not be longer than six (6) characters, otherwise the filenames of the original data and reconstructed data may be identical. To prevent overwriting data, the program checks for the existence of a file before creating it for saving data. If a file already exists, the user is prompted to authorize

the overwriting of an already existing data file or renaming the file where data will be saved. However, the length of the data files must be limited to the conditions listed above to prevent accidental loss of data.

Graphic output is available to users of FPID.C during different stages of data analysis provided an EGA card and Microsoft C compiler (version 5.1 or higher) are installed. Another feature of FPID.C is immediate printout of the timing file after the data has been analyzed.

APPENDIX C
PROGRAM LISTING: NASDAT.FOR

```

0001 C          PROGRAM NASDAT.FOR
0002 C
0003 C PROGRAM ARTIFICIALLY GENERATES MIDPOSITION GAZE AND
0004 C SINUSOIDAL PURSUIT EOG DATA
0005 C DEFINE VARIABLES
0010 REAL SR,MAXDEV,MINDEV,NYSAMP,FVEL,SVEL,SINMAG,SINFREQ
0011 INTEGER CHOICE,NYS,DIR
0012 C
0013 C DETERMINE WHICH TEST DATA TO GENERATE
0014 4 PRINT 5
0015 5 FORMAT (//,T10,'1- MIDPOSITION GAZE',//,T10,
0016 *      '2- SINUSOIDAL PURSUIT',//,
0017 *      T10,'3- EXIT',//,T10,'(ALWAYS USE CAPITAL LETTERS)')//)
0018 7 PRINT *,'ENTER TEST NUMBER FOR DATA TO BE GENERATED: '
0019 READ *, CHOICE
0020 C
0021 C INPUT THE APPROPRIATE INFORMATION
0022 IF (CHOICE.EQ.3) GOTO 200
0023 PRINT *, 'ENTER SAMPLING RATE (SAMPLES/SECOND): '
0024 READ *, SR
0025 IF (CHOICE.EQ.1.OR.CHOICE.EQ.2) THEN
0026 10 PRINT *, 'ENTER 1 FOR SUPERIMPOSED NYSTAGMUS: '
0027 READ *, NYS
0028 IF (NYS.LE.0.AND.NYS.NE.1) GO TO 10
0029 IF (NYS.EQ.1) THEN
0030 PRINT *, 'ENTER NYSTAGMUS AMPLITUDE (DEGREES): '
0031 READ *, NYSAMP
0032 PRINT *, 'ENTER FAST PHASE VELOCITY (DEGREES/SECOND): '
0033 READ *, FVEL
0034 IF (CHOICE.EQ.1) THEN
0035 PRINT *, 'ENTER SLOW PHASE VELOCITY (DEGREES/SECOND): '
0036 READ *, SVEL
0037 15 PRINT *, 'ENTER 1 IF RIGHT(UP)-DOWNBEATING OR 0 IF
0038 * LEFT(DOWN)-UPBEATING: '
0039 READ *, DIR
0040 IF (DIR.NE.0.AND.DIR.NE.1) GO TO 15
0041 ENDIF
0042 ELSEIF (NYS.EQ.0) THEN
0043 NYSAMP=0
0044 FVEL=0
0045 SVEL=0
0046 DIR=0
0047 ENDIF
0048 ENDIF
0049 IF (CHOICE.EQ.2) THEN
0050 PRINT *,'ENTER SINUSOID MAGNITUDE (DEGREES): '
0051 READ *, SINMAG
0052 PRINT*, 'ENTER FREQUENCY (HERTZ): '
0053 READ *, SINFREQ
0054 ENDIF
0055 IF (CHOICE.EQ.1) THEN
0056 CALL MIDPOSITION (SR,NYS,NYSAMP,FVEL,SVEL,DIR)
0057 ELSEIF (CHOICE.EQ.2) THEN
0058 CALL SINUSOID (SR,SINMAG,SINFREQ,NYS,NYSAMP,FVEL,SVEL)

```

```

0059     ENDIF
0060     GO TO 4
0061     200 END

```

FUNCTIONS AND SUBROUTINES REFERENCED

Type Name	Type Name
	MIDPOSITION
	SINUSOID
0001	C
0002	C
0003	SUBROUTINE MIDPOSITION (SR,NYS,NYSAMP,FVEL,SVEL,DIR)
0004	REAL SR,NYSAMP,FVEL,SVEL,TS,FS,SS,CY,POS(20000),TIME,FD,SD
0005	REAL
0006	FR,NOISMAG,SIGRMS,NOISRMS,SNRATIO,NOISFREQ,FTEMP,STEMP
0007	REAL X(-10:20000),Y(-10:20000),U(-10:20000),W(-10:20000),UB
0008	REAL A,B,C
0009	INTEGER NYS,DIR,TOTSAMP,FSAMP,SSAMP,CYCLES,I,J,K,M
0010	CHARACTER MIDHOR*20, MIDVER*20, ND*5, NZ*5, ANS*3, HEAD*3
0011	CHARACTER NOISE*3, FORMT*14, ACC*12, RANDFR*20, RANDMAG*20
0012	CHARACTER SACTIME*20,FL*5,TYPE*10
0013	C
0014	PRINT *, 'ENTER HORIZONTAL MIDPOSITION DATA FILE NAME'
0015	PRINT *, '(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): '
0016	READ *, MIDHOR
0017	PRINT *, 'ENTER FORM OF OUTPUT FILES (FORMATTED OR UNFORMATTED)'
0018	PRINT *, '(IN APOSTROPHES): '
0019	READ *, FORMT
0020	PRINT *, 'ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)'
0021	PRINT *, '(IN APOSTROPHES): '
0022	READ *, ACC
0023	PRINT *, 'DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE'
0024	PRINT *, '(Y OR N IN APOSTROPHES): '
0025	READ *, HEAD
0026	PRINT *, 'DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE'
0027	PRINT *, '(Y OR N IN APOSTROPHES): '
0028	READ *, ANS
0029	PRINT *, 'DO YOU WANT SUPERIMPOSED RANDOM NOISE'
0030	PRINT *, '(Y OR N IN APOSTROPHES): '
0031	READ *, NOISE
0032	OPEN (UNIT=2,FILE=MIDHOR,STATUS='NEW',ACCESS=ACC,
0033	* FORM=FORMT,RECL=65)
0034	TS=15.0*SR
0035	TOTSAMP=INT(TS)
0036	DO 10 I=1,TOTSAMP
0037	POS(I)=0
0038	10 CONTINUE
0039	FD=0
0040	SD=0
0041	IF (NYS.EQ.1) THEN
	PRINT *, 'ENTER OUTPUT NYSTAGMUS TIME FILENAME'


```

0042     PRINT *, '(IN APOSTROPHES): '
0043     READ *, SACTIME
0044     OPEN
          (UNIT=3, FILE=SACTIME, STATUS='NEW', ACCESS='SEQUENTIAL')
0045     FS=(NYSAMP*0.1)/((FVEL*0.1)/SR)
0046     FSAMP=INT(FS)
0047     SS=(NYSAMP*0.1)/((SVEL*0.1)/SR)
0048     SSAMP=INT(SS)
0049     FD=FS/SR
0050     SD=SS/SR
0051     CY=TOTSAMP/(FSAMP+SSAMP)
0052     CYCLES=INT(CY)
0053     WRITE (3,20)
0054 20    FORMAT (T5,'CYCLE',T15,'SCSTTIME',T25,'FCSTTIME')
0055     DO 50 J=1,CYCLES
0056         K=J-1
0057         STEMP=(1+(K*(SSAMP+FSAMP)))/SR
0058         FTEMP=((1+SSAMP)+(K*(SSAMP+FSAMP)))/SR
0059         WRITE (3,25) J,STEMP,FTEMP
0060 25    FORMAT (T5,I3,T15,F7.4,T25,F7.4)
0061         DO 30 I=1,SSAMP
0062             M=I+(K*(FSAMP+SSAMP))
0063             POS(M)=(SVEL*0.1*I)/SR
0064 30    CONTINUE
0065         DO 40 I=1,FSAMP
0066             M=(I+SSAMP)+(K*(FSAMP+SSAMP))
0067             POS(M)=(NYSAMP*0.1)-((FVEL*0.1*I)/SR)
0068 40    CONTINUE
0069 50    CONTINUE
0070         IF (DIR.EQ.1) THEN
0071             DO 60 I=1,TOTSAMP
0072                 POS(I)=-POS(I)
0073 60    CONTINUE
0074         ENDIF
0075     ENDIF
0076     SIGNRMS=0.0
0077     DO 61 I=1,TOTSAMP
0078         SIGNRMS=SIGNRMS+(POS(I)**2.0)
0079 61    CONTINUE
0080     NOISRMS=0.0
0081     IF (NOISE.EQ.'N') THEN
0082         FIL='N'
0083     ELSEIF (NOISE.EQ.'Y') THEN
0084         PRINT *, 'DO YOU WANT TO USE THE LOWPASS NOISE FILTER ?'
0085         PRINT *, '(Y OR N IN APOSTROPHES)'
0086         PRINT *, '(ONLY GOOD FOR 200 HZ SAMPLING RATE): '
0087         READ *, FIL
0088         IF (FIL.EQ.'Y') THEN
0089             PRINT *, 'DO YOU WANT MIT (35 HZ), UCLA (40 HZ), OR NASA
0090 *      (80 HZ) FILTER?'
0091             PRINT *, '(ENTER MIT, UCLA, OR NASA IN APOSTROPHES): '
0092             READ *, TYPE
0093         ENDIF
0094         PRINT *, 'ENTER UPPER MAGNITUDE BOUND: '

```

```

0095     READ *, UB
0096     PRINT *, 'ENTER RANDOM NOISE FREQUENCY FILENAME'
0097     PRINT *, '(IN APOSTROPHES): '
0098     READ *, RANDFR
0099     OPEN (UNIT=3,FILE=RANDFR,STATUS='OLD')
0100     PRINT *, 'ENTER RANDOM NOISE MAGNITUDE FILENAME'
0101     PRINT *, '(IN APOSTROPHES): '
0102     READ *, RANDMAG
0103     OPEN (UNIT=4,FILE=RANDMAG,STATUS='OLD')
0104     DO 70 I=1,TOTSAMP
0105         READ (3,68) NOISFREQ
0106         READ (4,68) NOISMAG
0107     68  FORMAT (T10,F10.7)
0108     69  IF (NOISMAG.GT.UB) THEN
0109         NOISMAG=NOISMAG/10.0
0110         GOTO 69
0111     ENDIF
0112         FR=(I*NOISFREQ*3.14152654)/SR
0113         X(I)=SIN(FR)*NOISMAG*0.1
0114         IF (FIL.EQ.'N') THEN
0115             NOISRMS=NOISRMS+(X(I)**2.0)
0116             POS(I)=POS(I)+X(I)
0117         ENDIF
0118     70  CONTINUE
0119         IF (FIL.EQ.'Y') THEN
0120             DO 71 I=-5,0
0121                 X(I)=0.0
0122                 Y(I)=0.0
0123                 U(I)=0.0
0124                 W(I)=0.0
0125     71  CONTINUE
0126             DO 72 I=1,TOTSAMP
0127                 IF (TYPE.EQ.'MIT') THEN
0128                     A=3.32588
0129                     B=-2.818231
0130                     C=4.507649
0131                 ELSEIF (TYPE.EQ.'UCLA') THEN
0132                     A=1.784893
0133                     B=-2.180351
0134                     C=3.604542
0135                 ELSEIF (TYPE.EQ.'NASA') THEN
0136                     A=-1.789568
0137                     B=-0.9373096
0138                     C=1.273122
0139                 ENDIF
0140                 Y(I)=(X(I)+2.0*X(I-1)+X(I-2)+A*Y(I-1)+B*Y(I-2))/C
0141     72  CONTINUE
0142             DO 74 I=1,TOTSAMP
0143                 IF (TYPE.EQ.'MIT') THEN
0144                     A=3.32588
0145                     B=-1.355153
0146                     C=5.970727
0147                 ELSEIF (TYPE.EQ.'UCLA') THEN
0148                     A=1.784893

```

```

0149         B=-0.9469664
0150         C=4.837927
0151     ELSEIF (TYPE.EQ.'NASA') THEN
0152         A=-1.789568
0153         B=-0.6464876
0154         C=1.563944
0155     ENDIF
0156     U(I)=(Y(I)+2.0*Y(I-1)+Y(I-2)+A*U(I-1)+B*U(I-2))/C
0157 74 CONTINUE
0158     DO 76 I=1,TOTSAMP
0159         IF (TYPE.EQ.'MIT') THEN
0160             A=3.32588
0161             B=-0.5104444
0162             C=6.815436
0163         ELSEIF (TYPE.EQ.'UCLA') THEN
0164             A=1.784893
0165             B=-0.2348712
0166             C=5.550022
0167         ELSEIF (TYPE.EQ.'NASA') THEN
0168             A=-1.789568
0169             B=-0.4785813
0170             C=1.73185
0171         ENDIF
0172         W(I)=(U(I)+2.0*U(I-1)+U(I-2)+A*W(I-1)+B*W(I-2))/C
0173         NOISRMS=NOISRMS+(W(I)**2.0)
0174         POS(I)=POS(I)+W(I)
0175 76 CONTINUE
0176     ENDIF
0177     ENDIF
0178     SIGNRMS=(SIGNRMS/TOTSAMP)**(1.0/2.0)
0179     SIGNRMS=10*SIGNRMS
0180     NOISRMS=(NOISRMS/TOTSAMP)**(1.0/2.0)
0181     NOISRMS=10*NOISRMS
0182     IF (NOISE.EQ.'Y') THEN
0183         SNRATIO=SIGNRMS/NOISRMS
0184     ELSEIF(NOISE.EQ.'N') THEN
0185         SNRATIO=0.0
0186     ENDIF
0187     IF (DIR.EQ.1) THEN
0188         ND='RIGHT'
0189         NZ='UP'
0190     ELSEIF (DIR.EQ.0) THEN
0191         ND='LEFT'
0192         NZ='DOWN'
0193     ENDIF
0194 90 FORMAT (1X,'C THE SAMPLING FREQUENCY IS ',F8.3,' HZ',/,
0195     *'C THE NYSTAGMUS AMPLITUDE IS ',F4.1,' DEGREES',/,
0196     *'C THE FAST PHASE VELOCITY IS ',F6.1,' DEGREES/SECOND',/,
0197     *'C THE SLOW PHASE VELOCITY IS ',F6.1,' DEGREES/SECOND',/,
0198     *'C THE FAST PHASE DURATION IS ',F7.4,' SECONDS',/,
0199     *'C THE SLOW PHASE DURATION IS ',F7.4,' SECONDS',/,
0200     *'C THE NYSTAGMUS DIRECTION IS ',A,'-BEATING',/,
0201     *'C THE SIGNAL RMS VALUE IS ',F9.4,/,
0202     *'C THE NOISE RMS VALUE IS ',F9.4,/,

```

```

0203      *C THE SIGNAL TO NOISE RATIO IS ',F9.4,/,
0204      *C WAS THE LOWPASS FILTER USED ? ',A,/,
0205      *C ALL DATA IS STORED IN 1/10 DEGREES',/)
0206      IF (ANS.EQ.'Y') THEN
0207      DO 120 I=1,TOTSAMP
0208          TIME=I/SR
0209          IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0210              WRITE (2,110,REC=I) TIME, POS(I)
0211          ELSEIF (HEAD.EQ.'N'.AND.ACC.EQ.'DIRECT') THEN
0212              WRITE (2,110,REC=I) TIME, POS(I)
0213          ELSEIF (ACC.EQ.'SEQUENTIAL') THEN
0214              WRITE (2,110) TIME, POS(I)
0215          ENDIF
0216      110  FORMAT (T10,F8.4,T25,F10.6)
0217      120  CONTINUE
0218          ELSEIF (ANS.EQ.'N') THEN
0219          DO 140 I=1,TOTSAMP
0220              IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0221                  WRITE (2,130,REC=I) POS(I)
0222              ELSEIF (HEAD.EQ.'N'.AND.ACC.EQ.'DIRECT') THEN
0223                  WRITE (2,130,REC=I) POS(I)
0224              ELSEIF (ACC.EQ.'SEQUENTIAL') THEN
0225                  WRITE (2,130) POS(I)
0226              ENDIF
0227      130  FORMAT (T10,F10.6)
0228      140  CONTINUE
0229          ENDIF
0230          IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0231              WRITE (2,90,REC=TOTSAMP+1) SR,NYSAMP,FVEL,SVEL,FD,SD,ND,
0232              *          SIGNRMS,NOISRMS,
0233              *          SNRATIO,FIL
0234          ELSEIF (HEAD.EQ.'Y'.AND.ACC.EQ.'SEQUENTIAL') THEN
0235              WRITE (2,90) SR,NYSAMP,FVEL,SVEL,FD,SD,ND,SIGNRMS,
0236              *          NOISRMS,SNRATIO,FIL
0237          ENDIF
0238      RETURN
0239      END

```

```

0001  C
0002  C
0003  SUBROUTINE SINUSOID
(SR,SINMAG,SINFREQ,NYS,NYSAMP,FVEL,SVEL)
0004  REAL SR,SINMAG,SINFREQ,NYSAMP,FVEL,SVEL,TS,FS,FR,DIFF
0005  REAL POS(0:20000),TIME,SIGNRMS,NOISRMS,SQSIGN,FD,TRESH
0006  REAL NOISMAG, NOISFREQ,SNRATIO,FP(0:500),FMAG
0007  REAL TP,X(-10:20000),Y(-10:20000),PTEMP,NSBG,NSED
0008  REAL U(-10:20000),W(-10:20000),UB,A,B,C
0009  INTEGER NYS,TOTSAMP,FSAMP,SSAMP,CYCLES,I,J,K,M,WEND,EDIR
0010  INTEGER INOIS(-20:5000),RLOPT,IRATIO,ISTART,IEND,NOISEND
0011  CHARACTER SINHOR*20,SINVER*20,ANS*3,HEAD*3,NOISE*3,FIL*5
0012  CHARACTER FORMT*14,ACC*12,RANDFR*20,RANDMAG*20,TYPE*10
0013  CHARACTER SACTIME*20

```

```

0014 C
0015 PRINT *, 'ENTER HORIZONTAL PURSUIT DATA FILE NAME'
0016 PRINT *, '(IN APOSTROPHES AND NO MORE THAN 10 CHARACTERS): '
0017 READ *, SINHOR
0018 PRINT *, 'ENTER FORM OF OUTPUT FILES (FORMATTED OR
UNFORMATTED)'
0019 PRINT *, '(IN APOSTROPHES): '
0020 READ *, FORMT
0021 PRINT *, 'ENTER ACCESS OF OUTPUT FILES (DIRECT OR SEQUENTIAL)'
0022 PRINT *, '(IN APOSTROPHES): '
0023 READ *, ACC
0024 PRINT *, 'DO YOU WANT A HEADER PRINTED WITH THE OUTPUT FILE'
0025 PRINT *, '(Y OR N IN APOSTROPHES): '
0026 READ *, HEAD
0027 PRINT *, 'DO YOU WANT TIME INCLUDED IN THE OUTPUT FILE'
0028 PRINT *, '(Y OR N IN APOSTROPHES): '
0029 READ *, ANS
0030 PRINT *, 'DO YOU WANT THE SINUSOID TO GO RIGHT (UP) OR LEFT
0031 * (DOWN) FIRST?'
0032 PRINT *, '(UP=1, DOWN=-1)'
0033 READ *, RLOPT
0034 PRINT *, 'DO YOU WANT SUPERIMPOSED RANDOM NOISE'
0035 PRINT *, '(Y OR N IN APOSTROPHES): '
0036 READ *, NOISE
0037 OPEN (UNIT=2,FILE=SINHOR,STATUS='NEW',ACCESS=ACC,
0038 * FORM=FORMT,RECL=65)
0039 IF (NYS.EQ.1) THEN
0040 PRINT *, 'ENTER THE DATA POINT RATIO (INT.>1): '
0041 READ *, IRATIO
0042 PRINT *, 'ENTER THE SLOPE DIFFERENCE THRESHOLD: '
0043 READ *, TRESH
0044 ENDIF
0045 TS=15.0*SR
0046 TOTSAMP=INT(TS)
0047 DO 10 I=0,TOTSAMP
0048 POS(I)=0
0049 10 CONTINUE
0050 SQSIGN=0.0
0051 DO 13 I=0,TOTSAMP+1
0052 FR=(I*SINFREQ*2*3.14152654)/SR
0053 POS(I)=SIN(FR)*SINMAG*0.1*RLOPT
0054 13 CONTINUE
0055 FD=0
0056 IF (NYS.EQ.1) THEN
0057 PRINT *, 'ENTER OUTPUT NYSTAGMUS TIME FILENAME'
0058 PRINT *, '(IN APOSTROPHES)'
0059 READ *, SACTIME
0060 OPEN
(UNIT=3,FILE=SACTIME,STATUS='NEW',ACCESS='SEQUENTIAL')
0061 FS=(NYSAMP*SR)/FVEL
0062 FSAMP=INT(FS)
0063 IF (FS.LT.1.0) FSAMP=1
0064 SSAMP=FSAMP*IRATIO
0065 FD=FSAMP/SR

```

```

0066      DO 17 I=1,FSAMP
0067          FP(I)=0.1*I*FVEL/SR
0068          IF (FP(I).GT.NYSAMP) FP(I)=NYSAMP
0069      17 CONTINUE
0070          FMAG=0.0
0071          ISTART=0
0072          IEND=SSAMP
0073          INOIS(0)=0
0074          INOIS(1)=IEND
0075          K=2
0076          WEND=TOTSAMP-FSAMP
0077      18 DO 19 I=ISTART,IEND
0078          POS(I)=POS(I)+FMAG
0079      19 CONTINUE
0080          TP=POS(IEND)-POS(IEND-1)
0081          IF (TP.GT.0) IDIR=-1
0082          IF (TP.LT.0) IDIR=1
0083          IF (ABS(TP).LT.TRESH) THEN
0084              ISTART=IEND+1
0085              IEND=ISTART+SSAMP
0086              INOIS(K-1)=IEND
0087              IF (IEND.GT.WEND) GO TO 22
0088              GO TO 18
0089          ENDIF
0090          PTEMP=POS(IEND+FSAMP)
0091          DO 20 J=1,FSAMP
0092              I=IEND+J
0093              POS(I)=POS(IEND)+IDIR*FP(J)
0094      20 CONTINUE
0095          ISTART=IEND+FSAMP+1
0096          IEND=ISTART+SSAMP
0097          INOIS(K)=ISTART
0098          INOIS(K+1)=IEND
0099          K=K+2
0100          FMAG=POS(I)-PTEMP
0101          IF (IEND.LE.WEND) GO TO 18
0102      22 DO 24 I=ISTART,TOTSAMP
0103          POS(I)=POS(I)+FMAG
0104      24 CONTINUE
0105          NOISEND=K-1
0106          INOIS(K-1)=TOTSAMP
0107          WRITE (3,30)
0108      30 FORMAT (T5,'CYCLE',T15,'FCSTTIME',T25,'FCEDTIME')
0109          J=0
0110          DO 40 I=1,NOISEND,2
0111              J=J+1
0112              NSBG=INOIS(I)/SR
0113              NSED=INOIS(I+1)/SR
0114              WRITE (3,35) J,NSBG,NSED
0115      35 FORMAT (T5,I3,T15,F7.4,T25,F7.4)
0116      40 CONTINUE
0117          ENDIF
0118          SIGNRMS=0.0
0119          DO 54 I=1,TOTSAMP

```

```

0120     SIGNRMS=SIGNRMS+(POS(I)**2.0)
0121 54 CONTINUE
0122     NOISRMS=0.0
0123     IF (NOISE.EQ.'N') THEN
0124         FIL='N'
0125     ELSEIF (NOISE.EQ.'Y') THEN
0126         PRINT *,'DO YOU WANT TO USE THE LOWPASS NOISE FILTER ?'
0127         PRINT *,'(Y OR N IN APOSTROPHES)'
0128         PRINT *,'(ONLY GOOD FOR 200 HZ SAMPLING RATE)'
0129         READ *, FIL
0130         IF (FIL.EQ.'Y') THEN
0131             PRINT *,'DO YOU WANT MIT (35 HZ), UCLA (40 HZ), OR NASA
0132 *           (80 HZ) FILTER?'
0133             PRINT *,'(ENTER MIT, UCLA OR NASA IN APOSTROPHES)'
0134             READ *, TYPE
0135         ENDIF
0136         PRINT *,'ENTER UPPER MAGNITUDE BOUND'
0137         READ *, UB
0138         PRINT *,'ENTER RANDOM NOISE FREQUENCY FILENAME'
0139         PRINT *,'(IN APOSTROPHES)'
0140         READ *, RANDFR
0141         OPEN (UNIT=3,FILE=RANDFR,STATUS='OLD')
0142         PRINT *,'ENTER RANDOM NOISE MAGNITUDE FILENAME'
0143         PRINT *,'(IN APOSTROPHES)'
0144         READ *, RANDMAG
0145         OPEN (UNIT=4,FILE=RANDMAG,STATUS='OLD')
0146         DO 63 I=1,TOTSAMP
0147             READ (3,61) NOISFREQ
0148             READ (4,61) NOISMAG
0149 61     FORMAT (T10,F10.7)
0150 62     IF (NOISMAG.GT.UB) THEN
0151             NOISMAG=NOISMAG/10.0
0152             GOTO 62
0153         ENDIF
0154         FR=(I*NOISFREQ*2*3.14152654)/SR
0155         X(I)=SIN(FR)*NOISMAG*0.1
0156         IF (FIL.EQ.'N') THEN
0157             NOISRMS=NOISRMS+(X(I)**2.0)
0158             POS(I)=POS(I)+X(I)
0159         ENDIF
0160 63 CONTINUE
0161         IF (FIL.EQ.'Y') THEN
0162             DO 64 I=-5,0
0163                 X(I)=0.0
0164                 Y(I)=0.0
0165                 U(I)=0.0
0166                 W(I)=0.0
0167 64 CONTINUE
0168             DO 65 I=1,TOTSAMP
0169                 IF (TYPE.EQ.'MIT') THEN
0170                     A=3.32588
0171                     B=-2.818231
0172                     C=4.507649
0173                 ELSEIF (TYPE.EQ.'UCLA') THEN

```

```

0174      A=1.784893
0175      B=-2.180351
0176      C=3.604542
0177      ELSEIF (TYPE.EQ.'NASA') THEN
0178          A=-1.789568
0179          B=-0.9373096
0180          C=1.273122
0181      ENDIF
0182      Y(I)=(X(I)+2.0*X(I-1)+X(I-2)+A*Y(I-1)+B*Y(I-2))/C
0183 65    CONTINUE
0184      DO 66 I=1,TOTSAMP
0185          IF (TYPE.EQ.'MIT') THEN
0186              A=3.32588
0187              B=-1.355153
0188              C=5.970727
0189          ELSEIF (TYPE.EQ.'UCLA') THEN
0190              A=1.784893
0191              B=-0.9469664
0192              C=4.837927
0193          ELSEIF (TYPE.EQ.'NASA') THEN
0194              A=-1.789568
0195              B=-0.6464876
0196              C=1.563944
0197          ENDIF
0198          U(I)=(Y(I)+2.0*Y(I-1)+Y(I-2)+A*U(I-1)+B*U(I-2))/C
0199 66    CONTINUE
0200      DO 67 I=1,TOTSAMP
0201          IF (TYPE.EQ.'MIT') THEN
0202              A=3.32588
0203              B=-0.5104444
0204              C=6.815436
0205          ELSEIF (TYPE.EQ.'UCLA') THEN
0206              A=1.784893
0207              B=-0.2348712
0208              C=5.550022
0209          ELSEIF (TYPE.EQ.'NASA') THEN
0210              A=-1.789568
0211              B=-0.4785813
0212              C=1.73185
0213          ENDIF
0214          W(I)=(U(I)+2.0*U(I-1)+U(I-2)+A*W(I-1)+B*W(I-2))/C
0215          NOISRMS=NOISRMS+(W(I)**2.0)
0216          POS(I)=POS(I)+W(I)
0217 67    CONTINUE
0218      ENDIF
0219      ENDIF
0220      SIGNRMS=(SIGNRMS/TOTSAMP)**(1.0/2.0)
0221      SIGNRMS=10*SIGNRMS
0222      NOISRMS=(NOISRMS/TOTSAMP)**(1.0/2.0)
0223      NOISRMS=10*NOISRMS
0224      IF (NOISE.EQ.'Y') THEN
0225          SNRATIO=SIGNRMS/NOISRMS
0226      ELSEIF (NOISE.EQ.'N') THEN
0227          SNRATIO=0.0

```



```

0228     ENDIF
0229     70 FORMAT (1X,'C  THE SAMPLING FREQUENCY IS ',F8.3,' HZ' /,
0230     *'C  THE NYSTAGMUS AMPLITUDE IS ',F4.1,' DEGREES' /,
0231     *'C  THE FAST PHASE VELOCITY IS ',F6.1,' DEGREES/SECOND' /,
0232     *'C  THE FAST PHASE DURATION IS ',F7.4,' SECONDS' /,
0233     *'C  THE SINUSOID MAGNITUDE IS ',F5.1,' DEGREES' /,
0234     *'C  THE SINUSOID FREQUENCY IS ',F8.3,' HZ' /,
0235     *'C  THE SIGNAL RMS VALUE IS ',F9.4 /,
0236     *'C  THE NOISE RMS VALUE IS ',F9.4 /,
0237     *'C  THE SIGNAL TO NOISE RATIO IS ',F9.4 /,
0238     *'C  WAS THE LOWPASS FILTER USED ? ',A /,
0239     *'C  ALL DATA IS STORED IN 1/10 DEGREES' /)
0240     IF (ANS.EQ.'Y') THEN
0241     DO 100 I=1,TOTSAMP
0242         TIME=I/SR
0243         IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0244             WRITE (2,80,REC=I) TIME, POS(I)
0245         ELSEIF (HEAD.EQ.'N'.AND.ACC.EQ.'DIRECT') THEN
0246             WRITE (2,80,REC=I) TIME,POS(I)
0247         ELSEIF (ACC.EQ.'SEQUENTIAL') THEN
0248             WRITE (2,80) TIME, POS(I)
0249         ENDIF
0250     80  FORMAT (T10,F8.4,T25,F10.6)
0251     100 CONTINUE
0252     ELSEIF(ANS.EQ.'N') THEN
0253     DO 120 I=1,TOTSAMP
0254         IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0255             WRITE (2,110,REC=I) POS(I)
0256         ELSEIF (HEAD.EQ.'N'.AND.ACC.EQ.'DIRECT') THEN
0257             WRITE (2,110,REC=I) POS(I)
0258         ELSEIF (ACC.EQ.'SEQUENTIAL') THEN
0259             WRITE (2,110) POS(I)
0260         ENDIF
0261     110  FORMAT (T10,F10.6)
0262     120 CONTINUE
0263     ENDIF
0264     IF (HEAD.EQ.'Y'.AND.ACC.EQ.'DIRECT') THEN
0265         WRITE (2,70,REC=TOTSAMP+1) SR,NYSAMP,FVEL,FD,SINMAG,
0266     *           SINFREQ,SIGNRMS,NOISRMS,
0267     *           SNRATIO,FIL
0268     ELSEIF (HEAD.EQ.'Y'.AND.ACC.EQ.'SEQUENTIAL') THEN
0269         WRITE (2,70) SR,NYSAMP,FVEL,FD,SINMAG,SINFREQ,
0270     *           SIGNRMS,NOISRMS,SNRATIO,FIL
0271     ENDIF
0272     RETURN
0273     END

```

COMPILATION STATISTICS

```

Run Time:      5.16 seconds
Elapsed Time:  8.33 seconds
Page Faults:   1308
Dynamic Memory: 1248 pages

```

APPENDIX D
PROGRAM LISTING: FPID.C

```

1 /* */
2 /*          fpid.c          */
3 /* */
4 /* */
5 /*      EOG Fast Phase identification and Slow Phase Reconstruction.      */
6 /* */
7 /*      Code conversion of fpid.for from FORTRAN to C: May 1990.      */
8 /*      Converted by: Karen J. Poindexter and Carlos A. Rodriguez-Garcia  */
9 /* */
10 /* */
11 /* */
12 /*******/
13
14
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include <conio.h>
19 #include <math.h>
20 #include <graph.h>
21
22 /*Global constants*/
23 #define SIZEA 100
24 #define SIZEB 3010
25
26
27 /*Global variables*/
28 FILE *fopen(), *in_file, *out_file;
29
30 float      *rtbeg, *rtend, *ltbeg, *ltend;
31 float      *heog, *yheog, *fdeog, *times;
32
33
34 main()
35 {
36     int      n, j, k, i, l, ln, p, ltcnt, rtcnt;
37     int      nco, var, frmt, istep, ilst, isrt, istp, islg, lterm;
38     int      isl, ilf, temp1, count, max_count, index=1;
39     short int yes, rtflag, ltflag, more=1;
40     float    length, xsum, xmean, xlgt, fdrms, fdrmsht, tht, hht,h;
41     float    ssmfd, ssfd, xn, dxsum, xilf, q, denom, slope, yint;
42     float    dysum, dxysum, dxssum, fsample, fpdur, vel, num_pts;
43     char     datafile[15], filename[15], ext[15], ch;
44     static char printer[]="PRN";
45
46     strcpy(ext,"exit");
47
48     printf("\n\n");
49
50     message();
51
52     do /*while (more != 0)*/
53     {
54         system("cls");

```

```

55  get_memory();
56
57  printf("\n\n\nENTER NAME OF EOG DATAFILE (without .ext) OR EXIT > ")
;
58  scanf("%s", filenam);
59  if (!(strcmpi (filenam,"exit")))
60  {
61    exit(0);
62  }
63
64
65  sprintf(datafile, "%s.dat", filenam);
66  printf("\n");
67
68  printf("Enter sampling rate (in Hz) > ");
69  scanf("%f", &fsample);
70
71  printf("Enter length of datafile (in seconds) > ");
72  scanf("%f", &length);
73  length=length*fsample;
74
75  system("cls");
76
77  open_read(datafile);
78
79  for (n = 0; n <= (int)length-1; n++)
80  {
81    fscanf(in_file, "%f", &heog[n]);
82    times[n]=(float)n*(1/fsample);
83  }
84
85  printf("Do you want to plot EOG now? (1- Yes) > ");
86  scanf("%d%c", &yes,&ch);
87  if (yes)
88    plot(length,fsample);
89
90
91  /*Computing the mean of the data - removing DC component*/
92  printf("\n\nDo you want to remove DC? (1- Yes) > ");
93  scanf("%d%c",&yes,&ch);
94
95  if (yes)
96  {
97    printf("Removing DC components...\n");
98    xsum = 0.0;
99    xlgt = length;
100   for(i = 1; i <= (int)length; i++)
101     xsum = xsum + heog[i];
102
103   xmean = xsum / length;
104
105   for(i = 1; i <= (int)length; i++)
106     heog[i] = heog[i] - xmean;
107

```

```

108     }
109
110
111
112 /* Averaging the data*/
113     smooth_data(length);
114
115
116 /*Filtering data*/
117     filter_data(length);
118
119 /* sprintf(datafile,"%s_s.dat",filenam);
120     open_write(datafile);
121 */
122
123     for (n=1; n<=(int)length ;n++)
124         fprintf(out_file,"%f\n",yheog[n]);
125     fclose(out_file);
126
127
128 /*First derivative*/
129     istep = 1;
130     h = (float)istep / fsample;
131     printf("\nComputing Fourth Order (5 pt.) Central Difference... \n
132 ");
133     for (i = 3; i <= (int)length-2; i++)
134         fdeog[i] = ((yheog[i-2] - yheog[i+2])
135             + (8.0*(yheog[i+1] - yheog[i-1]))) / (h * 12.0);
136     fdeog[1] = fdeog[3];
137     fdeog[2] = fdeog[3];
138     fdeog[(int)length] = fdeog[(int)length-2];
139     fdeog[(int)length-1] = fdeog[(int)length-2];
140
141     ssfd = 0.0;
142     for(i = 1; i <= (int)length-20; i++)
143         ssfd = ssfd + (fdeog[i] * fdeog[i]);
144     ssmfd = ssfd / ((length - 20));
145     fdrms = sqrt(ssmfd);
146
147
148 /* sprintf(datafile,"%s_d.dat",filenam);
149     open_write(datafile);
150 */
151
152     for (n=1; n<=(int)length ;n++)
153         fprintf(out_file,"%f\n",fdeog[n]);
154     fclose(out_file);
155
156
157 /*Identification of fast phases*/
158     printf("Identification of fast phases");
159     fdrmsth = fdrms * 2.0;
160

```

```

161  rctcnt = 0;
162  lctcnt = 0;
163  islg = 0;
164  tht = 0.0;
165  hht = 0.0;
166  do /*while(islg < length)*/
167      {
168          lterm = (int)length - islg;
169
170          j = 1;
171          do
172              {
173                  ++j;
174                  } while((abs(fdeog[islg+j])) <= (abs(fdrmsth)) && (j <= lterm
175  ));
176          itlst = j + islg;
177
178          if(fdeog[itlst] >= 0.0)
179              goto lp1;
180
181          /*printf("negative values start\n");*/
182          for(l = 1; l <= 10; l++)
183              if (fdeog[itlst-l] >= 0.0)
184                  goto lp2;
185
186  lp1: /*printf("positive values start.\n");*/
187          for(l = 1; l <= 10; l++)
188              if (fdeog[itlst-l] < 0.0)
189                  goto lp3;
190
191  lp2: /*printf("negative values end.\n");*/
192          istr = itlst - l;
193          ++lctcnt;
194          ltbeg[lctcnt] = times[istr+10];
195
196          for (k = 1; k <= 30; k++)
197              if (fdeog[itlst+k] >= 0.0)
198                  {
199                      istp = itlst + k;
200                      ltend[lctcnt] = times[istp+6];
201                      goto lp4;
202                  } /*end if*/
203
204  lp3: /*printf("positive values end.\n");*/
205          istr = itlst - l;
206          ++rctcnt;
207          rtbeg[rctcnt] = times[istr+10];
208
209          ln = 1;
210          do
211              {
212                  ++ln;
213                  } while((fdeog[itlst+ln] >= 0.0) && (ln <= 30));

```

```

214
215     istp = itlst + ln;
216
217     rtend[rtcnt] = times[istp+6];
218
219 lp4: /*Least Squares Fit*/
220     dxsum = 0.0;
221     dxssum = 0.0;
222     dysum = 0.0;
223     dxysum = 0.0;
224     q = 1.0;
225
226     for (isl = istr-10; isl < istr; isl++)
227     {
228         dxsum = dxsum + q;
229         dxssum = dxssum + (q * q);
230         dysum = dysum + yheog[isl];
231         dxysum = dxysum + (q * yheog[isl]);
232         q = q + 1.0;
233     } /*end for*/
234
235     denom = dxssum - (dxsum * dxsum) / 11.0;
236     if (denom != 0)
237         slope = (dxysum - (dxsum * dysum) / 11.0) / denom;
238     else
239         slope = 0.;
240     yint = (dysum / 11.0) - slope * (dxsum / 11.0);
241
242
243     xilf = 12.0;
244     for (ilf = istr+1; ilf <= istp; ilf++)
245     {
246         yheog[ilf] = (slope * xilf) + yint;
247         xilf = xilf + 1.0;
248     } /*end for*/
249
250
251     /*Reconstructing slow phases*/
252     hht = hht + yheog[istp] - yheog[istp+1];
253
254     for(i = islg+1; i <= istp; i++)
255         heog[i] = yheog[i] + tht;
256
257     tht = hht;
258     islg = istp;
259
260 lp5: printf(".");
261     } while(islg < (int)length);
262
263     if (ltflag)
264         ltcnt--;
265     else if (rtflag)
266         rtcnt--;
267

```

```

268     printf("\n\nNumber of Fast Phases Detected: %d... HIT [ENTER] TO
CONTINUE",lrcnt+rtcnt);
269     getchar();
270
271     system("cls");
272
273     printf("Do you want to plot the reconstructed EOG now? (1- Yes) >
");
274     scanf("%d%c", &yes,&ch);
275
276     if (yes)
277         plot(length-10,fsample);
278
279
280 /*5-Point Smoother*/
281     printf("\nDo you want to smooth reconstructed EOG? (1- Yes) > ");
282     scanf("%d%c", &yes,&ch);
283     printf("\n");
284
285     if (yes)
286         smooth_data(length);
287
288     printf("Do you want to plot the reconstructed EOG again? (1- Yes)
> ");
289     scanf("%d%c", &yes,&ch);
290
291     if (yes)
292         plot(length,fsample);
293
294
295
296 /*Creating timing file*/
297     sprintf(datafile, "%s.tim", filenam);
298     open_write(datafile);
299     write_timing(datafile,rtcnt,lrcnt);
300
301     printf("\n\nDo you want a hardcopy of timing file now? (1- Yes) ")
;
302     scanf("%d%c",&var,&ch);
303     if (var)
304     {
305         open_write(printer);
306         write_timing(datafile,rtcnt,lrcnt);
307     }
308
309
310 /*Creating data file*/
311     sprintf(datafile, "%s-r.dat", filenam);
312     open_write(datafile);
313
314     for (i = 1; i <= (int)length; i++)
315         fprintf(out_file, "%f\n",heog[i]);
316     fclose(out_file);
317

```



```

318  system("cls");
319
320  free_memory(); /*Freeing memory*/
321
322  printf("\n\nRun program again? (1- Yes) > ");
323  scanf("%d%c", &more,&ch);
324  printf("\n");
325
326
327  } while(more != 0);
328
329
330 } /*end main*/
331
332
333
334
335
336 /*Allocate memory needed for the data arrays*/
337 get_memory()
338 {
339
340  rtbeg = (float *)calloc(SIZEA, sizeof(float));
341  rtend = (float *)calloc(SIZEA, sizeof(float));
342  ltbeg = (float *)calloc(SIZEA, sizeof(float));
343  ltend = (float *)calloc(SIZEA, sizeof(float));
344
345  if (!rtbeg || !rtend || !ltbeg || !ltend)
346  {
347  printf("\n\n***MEMORY ALLOCATION ERROR A***\n");
348  exit(0);
349  }
350
351  heog = (float *)calloc(SIZEB, sizeof(float));
352  yheog = (float *)calloc(SIZEB, sizeof(float));
353  fdeog = (float *)calloc(SIZEB, sizeof(float));
354  times = (float *)calloc(SIZEB, sizeof(float));
355
356  if (!heog || !yheog || !fdeog || !times)
357  {
358  printf("\n\n***MEMORY ALLOCATION ERROR B***\n");
359  exit(0);
360  }
361
362 } /*end get_memory*/
363
364
365 /*Free memory allocated for data arrays*/
366 free_memory()
367 {
368  free(times); free(heog); free(yheog); free(fdeog);
369  free(rtbeg); free(rtend); free(ltbeg); free(ltend);
370 }
371

```

```

372
373 /*Pause the program until the user hits ENTER*/
374 message()
375 {
376
377 printf("   EOG FAST PHASE IDENTIFICATION and SLOW PHASE
RECONSTRUCTION")
;
378 printf("\n\n      Files Read      Type");
379 printf("\n      -----");
380 printf("\n      orig_filename.DAT  EOG Data - ASCII");
381 printf("\n      Files Generated    Type");
382 printf("\n      -----");
383 printf("\n      orig_filename.TIM   Formatted Report - ASCII"
);
384 printf("\n      orig_filename_R.DAT  EOG Data (no index)- ASCII
I");
385 printf("\n\n\nANY EOG DATA WILL BE SUCCESSFULLY ANALYZED (e.g., fast
phases timing) BUT");
386 printf("\nCORRECT VELOCITY CALCULATIONS AND DATA
RECONSTRUCTION CAN BE DONE ON SCALED");
387 printf("\nDATA ONLY (e.g., eog=degrees)");
388 printf("\n\n\nEOG FILENAME MUST HAVE A .dat EXTENSION");
389 printf("\n\n\nEOG FILENAME MUST ALSO BE LESS THAN -10- CHARACTERS IN
LENGTH");
390 printf("\n\n\nIF YOU HAVE TO RENAME DATAFILE EXIT PROGRAM AND
RENAME FILE");
391 printf("\n\n\n                        HIT [ENTER] TO CONTINUE...");
392 getchar();
393
394 }
395
396
397
398 /*open input datafile*/
399 open_read (file)
400 char file[15];
401 {
402   if((in_file = fopen(file, "r")) == NULL)
403   {
404     printf("\n***ERROR opening data file %s.....BYE!\n", file);
405     exit (0);
406   }
407
408   printf("\n\nOpened %s - reading data...\n", file);
409
410 }

```

open_read Local Symbols

Name	Class	Type	Size	Offset	Register
file.		param		0004	

```

411
412
413 /*open output datafile*/
414 open_write(file)
415 char file[15];
416 {
417 int ans;
418 char newfile[15];
419
420 if ((out_file = fopen(file, "r")) != NULL)
421 {
422 printf("\n\nFILE ALREADY EXISTS -- DO YOU WANT TO OVERWRITE? (1-Yes) ");
423 scanf("%d",&ans);
424 if (ans!=1)
425 {
426 printf("\n\nENTER NEW FILENAME (without .ext)");
427 scanf("%s",newfile);
428 sprintf(file,"%s.dat",newfile);
429 }
430 }
431
432 if ((out_file = fopen(file, "w")) == NULL)
433 {
434 printf("\n\n***ERROR opening file %s.....BYE!\n", file);
435 exit(0);
436 }
437
438 printf("\n\nWriting to file %s\n", file);
439
440 }
441
442
443 smooth_data(len)
444 float len;
445 {
446 int i;
447
448 printf("5-point smoother...\n");
449 for (i = 3; i <= (int)len-2; i++)
450 yheog[i] = .11*(heog[i-2] + heog[i+2]) + .22*(heog[i-1] + heog[i+1])
451 + .33*heog[i];
452
453 yheog[1] = yheog[3];
454 yheog[2] = yheog[3];
455 yheog[(int)len] = yheog[(int)len-2];
456 yheog[(int)len-1] = yheog[(int)len-2];
457
458 for (i = 1; i <= (int)len; i++)
459 heog[i] = yheog[i];
460
461 }
462

```

```

463
464 filter_data(len)
465 float len;
466 {
467 int i,j,p,n,nco=15;
468 float h,sum;
469 double COEF[15];
470
471 COEF[1] = .0037165603;
472 COEF[2] = .020235427;
473 COEF[3] = .013399956;
474 COEF[4] = -.040643737;
475 COEF[5] = -.066073574;
476 COEF[6] = .061152663;
477 COEF[7] = .30294424;
478 COEF[8] = .43047285;
479 for(n = 1; n <= 7; n++)
480 COEF[16-n] = COEF[n];
481
482 printf("15-point FIR filter...\n");
483
484 for (i = 2; i <= (int)len+nco; i++)
485 {
486 sum = 0.0;
487 for (j = 1; j <= nco; j++)
488 {
489 if (j < i)
490 {
491 if (i-j <= (int)len)
492 {
493 h = COEF[j] * heog[i-j];
494 sum = sum + h;
495 } /*end if*/
496 } /*end if*/
497 } /*end for*/
498 yheog[i-1] = sum;
499 } /*end for*/
500
501 p = 14;
502 for (j = 15; j <= (int)len+p; j++)
503 yheog[j-p] = yheog[j];
504 for (i = 1; i <= 14; i++)
505 yheog[(int)len-i-1] = yheog[(int)len-14];
506
507 }
508
509
510
511 write_timing(file,num_rt,num_lt)
512 int num_rt,num_lt;
513 char file[15];
514 {
515 int n;
516

```

```

517 fputc('\014', out_file);
518 fprintf(out_file, "FAST PHASE TIMING INFORMATION FOR: %s\n\n", file);
519 fprintf(out_file, "LEFT PHASES ARE DOWNWARD PHASES RIGHT PHASES
ARE UPWARD PHASES\n\n\n");
520 fprintf(out_file, " RIGHT      TIMING\n");
521 fprintf(out_file, "PHASE # START  END\n");
522 fprintf(out_file, "-----\n");
523 for (n = 1; n <= num_rt; n++)
524   fprintf(out_file, "%3d    %5.3f  %5.3f\n", n, rtbeg[n], rtend[n
]);
525
526 fprintf(out_file, "\n\n\n");
527 fprintf(out_file, "LEFT      TIMING\n");
528 fprintf(out_file, "PHASE # START  END\n");
529 fprintf(out_file, "-----\n");
530 for (n = 1; n <= num_lt; n++)
531   fprintf(out_file, "%3d    %5.3f  %5.3f\n", n, ltbeg[n], ltend[n
]);
532
533 fclose(out_file);
534
535 }
536
537
538
539 plot(len,fs)
540 float len, fs;
541 {
542 int i,n;
543 long unsigned int g;
544 float maxeog,maxtime;
545 short oldcolor;
546
547 maxeog=0;
548 for (i=1; i<=(int)len ;i++)
549   {
550   if (abs(heog[i]) > maxeog)
551     maxeog=abs(heog[i]);
552   }
553
554
555 maxtime=len/fs;
556 for (i=1; i<=(int)len ;i++)
557   {
558   g = (long)i * 560 / (long int)len;
559   yheog[i] = heog[i] * 120. / maxeog;
560   }
561
562
563 _setvideomode(_ERESCOLOR);
564
565 _moveto(54, 30);
566 _lineto (614, 30);
567 _lineto (614, 284);

```

```

568 _lineto (54, 284);
569 _lineto (54, 30);
570
571
572 n=94;
573 do
574 {
575   _moveto ((short)n, 30);
576   _lineto ((short)n, 284);
577   n=n+40;
578 } while(n<=594);
579
580 oldcolor=_getcolor();
581 n=55;
582 do
583 {
584   _moveto (54, (short)n);
585   if (n==155)                               /*0 degrees=155*/
586     _setcolor(5);
587   else
588     _setcolor(oldcolor);
589   _lineto (614, (short)n);
590   n=n+25;
591 } while(n<=255);
592
593
594 _setcolor(10);
595 _moveto(54,155);
596 for (i=1; i<=(int)len ;i++)
597 {
598   g = (long)i * 560 / (long int)len;
599   _lineto (54+(short)g, 155-(int)yeog[i]);
600 }
601
602
603 _moveto(54,145);
604 printf("\n %-3.2f\n\n %-3.2f\n\n %-3.2f\n\n %-3.2f\n", maxe
og, maxeog/5*4, maxeog/5*3, maxeog/5*2, maxeog/5*1);
605 printf("E\nO 0\nG");
606 printf("\n%-3.2f\n%-3.2f\n\n%-3.2f\n\n%-3.2f\n", maxeog/5*-1
, maxeog/5*-2, maxeog/5*-3, maxeog/5*-4, maxeog*-1);
607 printf(" 0 %2.2f %-2.2f %-2.2f %-2.2f %-2.2f %-2.2f %-2.2f %-2.2f
f %-2.2f %-2.2f %-2.2f %-2.2f %-2.2f %-2.1f",maxtime/14*1,maxtime/14*2
,maxtime/14*3,maxtime/14*4,maxtime/14*5,maxtime/14*6,maxtime/14*7,maxt
ime/14*8,maxtime/14*9,maxtime/14*10,maxtime/14*11,maxtime/14*12,maxtim
e/14*13,maxtime/14*14);
608 printf("\n\n                               TIME");
609
610
611 while (!kbhit());
612 _setvideomode(_DEFAULTMODE);
613
614 }

```

Code size = 158a (5514)
Data size = 0aa1 (2721)
Bss size = 0000 (0)

No errors detected

/* VARIABLE DICTIONARY: a listing of all variables used and their function.

Global:

SIZEA constant; the number of elements allocated for rtbeg, rtend, ltbeg, and ltend;
SIZEB constant; the number of elements allocated for heog, yheog, fdeog, and times;
***fopen()** a C pointer used to open an outside file;
***in_file** a pointer set to the input EOG file;
***out_file** a pointer set first to the output Timing File, then to the output Data File;
***rtbeg** an array of 100 floating-point elements allocated dynamically; used to mark the end of the right phases;
***rtend** an array of 100 floating-point elements allocated dynamically; used to mark the beginning of the right phases;
***ltbeg** an array of 100 floating-point elements allocated dynamically; used to mark the end of the right phases;
***ltend** an array of 100 floating-point elements allocated dynamically; used to mark the beginning of the right phases;
***heog** an array of 3100 floating-point elements allocated dynamically; holds the EOG data, read in from the input file; stores most of the changes made to the wave and finally stores the reconstructed slow phase wave;
***yheog** an array of 3100 floating-point elements allocated dynamically; used to temporarily store some of the changes made to the wave;
***fdeog** an array of 3100 floating-point elements allocated dynamically; holds the first derivative of the wave;
***times** an array of 3100 floating-point elements allocated dynamically; holds the timing information of the wave; read in from the EOG input file;

Local:

length the number of points to read; equal to sampling rate*seconds; read in from the keyboard;
filename array holding the name (with no extension) of the EOG input file; read in from the keyboard;
datafile array holding the name (including the extension) of the file to be opened;
more boolean (0 or 1); set to 1 if the main loop is to be repeated (run program with another EOG file); set to 0 if main loop is not to be repeated, either because of an error or by the

user's choice;
 yes boolean (0 or 1); set to 1 if the user wants 5-point smoothing and/or another 15-point FIR filter;
 xmean the mean of the EOG data; used to remove the DC component;
 xsum used in finding the mean of the EOG data;
 xlgt used in finding the mean of the EOG data;
 nco the number of the FIR filter being used; equal to 15;
 COEF array of constants; used in filtering the data;
 h factor used in the first 15-point FIR filter;
 dh factor used in the second 15-point FIR filter;
 sum used to hold the sum in the first 15-point FIR filter;
 dsum used to hold the sum in the second 15-point FIR filter;
 istep used in Differentiating; equal to 1;
 ssmfd used in calculating the RMS of the first derivative;
 ssfd used in calculating the RMS of the first derivative;
 fdrms the RMS of the first derivative;
 fdrmssth holds the first derivative RMS threshold value, 2*fdrms;
 itlst used in Identification of fast phases;
 islg used in Identification of fast phases;
 lterm used to mark the position in Identification of fast phases;
 istrt used to mark the beginning of a phase;
 istp used to mark the end of a phase;
 rtcnt counter used when identifying right phases;
 ltcnt counter used when identifying left phases;
 isl used to mark a position in Least Squares Fit;
 ilf used to mark a position in Least Squares Fit;
 xn factor used in Least Squares Fit; equal to 10;
 xilf factor used in Least Squares Fit;
 q factor used in Least Squares Fit;
 dxsum the sum of the x values in the Least Squares Fit;
 dysum the sum of the y values in the Least Squares Fit;
 dxssum the sum of the x values squared in the Least Squares Fit;
 dxysum (the sum of the x values)*(the sum of the y values) in the Least Squares Fit;
 denom the value of the denominator in the Least Squares Fit;
 slope the value of the slope in the Least Squares Fit;
 yint the value of the y intercept in the Least Squares Fit;
 hht used in finding the height correction factor for the reconstructed wave;
 tht the height correction factor for the reconstructed wave;

n, j, k, i, l, ln, p counters used throughout the program

*/

APPENDIX E
PROGRAM LISTING: FPID.FOR

```

0001 C      PROGRAM FPID.FOR
0002 C
0003 INTEGER LENGTH,RTCNT,LTCNT
0004 REAL RTBEG(200),RTEND(200),LTBEG(200),LTEND(200),TIME(10000)
0005 CHARACTER * 20 FILENAM
0006 CHARACTER * 30 INFILE, TIMFILE, OUTFILE
0007 CHARACTER TEXT*5, REXT*5
0008 REAL FSAMPLE, HEOG(10000), XT(10000), COEF(15), YHEOG(10000),
0009 1  SPEOG(10000), SPYHEOG(10000), FDEOG(10000)
0010 C
0011 C
0012 COEF(1) = .0037165603
0013 COEF(2) = .020235427
0014 COEF(3) = .013399956
0015 COEF(4) = -.040643737
0016 COEF(5) = -.066073574
0017 COEF(6) = .061152663
0018 COEF(7) = .30294424
0019 COEF(8) = .43047285
0020 DO N = 1, 7
0021     COEF(16-N) = COEF(N)
0022 END DO
0023 C
0024 C
0025 TEXT = '.TIM'
0026 REXT = '-R'
0027 PRINT *, '          FPID'
0028 PRINT *, ' '
0029 PRINT *, ' EOG FAST PHASE IDENTIFICATION AND'
0030 PRINT *, ' SLOW PHASE RECONSTRUCTION'
0031 PRINT *, ' '
0032 12 PRINT *, ' ENTER EOG DATAFILE ENCLOSED IN APOSTROPHES (no .ext)'
0033 READ *, INFILE
0034 CALL CHARCONCAT(INFILE,TIMFILE,TEXT)
0035 CALL CHARCONCAT(INFILE,OUTFILE,REXT)
0036 PRINT 5, INFILE, OUTFILE, TIMFILE
0037 5  FORMAT (T5,'INPUT FILE (.dat): ',A,/,
0038 *         T5,'RECONSTRUCTED FILE (.dat): ',A,/,
0039 *         T5,'TIMING FILE: ',A,/)
0040 PRINT *, ' ENTER SAMPLING RATE (in Hz)'
0041 READ *, FSAMPLE
0042 PRINT *, ' ENTER LENGTH OF FILE (in seconds)'
0043 READ *,LENGTH
0044 C
0045 LENGTH = LENGTH * FSAMPLE
0046 C
0047 TYPE *, ' OPENING INPUT DATAFILE'
0048 OPEN (UNIT=2,FILE=INFILE,STATUS='OLD')
0049 TYPE *, ' OPENED DATAFILE- READING DATA'
0050 DO N=1,LENGTH
0051     READ (2,*) HEOG(N)
0052 19  FORMAT (F10.6)
0053     TIME(N) = N / FSAMPLE
0054 END DO

```

```

0055     CLOSE (UNIT=2,STATUS='KEEP')
0056     C
0057     C
0058     ISTEP = 1
0059     C
0060     XLGT = FLOAT(LENGTH)
0061     C
0062     C
0063     C COMPUTE THE MEAN OF THE DATA- REMOVE DC
0064     C
0065     XSUM = 0.0
0066     DO 58 I = 1, LENGTH
0067         XSUM = XSUM + HEOG(I)
0068     58 CONTINUE
0069     C
0070     XMEAN =XSUM/XLGT
0071     DO 59 I = 1, LENGTH
0072         HEOG(I) = HEOG(I) - XMEAN
0073     59 CONTINUE
0074     C
0075     C
0076     C AVERAGE DATA
0077     C
0078     62 DO 500 I = 3, LENGTH-2
0079         YHEOG(I) = .11*(HEOG(I-2)+HEOG(I+2))+.22*(HEOG(I-1)+HEOG(I+1))
0080         1         + .33*HEOG(I)
0081     500 CONTINUE
0082         YHEOG(1) = YHEOG(3)
0083         YHEOG(2) = YHEOG(3)
0084         YHEOG(LENGTH) = YHEOG(LENGTH-2)
0085         YHEOG(LENGTH-1) = YHEOG(LENGTH)
0086     C
0087         DO 510 I = 1, LENGTH
0088             HEOG(I) = YHEOG(I)
0089     510 CONTINUE
0090     C
0091     162         NCO = 15
0092     TYPE *, ' FILTERING DATA WITH 15-POINT FIR FILTER!'
0093     C
0094         DO 80 I = 2, LENGTH+NCO
0095             SUM =0.0
0096             DO 70 J = 1,NCO
0097                 L = J
0098                 IF (J .GE. 1) GO TO 70
0099                 IF (I-J .GT. LENGTH) GO TO 70
0100                 H = COEF(J) * HEOG(I-J)
0101                 SUM = SUM + H
0102     70 CONTINUE
0103         YHEOG(I-1) = SUM
0104     80 CONTINUE
0105     C
0106         J2 = 14
0107         DO 81 J = 15, LENGTH+J2
0108             YHEOG(J-J2) = YHEOG(J)

```

```

0109 81 CONTINUE
0110 C
0111 DO 630 I = 1, 14
0112 YHEOG(LENGTH-I-1)=YHEOG(LENGTH-14)
0113 630 CONTINUE
0114 C
0115 C
0116 83 H = FLOAT(ISTEP)/FSAMPLE
0117 L = LENGTH
0118 TYPE *, ' COMPUTING FOURTH ORDER FIRST DIFFERENCE'
0119 CCC
0120 86 DO 87 I = 3, L-2
0121 FDEOG(I) = ((YHEOG(I-2) - YHEOG(I+2)) + (8.*(YHEOG(I+1) -
0122 1 YHEOG(I-1))))/(H*12.)
0123 87 CONTINUE
0124 FDEOG(1) = FDEOG(3)
0125 FDEOG(2) = FDEOG(3)
0126 FDEOG(L) = FDEOG(L-2)
0127 FDEOG(L-1) = FDEOG(L-2)
0128 C
0129 FDRMS = 0.
0130 SSFD = 0.
0131 DO 700 I = 1, LENGTH-20
0132 SSFD = SSFD + FDEOG(I)**2.
0133 700 CONTINUE
0134 SSMFD = SSFD/FLOAT(LENGTH-20)
0135 FDRMS = SQRT(SSMFD)
0136 TYPE 725,FDRMS
0137 725 FORMAT (' RMS OF 1ST DERV. =',F8.3)
0138 C
0139 C
0140 C IDENTIFICATION OF FAST PHASES
0141 C
0142 90 TYPE *, ' IDENTIFICATION OF FAST PHASES'
0143 FDRMSTH = FDRMS*2.
0144 TYPE 410, FDRMSTH
0145 410 FORMAT (' 1ST DERV. RMS THRESHHOLD =',F9.3)
0146 C
0147 RTCNT = 0
0148 LTCNT = 0
0149 ISLG = 0
0150 THT = 0.
0151 HHT = 0.
0152 110 LTERM = LENGTH - ISLG
0153 DO 112 J = 1, LTERM
0154 IF (ABS(FDEOG(ISLG+J)) .GT. ABS(FDRMSTH)) GO TO 115
0155 112 CONTINUE
0156 C
0157 115 ITLST = J + ISLG
0158 CC TYPE 411,ITLST
0159 411 FORMAT (' INDEX TSH EXCD AT I =',I5)
0160 C
0161 IF (FDEOG(ITLST) .GE. 0.) GO TO 120
0162 CC TYPE *, ' NEGATIVE VALUES START'

```

```

0163     DO 117 L = 1, 10
0164     IF (FDEOG(ITLST - L) .GE. 0.) GO TO 124
0165 117     CONTINUE
0166 CC TYPE *, ' POSITIVE VALUES START'
0167 120     DO 122 L = 1, 10
0168     IF (FDEOG(ITLST - L) .LT. 0.) GO TO 128
0169 122     CONTINUE
0170 CC TYPE *, ' NEGATIVE VALUES END'
0171 124     ISTRT = ITLST - L
0172 CC TYPE 412, ISTRT
0173 412     FORMAT (' STRT NEG INDEX AT I = ', I4)
0174     LTCNT=LTCNT+1
0175     LTBEGL(LTCNT)=TIME(ISTRT+10)
0176 C
0177     DO 126 K = 1, 30
0178     IF (FDEOG(ITLST + K) .GE. 0.) GO TO 135
0179 126     CONTINUE
0180 CC TYPE *, ' POSITIVE VALUES END'
0181 128     ISTRT = ITLST - L
0182 CC TYPE 414, ISTRT
0183 414     FORMAT (' STRT POS INDEX AT I = ', I4)
0184     RTCNT=RTCNT+1
0185     RTBEGL(RTCNT)=TIME(ISTRT+10)
0186 C
0187     DO 130 LN = 1, 30
0188     IF (FDEOG(ITLST + LN) .LT. 0) GO TO 134
0189 130     CONTINUE
0190 134     ISTP = ITLST + LN
0191 CC TYPE 415, ISTP
0192 415     FORMAT (' END POS INDEX AT I = ', I4)
0193     RTEND(RTCNT)=TIME(ISTP+6)
0194     GOTO 136
0195 C
0196 135     ISTP = ITLST + K
0197 CC TYPE 416, ISTP
0198 416     FORMAT (' END NEG INDEX AT I = ', I4)
0199     LTEND(LTCNT)=TIME(ISTP+6)
0200 C
0201 C LEAST SQUARES FIT
0202 136     XN = 10.
0203     DXSUM = 0.
0204     DXSSUM = 0.
0205     DYSUM = 0.
0206     DXYSUM = 0.
0207     Q = 1.0
0208 C
0209     DO 140 ISL = ISTRT-10, ISTRT
0210     DXSUM = DXSUM + Q
0211     DXSSUM = DXSSUM + Q*Q
0212     DYSUM = DYSUM + YHEOG(ISL)
0213     DXYSUM = DXYSUM + Q*YHEOG(ISL)
0214     Q = Q + 1.0
0215 140     CONTINUE
0216 C

```

```

0217     DENOM = DXSSUM - (DXSUM*DXSUM) / 11.0
0218     SLOPE = (DXYSUM - (DXSUM*DYSUM) / 11.0) / DENOM
0219     YINT = (DYSUM / 11.0) - SLOPE * (DXSUM / 11.0)
0220 CC TYPE 417, SLOPE, YINT
0221 417     FORMAT (' SLOPE =',F12.5,' YINT =',F12.5)
0222
0223     XILF = 12.0
0224     DO 142 ILF = ISTRT, ISTOP
0225         YHEOG(ILF) = SLOPE*XILF + YINT
0226     XILF = XILF + 1.0
0227 142     CONTINUE
0228 C
0229 CC TYPE *, ' RECONSTRUCT SLOW PHASE'
0230 C
0231     HHT = HHT + YHEOG(ISTP) - YHEOG(ISTP+1)
0232 CC TYPE 420, HHT
0233 420     FORMAT(' HEIGHT CORRECTION =',F12.5)
0234 C
0235 C336     TYPE 421, ISLG, ISTOP
0236 421     FORMAT (' STRT SP INDX = ',I7,' STP SP INDX = ',I7)
0237 C
0238     DO 145 I = ISLG+1, ISTOP
0239         SPEOG(I) = YHEOG(I) + THT
0240 145     CONTINUE
0241 C
0242     THT = HHT
0243     ISLG = ISTOP
0244 CC TYPE 425,ISLG
0245 425     FORMAT(' NEW SEARCH STARTS AT I = ',I4)
0246 CC TYPE *, ' TO CONTINUE TYPE: 1'
0247 CC ACCEPT *, IXS
0248     IF (ISLG .LT. LENGTH) GO TO 110
0249 C
0250     RTCNT=RTCNT-1
0251     LTCNT=LTCNT-1
0252     TYPE 13, RTCNT,LTCNT
0253 13 FORMAT (' # RT PHASES: ',I7,5X,'#LT PHASES: ',I7)
0254 C
0255     OPEN (UNIT=3, FILE=TIMFILE, STATUS='NEW')
0256     WRITE (3,11)
0257 11 FORMAT (' UPBEATING PHASES',//
0258 *           ' PHASE #',T15,'START',T30,'END')
0259     DO N=1,RTCNT
0260         WRITE (3,14) N, RTBEG(N), RTEND(N)
0261 C     TYPE 14,N,RTBEG(N),RTEND(N)
0262 14     FORMAT (T2,I4,T15,F8.3,T30,F8.3)
0263     END DO
0264     WRITE (3, 16)
0265 16 FORMAT (///,' DOWNBEATING PHASES',//
0266 *           ' PHASE #',T15,'START',T30,'END')
0267     DO N=1,LTCNT
0268         WRITE (3,17) N, LTBEAG(N), LTEND(N)
0269 C     TYPE 17,N,LTBEAG(N),LTEND(N)
0270 17     FORMAT(T2,I4,T15,F8.3,T30,F8.3)

```

```

0271     END DO
0272     CLOSE (UNIT=3, STATUS='KEEP')
0273  C
0274  C
0275  149     OPEN (UNIT=4,FILE=OUTFILE, FORM='FORMATTED',
STATUS='NEW')
0276     DO I=1,LENGTH
0277     WRITE (4,100) SPEOG(I)
0278  100     FORMAT(F10.3)
0279     END DO
0280     CLOSE (UNIT=4, STATUS='KEEP')
0281  C
0282  C
0283  150     PRINT *, 'ANALYZE ANOTHER FILE? (1- YES, 0- NO)'
0284     ACCEPT *, MORE
0285     IF (MORE .EQ. 1) GO TO 12
0286  C
0287     STOP
0288  1000     END

```

FUNCTIONS AND SUBROUTINES REFERENCED

Type Name

CHARCONCAT

```

0001  C
0002  C
0003     subroutine charconcat(filename,cfilename,ext)
0004     character filename*20,cfilename*30,ext*5
0005     do i=1,20
0006     if (filename(i:i).eq.' ') go to 14
0007     enddo
0008  14  cfilename=filename(1:i-1)//ext(:)
0009     return
0010     end

```

COMPILATION STATISTICS

```

Run Time:      2.27 seconds
Elapsed Time:  3.95 seconds
Page Faults:   1023
Dynamic Memory: 1076 pages

```

FIGURES

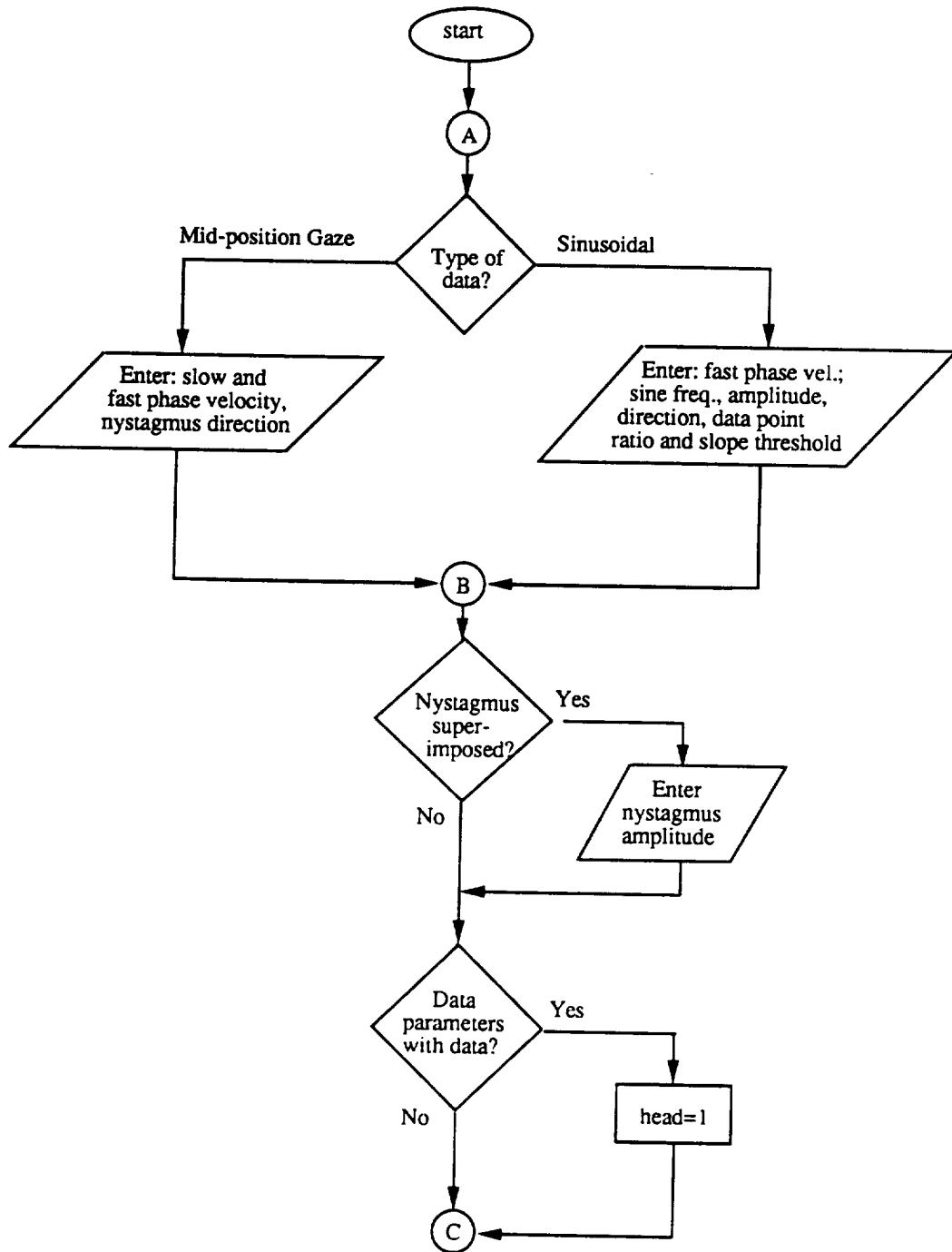


Figure 1A. Flowchart of NASDAT Program.

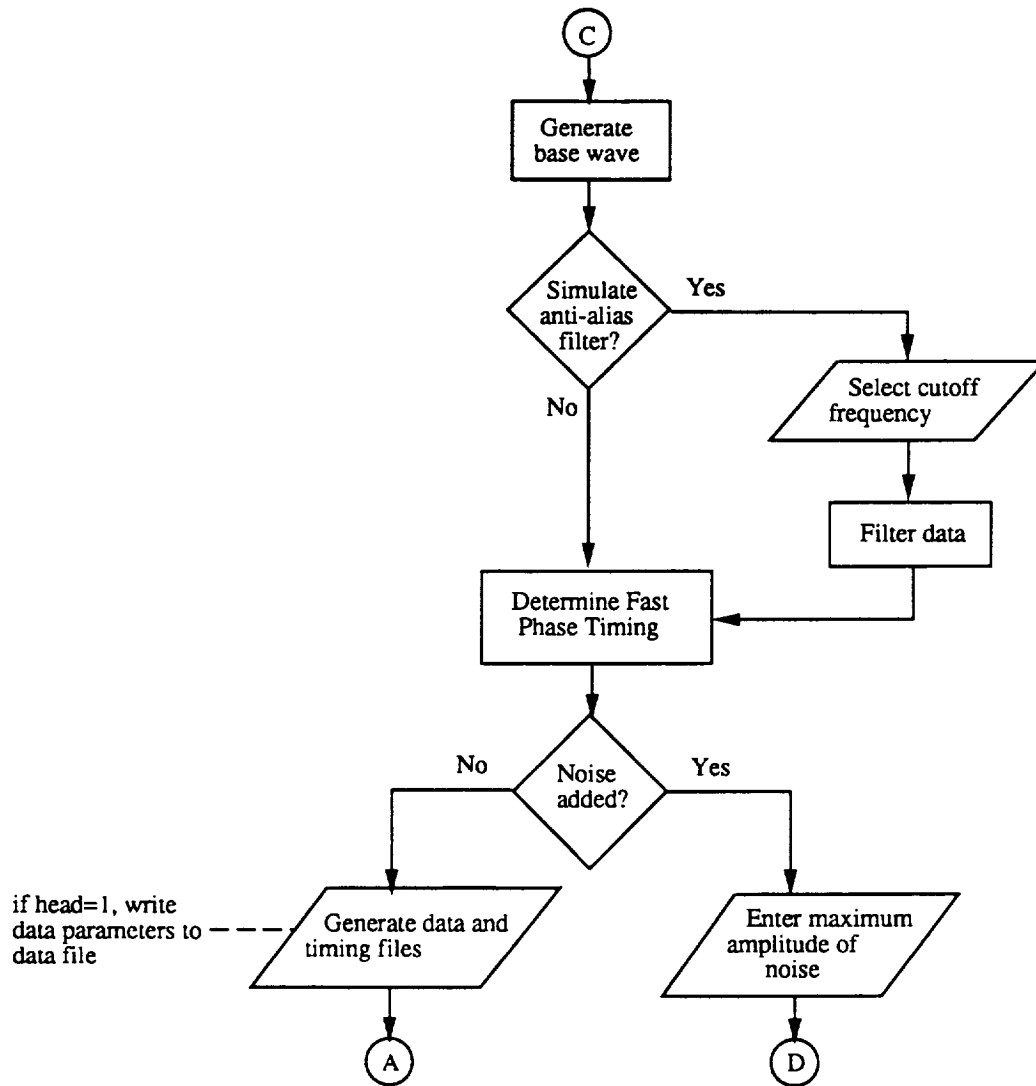


Figure 1B. Flowchart of NASDAT Program (continued).

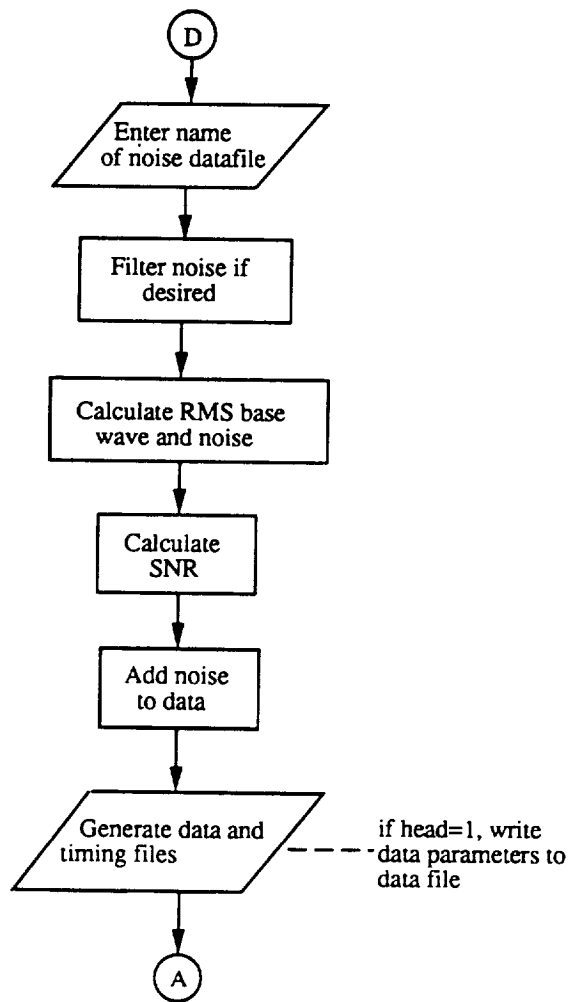


Figure 1C. Flowchart of NASDAT Program (continued).

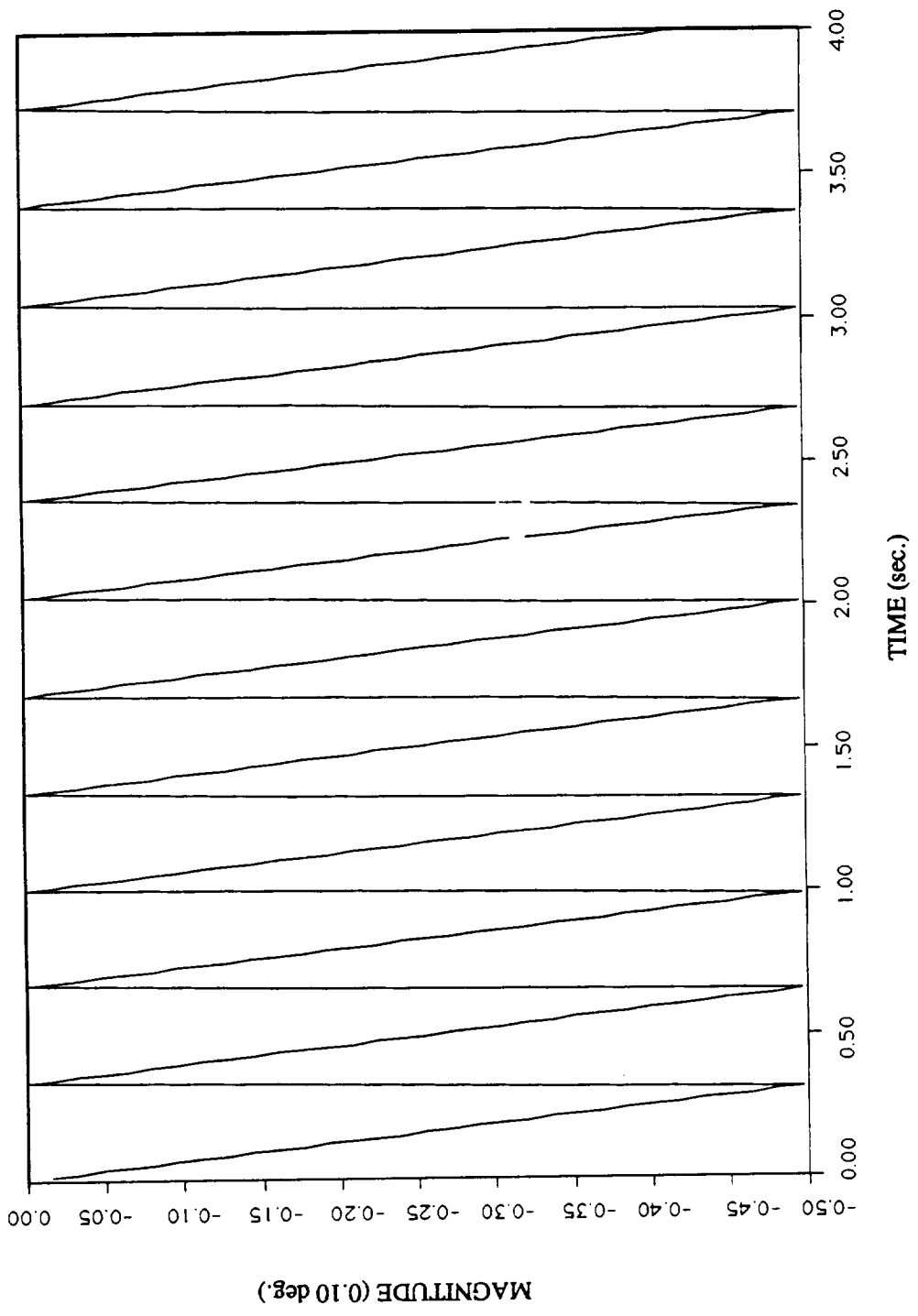


Figure 2. Sample synthetic midposition gaze data with five (5) degrees of saccadic jump (without noise).

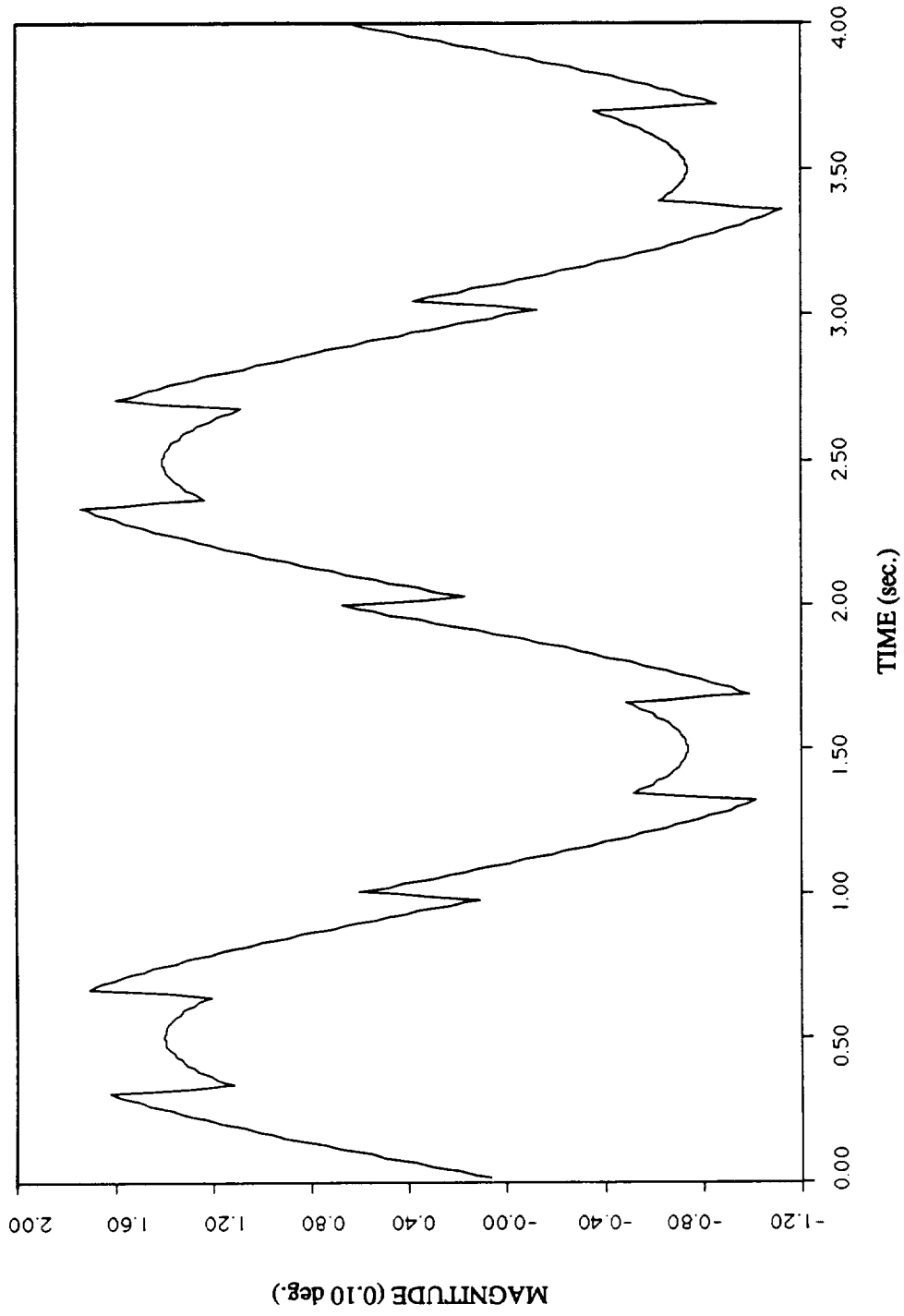


Figure 3. Sample synthetic 0.5 Hz sinusoidal pursuit data with five (5) degrees of saccadic jump (without noise).

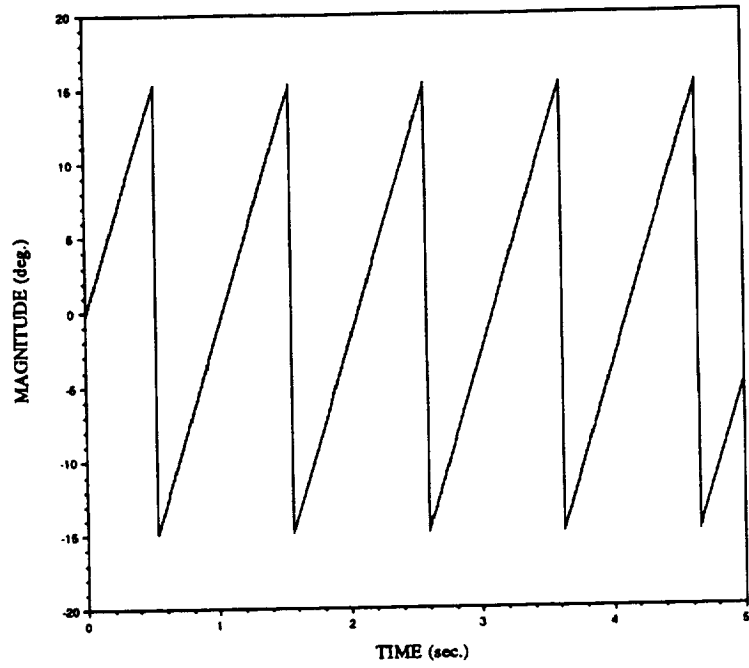


Figure 4A. Plot of optokinetic stimulus.

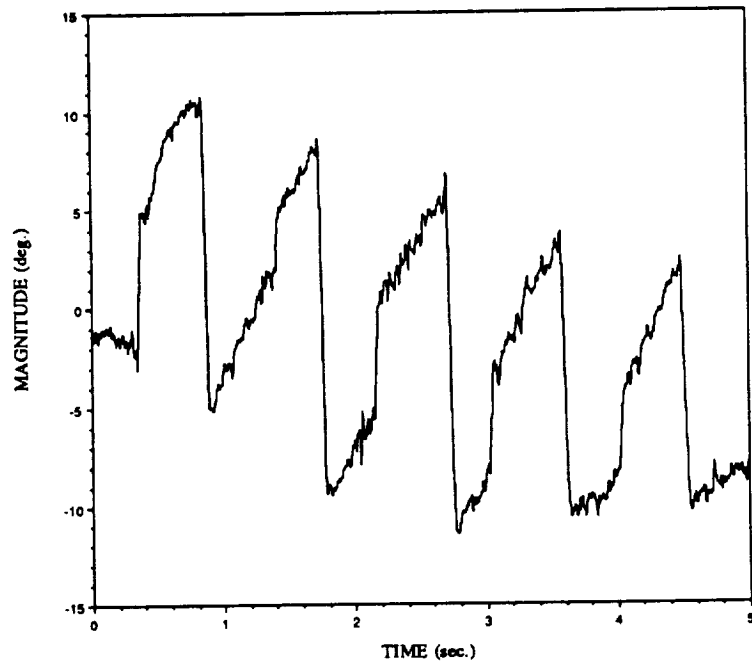


Figure 4B. Plot of human optokinetic EOG data set.

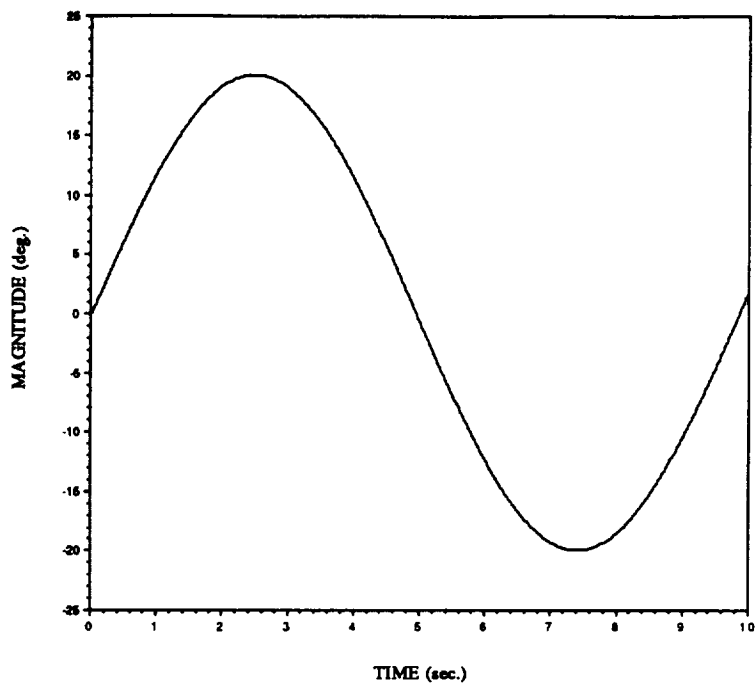


Figure 5A. Plot of 0.10 Hz sinusoidal pursuit stimulus.

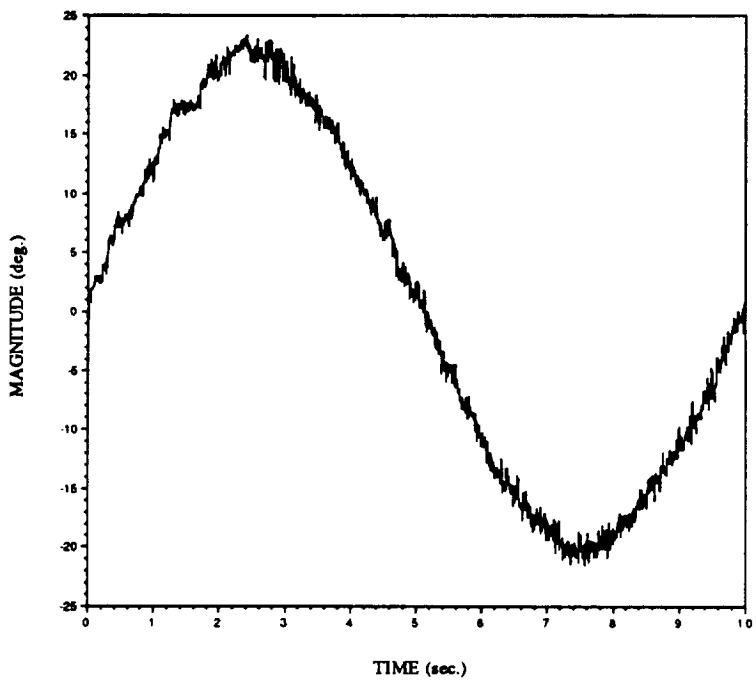


Figure 5B. Plot of human sinusoidal pursuit EOG data set.

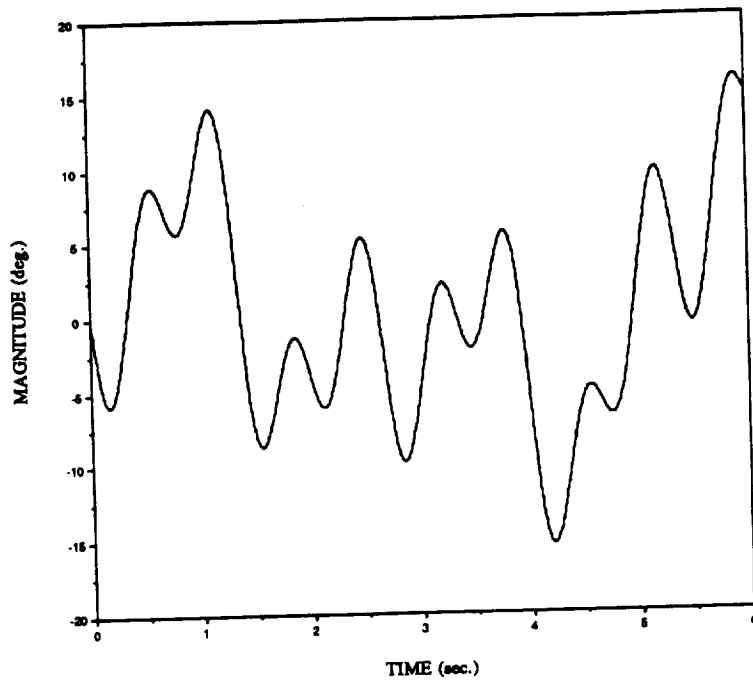


Figure 6A. Plot of pseudorandom stimulus.

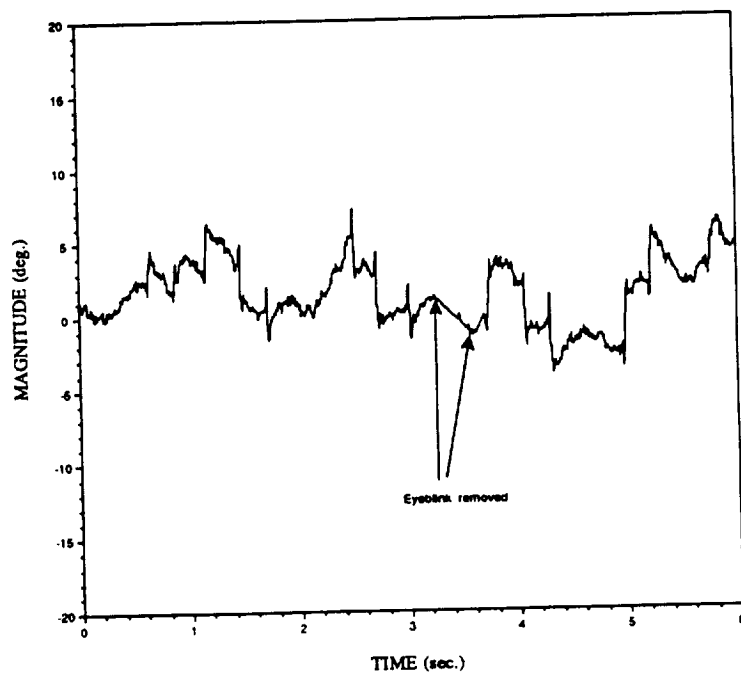


Figure 6B. Plot of human pseudorandom EOG data set.

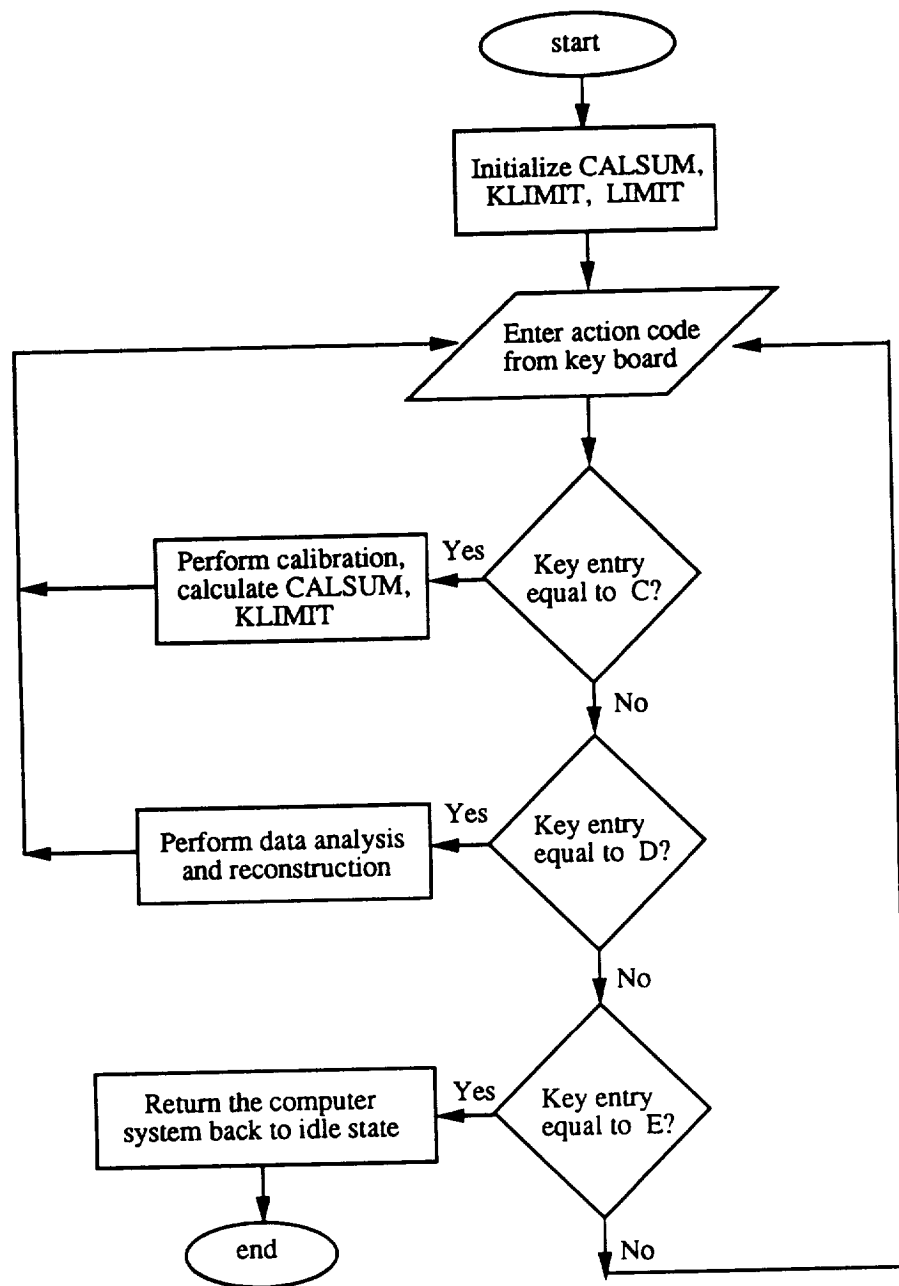


Figure 7. Overall flowchart of US Air Force Program.

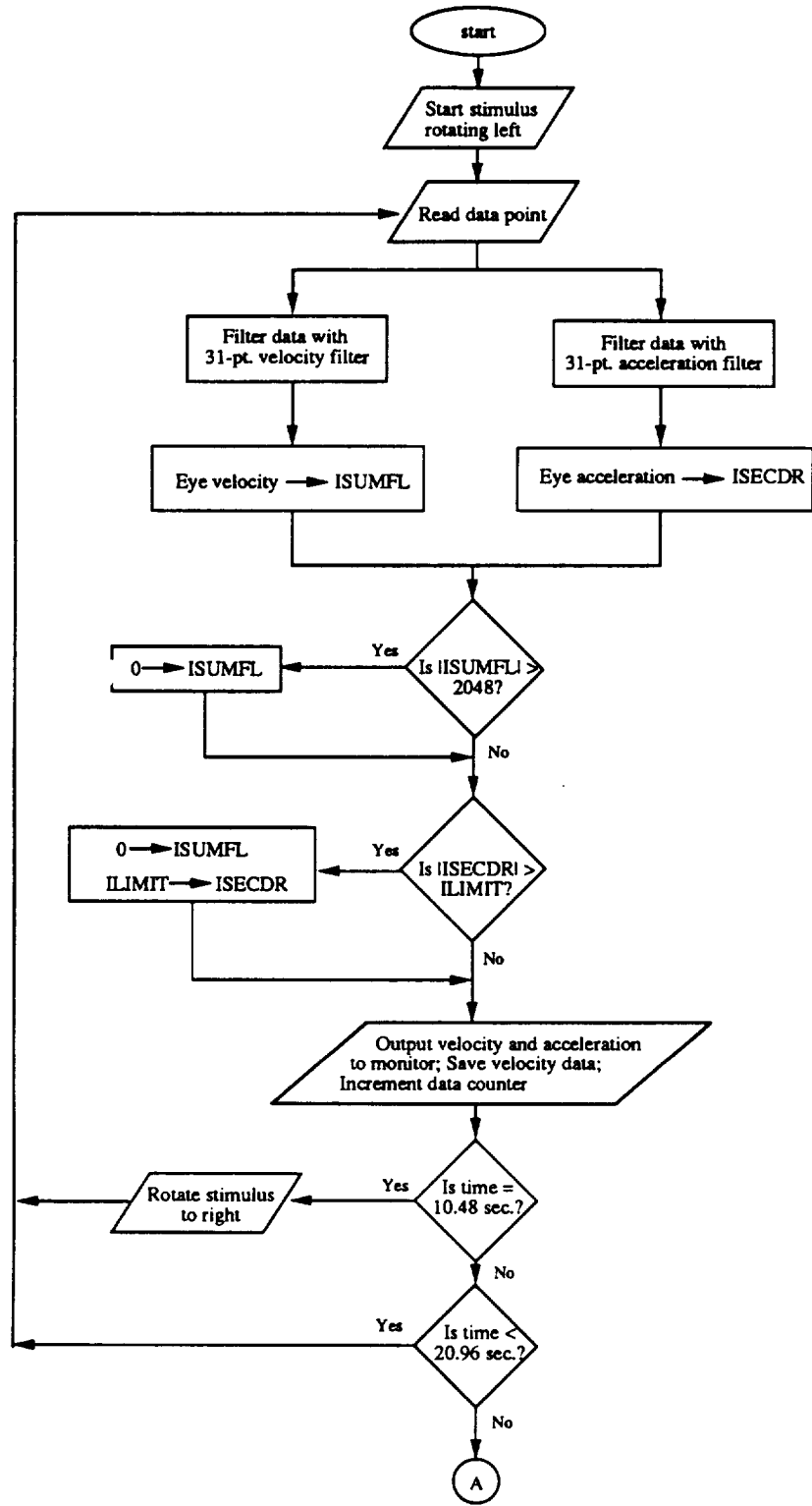


Figure 8A. Flowchart of calibration section of US Air Force Program.

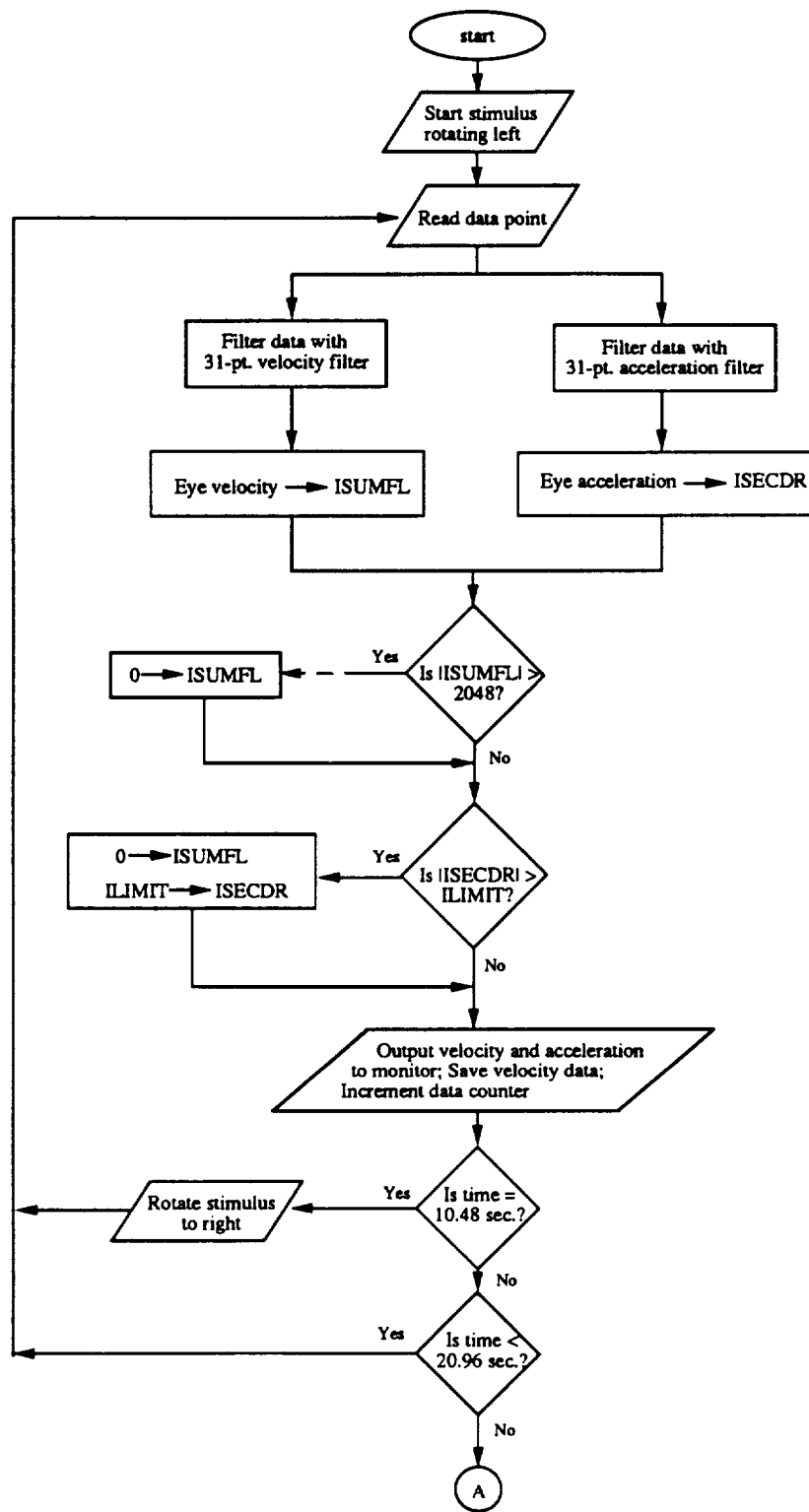


Figure 8A. Flowchart of calibration section of US Air Force Program.

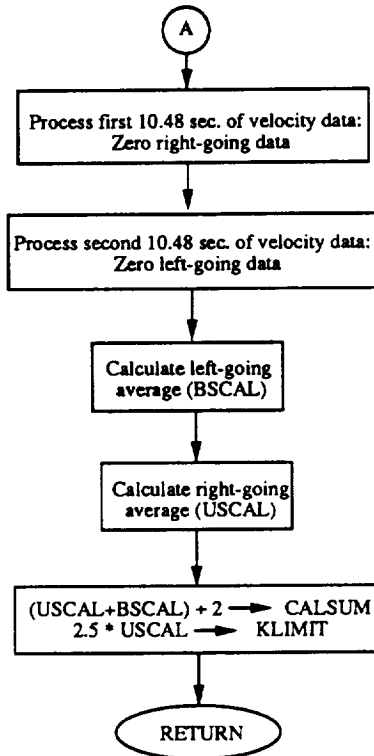


Figure 8B. Flowchart of calibration section of US Air Force Program (continued).

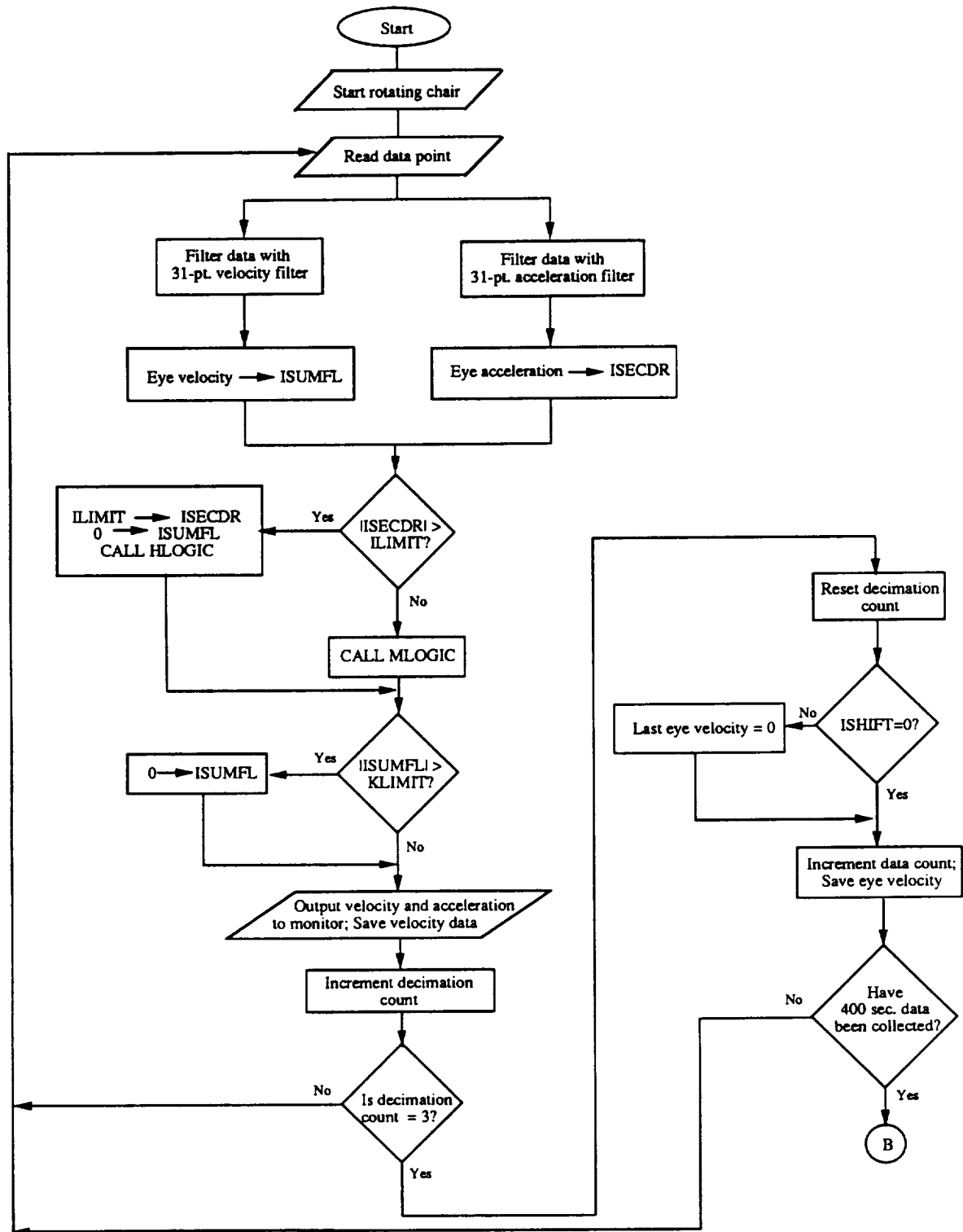


Figure 9A. Flowchart of analysis section of US Air Force Program.

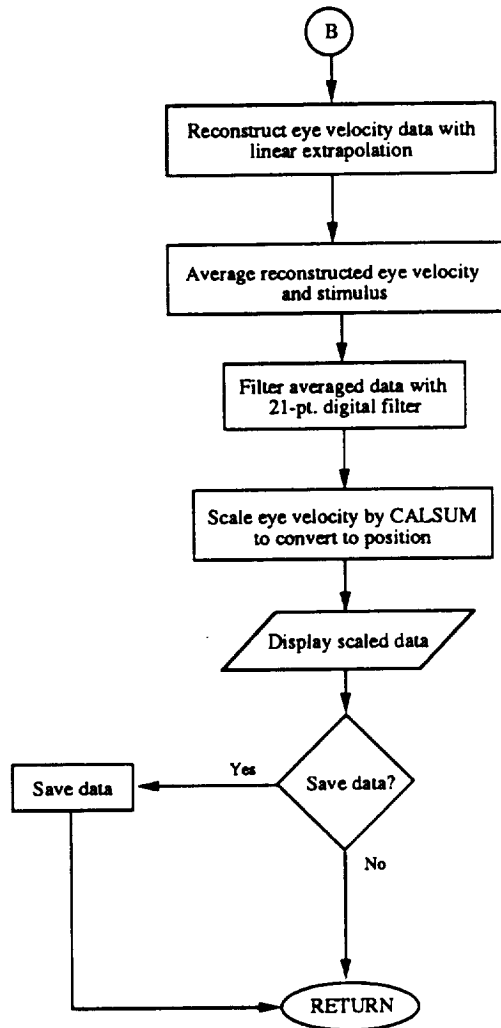


Figure 9B. Flowchart of analysis section of US Air Force Program (continued).

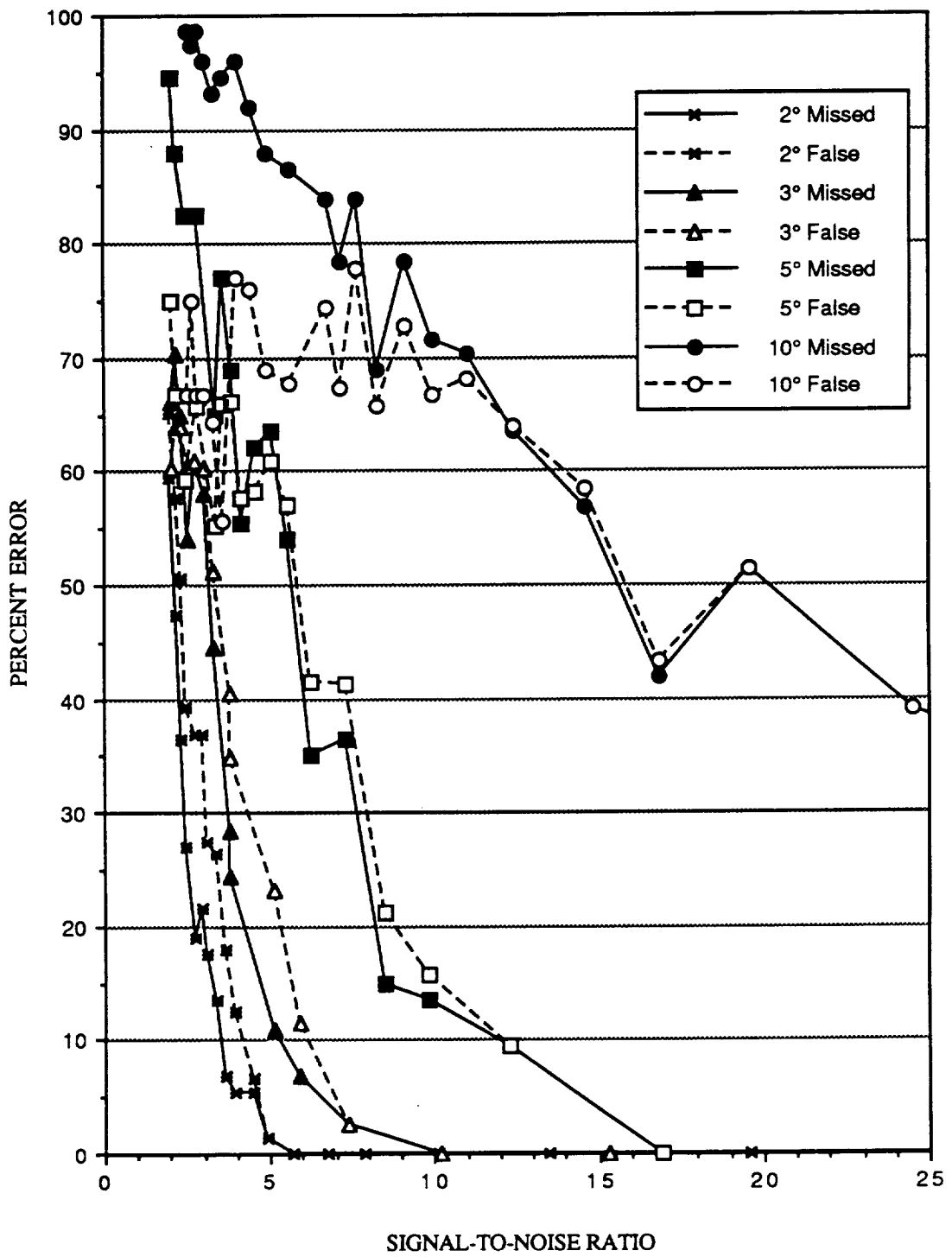


Figure 10. Plot of individual errors by USAF Program for midposition gaze data (maximum SNR of 25).

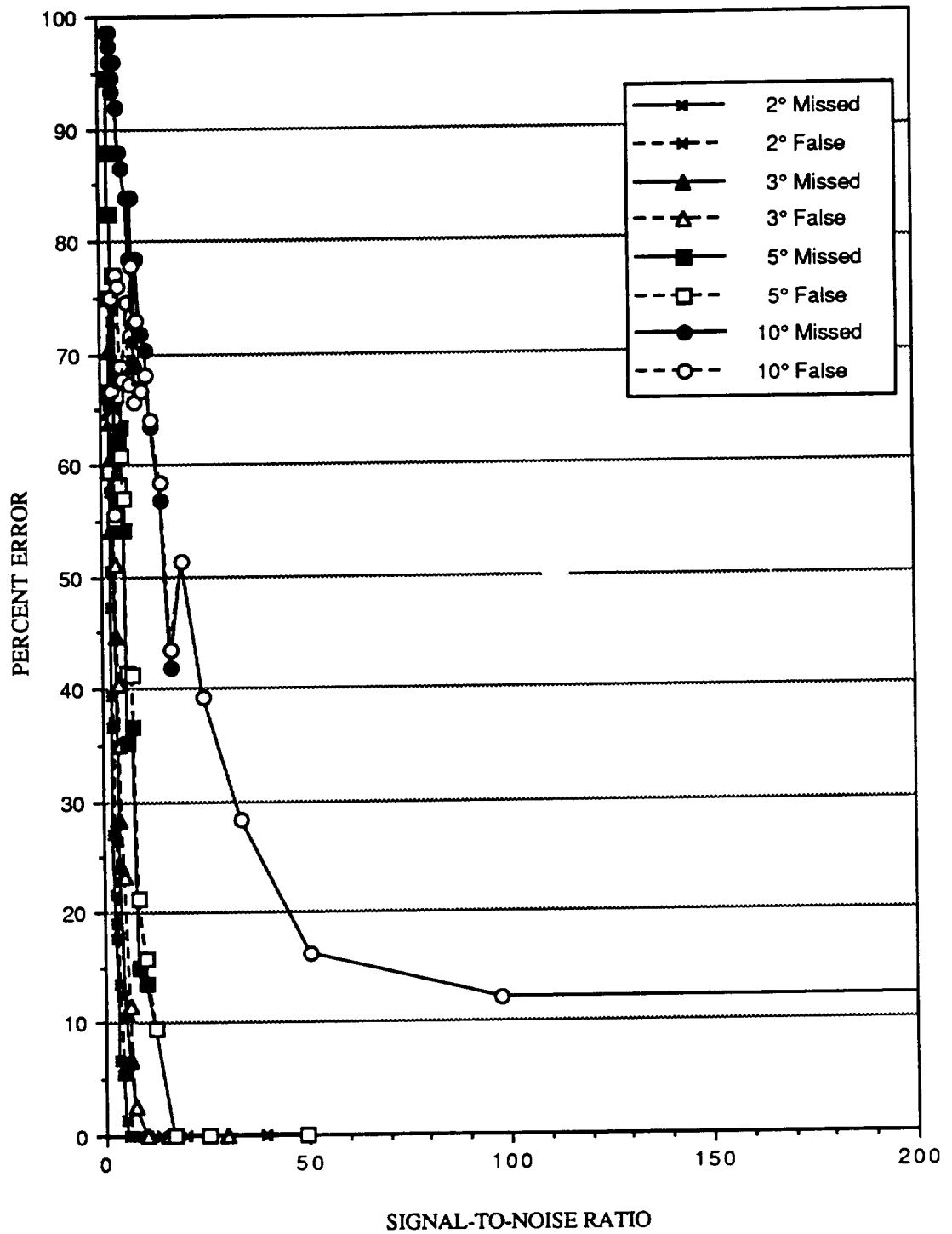


Figure 11. Plot of individual errors by USAF Program in midposition gaze data (maximum SNR of 200).

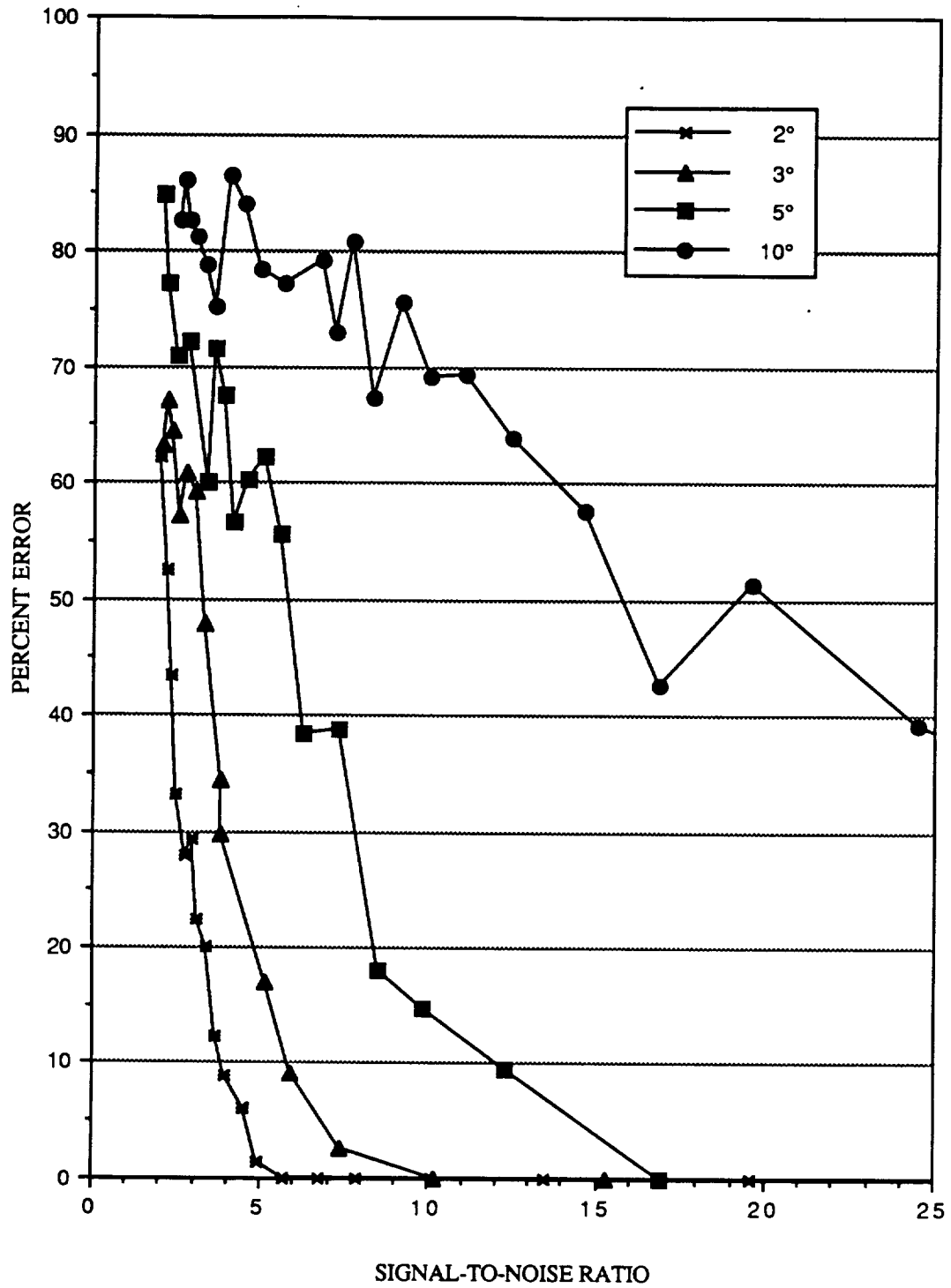


Figure 12. Plot of Error Index by USAF Program for midposition gaze data.

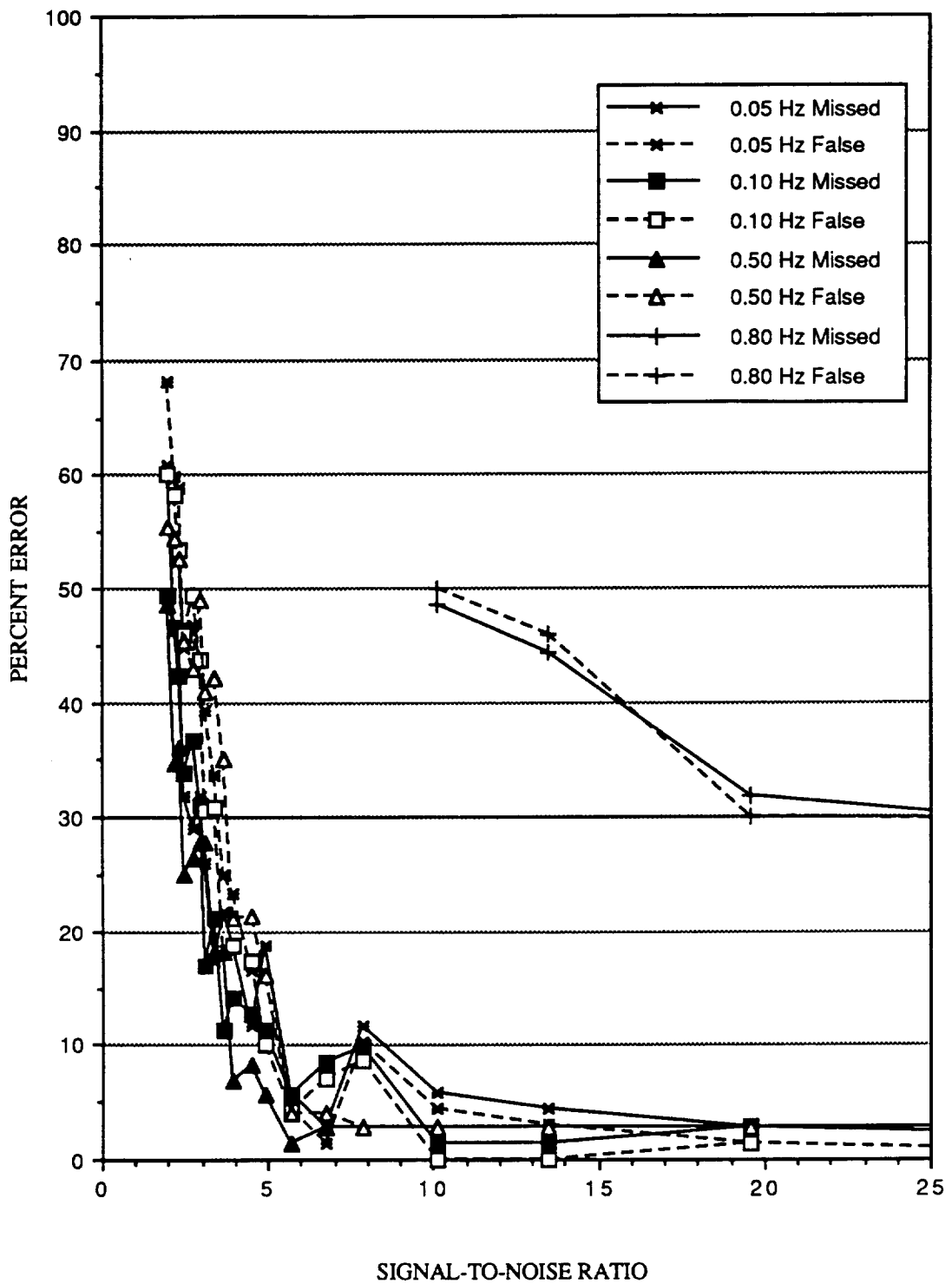


Figure 13. Plot of individual errors by USAF Program for sinusoidal pursuit data with two (2) degrees of saccadic jump.

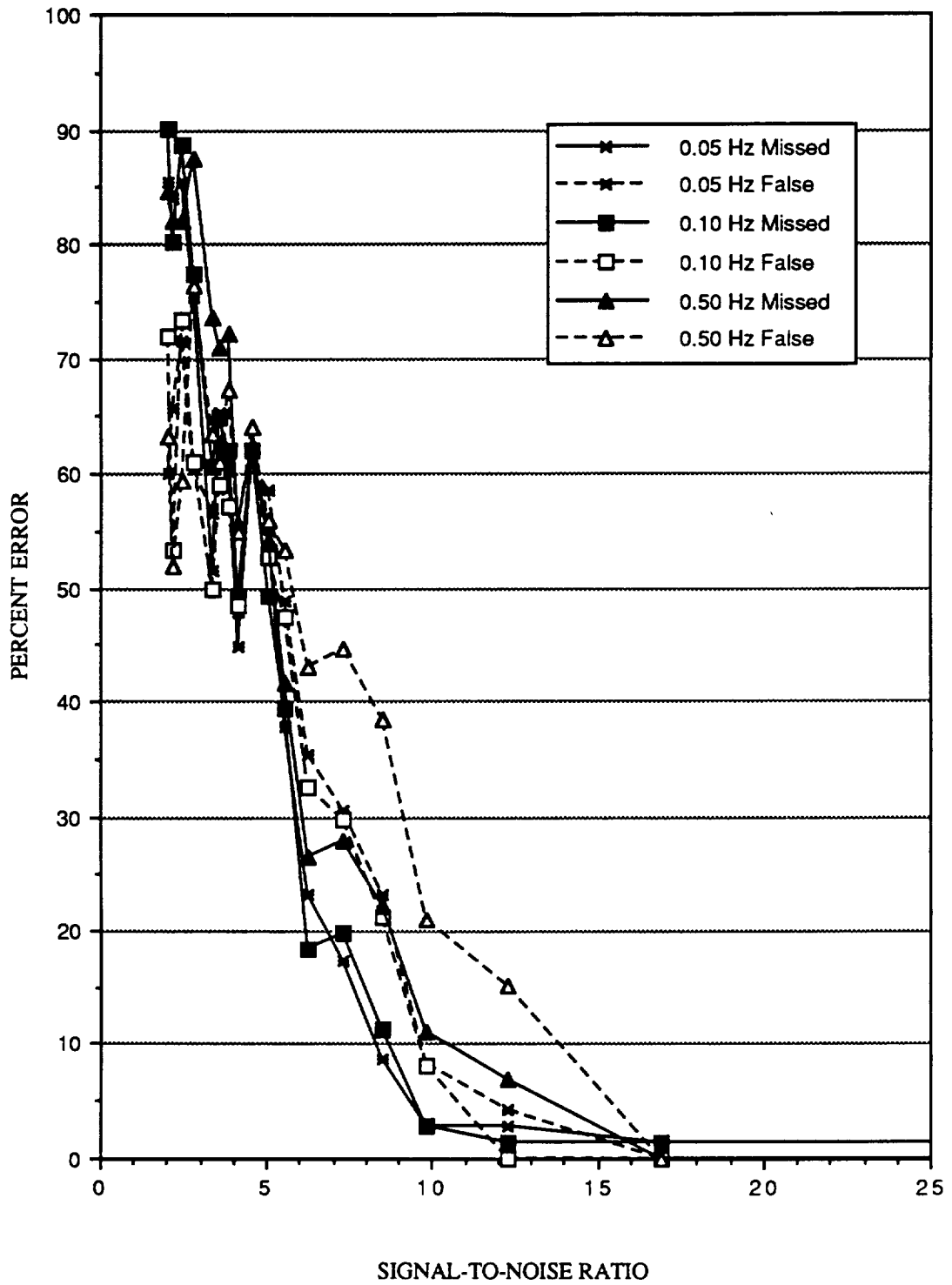


Figure 14. Plot of individual errors by USAF Program for sinusoidal pursuit data with five (5) degrees of saccadic jump.

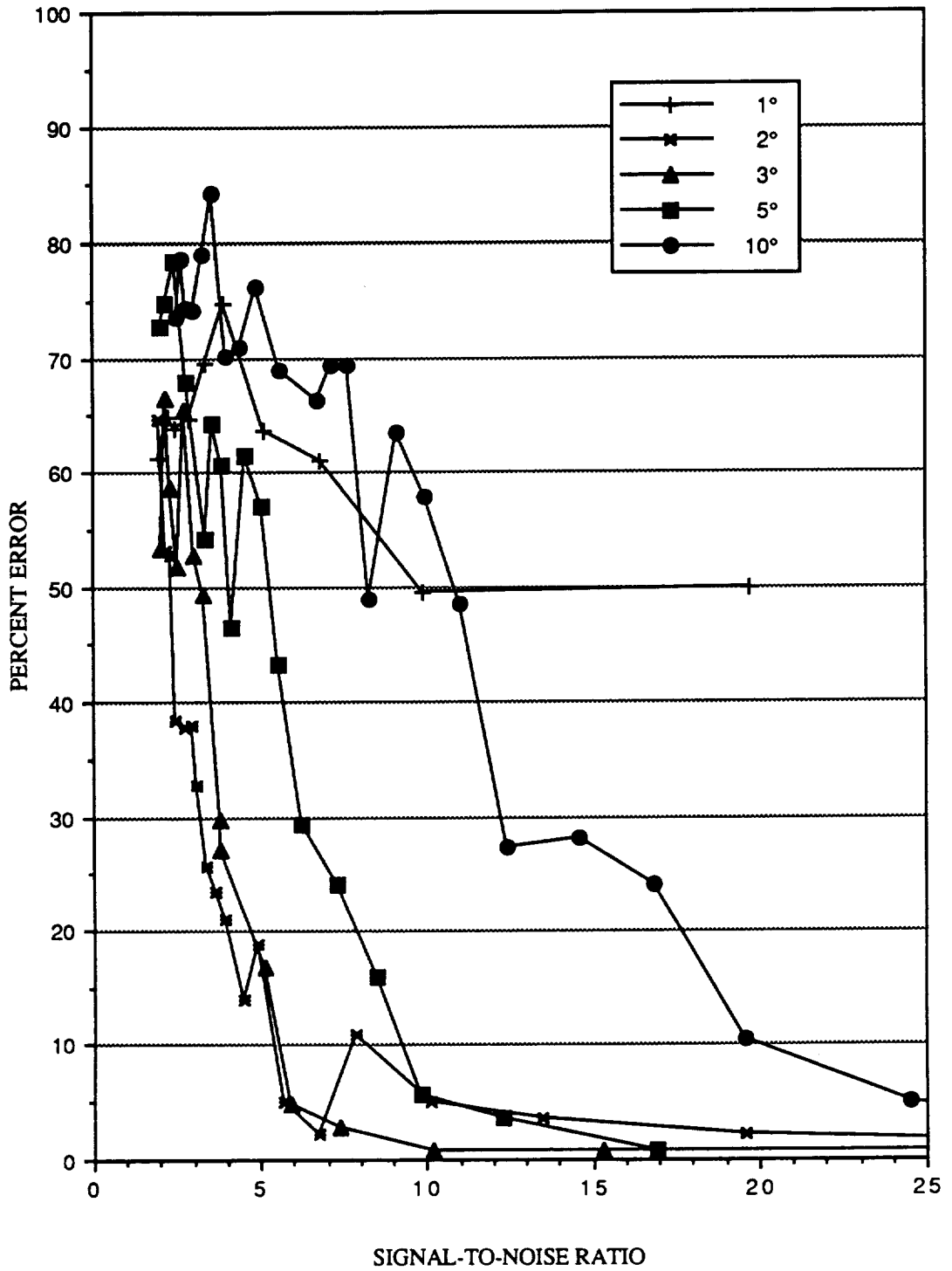


Figure 15. Plot of Error Index by USAF Program for 0.05 Hz sinusoidal pursuit data.

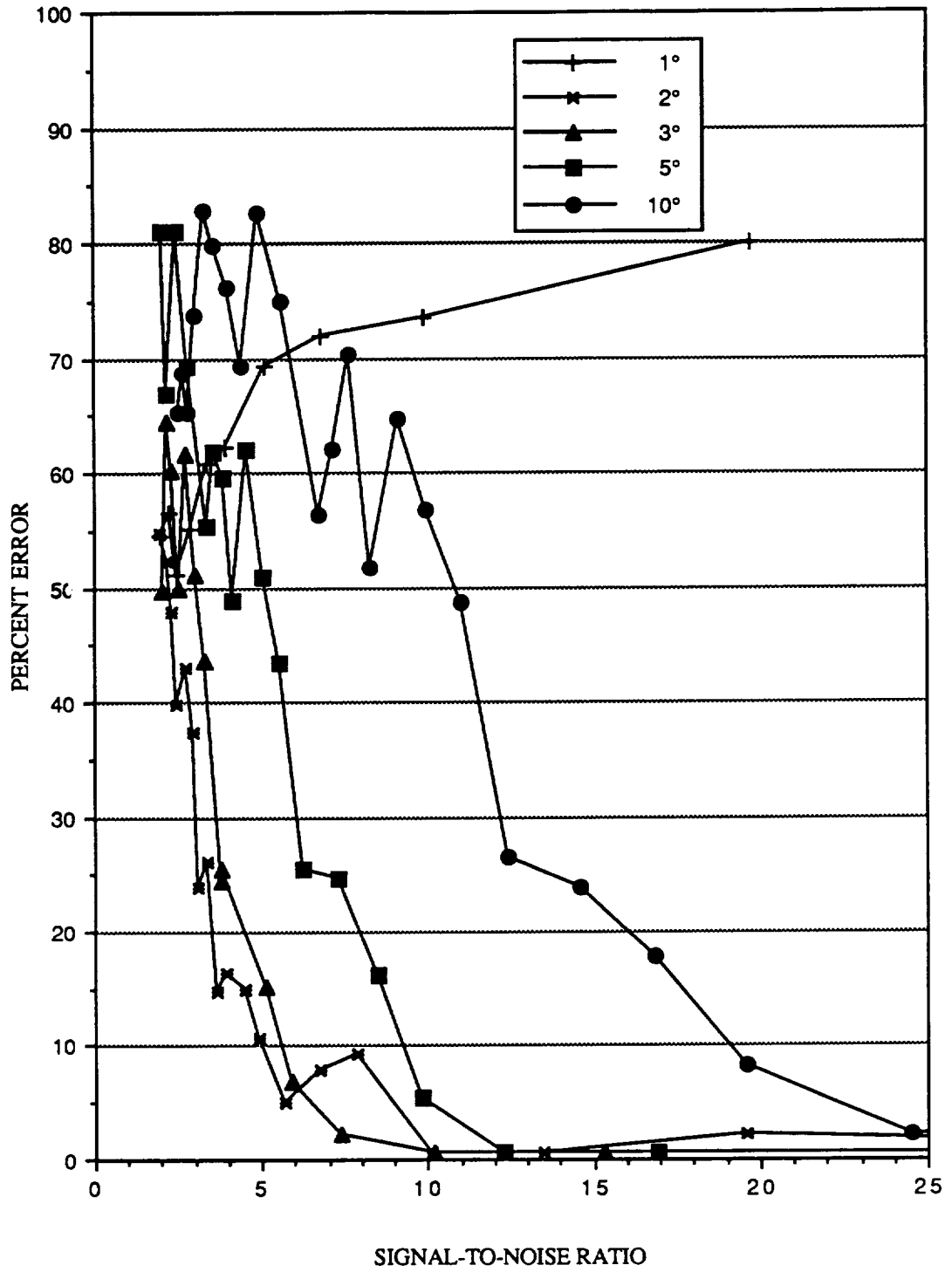


Figure 16. Plot of Error Index by USAF Program for 0.10 Hz sinusoidal pursuit data.

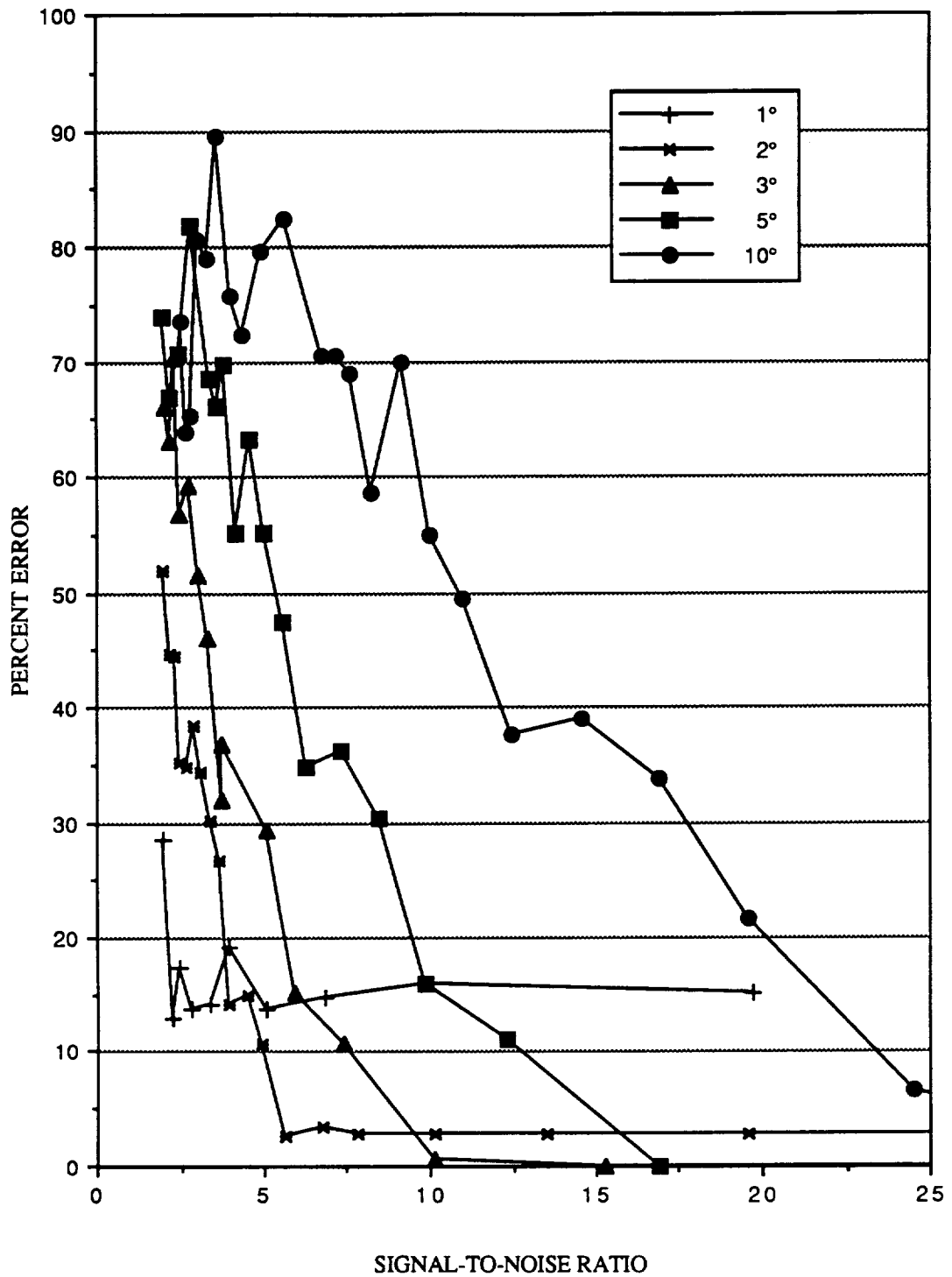


Figure 17. Plot of Error Index by USAF Program for 0.50 Hz sinusoidal pursuit data.

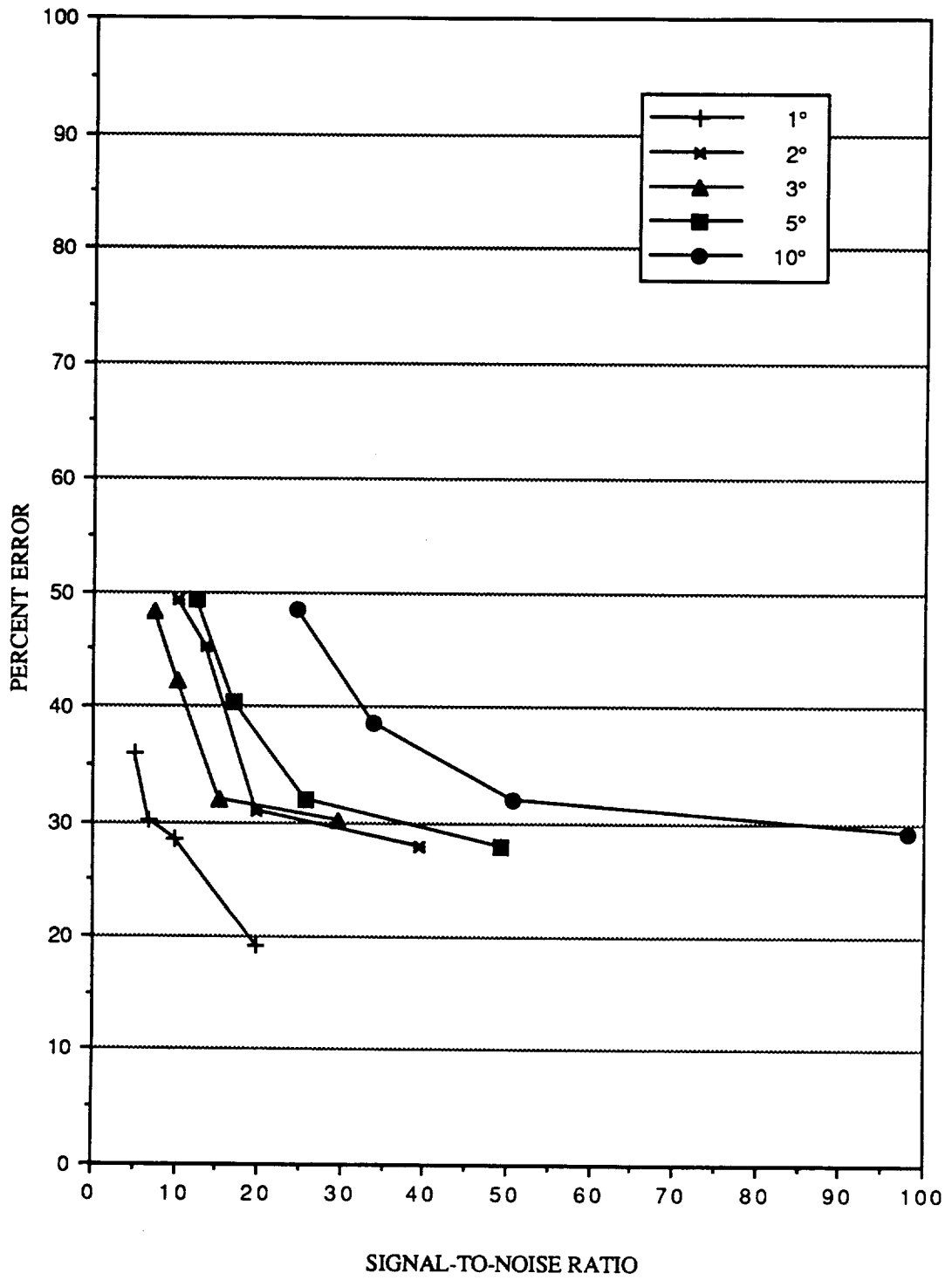


Figure 18. Plot of Error Index by USAF Program for 0.80 Hz sinusoidal pursuit data.

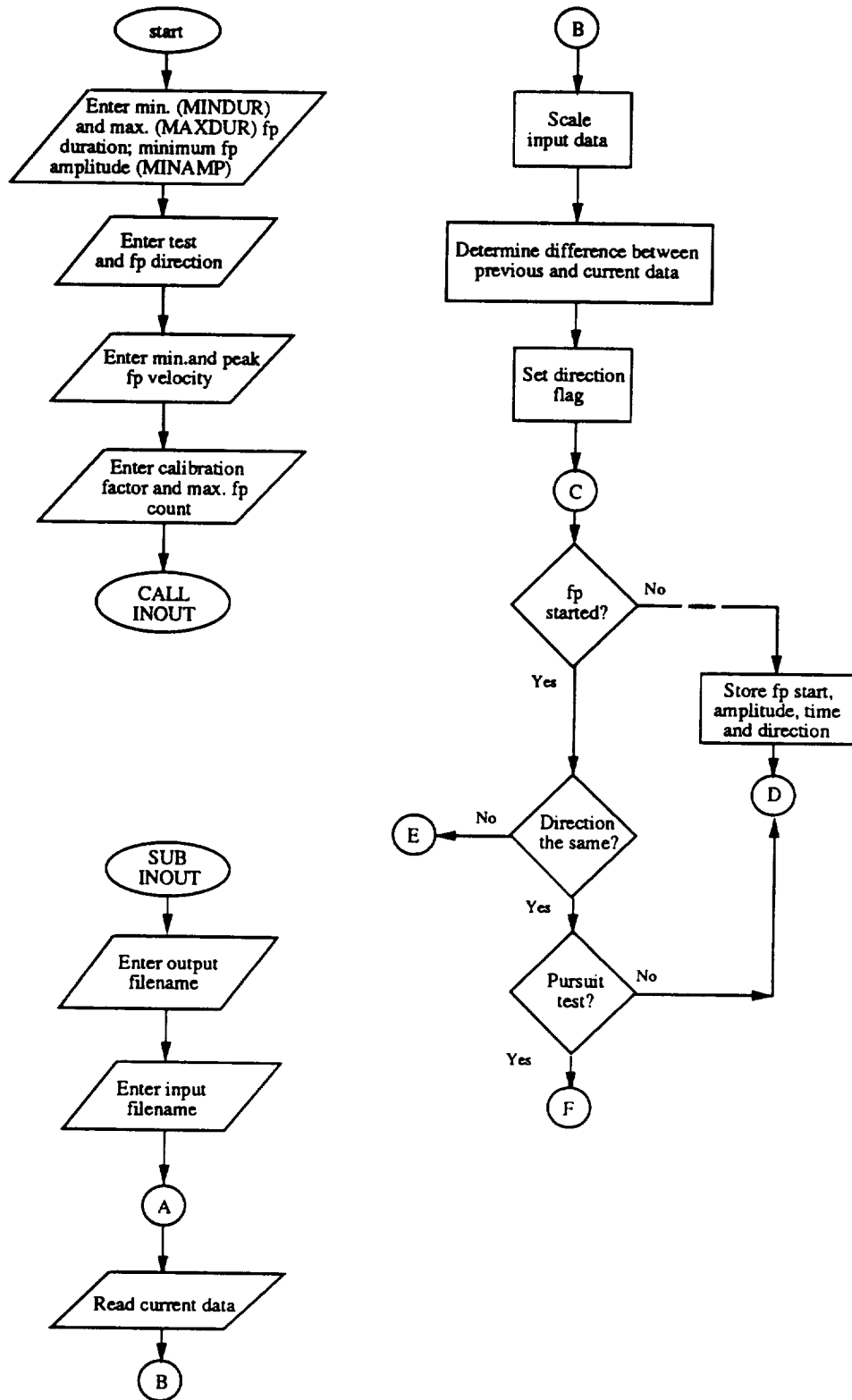


Figure 19A. Flowchart of UCLA SINUXEC Program.

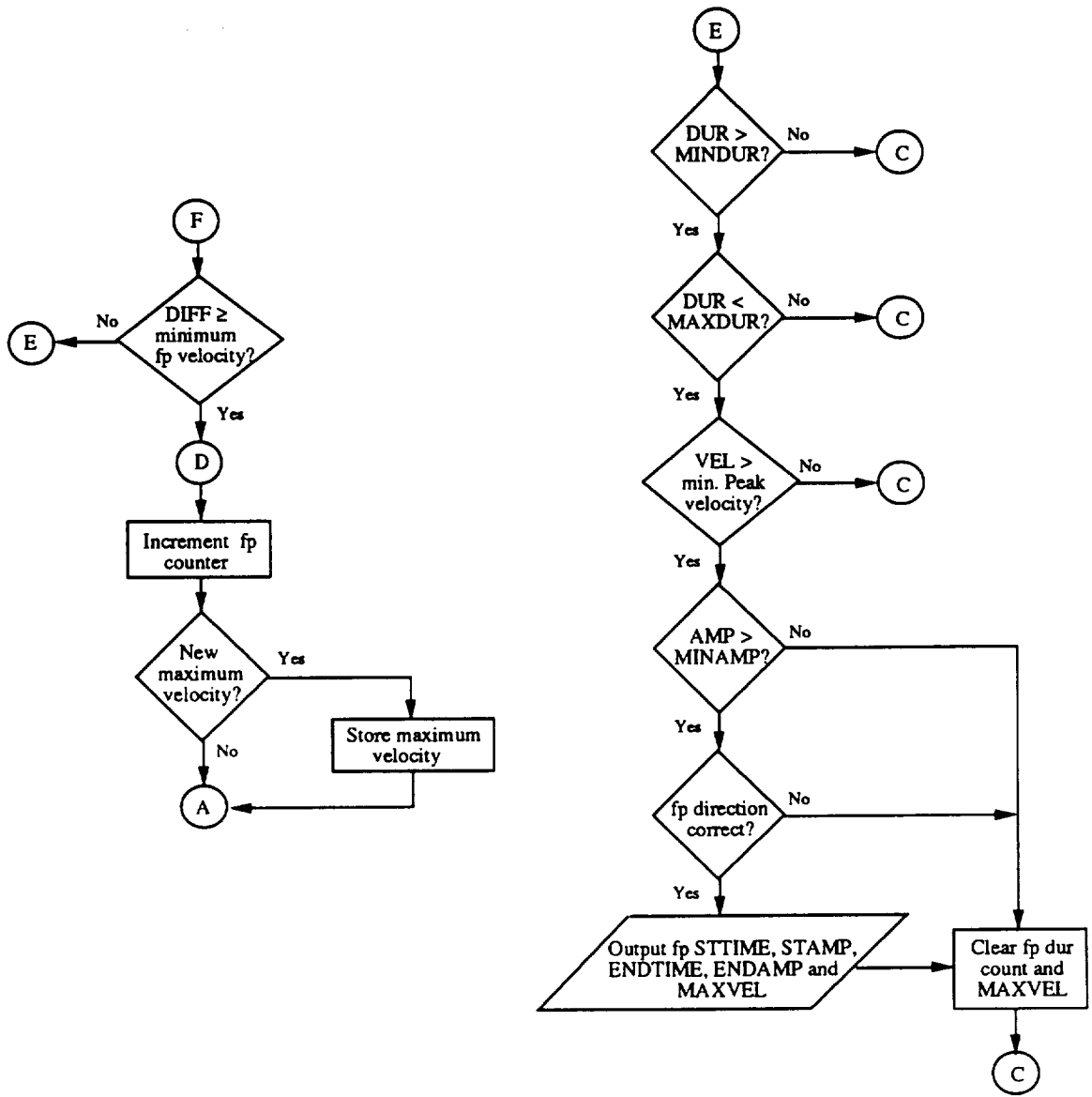


Figure 19B. Flowchart of UCLA SINUXEC Program (continued).

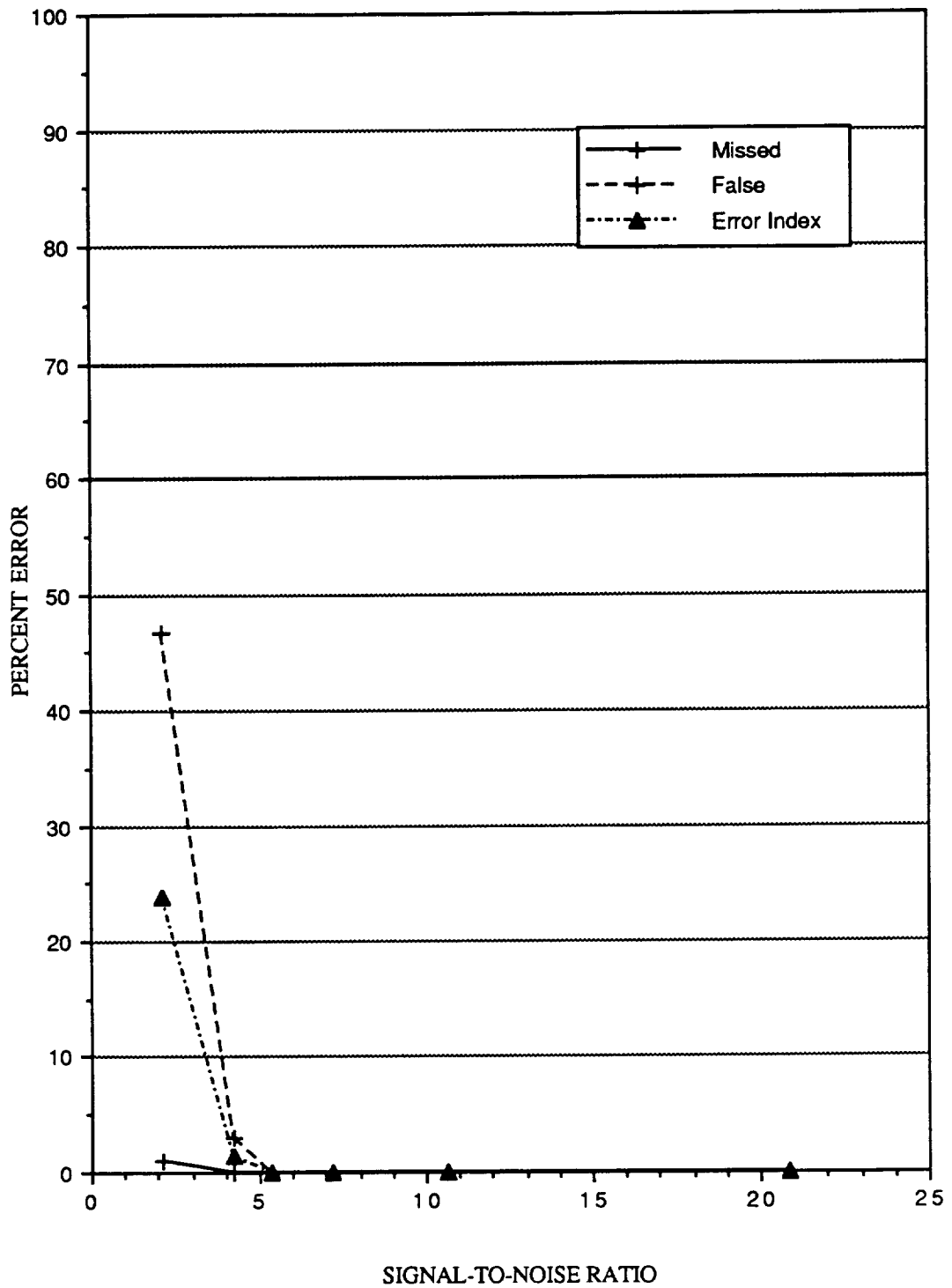


Figure 20. Plot of individual errors and Error Index by SINUXEC Program for midposition gaze data with one (1) degree of saccadic jump.

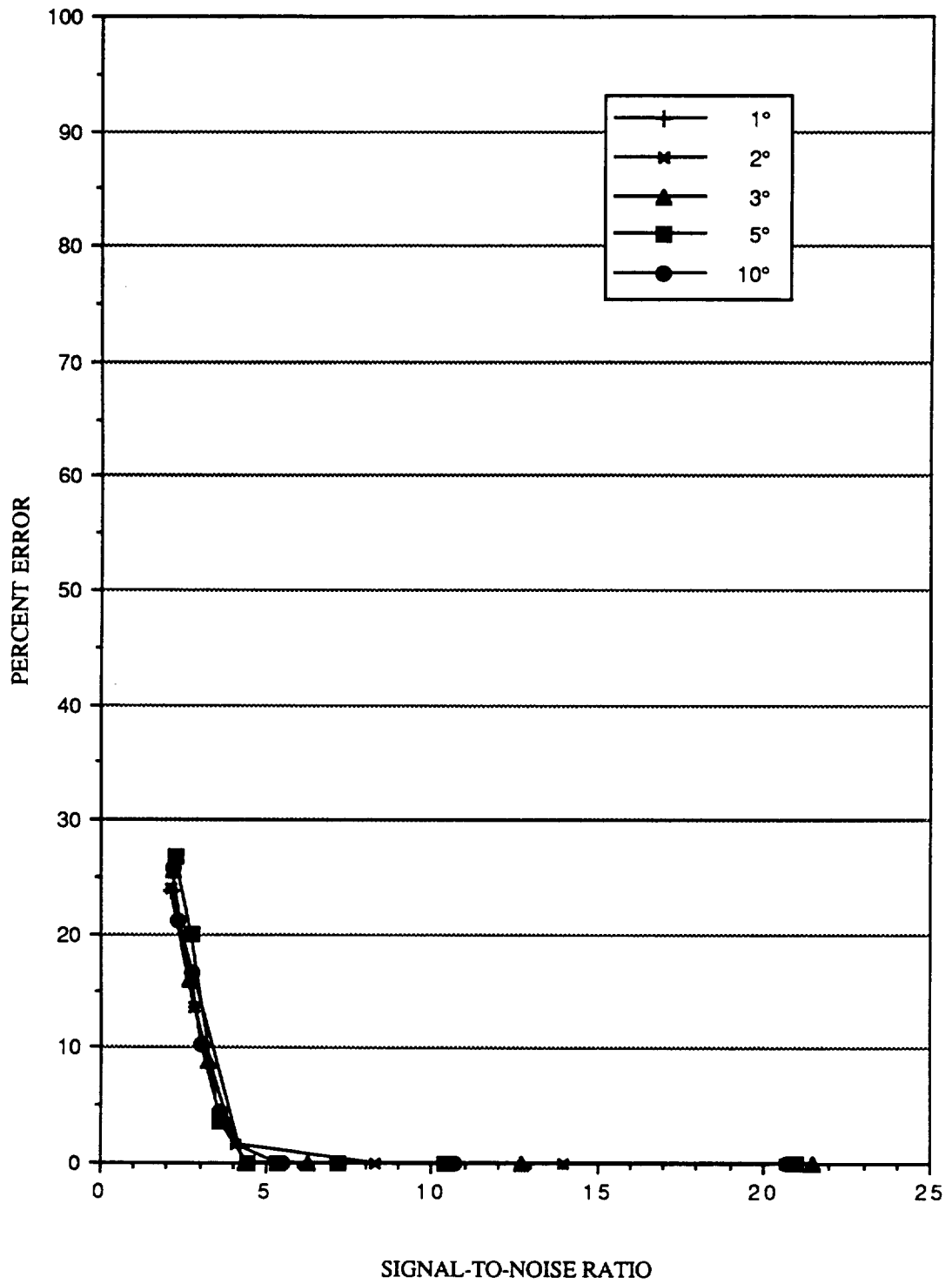


Figure 21. Plot of Error Index by SINUXEC Program for midposition gaze data.

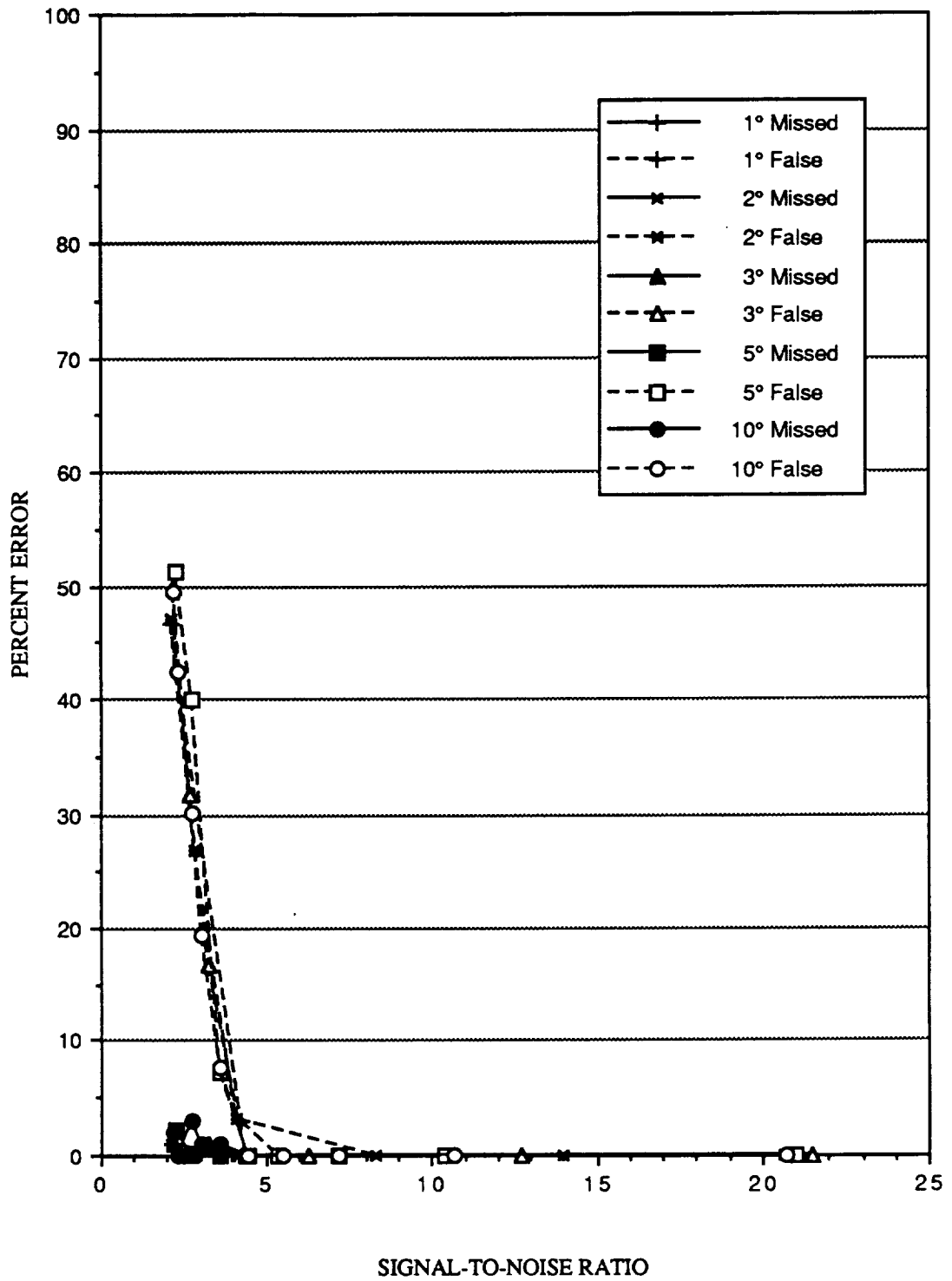


Figure 22. Plot of individual errors by SINUXEC Program for midposition gaze data.

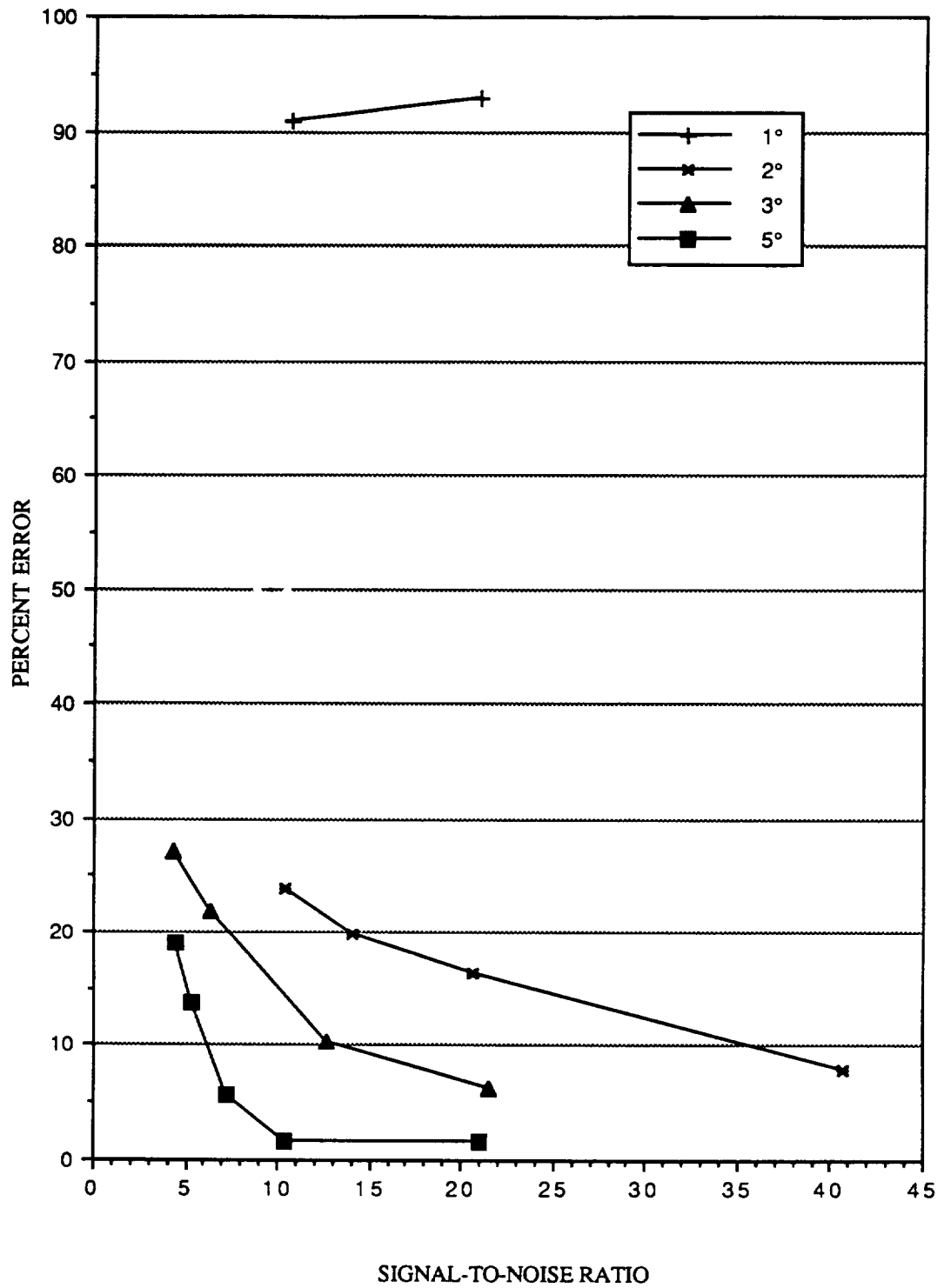


Figure 23. Plot of Error Index by SINUXEC Program for 0.50 Hz sinusoidal pursuit data.

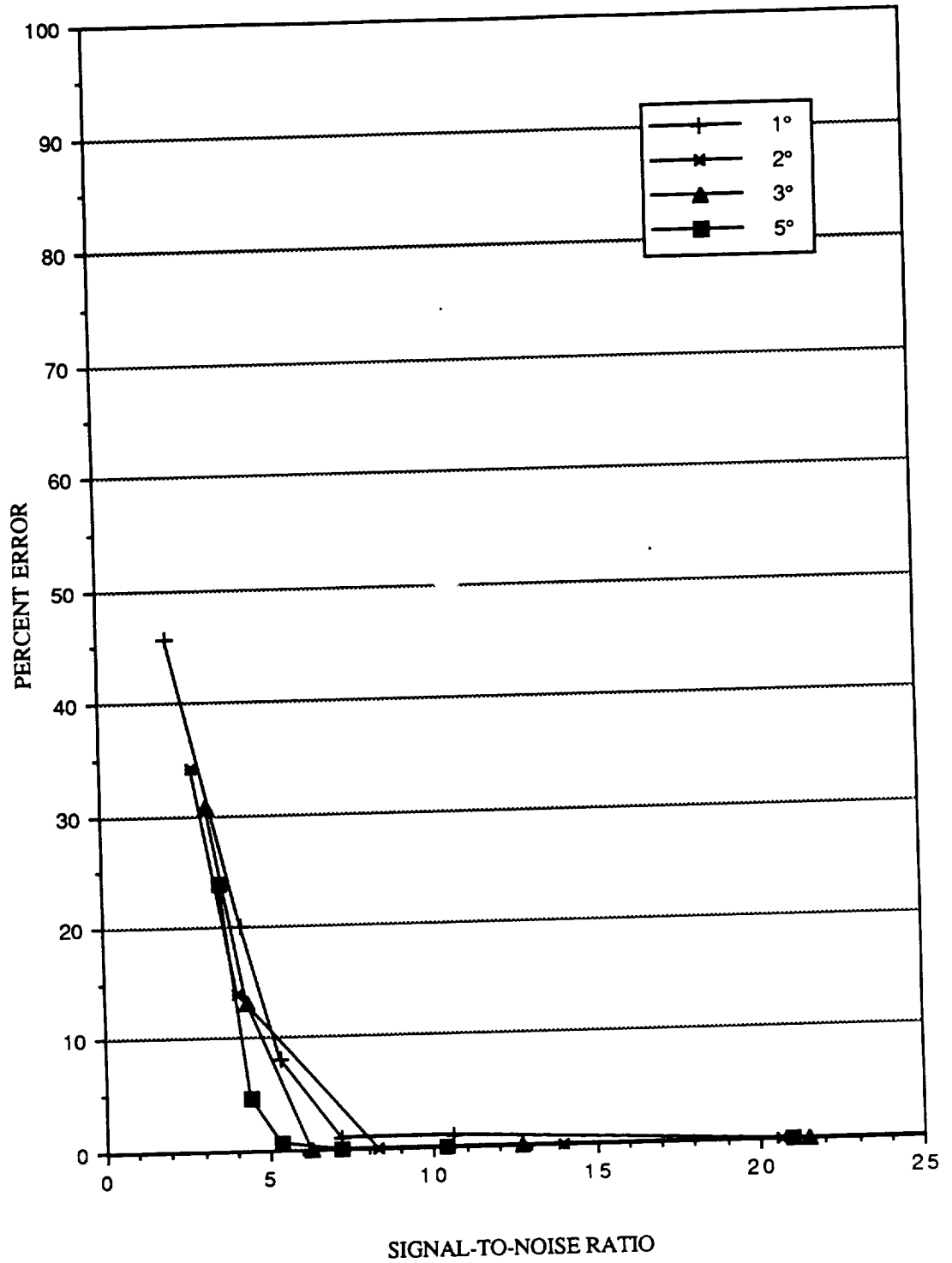


Figure 24. Plot of Error Index by SINUXEC Program for 0.05 Hz sinusoidal pursuit data.

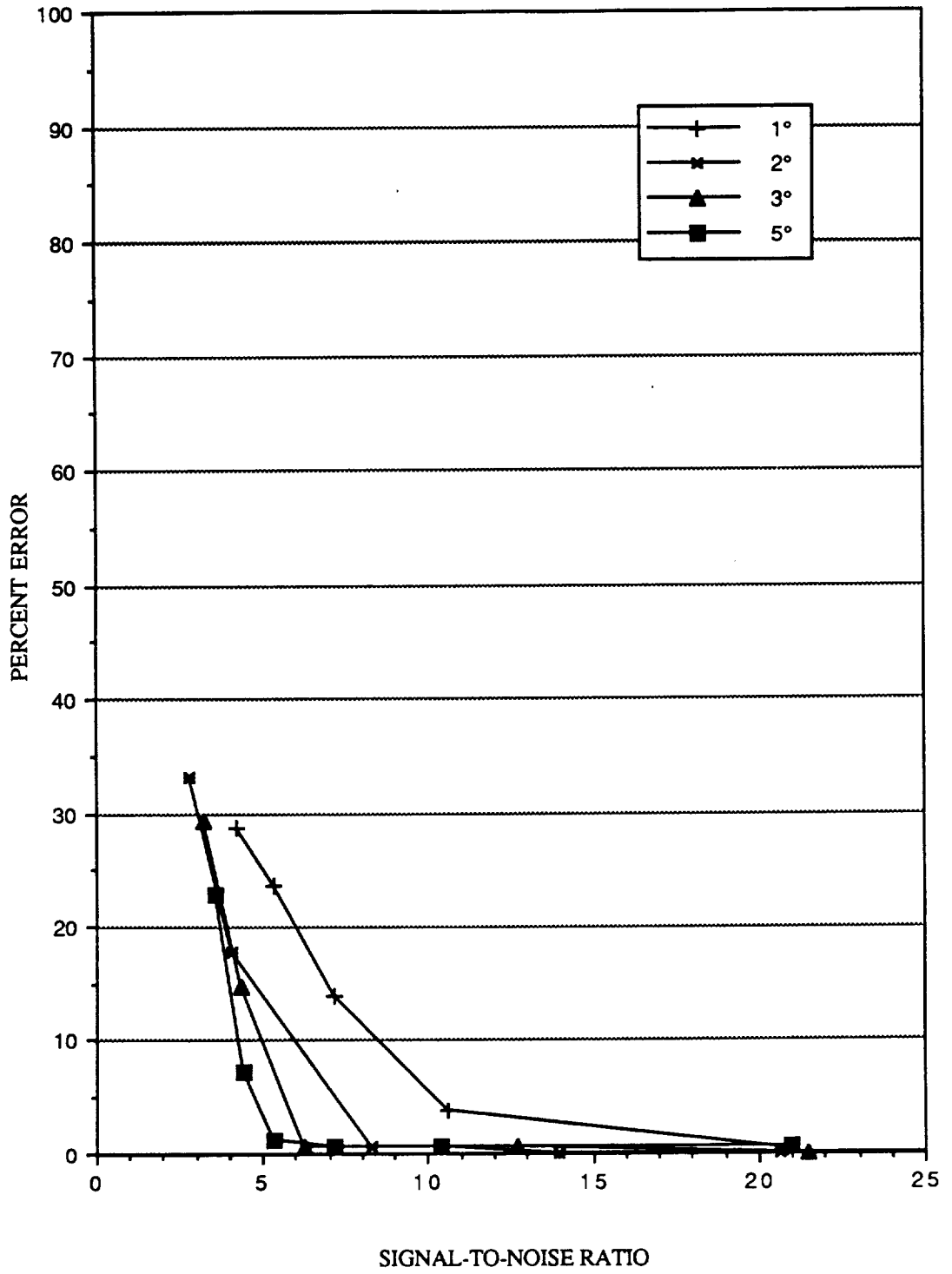


Figure 25. Plot of Error Index by SINUXEC Program for 0.10 Hz sinusoidal pursuit data.

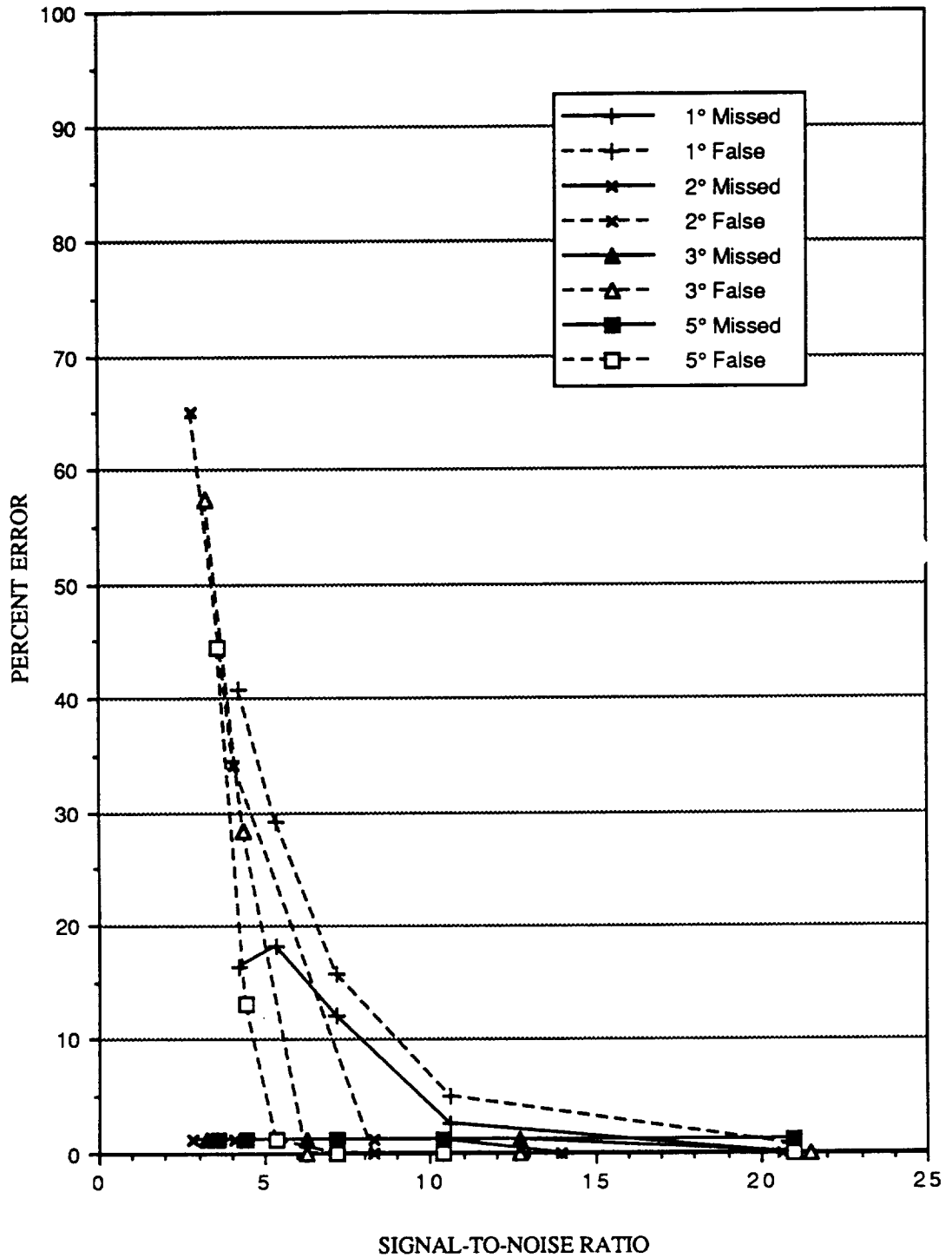


Figure 26. Plot of individual errors by SINUXEC Program for 0.10 Hz sinusoidal pursuit data.

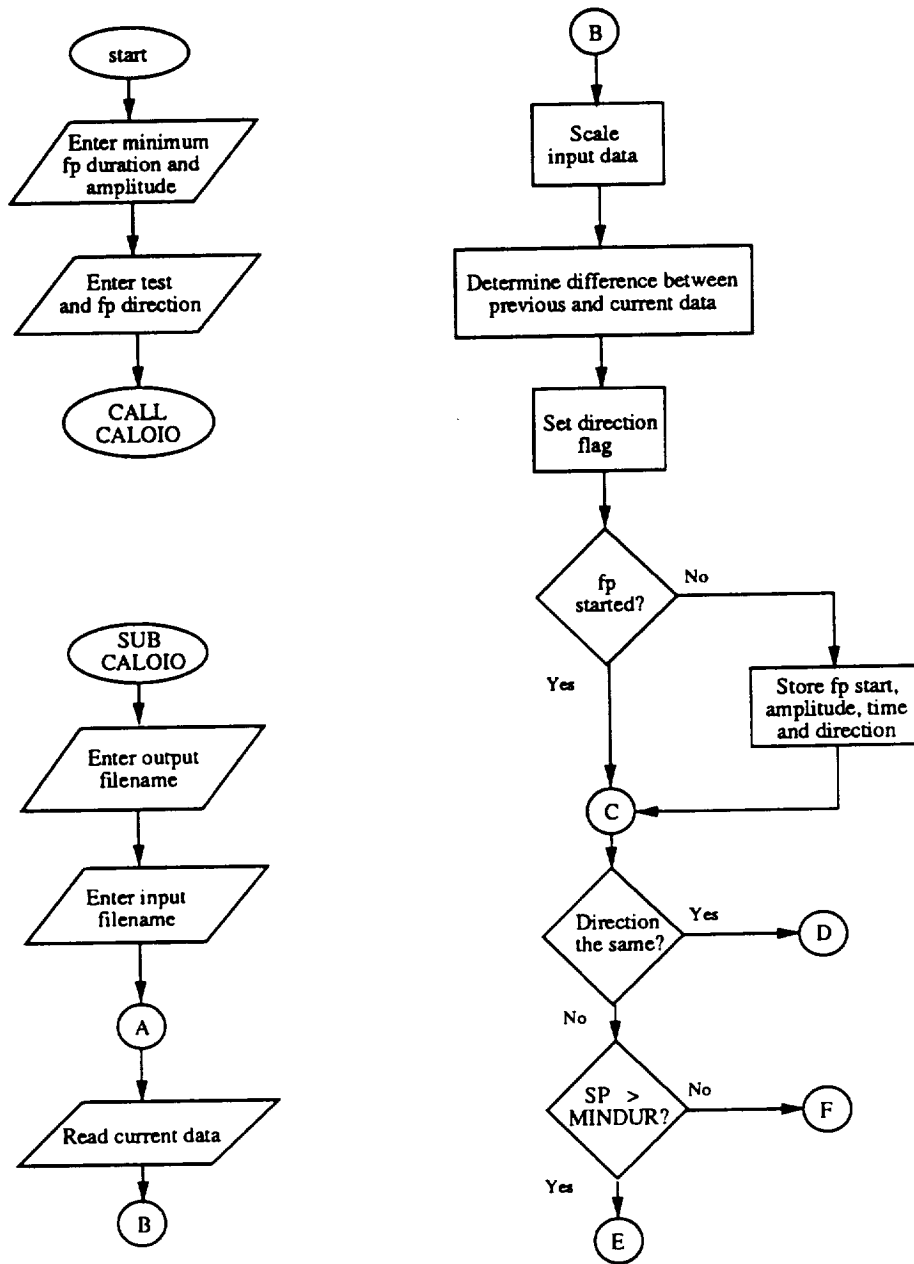


Figure 27A. Flowchart of UCLA CALORXEC Program.

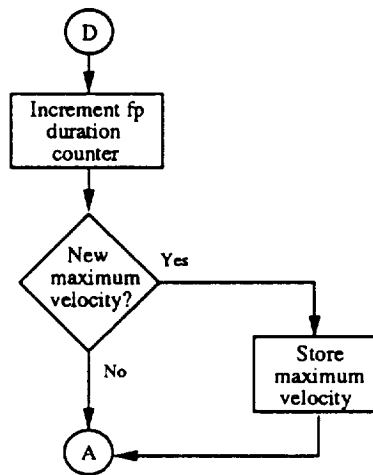
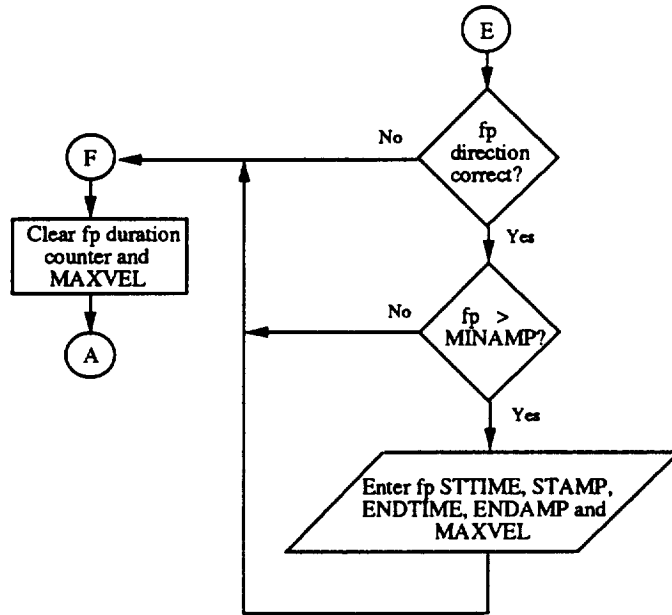


Figure 27B. Flowchart of UCLA CALORXEC Program (continued).

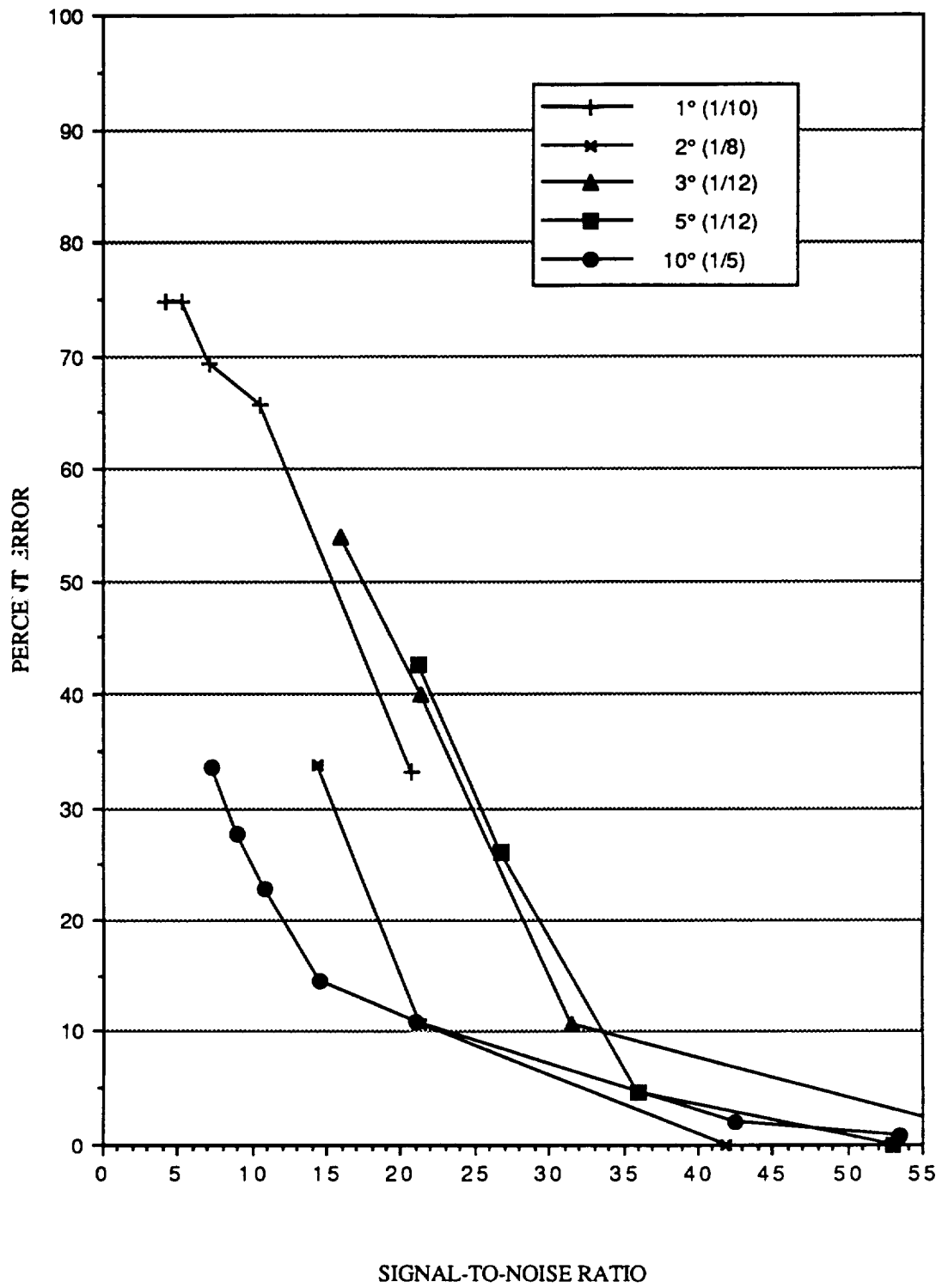


Figure 28. Plot of Error Index by CALORXEC Program for midposition gaze data.

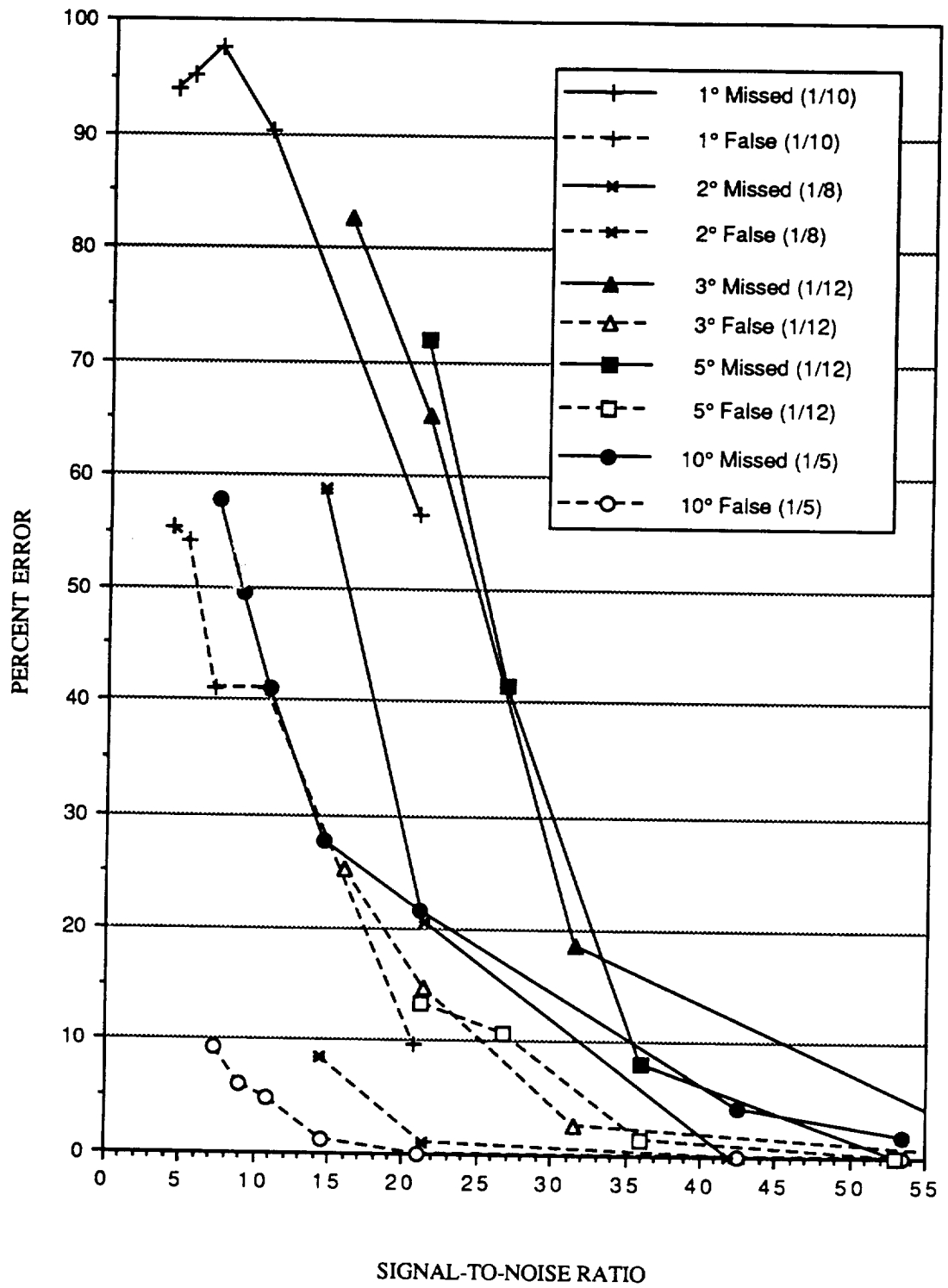


Figure 29. Plot of individual errors by CALORXEC Program for midposition gaze data.

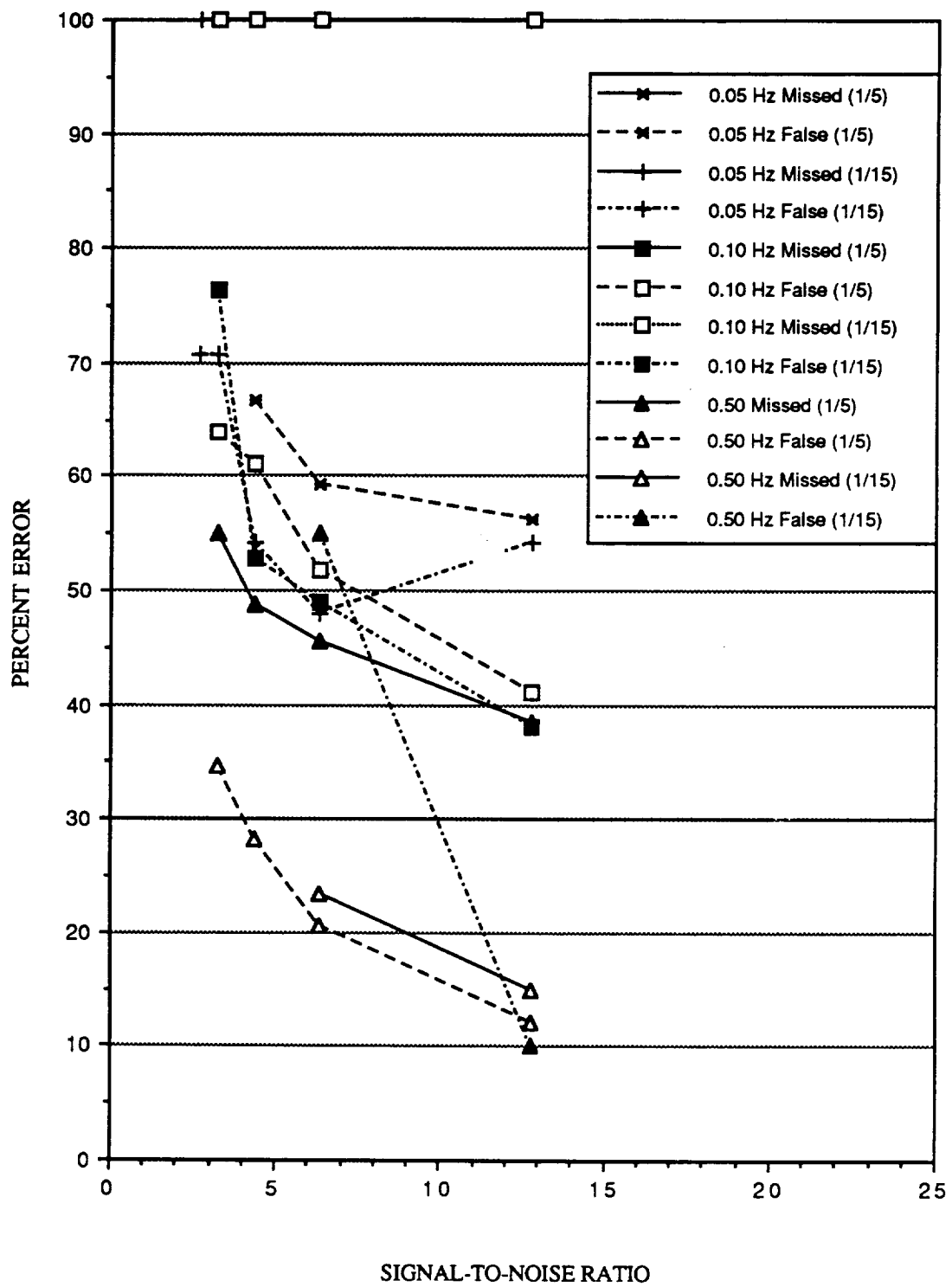


Figure 30. Plot of individual errors by CALORXEC Program for sinusoidal pursuit data with 1:5 and 1:15 data point ratios.

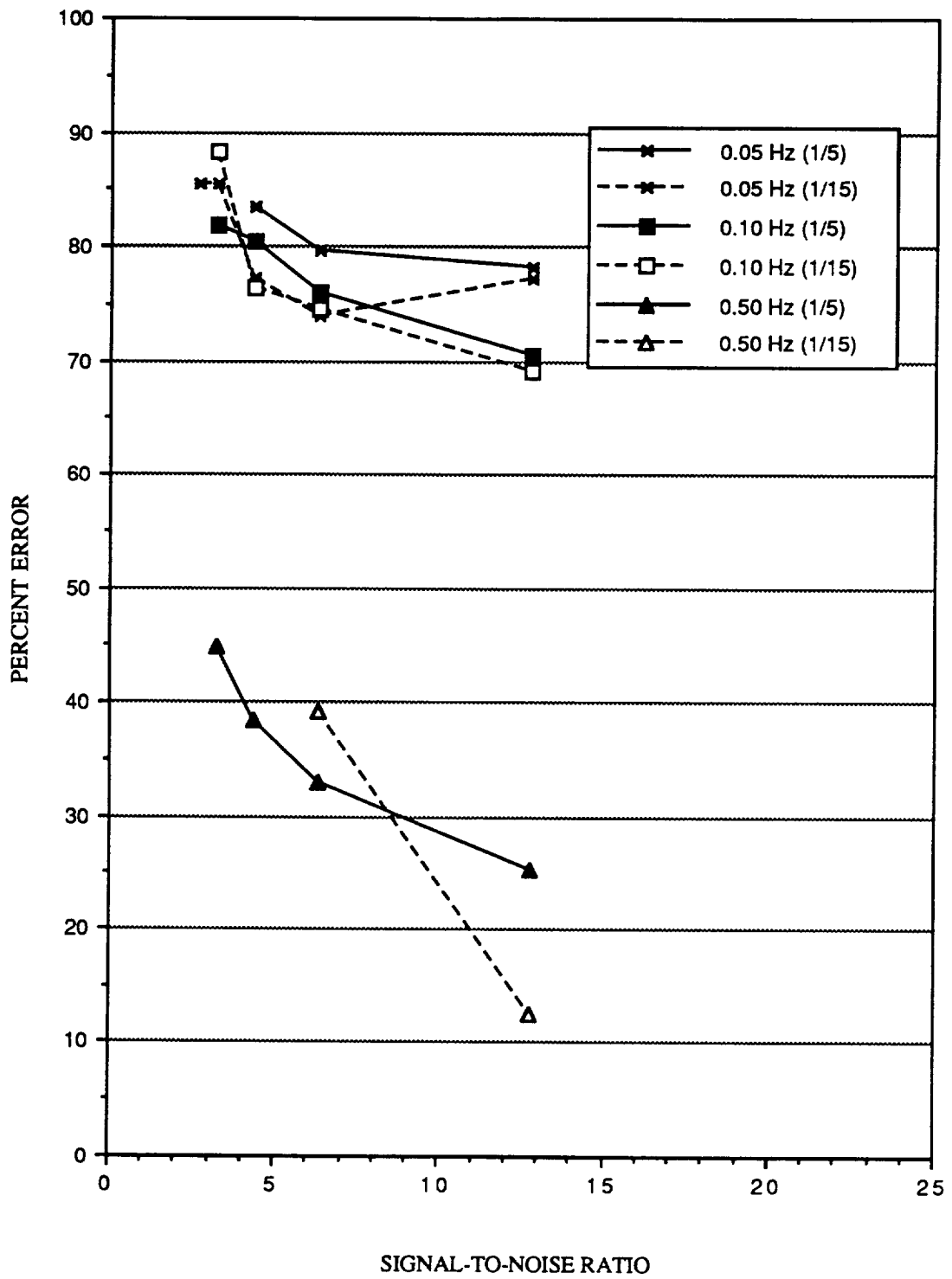


Figure 31. Plot of Error Index by CALORXEC Program for sinusoidal pursuit data with 1:5 and 1:15 data point ratios.

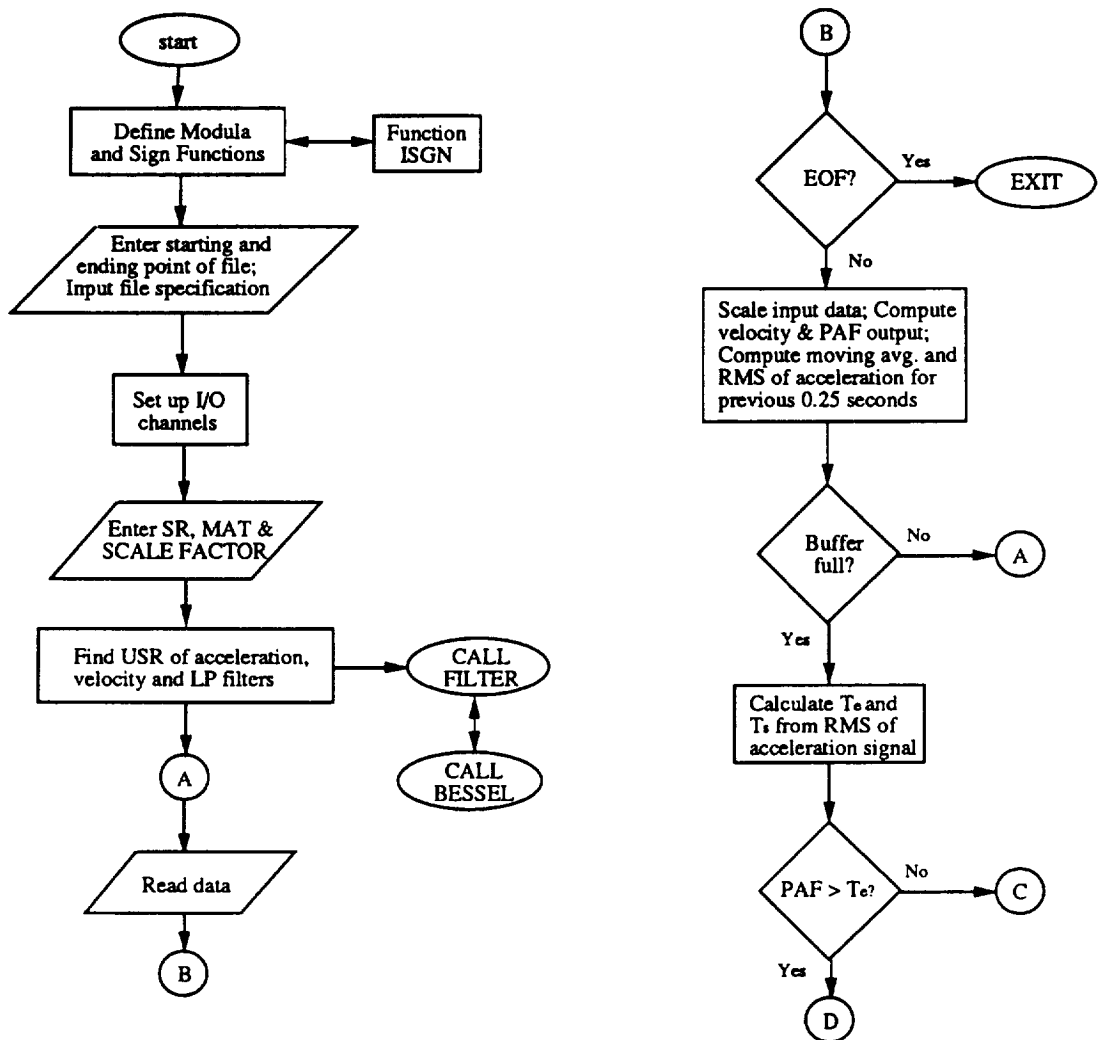


Figure 32A. Flowchart of MIT M2MI86 Program.

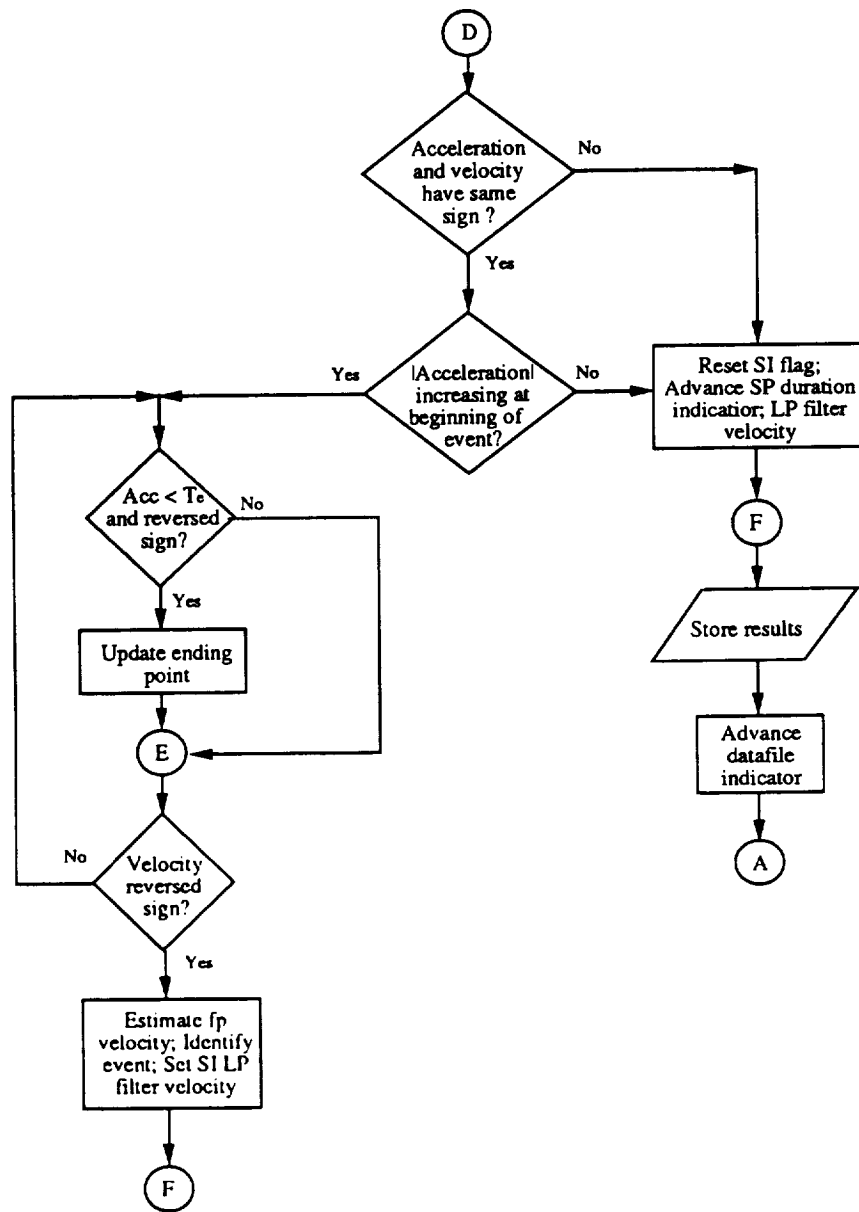


Figure 32B. Flowchart of MIT M2MI86 Program (continued).

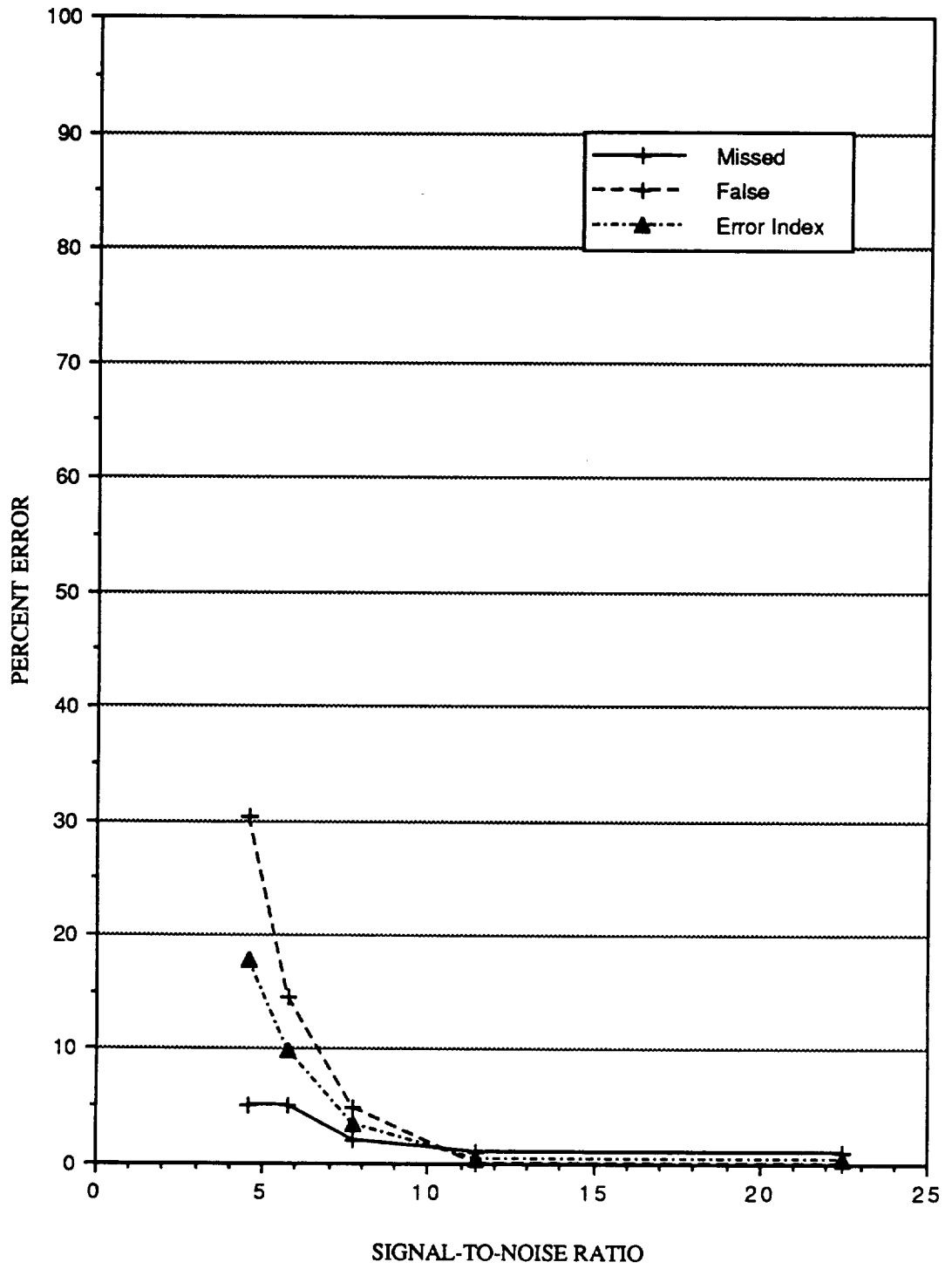


Figure 33. Plot of individual errors and Error Index by MIT M2MI86 Program for midposition gaze data with one (1) degree of saccadic jump.

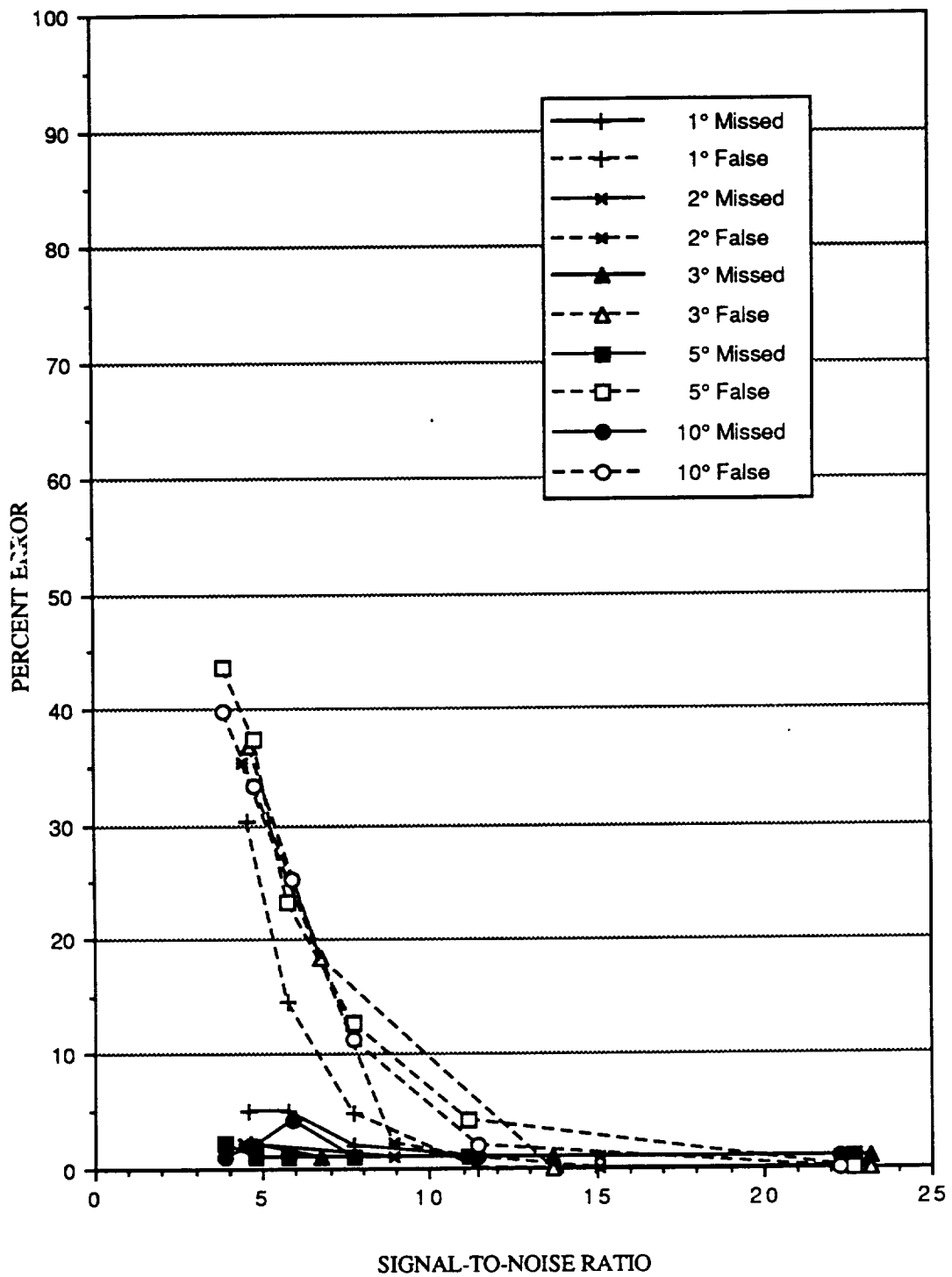


Figure 34. Plot of individual errors by MIT M2MI86 Program for midposition gaze data.

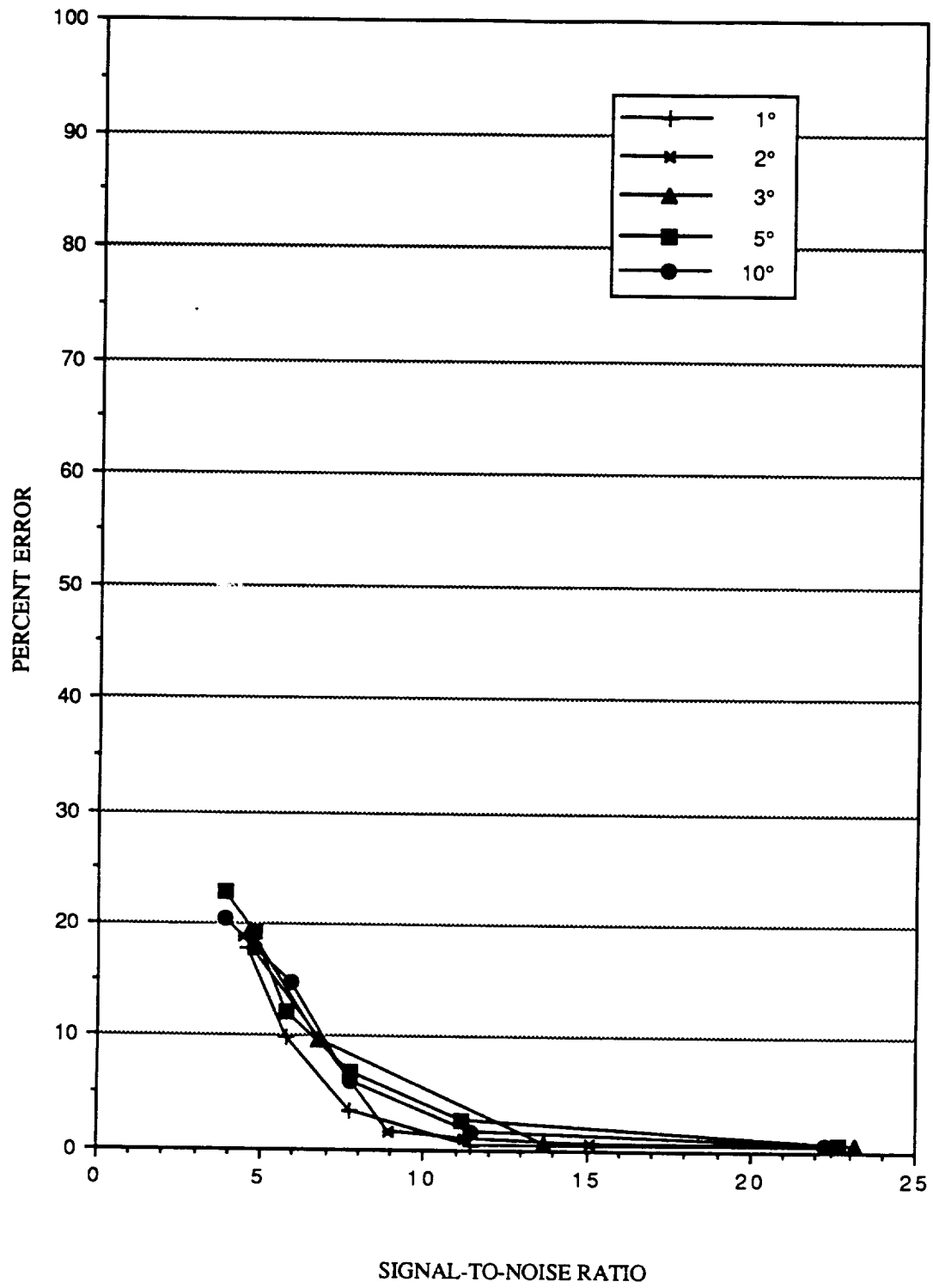


Figure 35. Plot of Error Index by MIT M2MI86 Program for midposition gaze data.

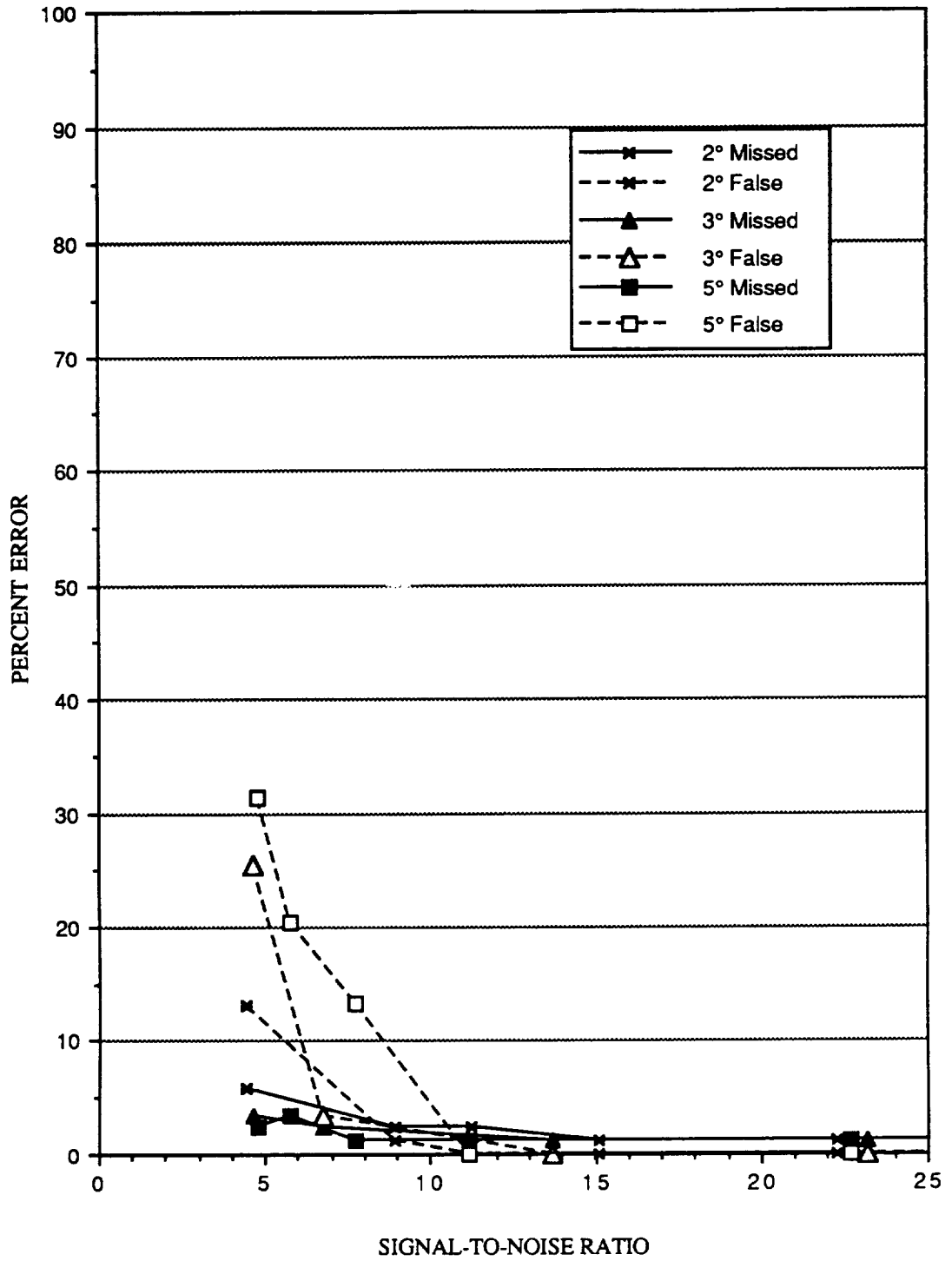


Figure 36. Plot of individual errors by MIT M2MI86 Program for 0.50 Hz sinusoidal pursuit data.

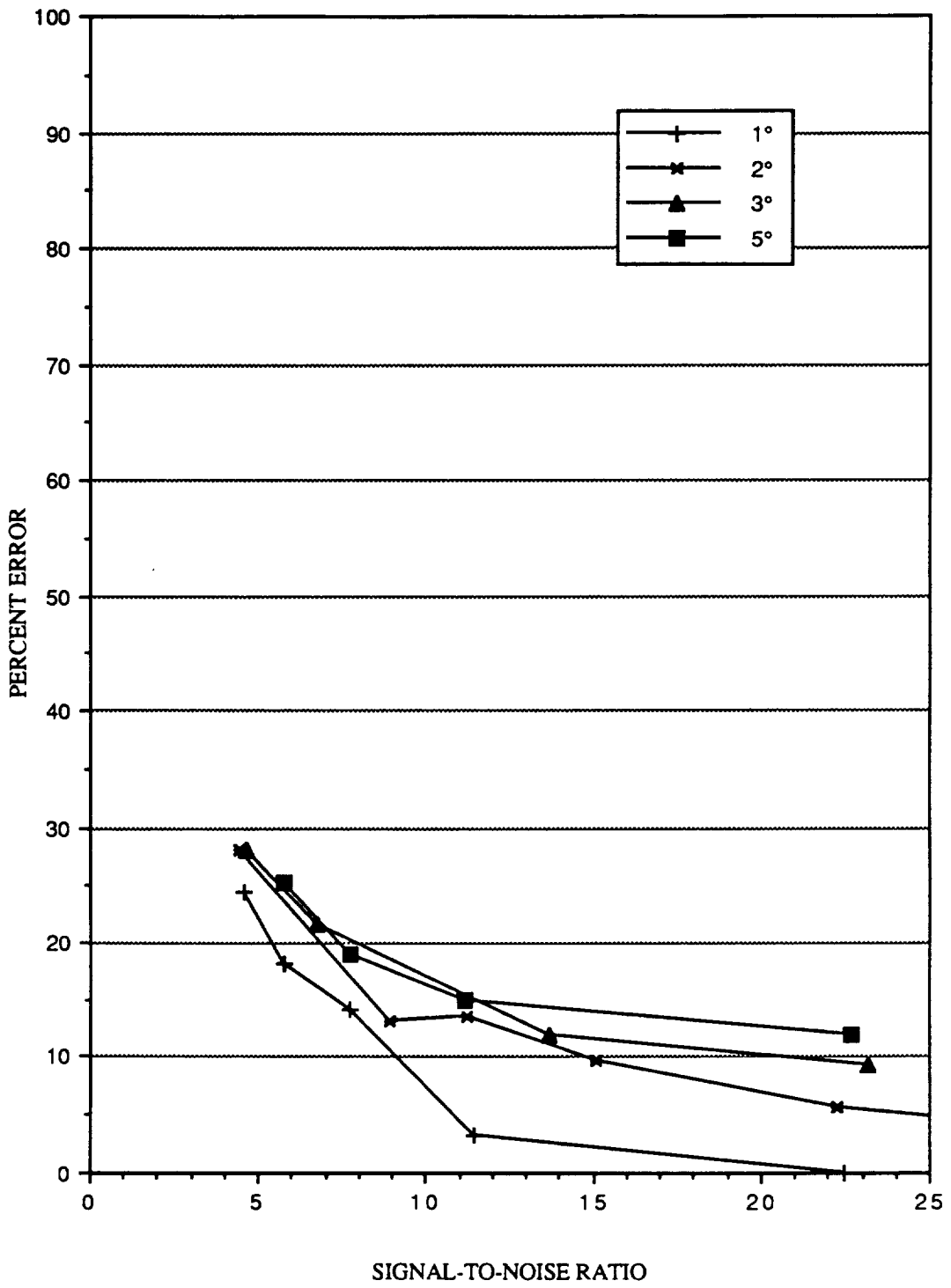


Figure 37. Plot of Error Index by MIT M2MI86 Program for 0.05 Hz sinusoidal pursuit data.

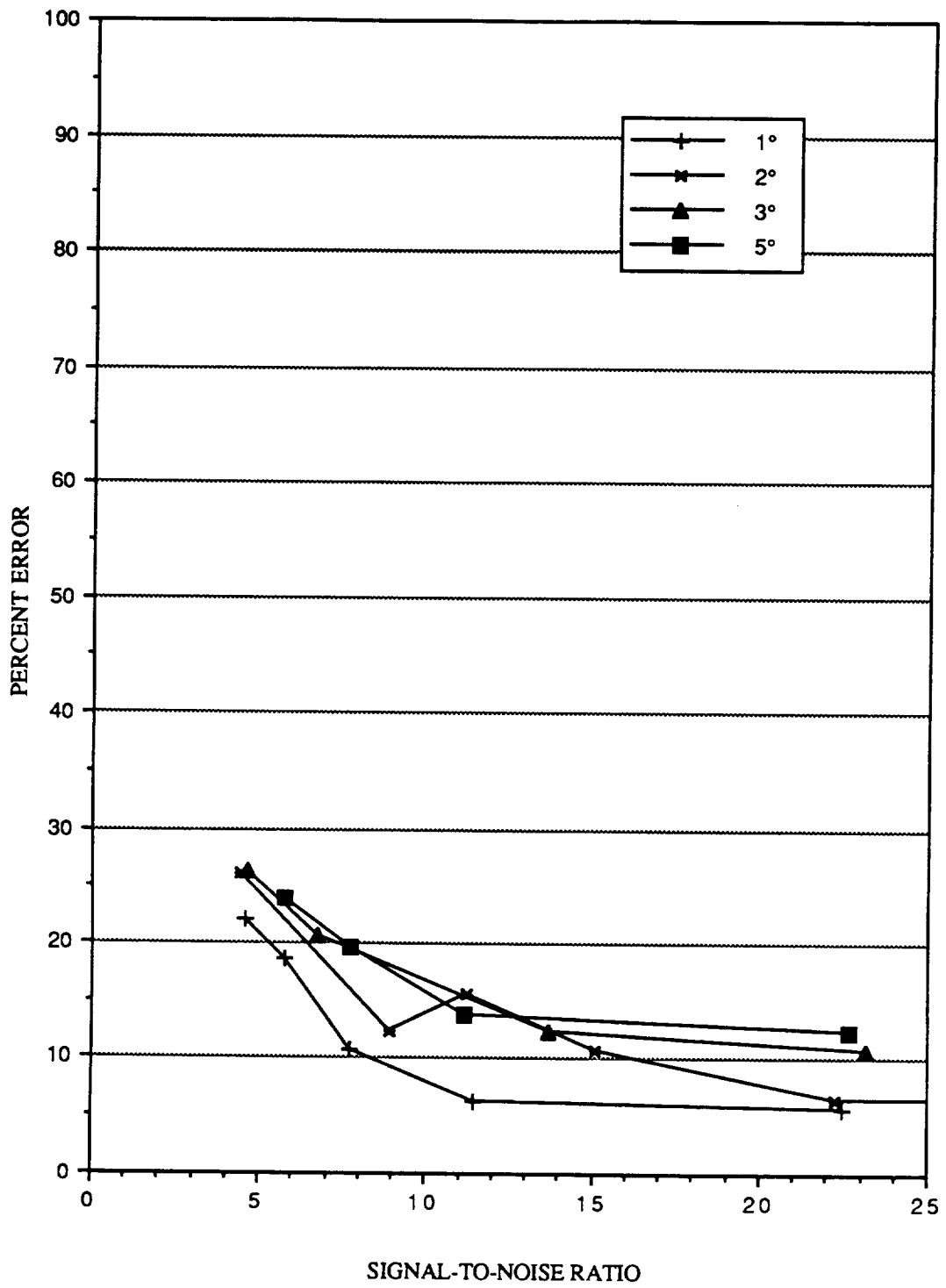


Figure 38. Plot of Error Index by MIT M2MI86 Program for 0.10 Hz sinusoidal pursuit data.

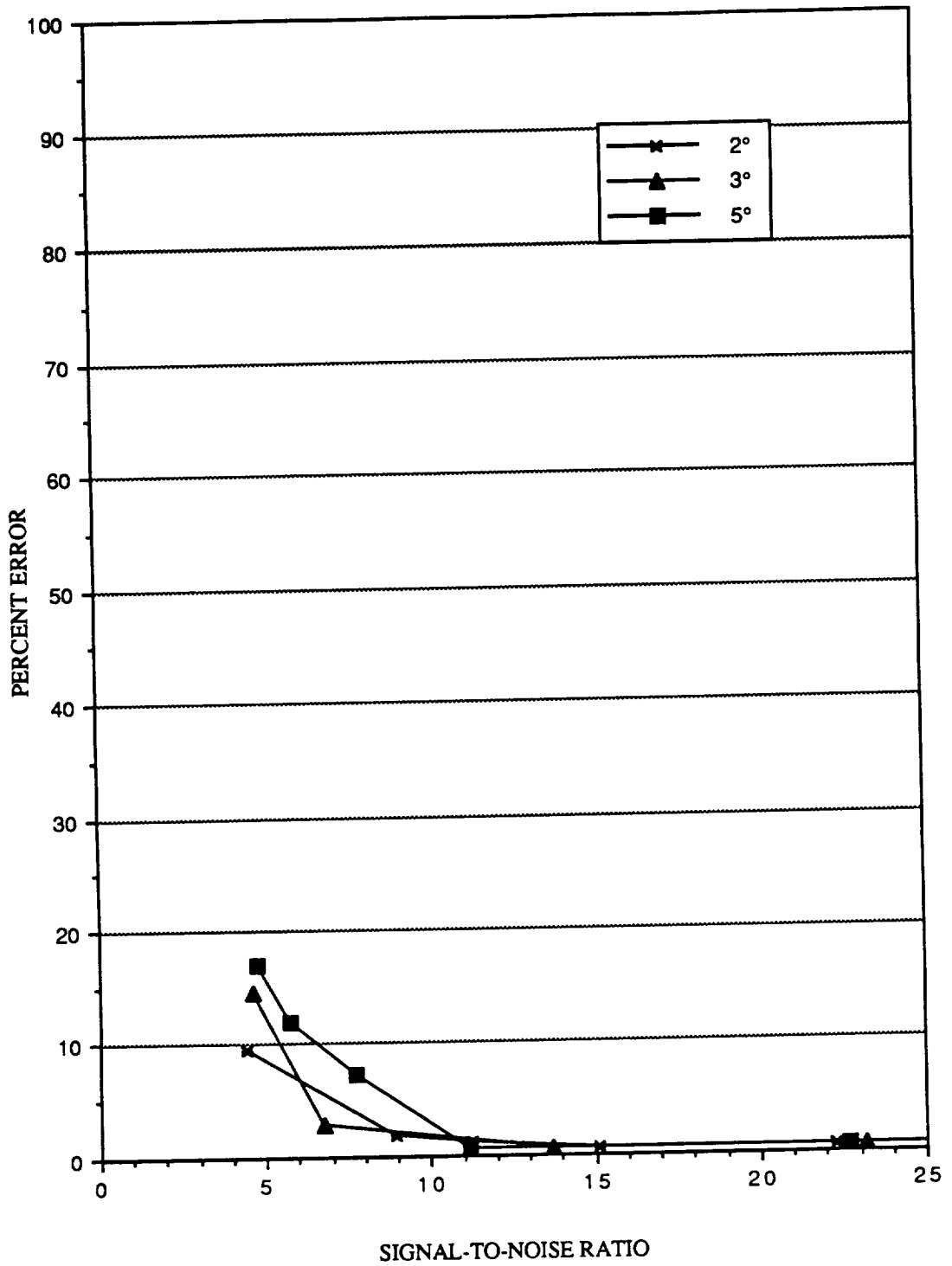


Figure 39. Plot of Error Index by MIT M2MI86 Program for 0.50 Hz sinusoidal pursuit data.

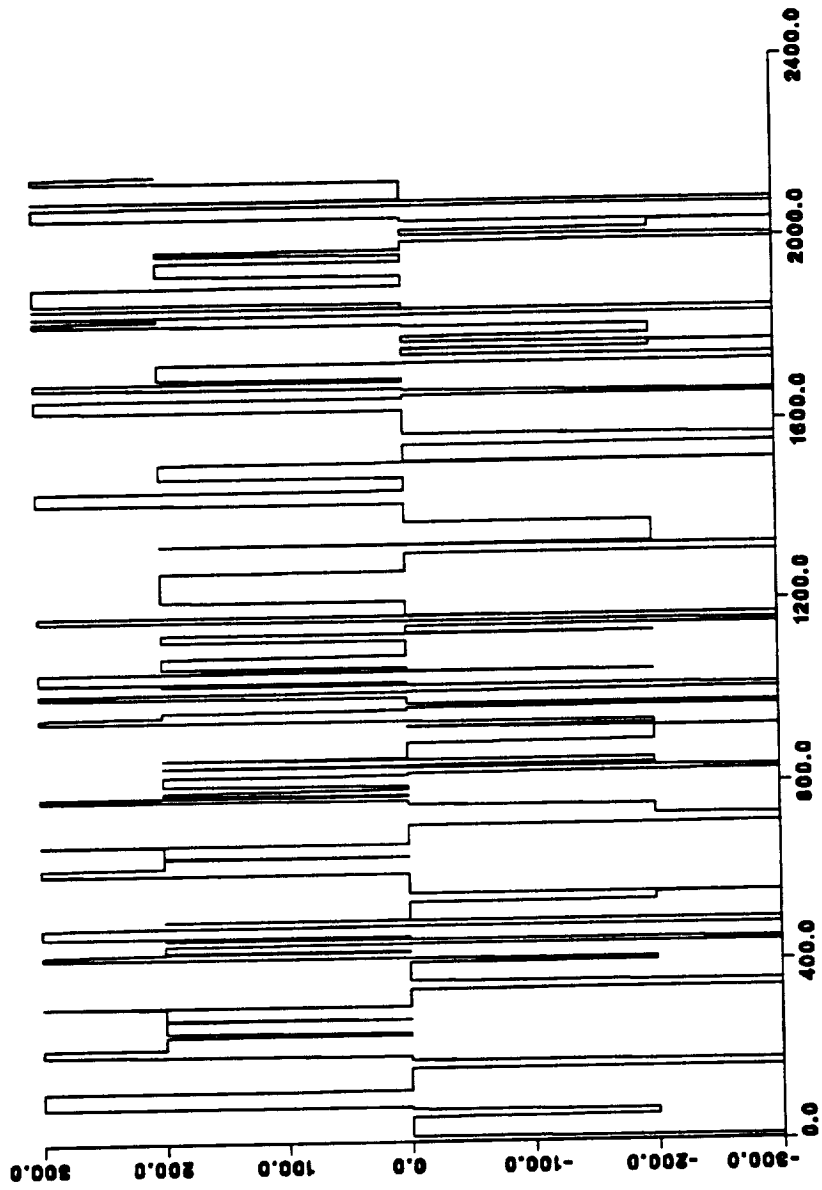


Figure 40. Plot of output for optokinetic data analysis by MIT M2MI86 Program.

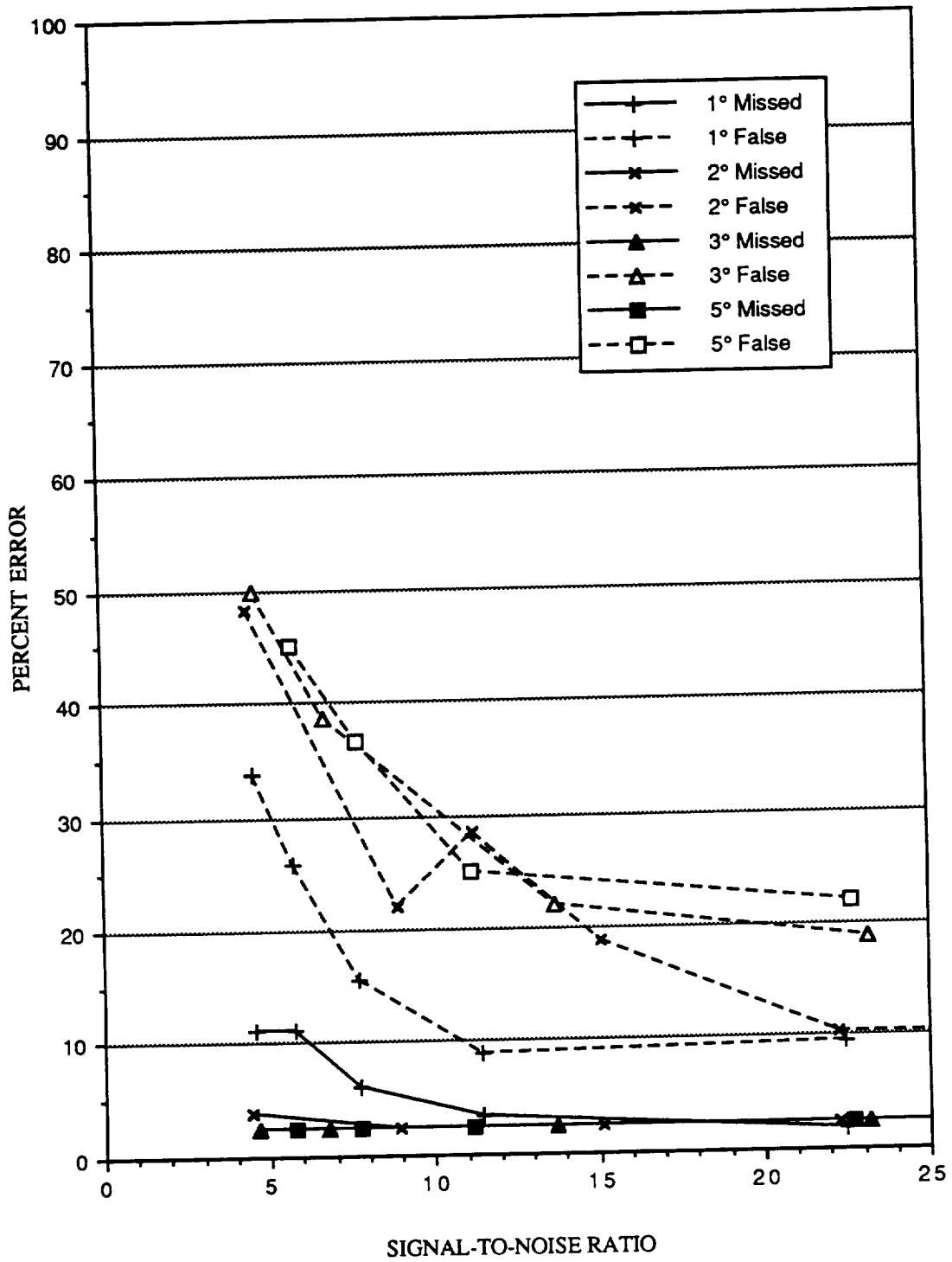


Figure 41. Plot of individual errors by MIT M2MI86 Program for 0.10 Hz sinusoidal pursuit data.

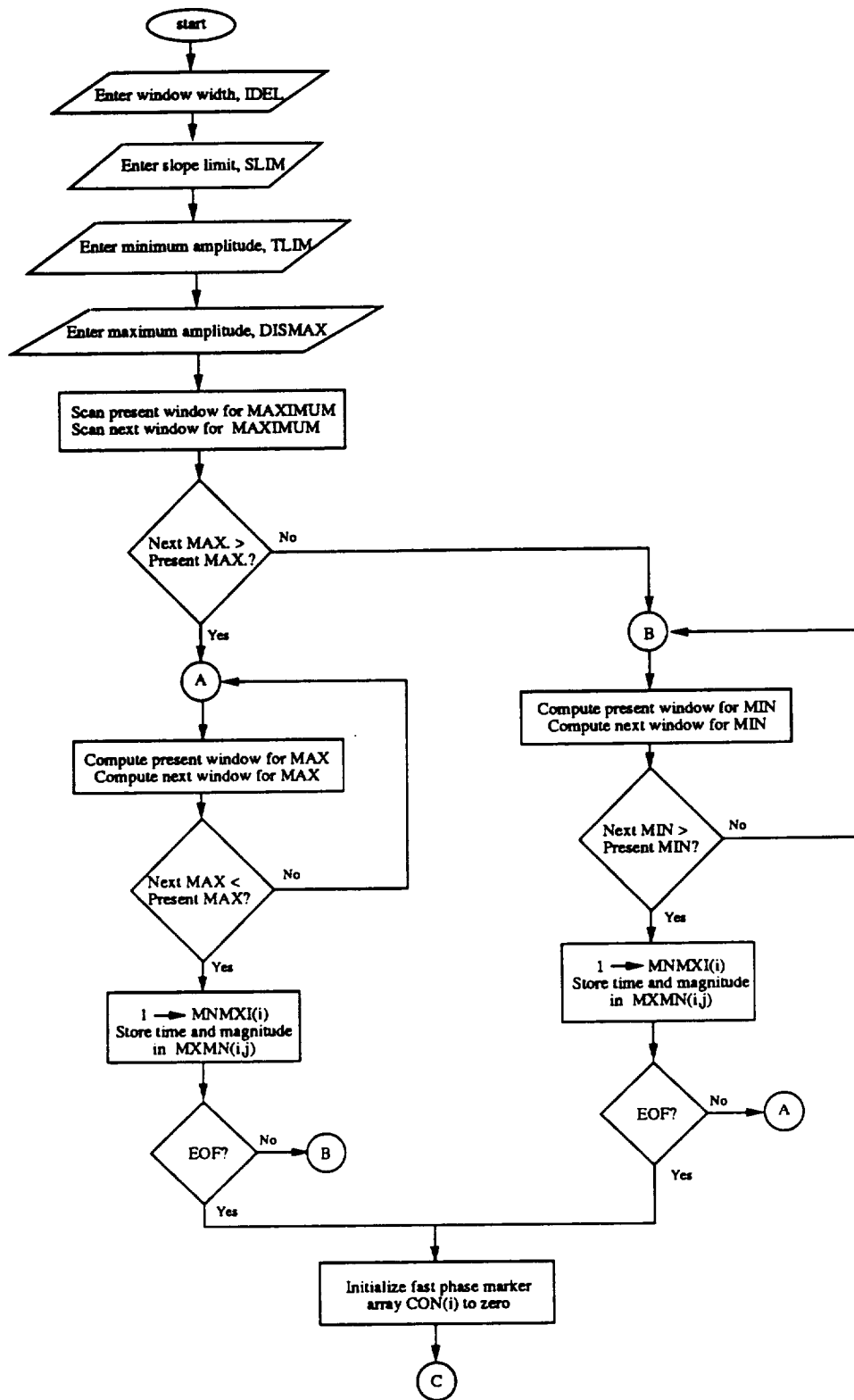


Figure 42A. Flowchart of Harvard Program.

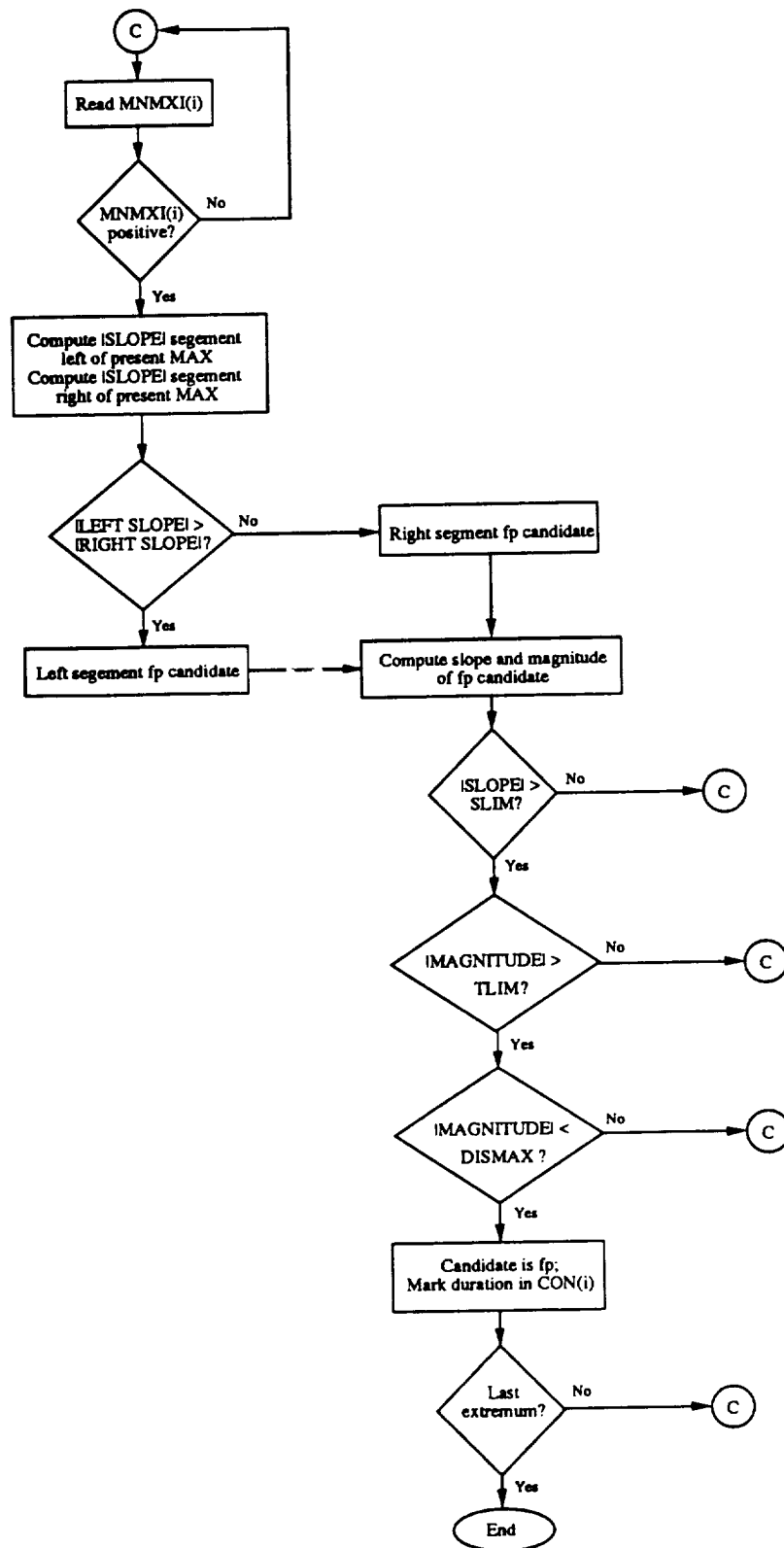


Figure 42B. Flowchart of Harvard Program (continued).

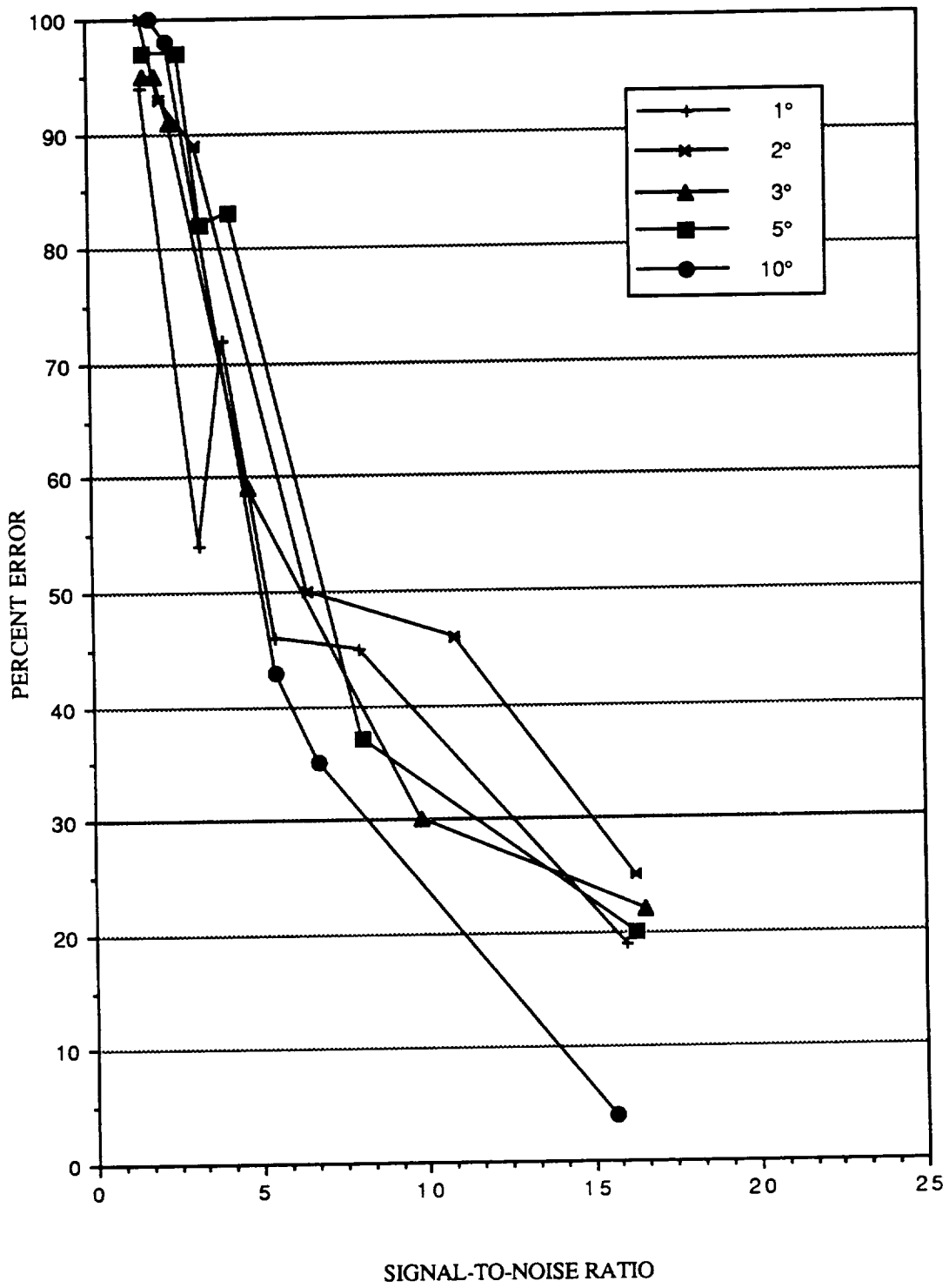


Figure 43. Plot of Error Index by Harvard Program for midposition gaze data.

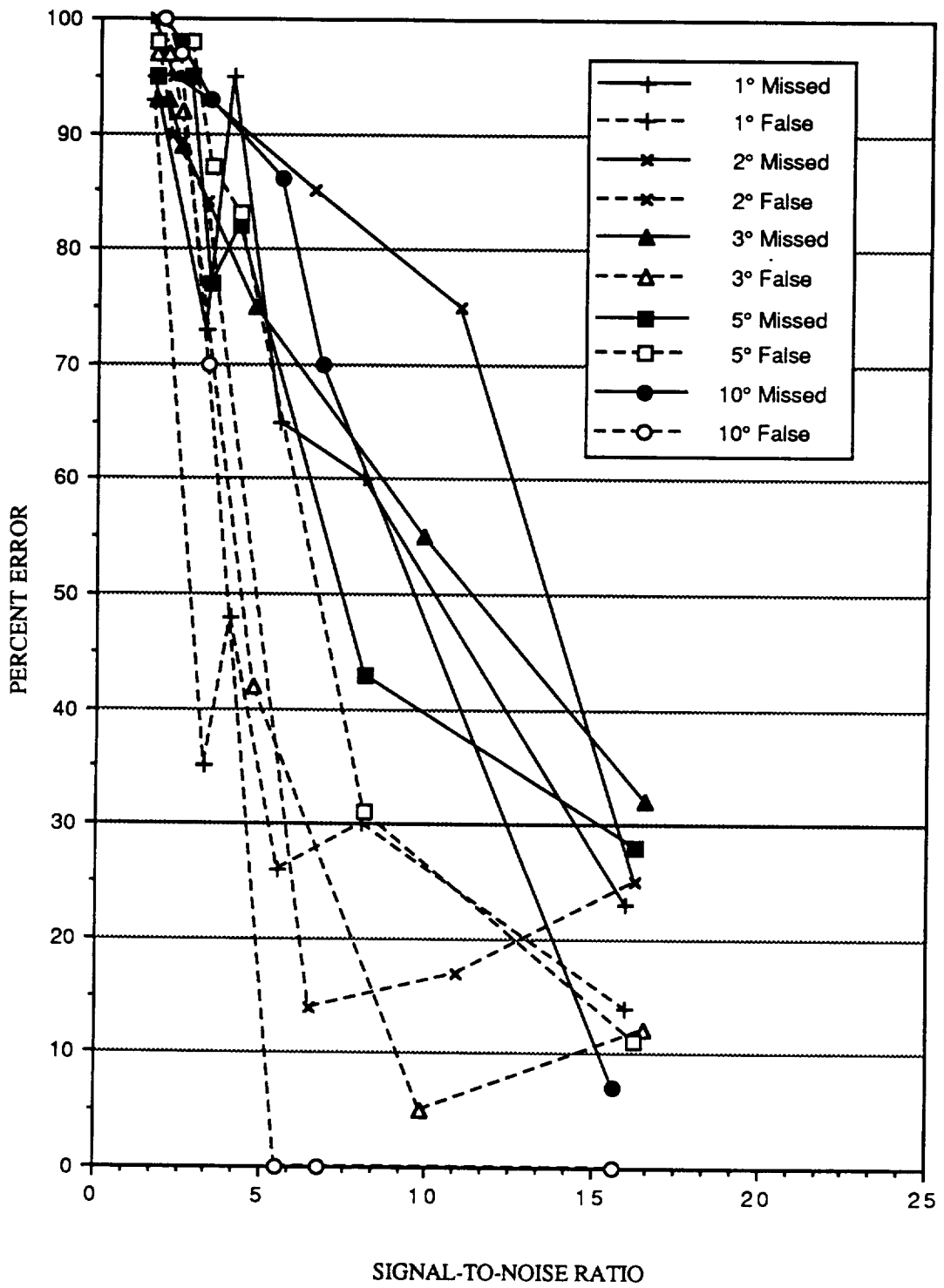


Figure 44. Plot of individual errors by Harvard Program for midposition gaze data.

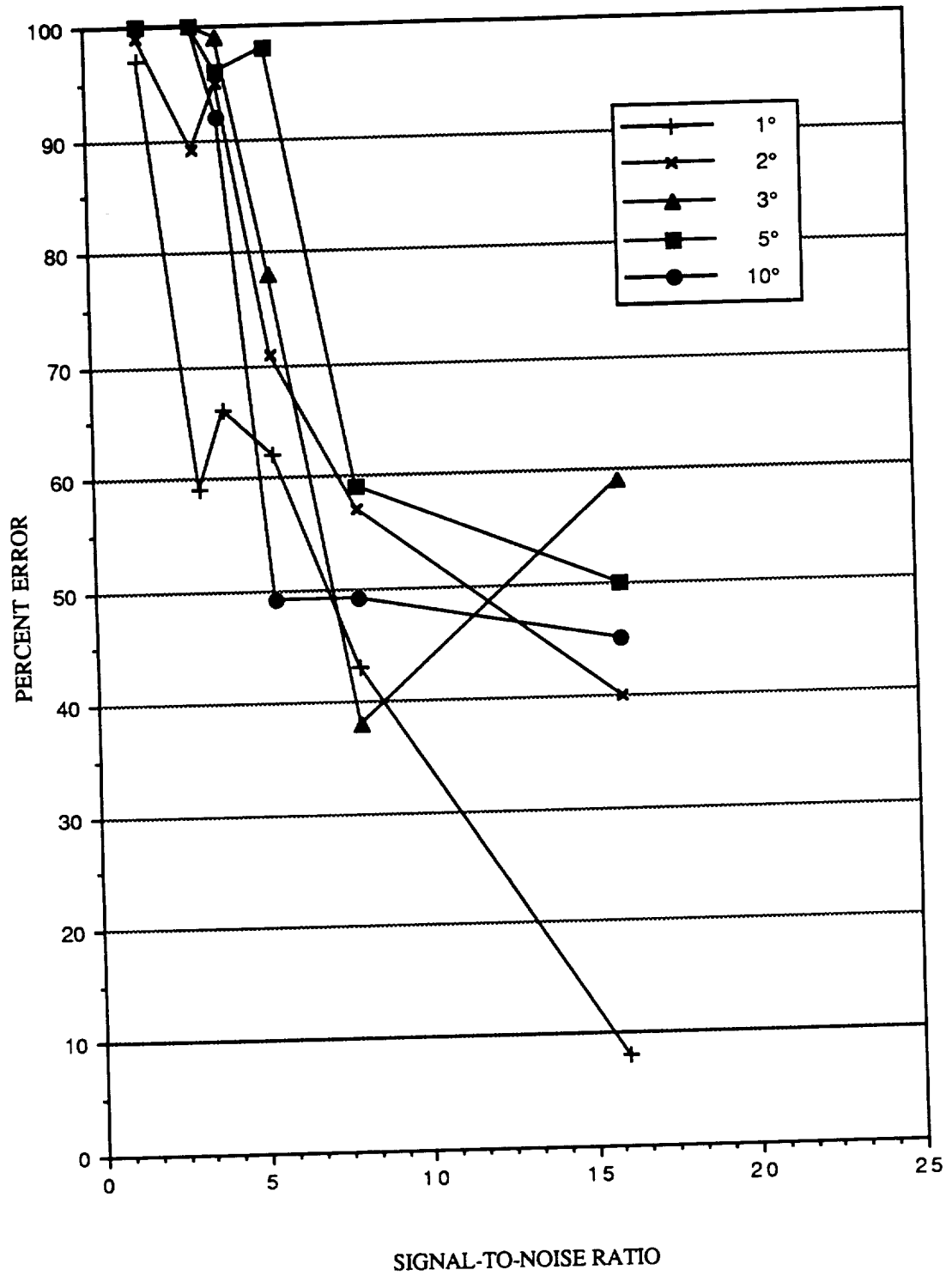


Figure 45. Plot of Error Index by Harvard Program for 0.05 Hz sinusoidal pursuit data.

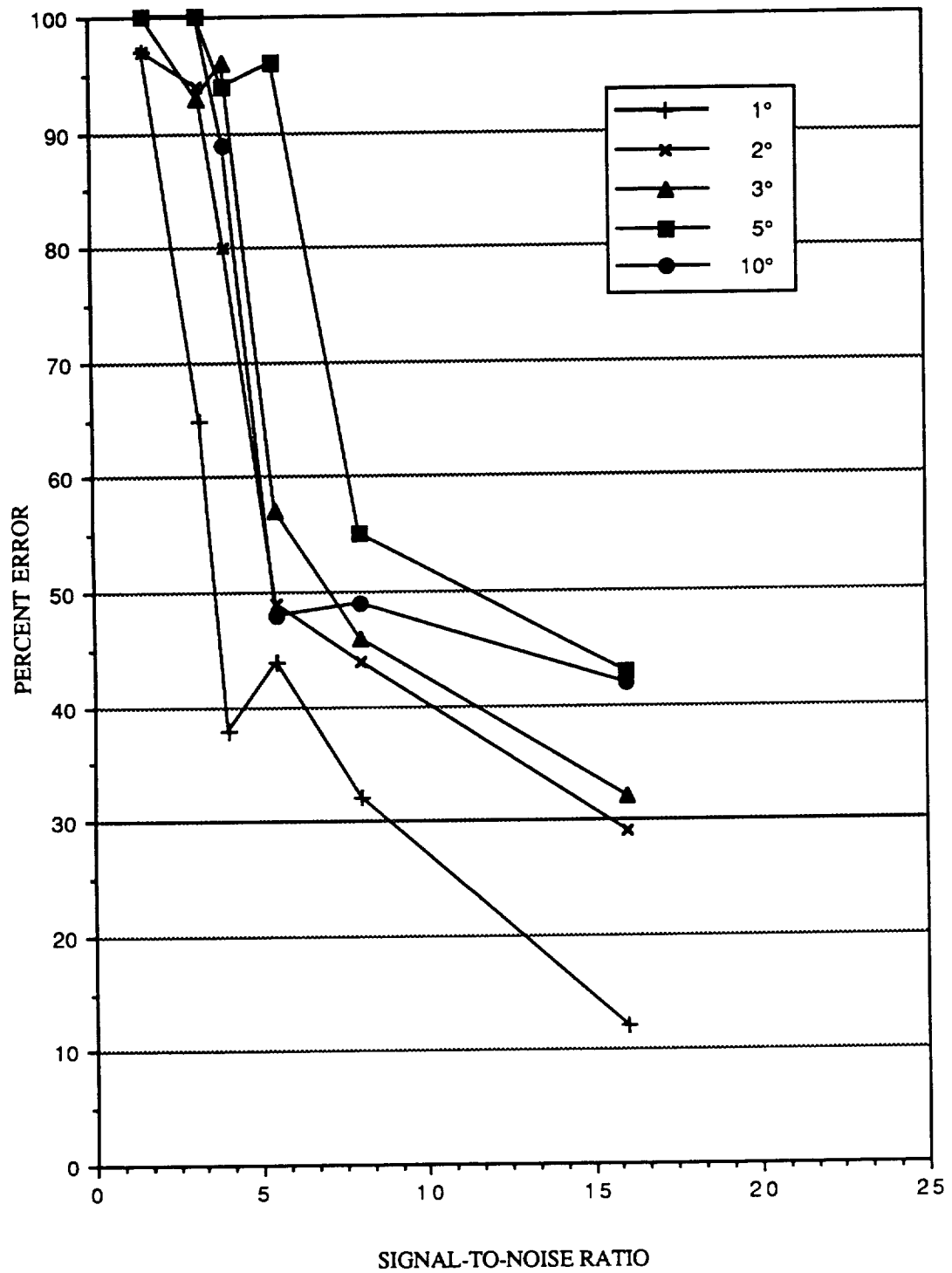


Figure 46. Plot of Error Index by Harvard Program for 0.10 Hz sinusoidal pursuit data.

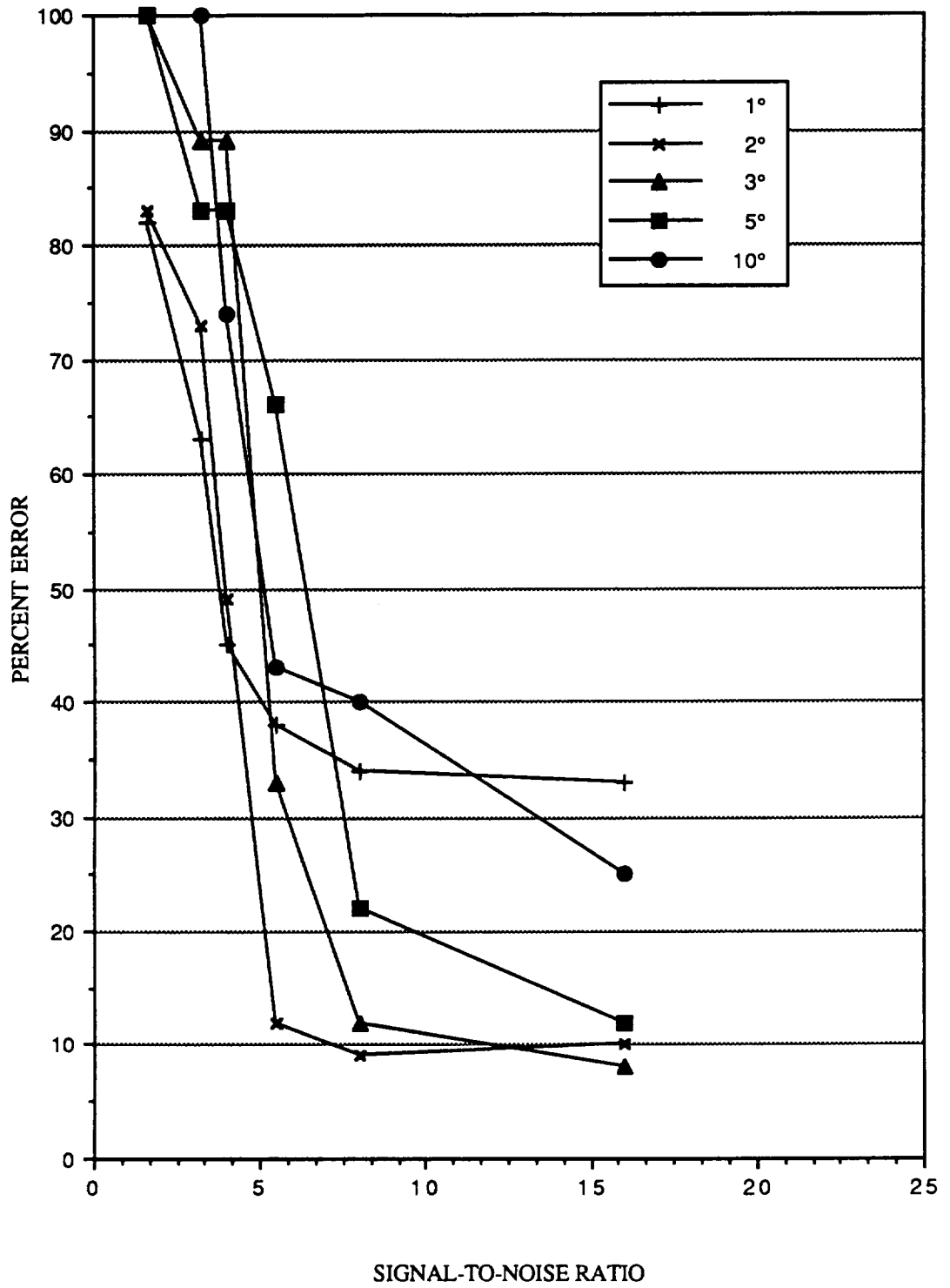


Figure 47. Plot of Error Index by Harvard Program for 0.50 Hz sinusoidal pursuit data.

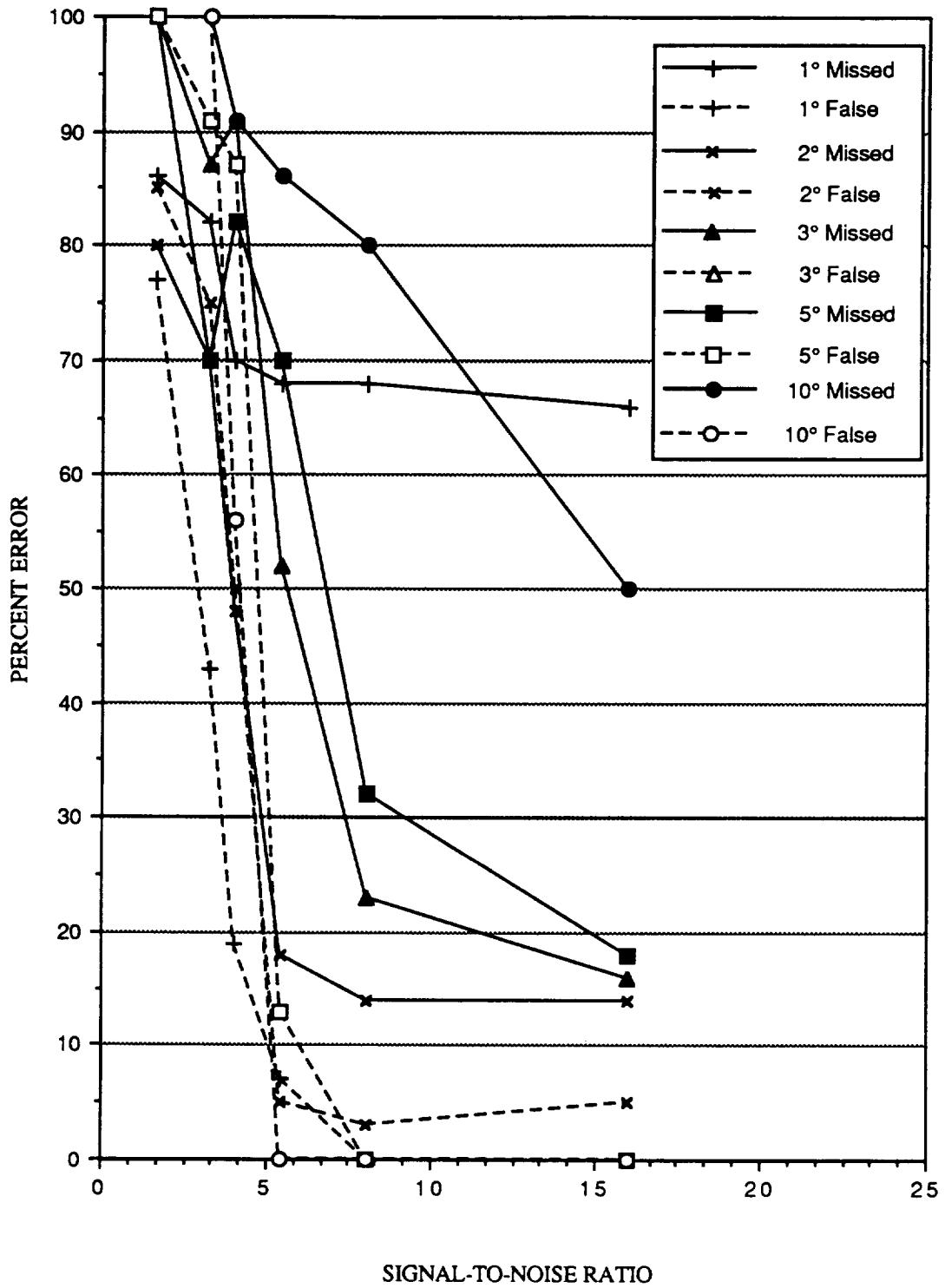


Figure 48. Plot of individual errors by Harvard Program for 0.50 Hz sinusoidal pursuit data.

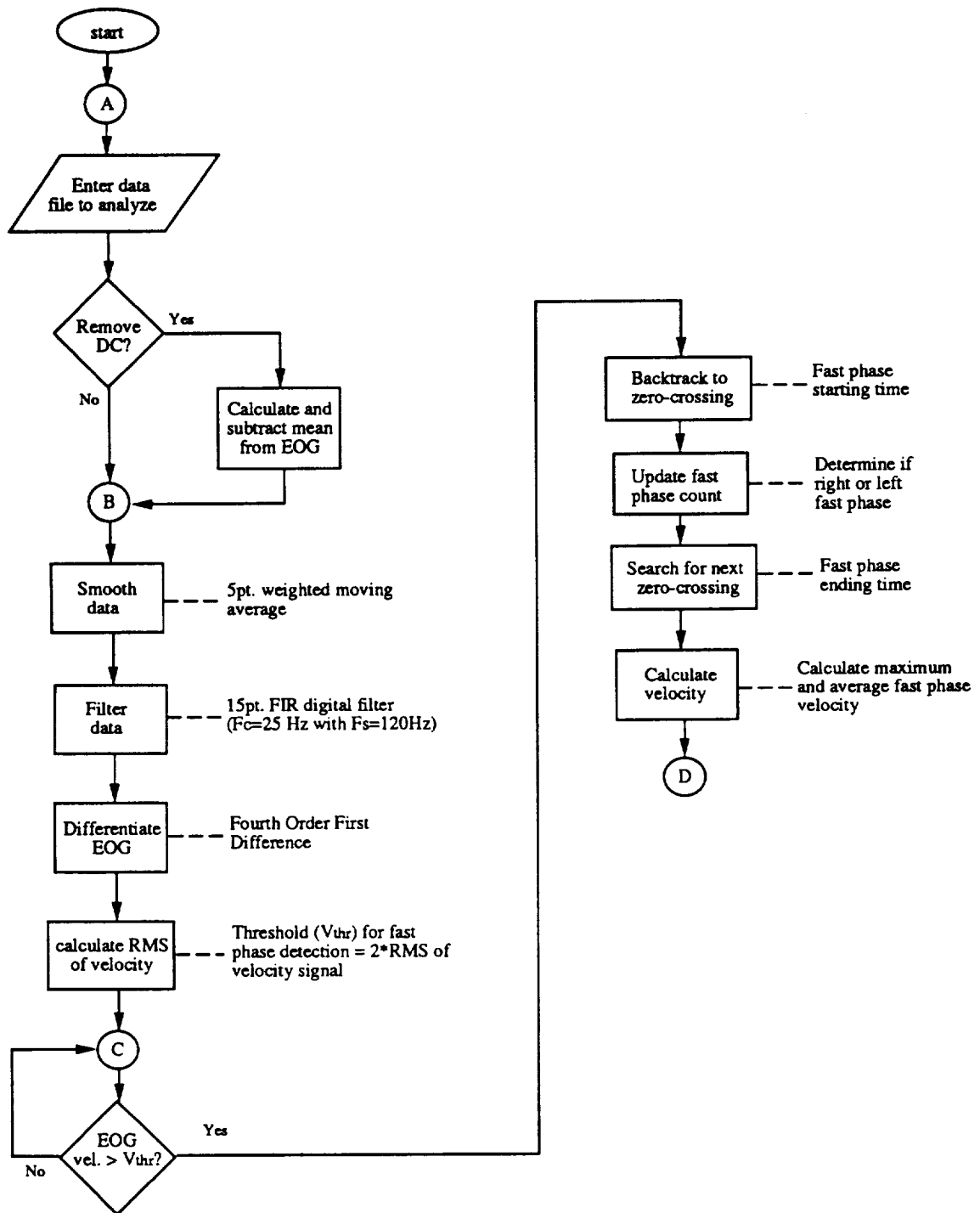


Figure 49A. Flowchart of FPID Program.

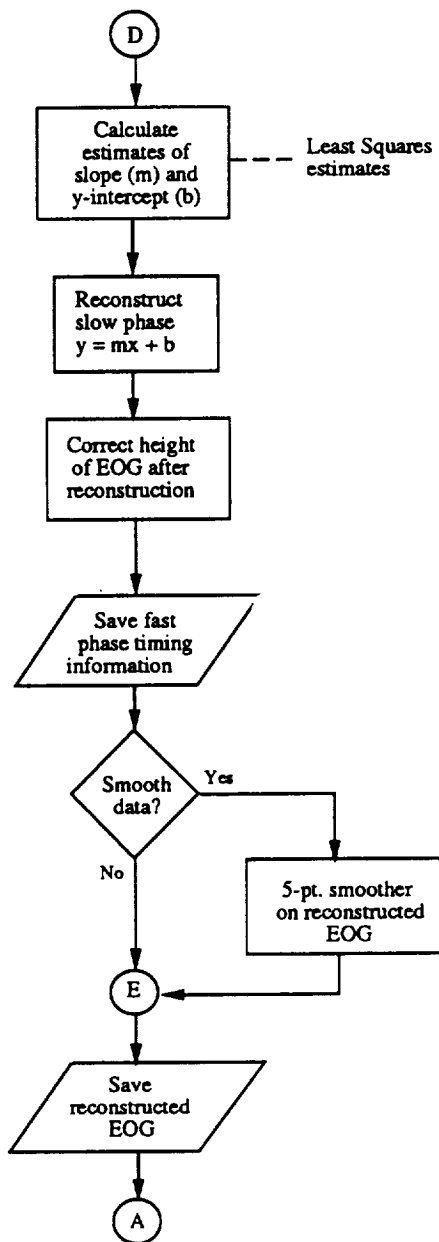


Figure 49B. Flowchart of FPID Program (continued).

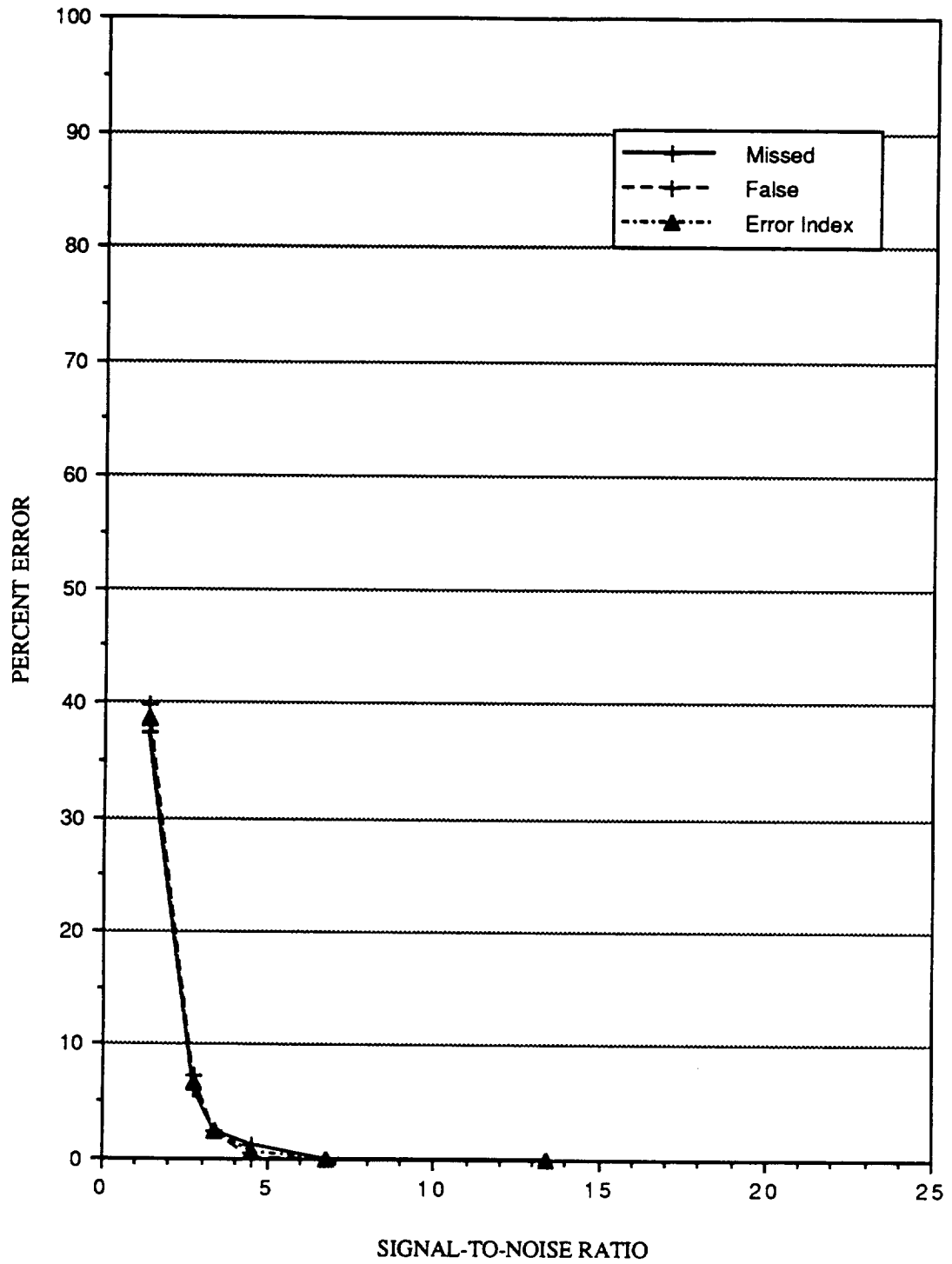


Figure 50. Plot of individual errors and Error Index by FPID Program for midposition gaze data with one (1) degree of saccadic jump.

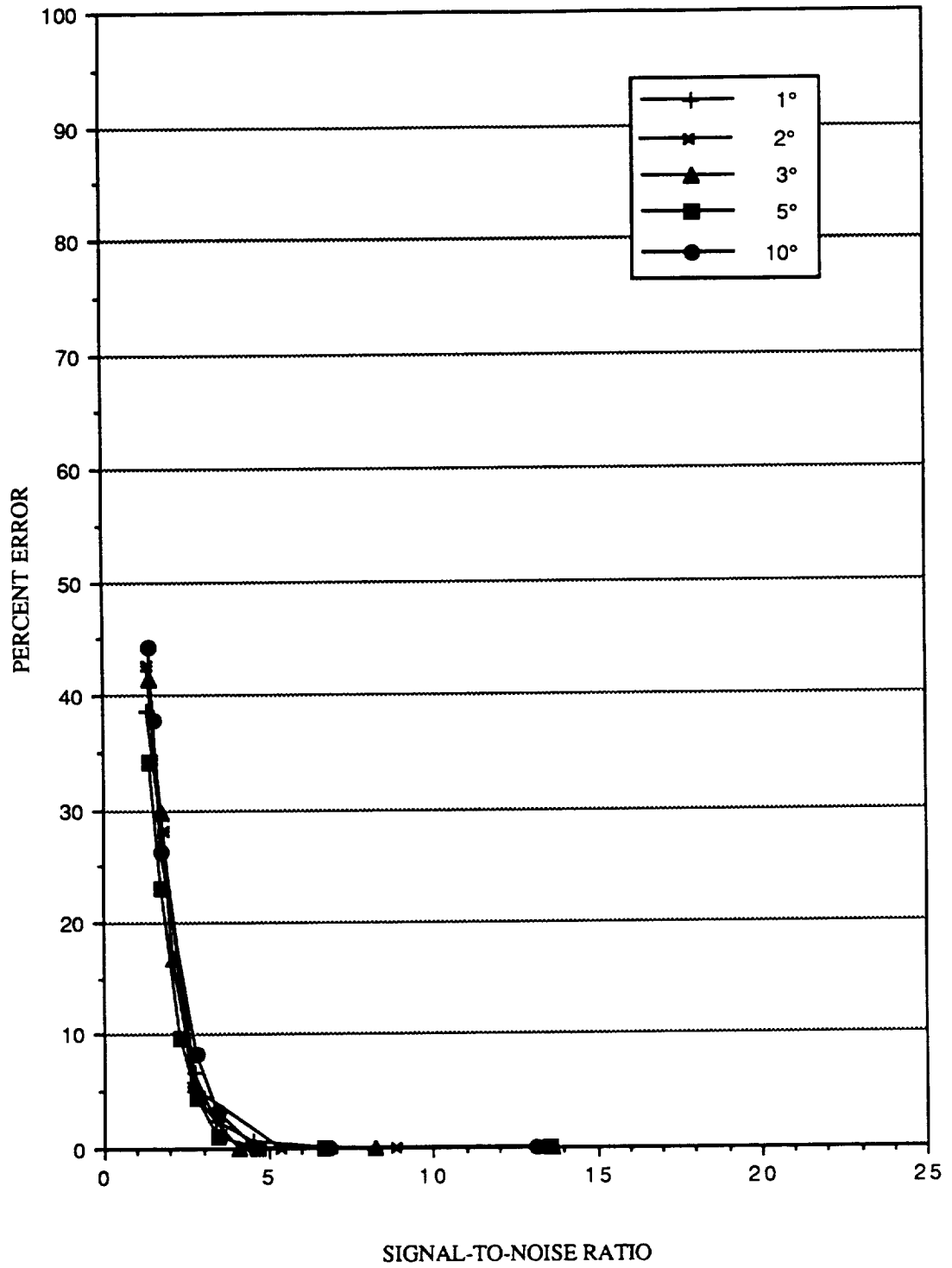


Figure 51. Plot of Error Index by FPI Program for midposition gaze data.

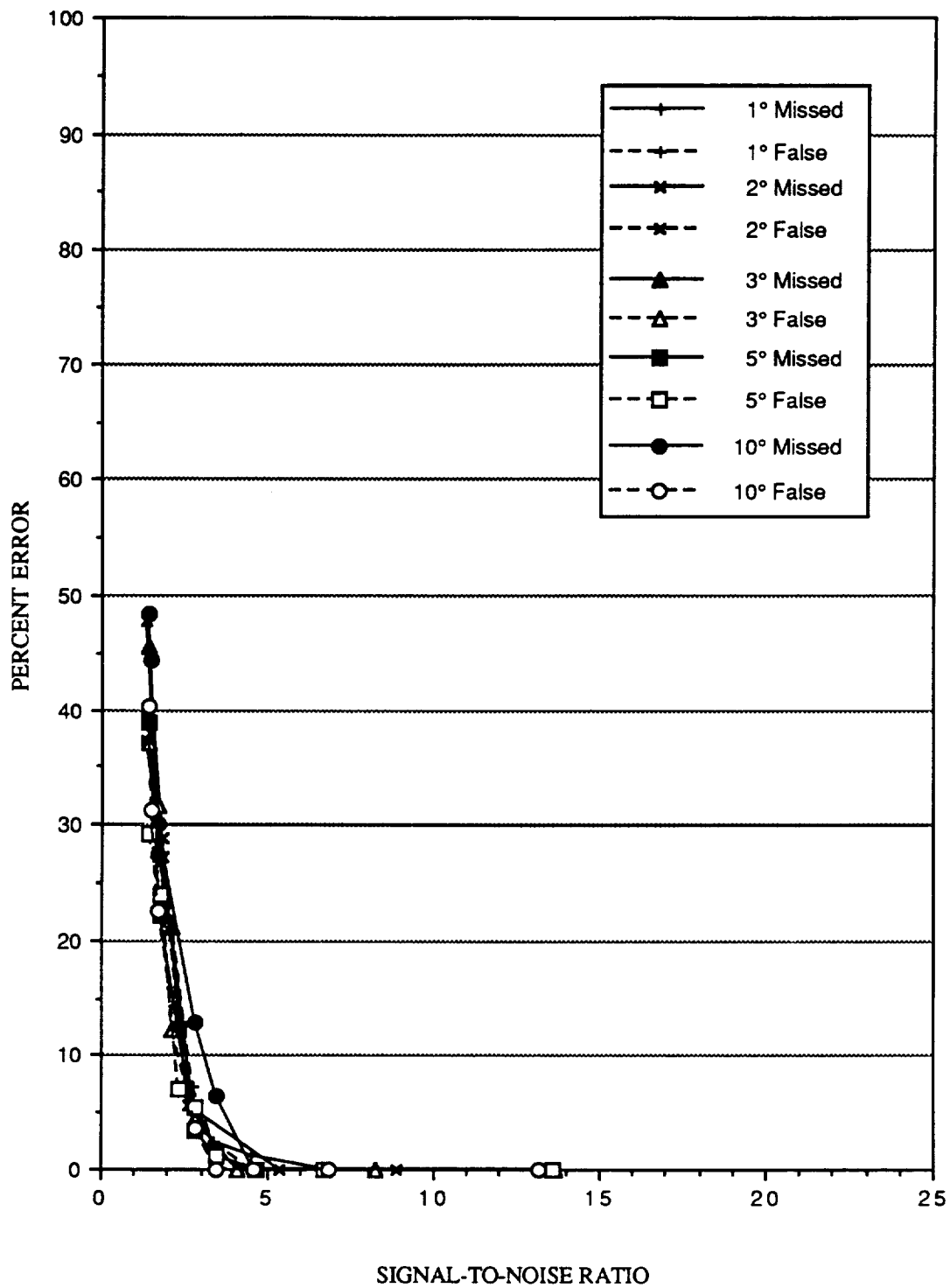


Figure 52. Plot of individual errors by FPID Program for midposition gaze data.

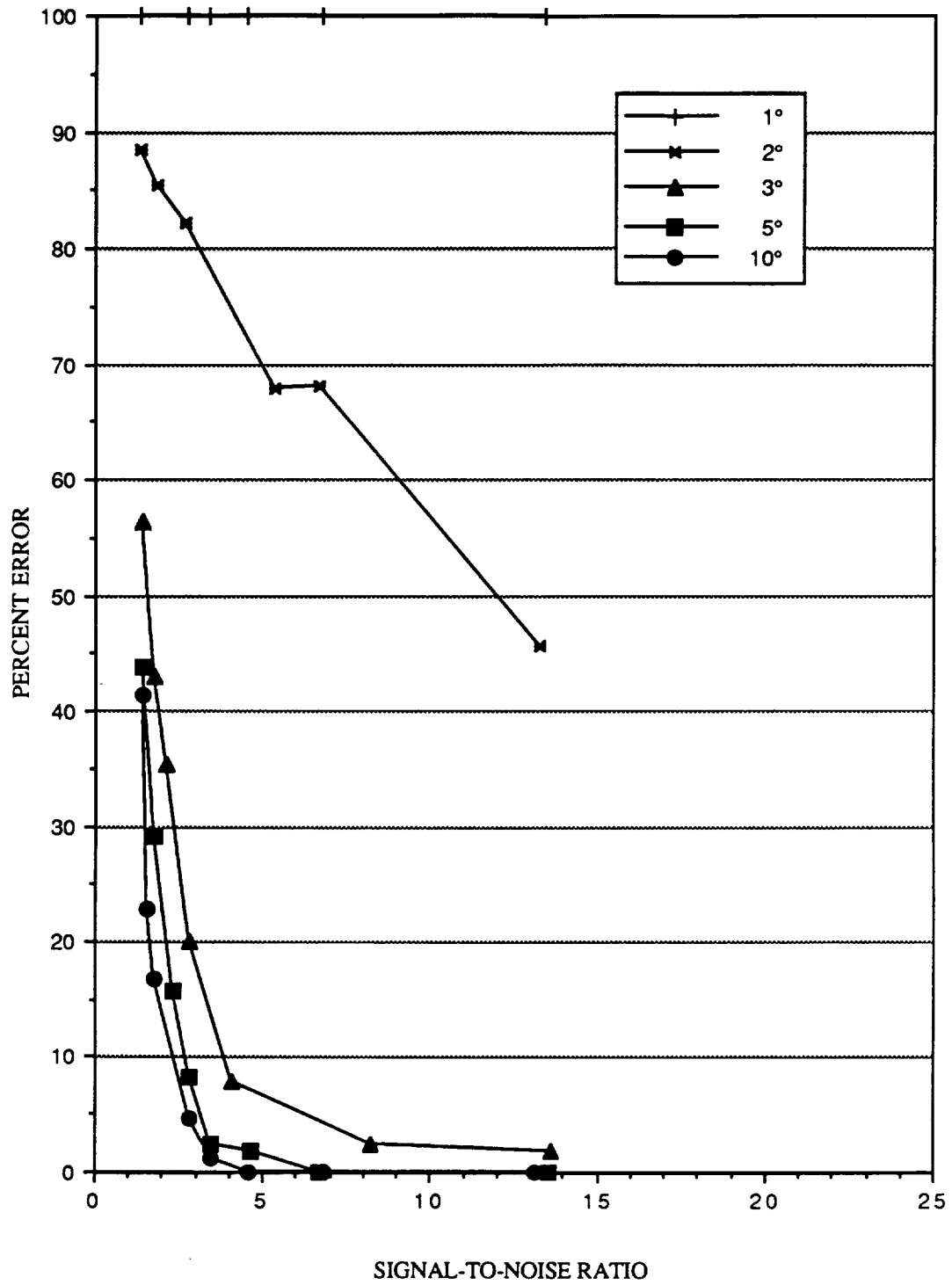


Figure 53. Plot of Error Index by FPID Program for 0.50 Hz sinusoidal pursuit data.

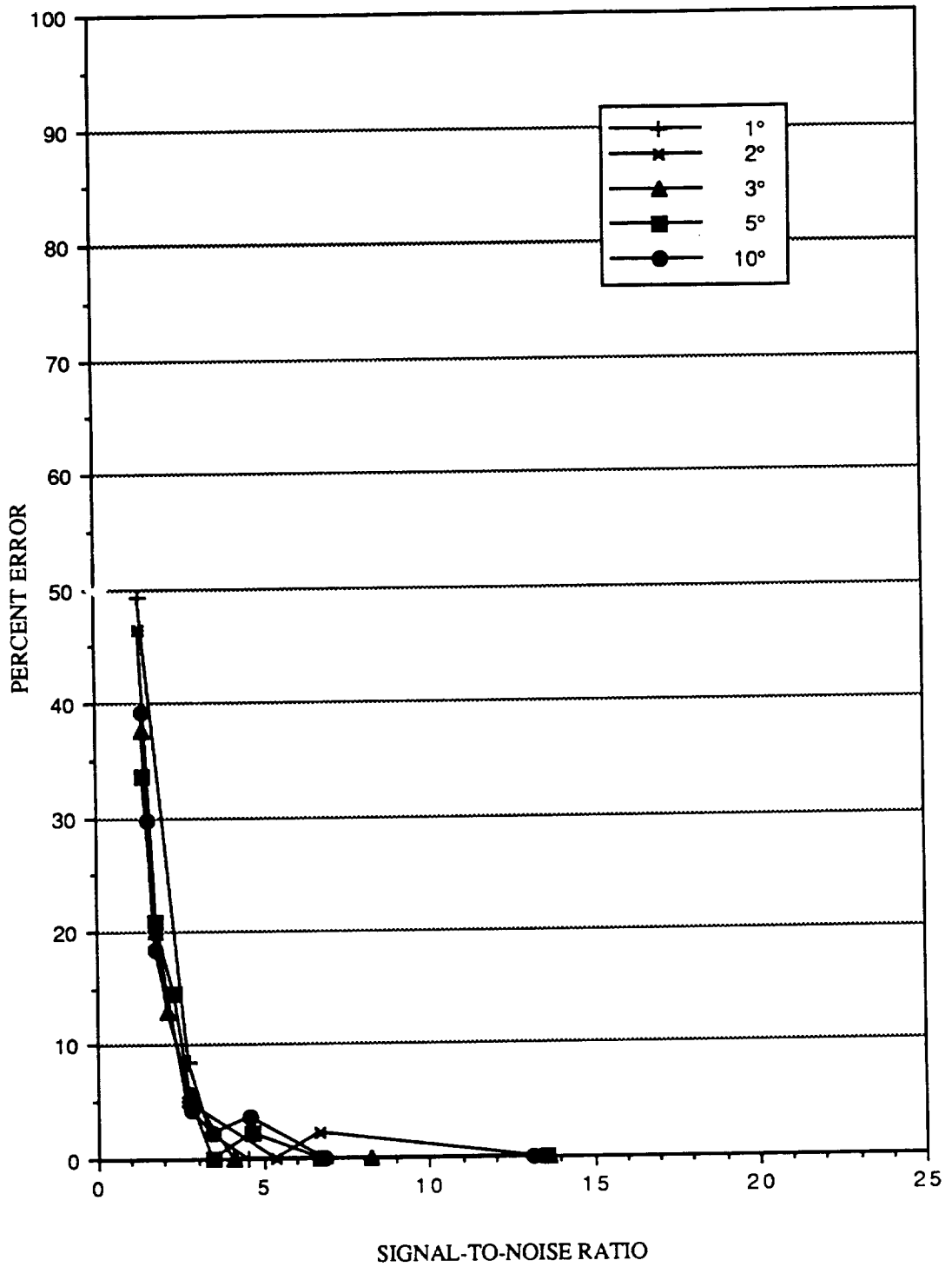


Figure 54. Plot of Error Index by FPID Program for 0.05 Hz sinusoidal pursuit data.

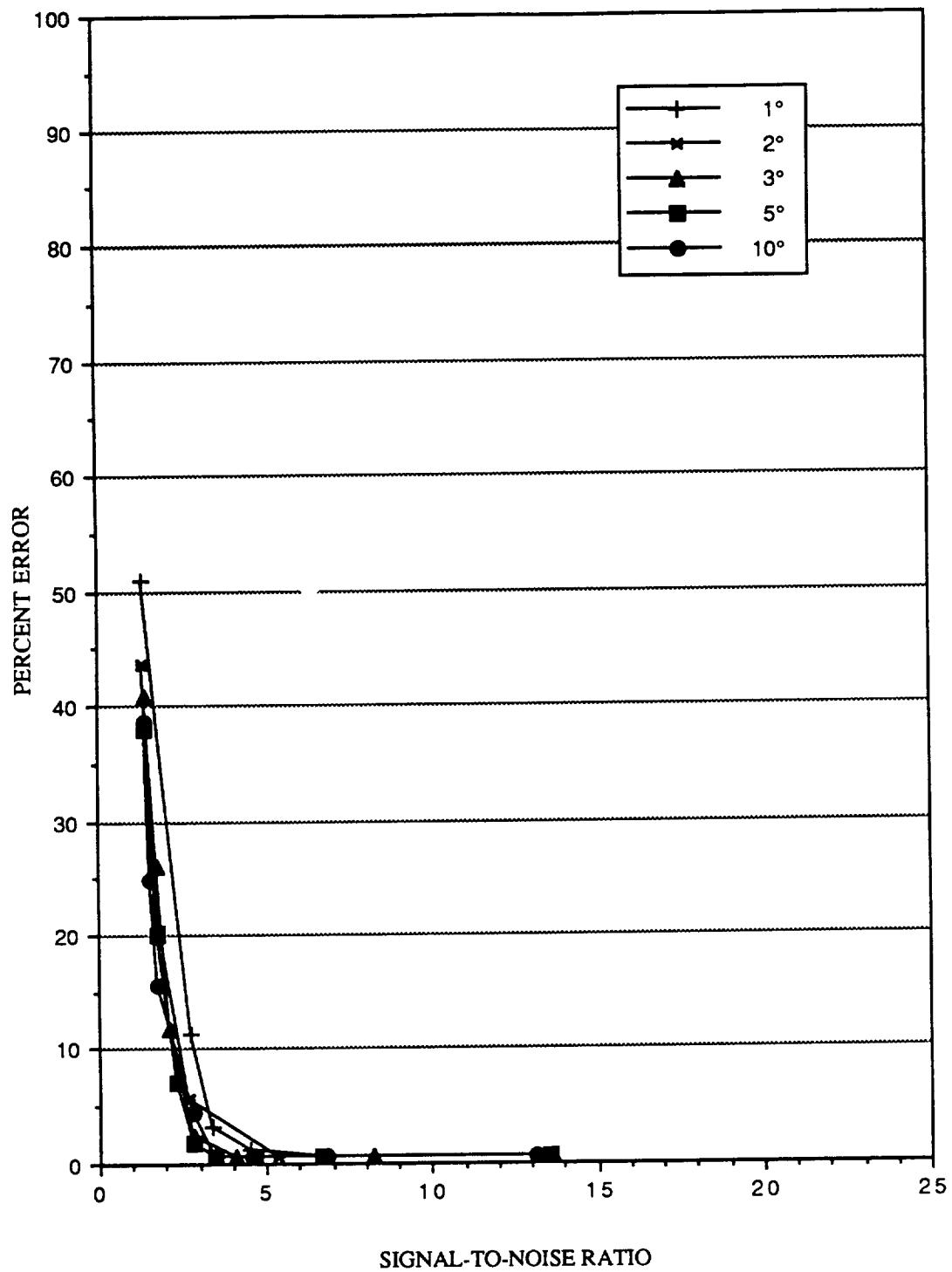


Figure 55. Plot of Error Index by FPID Program for 0.10 Hz sinusoidal pursuit data.

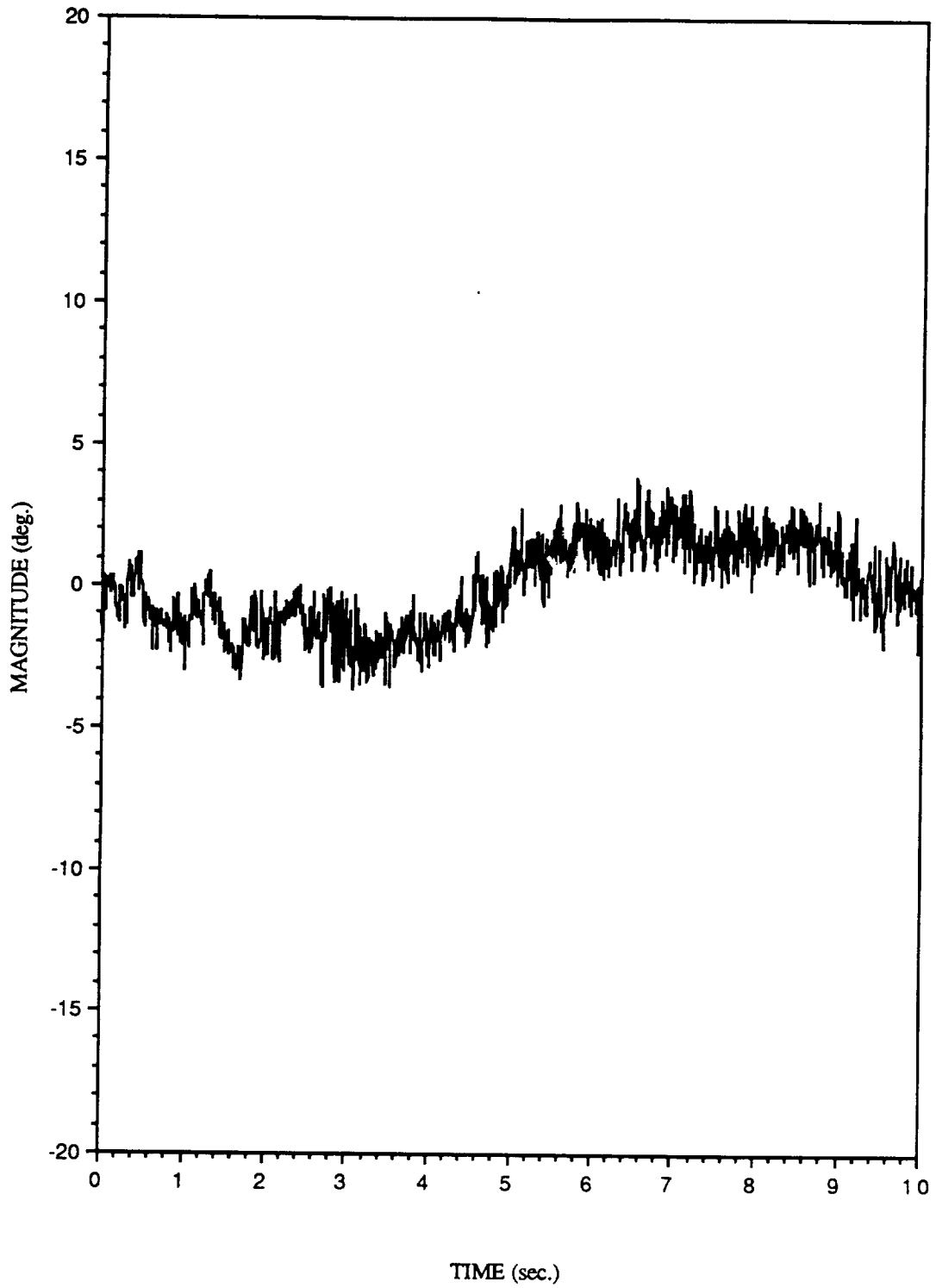


Figure 56. Plot of noise on the human sinusoidal pursuit data set.

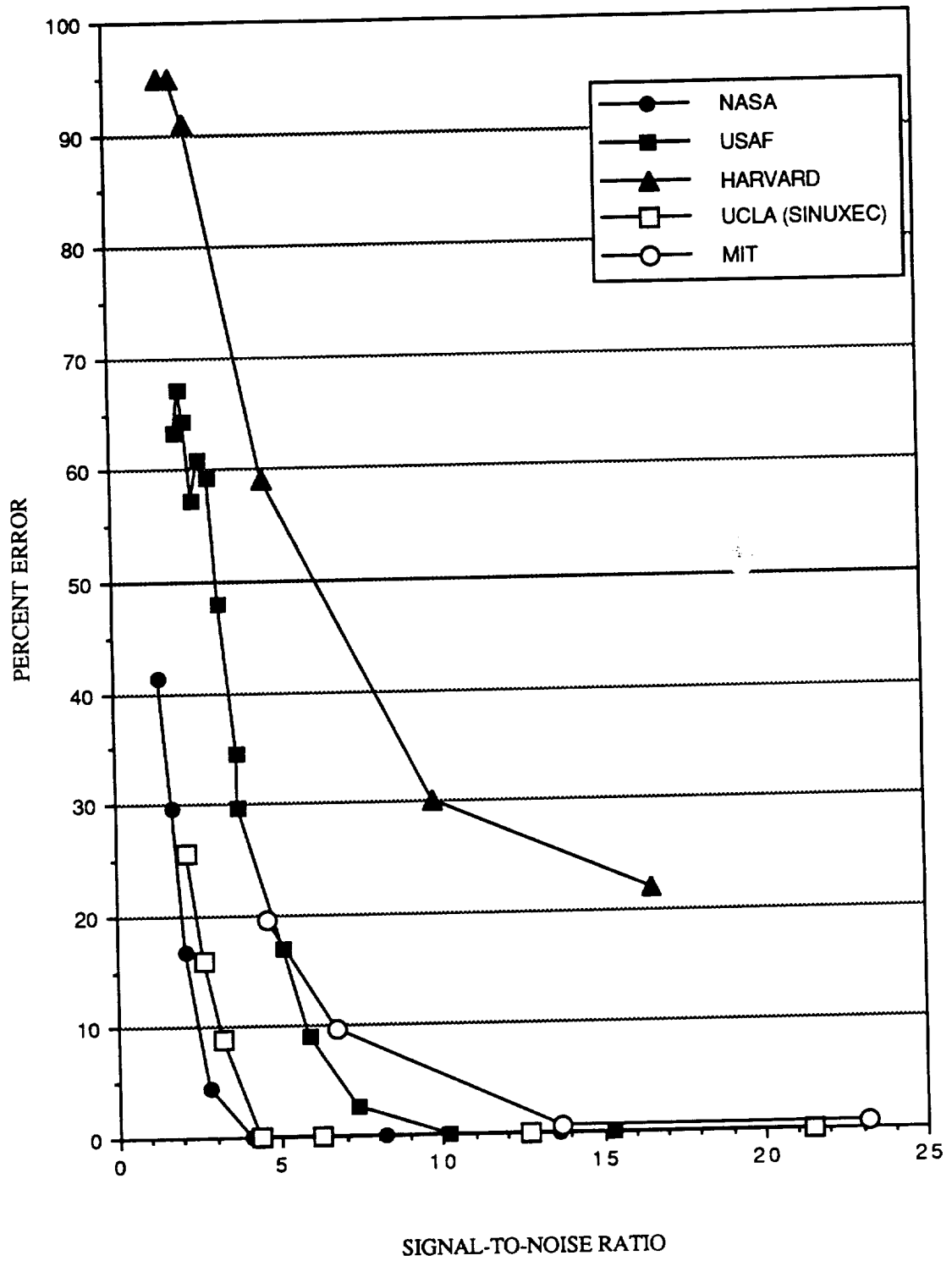


Figure 57. Plot of Error Index by five (5) of the programs (USAF/SAM, UCLA SINUXEC, MIT M2MI86, Harvard and NASA FPID) for midposition gaze data with three (3) degrees of saccadic jump.

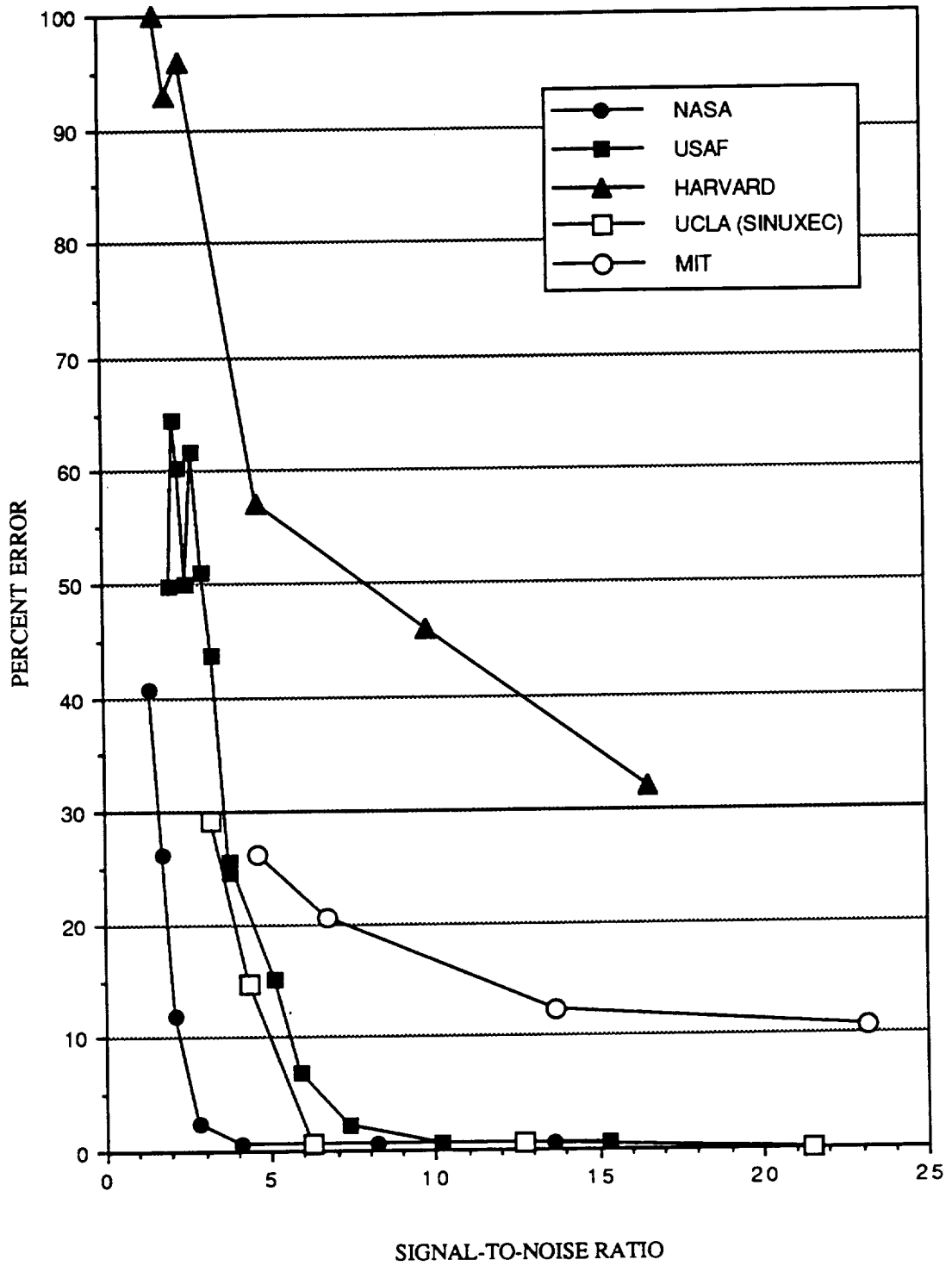


Figure 58. Plot of Error Index by five (5) of the programs (USAF/SAM, UCLA SINUXEC, MIT M2MI86, Harvard and NASA FPID) for 0.10 Hz sinusoidal pursuit data with three (3) degrees of saccadic jump.

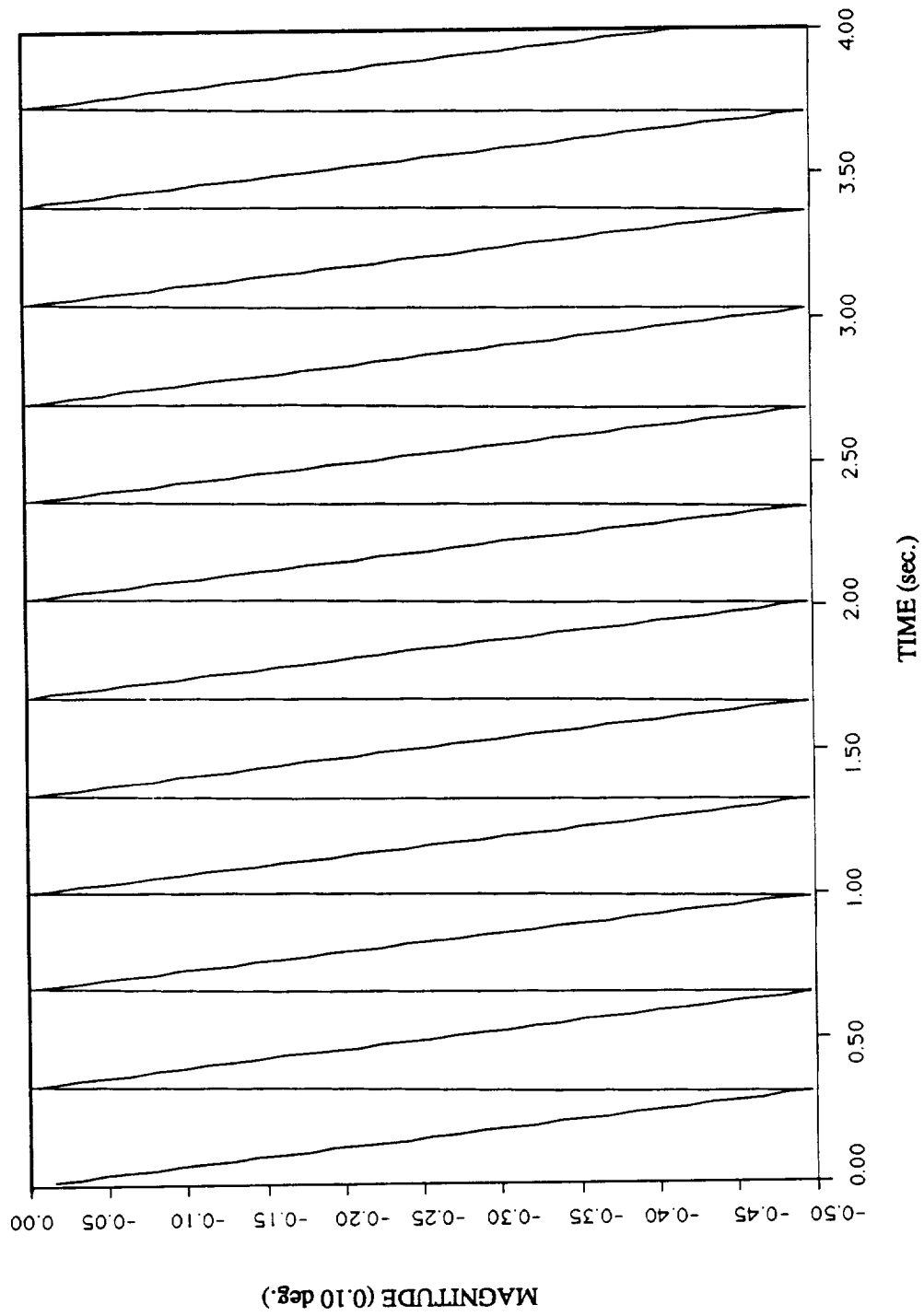


Figure A1. Plot of simulated midposition gaze data with five (5) degrees of saccadic jump without noise.

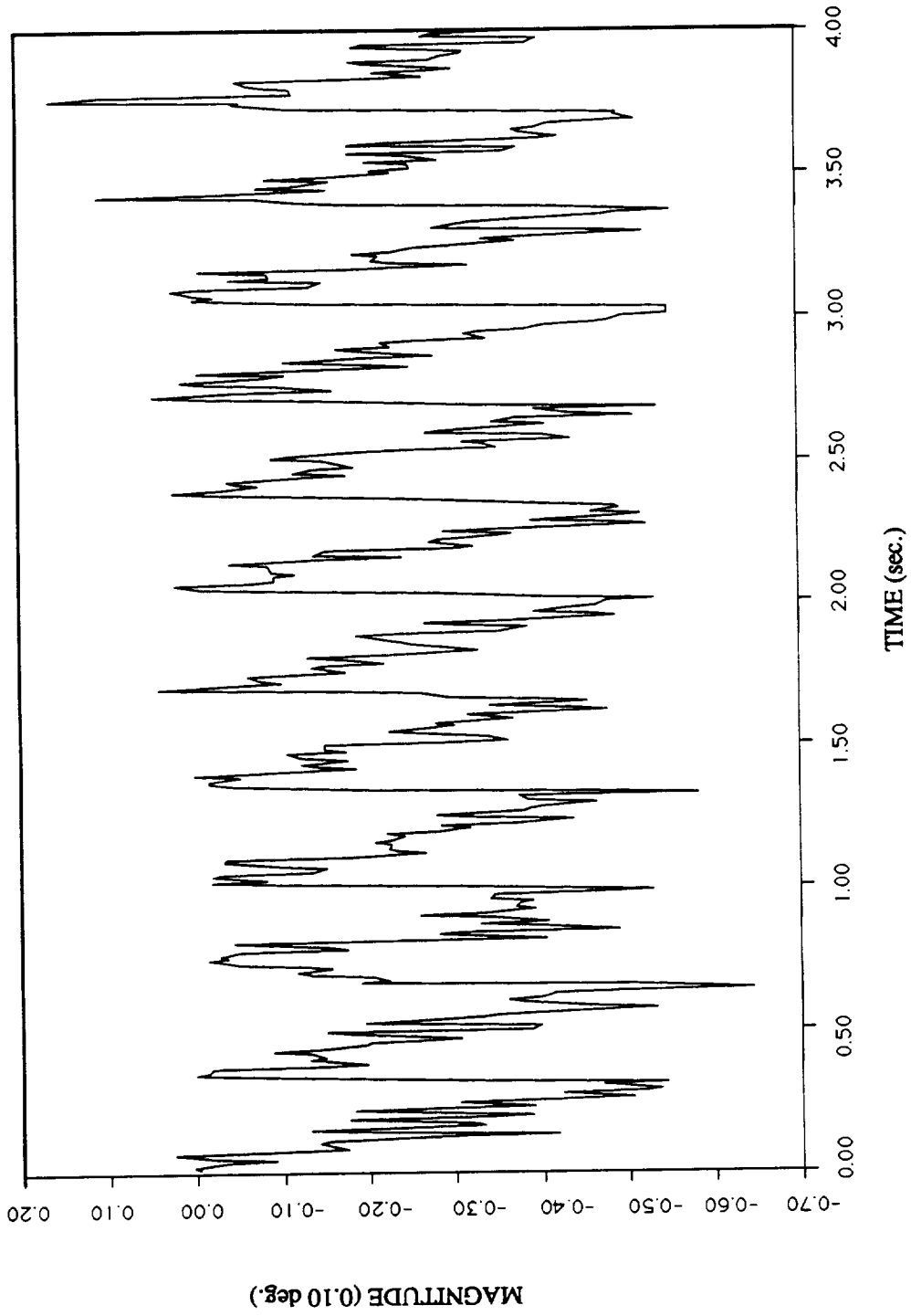


Figure A2. Plot of simulated midposition gaze data with five (5) degrees of saccadic jump with 0.5 degree (maximum amplitude) noise.

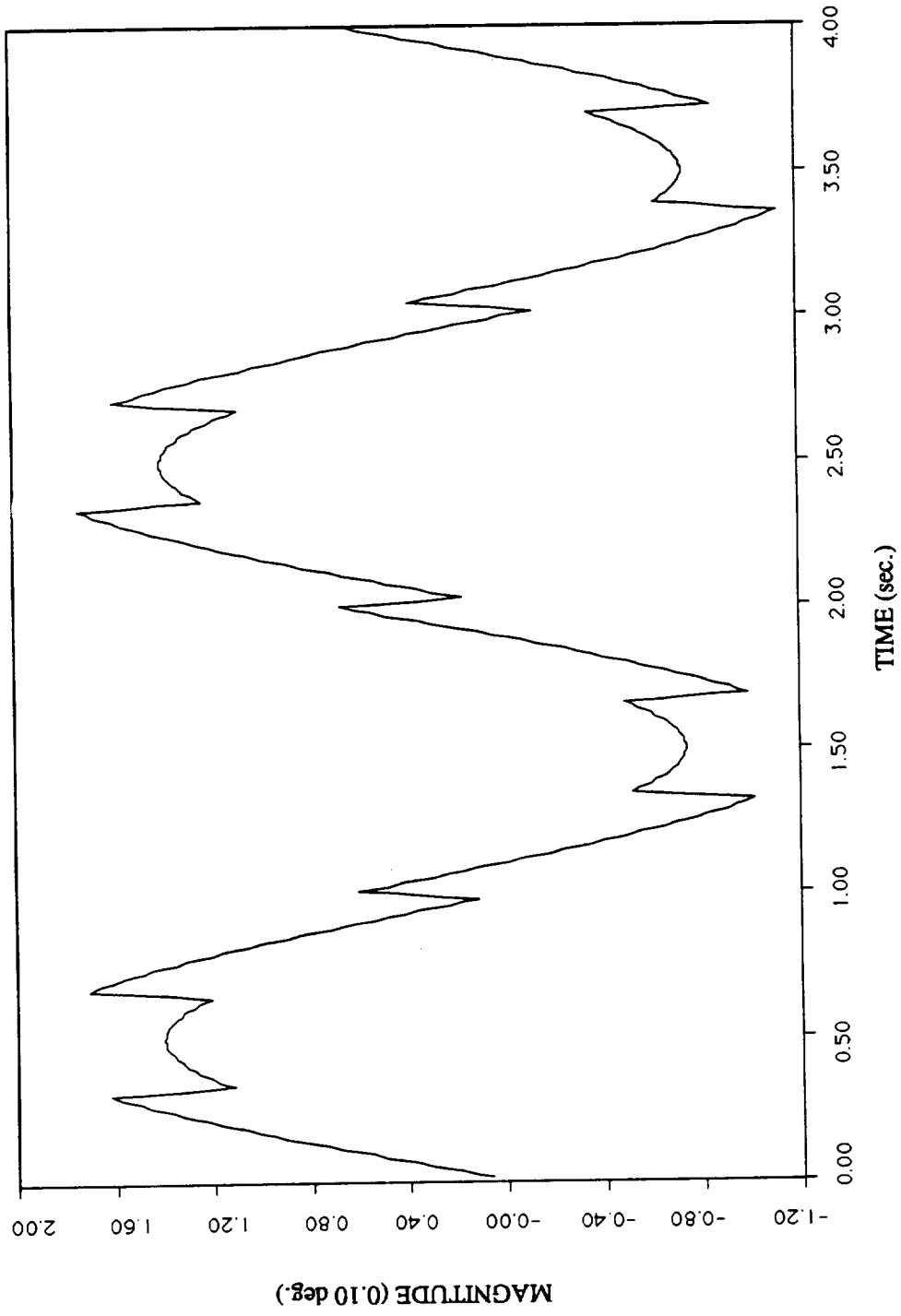


Figure A3. Plot of simulated sinusoidal pursuit data with five (5) degrees of saccadic jump without noise.

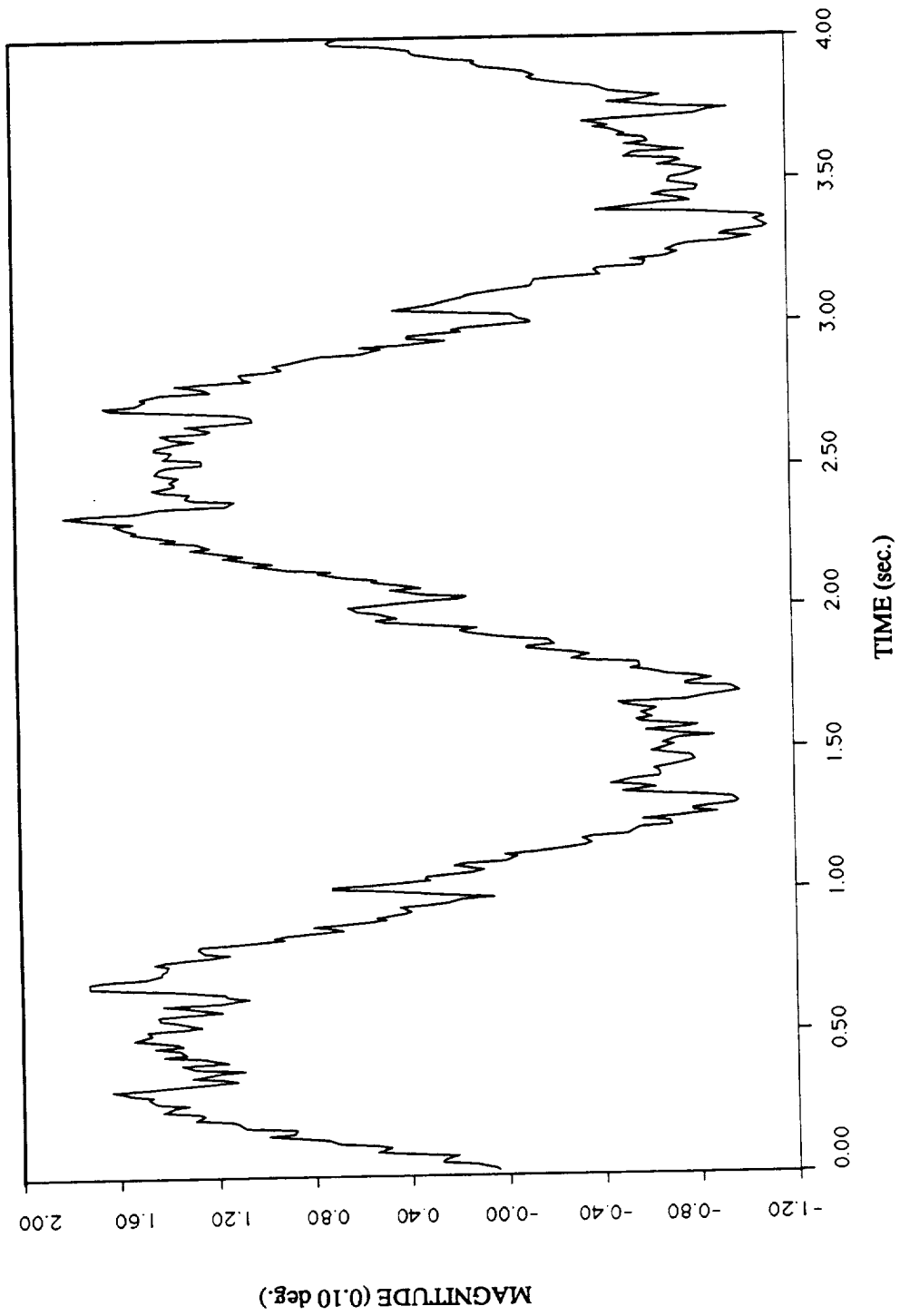


Figure A4. Plot of simulated sinusoidal pursuit data with five (5) degrees of saccadic jump with 0.5 degree (maximum amplitude) noise.

