ABSTRACT

MODEL PARAMETER IDENTIFICATION FOR ROD AND CONE OSCILLATORY POTENTIALS

by
Kehur Banker

The use of signal modeling of the oscillatory potential (OP) of the electroretinogram (ERG), in the study of cone-rod interaction is investigated. ERG response data were analyzed for red, orange, blue and white flashes on no background and with red flashes on a blue background (to suppress rod responses). The OP signals were extracted from the ERG by digital bandpass filtering and a signal model was fitted through a simplex algorithm to produce the parameters including "OP envelope-amplitude", latency in terms of "center-time" of the OP-envelope and OP frequency. Amplitude for red flashes with or without a blue background showed similar increases at high stimulus intensities. White and orange flashes produced higher amplitudes at all stimulus intensities, thus demonstrating the presence of rod OP within the signal. Latencies changed relatively little for pure cone stimuli with increasing intensities, while latency sharply reduced for responses for blue stimuli. Use of signal modeling provides a simple procedure for summarizing the characteristics of the OP in rods and cones over a range of amplitudes.

MODEL PARAMETER IDENTIFICATION FOR ROD AND CONE OSCILLATORY POTENTIALS

by
Kehur Banker

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Biomedical Engineering Committee

January 1995

APPROVAL PAGE

MODEL PARAMETER IDENTIFICATION FOR ROD AND CONE OSCILLATORY POTENTIALS

Kehur Banker

[illegible]

Dr. Andrew U. Meyer, Thesis Adviser — Date
Professor of Electrical Engineering, NJIT

[illegible]

Dr. Edward J. Haupt, Committee Member — Date
Assistant Professor of Ophthalmology,
New Jersey Medical School

Dr. David Kristol, Committee Member — Date
Professor of Chemistry,
Director, Biomedical Engineering, NJIT

BIOGRAPHIC SKETCH

Author: Kehur Banker

Degree: Master of Science in Biomedical Engineering

Date: January 1995

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering, New Jersey Institute of Technology, Newark, NJ, 1995

- Bachelor of Science in Electrical Engineering, Birla Vishvakarma Mahavidyalaya, Sardar Patel University, Vallabh Vidyanagar, India, 1991

Major: Biomedical Engineering

ACKNOWLEDGMENT

I am deeply indebted to my thesis advisor Dr. Andrew Meyer for his excellent guidance and help. His constant encouragement and careful thoroughness contributed to the completion of this thesis.

Special thanks to Dr. Edward J. Haupt for his advice and assistance on the medical aspects of this work.

My sincere gratitude to Dr. King-Smith for providing his data and his timely help and suggestions to complete this work.

Thanks to Mr. Ray Favis, working with him was a great experience.

Finally, I am thankful to everyone who helped me directly or indirectly in completing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the eye, an image is converted to electrical signals by sensory cells and photochemical processes, producing signals feeding into the optic nerve fibers. These signals are processed in various centers of brain. This complex sequence of events is known as a vision.

Electrophysiological testing has become an important diagnosis tool in clinical ophthalmology. It involves the recording of electrical responses, mainly the Electroretinogram (ERG), Electro-occulogram (EOG), Visual Evoked Response (VER). This thesis is concerned with the electroretinogram.

The electroretinogram (ERG) is a response of electrical activity in the retina. More than hundred years ago, in 1865, Frithiof Holmgren had published the first report on the production of electrical potentials by the action of light upon the eye. Dewar & McKendrick [1], in 1873, studied the effect of variation in stimulus intensity upon the amplitude of the response and derived the conclusion that there was a relationship between the height of electrical response and the logarithm of stimulus intensity. In 1908, Einthoven and Jolly [2] described the three sub-components of the ERG recorded form frog retina. The first negative segment is called the *a-wave*, which is followed by a larger positive deflection *b-wave* and slower positive potential *c-wave*. The ERG, the recorded potential, actually represents the summation of electrical activity produced in different layers of the retina. Granit [3] postulated the existence of three processes which he called

PI, PII, PIII. Meanwhile, Kahn & Lowenstein [4] attempted the first clinical use of the ERG, by employing string galvanometer and leads from corneas and temporal point on anesthetized eye ball. Their method was too complicated and difficult to be used in clinical settings. Later, Hartline [5] was able to record human ERGs successfully with a string galvanometer. Granit [6], in 1947, in reviewing all the available knowledge at that time, recognized that fast developing corneal negative PIII forms the *a-wave*. The corneal positive PII (which is much larger) then develops, and resultant PIII and PII produces the *b-wave*. As PII decreases, PI grows slowly and thus produces the *c-wave*. Granit also stated that PII (the *b-wave*) originated somewhere in the neural pathway between the receptors and ganglion cells. The short latency of PIII (the *a-wave*) indicated that it developed very early in the chain of the events constituting retinal activity in receptors.

In addition to the major components a number of other components contribute to the detailed form of the ERG. In 1954, Cobb & Morton [7] described rhythmic wavelets, known as Oscillatory Potentials (OP)[1], on the ascending limb of the *b-wave*. The nature of wavelets, the OP, is still obscure. The origin of the OP has been the subject of numerous investigations. Brown [8], in 1968, recorded the maximum amplitude of the OP in the inner part of the monkey retina and demonstrated their dependence on the retinal circulation. He concluded that they could not originate in cellular structures of outermost layers of retina such as photoreceptor cells or horizontal cells. In 1973, Ogden [9] was the first to record the laminar profile of voltage of the OP-wavelets. He found

1 In the literature, the OP - wavelets are generally referred as "Oscillatory Potentials" (OP). In the present work we denote the entire wavelets as "Oscillatory Potential" (OP)

the origin of the OP-wavelets separate from the *b-wave*. The maximum amplitude of the first three wavelets was found at inner neuronal layers indicating the involvement of the cells of inner plexiform layer. In addition, Wachtmeister and Dowling [10] (1978) found neurotransmitters such as glycine, gama-aminobutyric acid (GABA), dopamine, antagonist haloperidol, B-alanine, glutamate depress the Oscillatory Potentials. These findings suggest that the OP may be an independent component of the ERG, reflecting activity of inhibitory feedback synaptic circuits within the retina. The OP thus seems to reflect neuronal activity in inner plexiform layer and interaction the rod and cone mechanism in the inner part of retina.

Yonemura [11] found that in diabetic retinopathy, main components of the ERG, *a-wave* and *b-wave*, were normal in amplitude and implicit time. But the oscillatory potentials disappeared or decreased greatly. In 1969, Berson, Gouras and Hoff [15] recorded distinguishable ERG responses from a normal subject, from a patient with night blindness and one with rod monochromatism. From these results they concluded that different light stimuli could excite either the rod system or the cone system or both. Neal Peachey, Alexander and Fishman [18] explained the effect of conditioning flash to the rod and cone systems and their contribution to oscillatory potentials. King-Smith, Loffing and Jones [16] also recorded the ERG responses from the various stimuli, and concluded that the first negative peak of the oscillatory potentials might be the rod system response.

In summary, the exact origin of the oscillatory potential is unknown. The oscillatory potential is probably generated in the inner layers of retina and may indicate the condition of inner layer of the retina. Therefore, it is used as a diagnosis tool for patients with severe cases of glaucoma, obstruction of central artery and diabetic retinopathy. Studies of the OP in humans have demonstrated that under specific stimulus conditions it can reflect activity in either the rod and cone systems of the retina or the interaction between these two receptor systems.

This thesis deals with the modeling of the ERG and the OP, and the estimation of the model parameters, obtained by fitting experimental data. The model represents the original OP data in terms of a finite set of parameters. Previous work on parameter identification of the Oscillatory Potential was performed by Pan and Jang individually. Huizhong Pan [12] presented an initial mathematical model for the Oscillatory Potential. Later, Jang [13] modified the model and described the changes of the parameters for diabetic patients. The goal of the present work is to further improve the OP modeling and to examine its utility to distinguish rod and cone system contributions.

CHAPTER 2

THE PHYSIOLOGY OF THE EYE

A cross section of the eye is shown in Figure 2.1 The eye is filled with a clear jelly in its center, called vitreous humor. There are three layers of tissue surround the vitreous humor. The outer layer has tough tissues, sclera, that protects the eye, cornea and conjunctiva. Middle layer has pupil, iris, lens and choroid, the blood vessels that nourish the eye. Third inner layer contains the retina and the fovea. The retina is the light sensitive portion of the eye, containing millions of photoreceptor cells and neurons. When the photoreceptor cells are excited, signals are transmitted through successive neurons in the retina, into nerve fibers and to cerebral cortex.

2.1 Anatomy of the Structural Elements of the Retina

Figure 2.2 shows the functional components of the retina arranged in layers. After light passes through the lens system of the eye and then the vitreous humor, it enters the retina from the inside, that is, it passes through the ganglion cells, the plexiform layer, the nuclear layer and limiting membranes before it reaches the layer of photoreceptors. Photoreceptors contain pigments that change the color when exposed to light.

2.2 Photoreceptors

Figure 2.3 represents a diagrammatic presentation of photoreceptors. There are two kinds

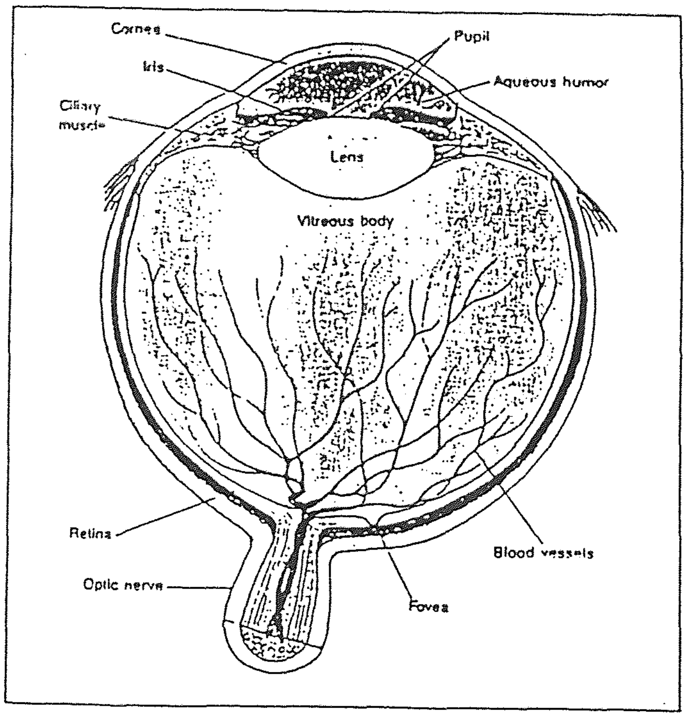


Figure 2.1 The cross section of the eye.

of photoreceptors cones and rods. The cones are distinguished by having a conical upper end. The rods are narrower and usually longer than cones. Photoreceptor cell is segmented into four major functional segments (1) Outer segment, (2) Inner segment, (3) Nucleus and (4) Synaptic body.

In the outer segment the light sensitive photochemical is found. In the case of rods, this is rhodopsin and in the cones it is one of several photochemicals collectively called iodopsin, which are almost exactly same as rhodopsin except for differences in spectral sensitivity. Both rhodopsin and iodopsin are conjugated proteins. These are incorporated into the membranes of the discs in the form of transmembrane proteins. These photosensitive pigment constitute nearly 40 per cent of entire mass of the outer

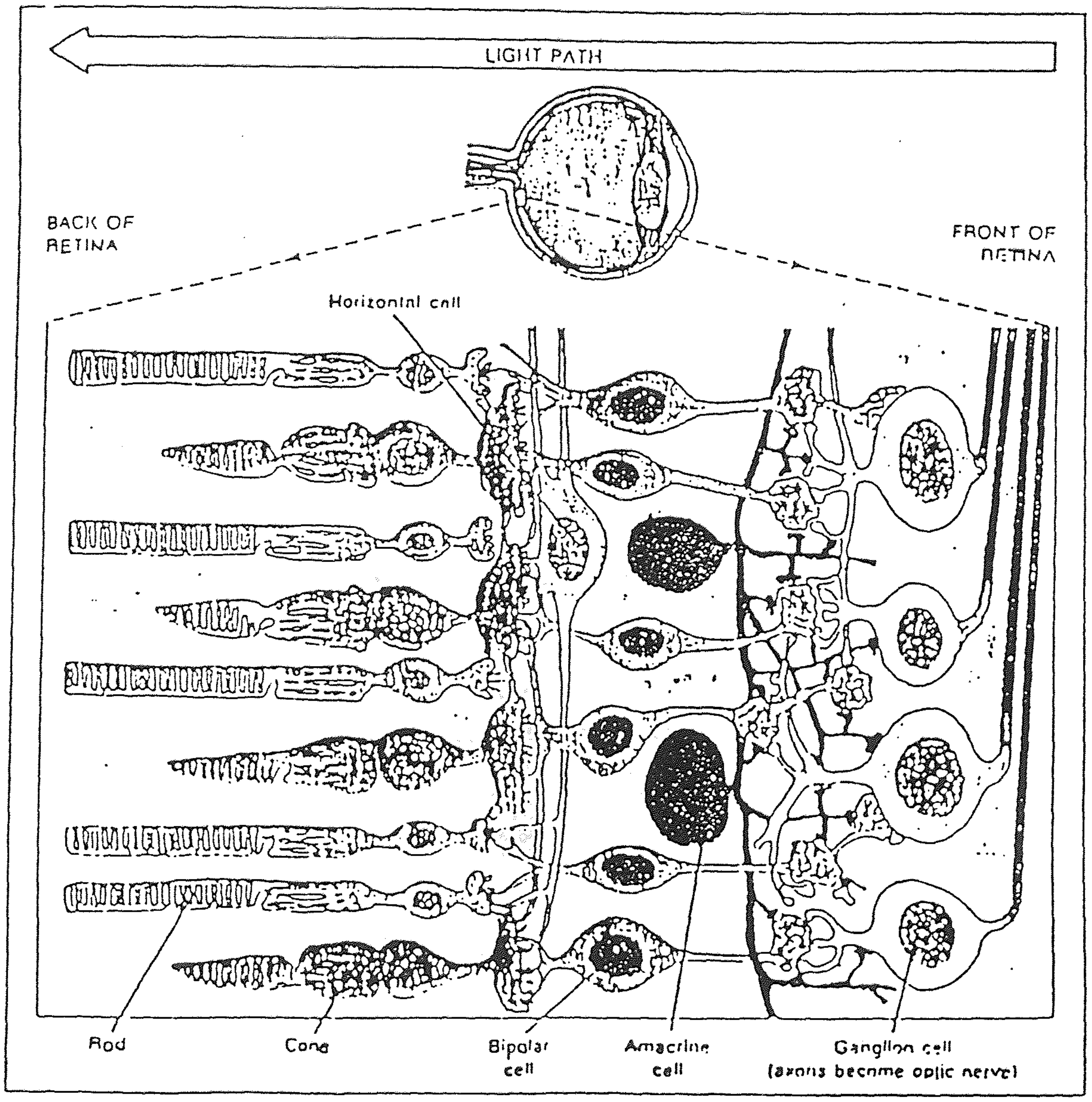


Figure 2.2 Organization of the retina. Light enters the eye through the lens and passes through the vitreous body and the font surface of retina. Since the tips of the rods and cones are on the side of the retina opposite to the light entry, the light must pass through all the layers before reaching the photoreceptors and stimulating them. (From Human Physiology, Fifth edition, McGraw-Hill Publishing Company, 1990.)

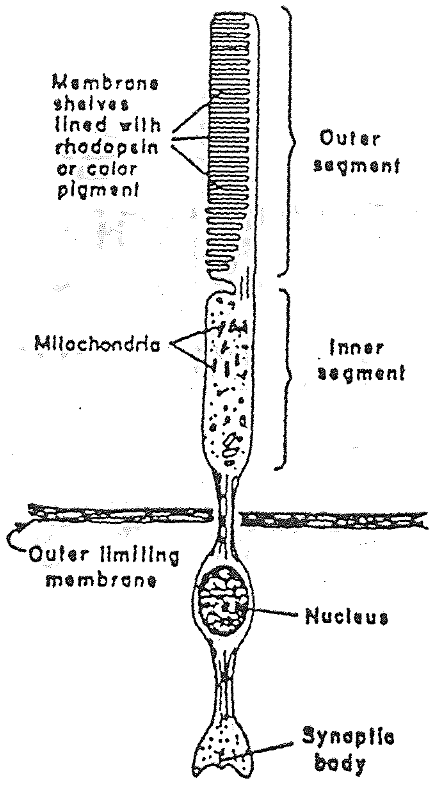


Figure 2.3 The schematic drawings of the functional parts of rod and cone.

segment. The inner segment contains the cytoplasm of the cell with the cytoplasmic organelles and mitochondria. Mitochondria plays an important role in providing an energy for the function of the photoreceptors. The synaptic body connects the rods and cones to subsequent neuronal cells, the horizontal and bipolar cells. They act as a relay between receptors and optic nerve fiber. The black pigment melanin, layer prevents light reflection through out the globe of the eye ball.

2.3 The Electroretinogram and the Oscillatory Potentials

The Electroretinogram (ERG) is a light evoked response of the eye, obtained from contact lens on pupil and ground at ear. The ERG is a total retinal activity, consists the responses created from different parts of retina. Granit [3] had analyzed the ERG in *a-wave*, *b-wave* and *c-wave*. The *a-wave* is a response of the receptor layer and *b-wave* is a response of neural pathway between receptors and ganglion cells.

Cobb and Morton [7] had described the wavelets on the ascending limb of the *b-wave*, known as oscillatory potential (OP). Figure 2.4 shows the unfiltered ERG response obtained from the normal subject for scotopic white flash. It contains *a-wave, b-wave, c-wave* and oscillatory potentials. This ERG response can be broken down into two categories. (1) The low frequency response, obtained by filtering the ERG response with low pass filter (< 70 Hz), mainly represents the *a-wave, b-wave* and *c-wave*. (2) The high frequency response, obtained by filtering the ERG with a high pass filter (100-200 Hz), produces the oscillatory potential. Figure 2.5 gives the representation of low and high frequency responses.

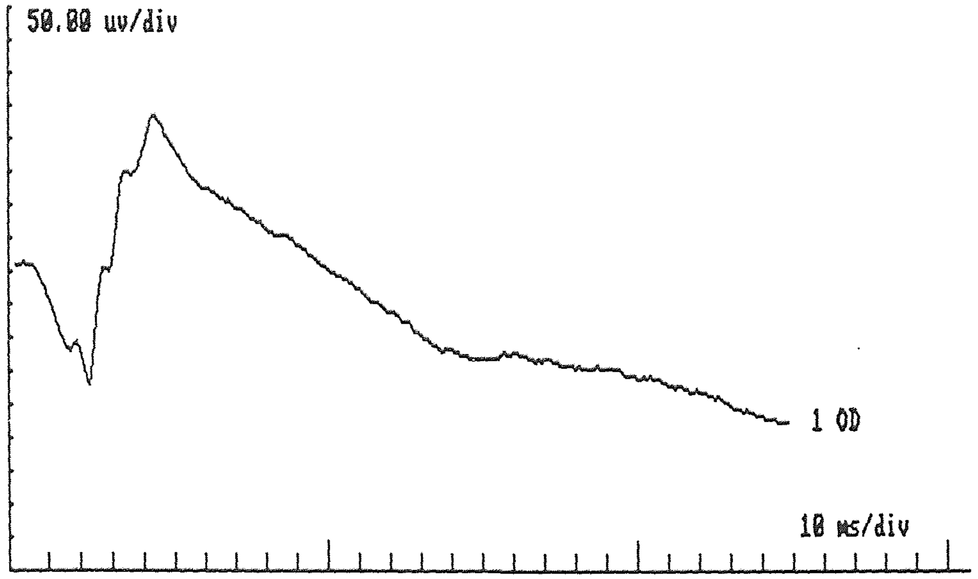


Figure 2.4 The unfiltered ERG response, for scotopic white flash.

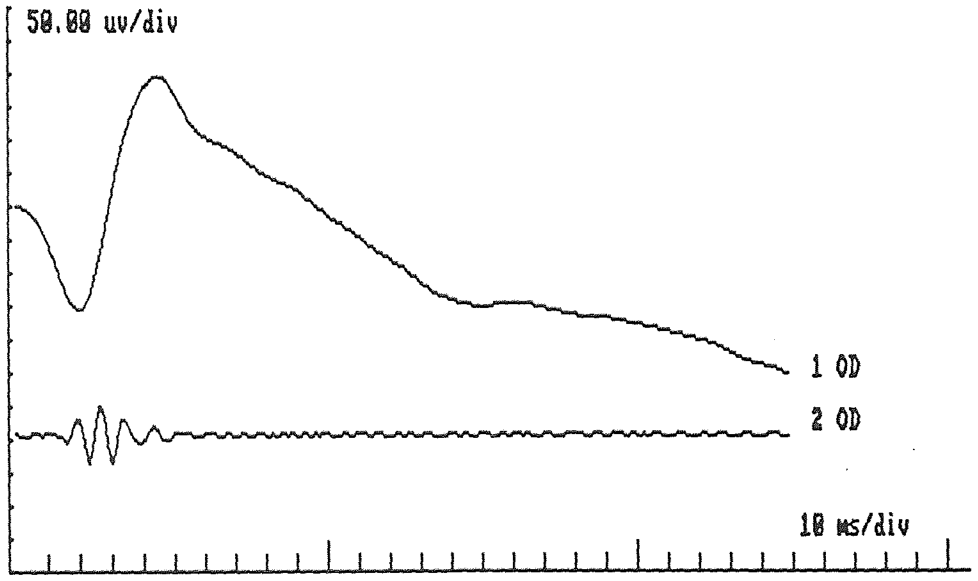


Figure 2.5 The representation of low and high frequency response of unfiltered ERG.

Figure 2.6 represents a power density spectrum in the frequency domain for a typical ERG obtained from blue stimuli. Most of the power is concentrated below 60 Hz. However, small additional components appears between frequencies of 65 to 170 Hz, which matches the range of the oscillatory potential. Therfore, the oscillatory potentials are extracted form the ERGs by digital bandpass filtering between 65 to 170 Hz (the results are in Chapter 6). By reviewing the power density spectrum for other stimuli, it is found that, same as blue stimuli, the major power concentration is below 60 Hz, but a small power is concentrated either between 65 to 200 Hz or between 100 to 200 Hz. Therefore, second set of oscillatory potentials are extracted from their ERG responses by treating them individually (the results are in Appendix-C).

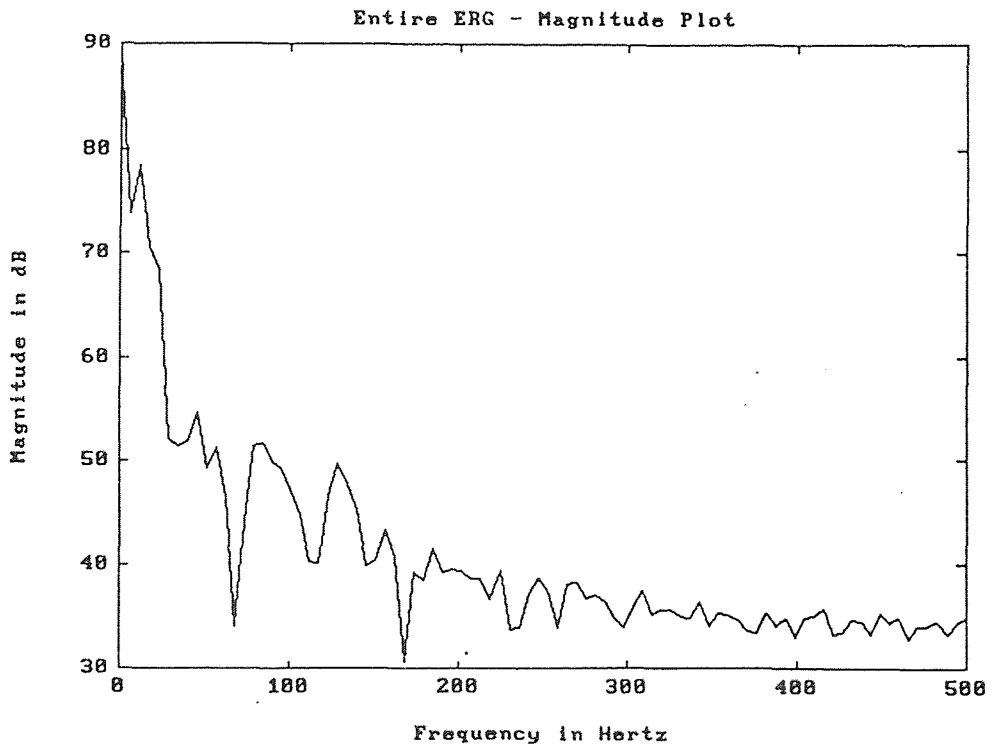


Figure 2.6 Power density spectrum of the typical ERG, in frequency domain.

CHAPTER 3

ROD AND CONE COMPONENTS OF THE ERG

Figure 3.1 presents (1) the scotopic luminosity curve and (2) the photopic luminosity curve, derived from the psychophysiological measurements of peripheral retinal function. It can be concluded from the graph that rods are more sensitive than the cones almost through out entire visible spectrum. But at longer wavelength the difference between curves diminishes. This means that the cone response can be isolated with stimuli of longer wavelength. The scotopic luminosity curve represents the spectral sensitivity function of the rod system. The rod contribution to the ERG can be obtained by recording the response to stimulation of the dark-adapted eye to dim light-flashes of relatively short or long wavelength. From the spectrum of the visible light, dim short wavelength gives blue color while dim long wavelength gives orange-red color, and $\lambda > 680$ nm is deep red. [19]

This leads to a supposition that the rod contribution to the ERG can be generated by recording the ERG with blue or orange-red flashes. Separate cone response ERG can be obtained by stimulating the eye with single flashes of deep red light. But with the Grass Photosimulator as the light source, deep red light is too dim to elicit an easily detectable cone ERG. The cone ERG can be obtained by broad band filters like wratten 26 ($\lambda > 600$ nm). It can also be recorded, even more precisely, by suppressing rod contribution when the eye is stimulated in the presence of a blue background light.

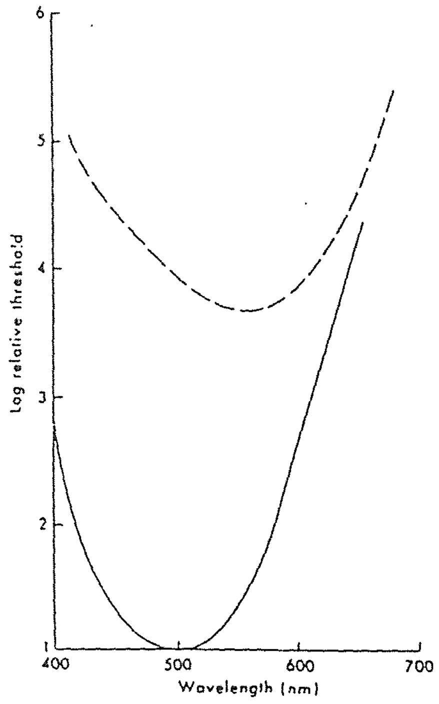
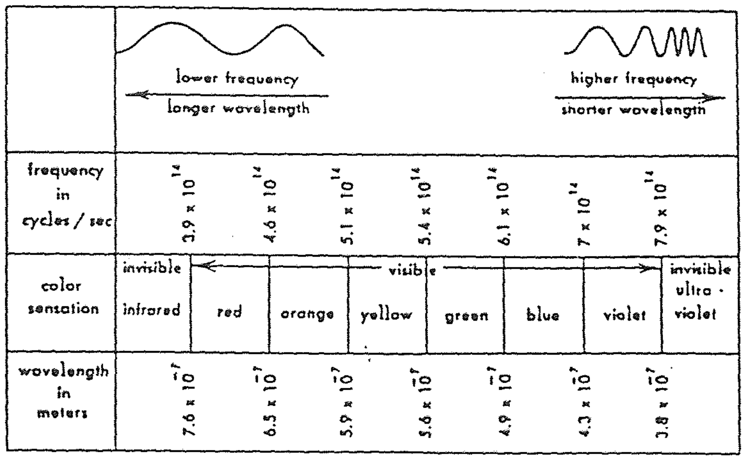


Figure 3.1 Continuous line is CIE (Commission International de l'Eclairage) scotopic luminosity curve (rod spectral sensitivity function)derived from psychophysical measurements and placed at level for normal human subjects; dashed line is Wald's photopic luminosity curve (spectral sensitivity function for the cone mechanisms under photopic conditions) derived from psychophysical measurements of peripheral retinal function. ERG spectral sensitivity curves for normal rod and cone systems also respectively approximate solid line and dashed line curves. (From Berson, Adler's physiology of the eye. 460,1970.)


lower frequency
longer wavelength
higher frequency
shorter wavelength

frequency in cycles / sec	3.9×10^{14}	4.6×10^{14}	5.1×10^{14}	5.4×10^{14}	6.1×10^{14}	7×10^{14}	7.9×10^{14}	
color sensation	invisible infrared	red	orange	yellow	green	blue	violet	invisible ultra-violet
		←	visible				→	
wavelength in meters	7.6×10^{-7}	6.5×10^{-7}	5.9×10^{-7}	5.6×10^{-7}	4.9×10^{-7}	4.3×10^{-7}	3.8×10^{-7}	

Figure 3.2 The spectrum of the visible light.

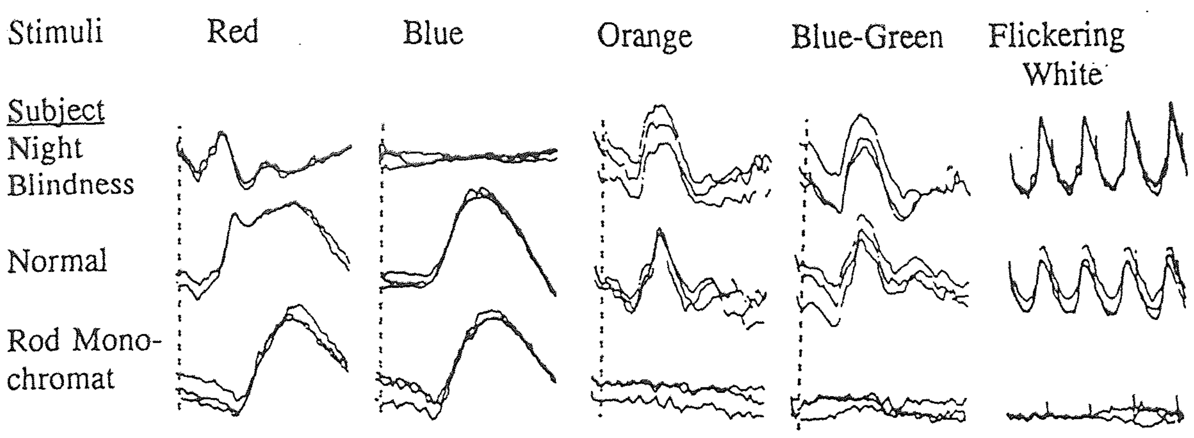


Figure 3.3 ERG responses to scotopically balanced red and blue light stimuli, to photopically balanced orange and blue-green stimuli in presence of 5 to 10 ft-L background light, and to wickering (30 Hz) white stimuli are shown successively from top to bottom for patient with night blindness, normal subject and congenital rod monochromat. (From Berson EL, Gouras P, Hoff M: Arch Ophthalmol 81:207,1969.)

In 1969, Berson et al [15] illustrated the cone and rod contribution to the total ERG. They represented the comparison of results recorded from a normal subject, from a patient with dominant stationary night blindness and from a patient with congenital rod monochromatism. The different ERG responses were recorded for five different stimuli, two scotopic (1) stimuli of red ($\lambda > 600$ nm) and (2) blue ($\lambda < 470$ nm), two photopic (3) stimuli of orange ($\lambda > 550$ nm) and (4) blue-green ($\lambda < 550$ nm), in presence of 5 to 10 ft-L background light and last one (5) with flickering (30 Hz) white stimuli. For their special set of experiments, they obtained the responses shown in Figure 3.3. (1) For short wavelength light (i.e. rods) elicited responses were normal in amplitude and implicit time for the rod monochromat and normal subject. But the response of night blindness patient was poor. (2) For long wavelength light, only cone components were elicited in the night blindness patient, only rod components in rod monochromat and both components in the normal subject. (3) For stimulus conditions which eliminate rod response, but stimulate long (orange) and medium (blue-green) wavelength cones, the responses were similar in implicit time and amplitude for the night blindness patient and normal subject, but no significant response from the rod monochromat was observed. (4) For flickering light, responses were similar to those received for photopic condition. From the results they concluded that short wavelength light stimulated rod system, long wavelength light stimulated both rod and cone both systems, while photopic flash and flickering light stimulated only the cone system.

As the ERG is a response of total retinal activity, the changes in the ERG for different light stimuli, may also cause a considerable changes in the oscillatory potential. Thus, there have been attempts to investigate and distinguish the OP generated in rod and cone system. King-Smith et al. [16] and Neal et al. [18] presented the same idea in 1986 and 1987. These groups of investigators focused on recording the responses of eye to different stimuli, to stimulate the responses of rod and cone systems separately; they concluded that the differences between red and blue responses were due to responses generated by rods. While comparing blue, orange, and white flash responses, they [16] found the largest negative peak at similar implicit time, and interpreted that as a rod system response, while the following waves were considered to be due to the cone system response.

The present study is based on the original responses recorded by Dr. King-Smith. The objective is to continue his initial ideas and to fit the rod and cone system generated oscillatory potentials to our mathematical model and to represent the rod and cone OP by our set of model parameters.

CHAPTER 4

MODELING OF THE OSCILLATORY POTENTIALS

4.1 A Model of the ERG.

The total ERG can be considered to be the combination of the *a-wave, b-wave*, minor components *c-wave* and *d-wave,* as well as the high frequency component called oscillatory potential. This implies that recorded unfiltered ERG can be represented as a combination of major components *a-wave, b-wave* and OP. Researchers and clinical professionals have emphasis on *a-wave, b-wave* and OP due to their significant physiological and clinical meanings.

$$ERG(t) = E_a(t) + E_b(t) + OP(t) \tag{4.1}$$

Juan Castro [17] had established the model for separating *a-wave* and *b-wave* from the filtered ERG response. He represented the filtered ERG as:

$$ERG(t) = E_a(t) + E_b(t) \tag{4.2}$$

and represented the *a-wave* as:

$$E_a(t) = k_1 \cdot [1 - e^{-[\alpha_1 \cdot (\frac{t}{T}) \cdot e^{(1 - \frac{t}{T})^5}]}] \tag{4.3}$$

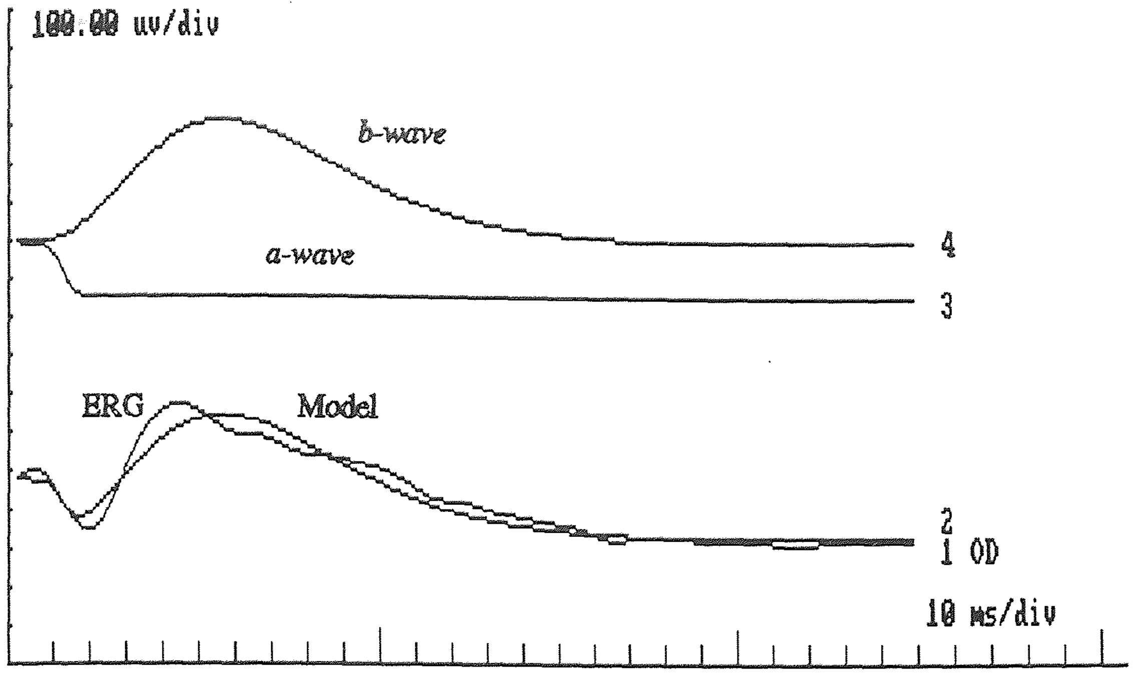


Figure 4.1 The filtered ERG and its model response (Equation 4.2)

and the *b-wave* as:

$$E_b(t) = k_2 \cdot t^3 \cdot e^{(-3\ \alpha_2\ t)} \cdot \sin\ (w\ t) \qquad (4.4)$$

Fig. 4.1 illustrates the model of *a-wave* and *b-wave* of the ERG, represented by Juan.

4.2 The Model of the Oscillatory Potential

The oscillatory potential appears as a series of wavelets on ascending limb of the *b-wave*. When the ERG response is filtered at higher frequency the OP resembles as a sinusoidal function whose amplitude rises and then flattens out.

Pan [12] has described his model according to above explained shape of the OP.

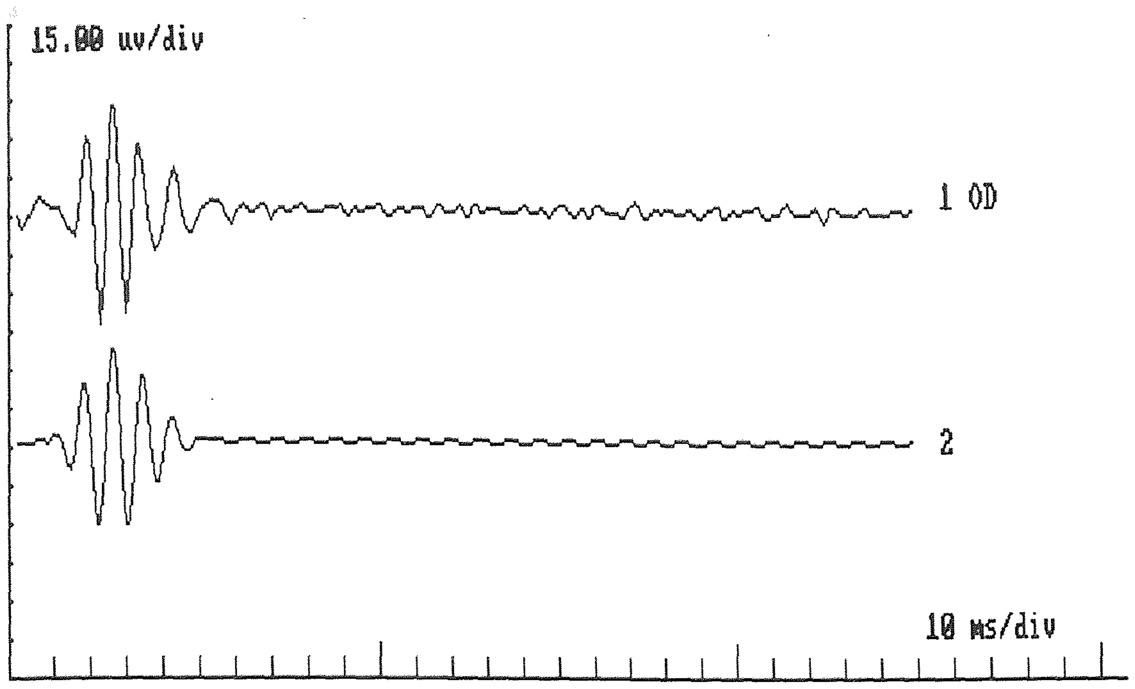


Figure 4.2 The oscillatory potential (top) and the model $OP_1(t)$ (bottom).

$$OP_1(t) = k_1 \cdot t^n \cdot e^{-\alpha_1 t} \cdot \sin (2 \pi f_1 t + \Phi_1) \tag{4.5}$$

This model is illustrated in Figure 4.2.

For a particular run, for which the parameter set is

$\{ k_1, n, \alpha_1, f_1, \Phi_1 \} = \{ 0.00447\ \mu v/(msec)^n, 4, 0.1537\ sec^{-1}, 118.8\ Hz, 0.726°\ deg \}$

A second model was presented as a function of two sine waves

$$OP_2(t) = k_2 \cdot t^n \cdot e^{-\alpha_2 \cdot t} \cdot \sin (wt) \cdot \sin (2 \pi f_2 t + \Phi_2) \tag{4.6}$$

where $k_2 \cdot t^n \cdot e^{-\alpha_2 t} \cdot \sin (wt)$ is the *b-wave* model of Castro, and the term

$\sin(2\pi f_2 t + \Phi_2)$ represents the high frequency oscillation.

For very small w, $\sin(w \cdot t) \approx w \cdot t$ results in,

$$OP_2(t) \cong k_2 \cdot w \cdot t^{n+1} \cdot e^{-\alpha_2 t} \cdot \sin(2 \pi f_2 t + \Phi_2) \tag{4.7}$$

Starting with Pan's OP model Jang [10] established an improved OP model

$$OP_3(t) = \frac{k_2 \cdot w}{\alpha_2^{n+1}} \cdot (k_2 \cdot t)^{n+1} \cdot e^{-\alpha_2 t} \cdot \sin(2 \pi f_2 t + \Phi_2) \tag{4.8}$$

Substituting, $k_o = k_2 \cdot w/(\alpha_2^{n+1})$, $\alpha_o = \alpha_2$, $f_o = f_2$, $\Phi_o = \Phi_2$ and n+1 representing as n, he obtained

$$OP_3(t) = k_o \cdot (\alpha_o \cdot t)^n \cdot e^{-\alpha_o t} \cdot \sin(2 \pi f_o t + \Phi_o) \tag{4.9}$$

Using a simplex method for parameter identification, he found $n = 7$ to provide a good fit. He disregarded the phase angle Φ_o as it appeared to be randomly distributed. As his final OP model, Jang used:

$$OP_4(t) = k_o \cdot (\alpha_o \cdot t)^7 \cdot e^{-\alpha_o t} \cdot \sin(2 \pi f_o t) \tag{4.10}$$

For same particular run, the parameter set was found to be

$\{ k_o, \alpha_o, f_o \} = \{ 0.04726\ \mu v,\ 0.2327\ \text{msec}^{-1},\ 108.8\ \text{Hz} \}$

A generalization of $OP_4(t)$ is

$$OP(t) = k_o \cdot (\alpha_o \cdot t)^n \cdot e^{-\alpha_o t} \cdot \sin(2 \pi f_o t + \Phi_o) \tag{4.11}$$

This will be the general model considered in this thesis, whose parameters are

$\{ k_o, \alpha_o, f_o, \Phi_o, n \}$

Jang and Pan's mathematical models were tested with white flash oscillatory potentials. The recorded data from Dr. King-Smith, for flashes of different colors shows a characteristic of reduction in amplitude and increase in time delay with decrease in the luminance intensity. The parameters $\{ k_o, \alpha_o, f_o, \Phi_o, n \}$ can be transformed into another set which better illustrates the features of the waveform, shown in Figure 4.3. That set is defined below:

α	=	amplitude of envelope of OP(t) [μV]
T_c	=	"Center-time" = time at which envelope of OP(t) is maximum [msec]
f_o	=	frequency of oscillatory potential [Hz]
Φ_o	=	phase-angle of oscillatory potential
n	=	exponent, the order of the model equation.

In terms of these parameters, OP(t) can be written as

$$OP(t) = \alpha \cdot (e \cdot \frac{t}{T_c})^n \cdot e^{-n\frac{t}{T_c}} \cdot \sin(2 \pi f_o t + \Phi_o) \tag{4.12}$$

where e = 2.718... Parameters f_o and Φ_o are the same for equation (4.11) and (4.12).

The relations between the remaining two parameters in equations (4.11) and (4.12) are

$$\alpha = k_o\,(n / e)^n, \qquad T_c = n / \alpha_o \tag{4.13}$$

In Appendix-A it is shown that α represents the maximum value of the OP-wave envelope and T_c represents the time when it occurs. A simplex method was used for parameter estimation for the OP model of equation (4.12).

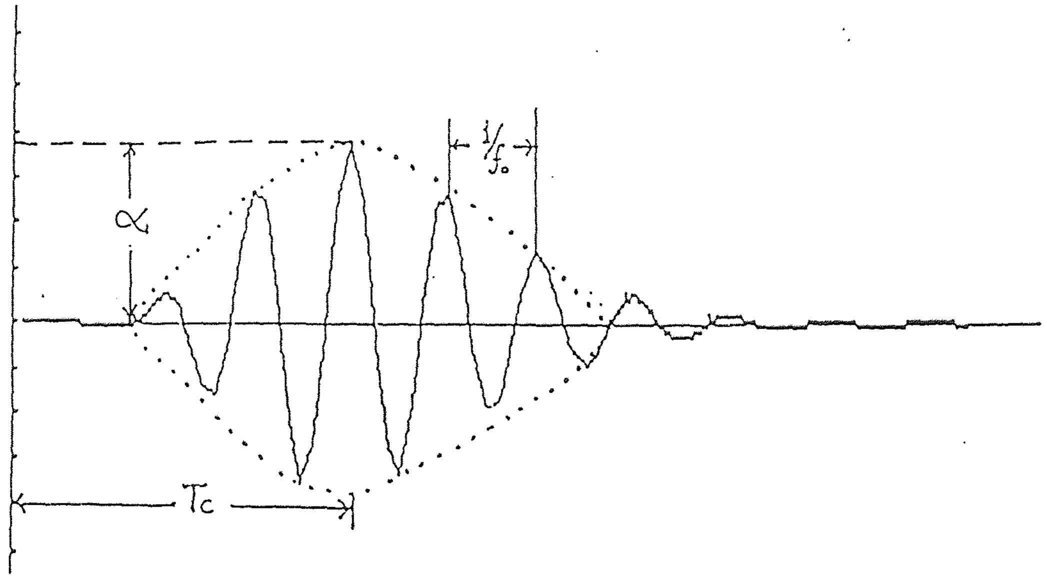


Figure 4.3 The wave shape of OP(t)

The Parameter Identification program is based on minimization of the Standard Deviation Error (ERR STD) and Estimated Error Function (EEF)

$$\text{ERR STD} = \sqrt{\frac{\sum_{i-1}^{n} (X_i - u)^2}{n - 1}} \qquad \text{EEF} = \frac{ERR\ STD}{\sqrt{n - m}}$$

where X_i is the error between data and the model at time i, n is number of samples used in each response. u is the average value of the error X_i and m is number of parameters.

Parameter estimation was performed in two ways:

(1) The entire parameter set $\{\alpha, T_c, f_o, \Phi_o, n\}$, was estimated, (2) For several fixed values of n, the remaining set, $\{\alpha, T_c, f_o, \Phi_o\}$, was estimated. As one would expect, estimation of the full parameter set resulted in lower standard deviation error than estimation of the partial set (with n fixed) (Table 4.1).

Table 4.1 Comparison of parameters and standard deviation error for (fixed) n = 6,7,8,9 and for n as a estimated parameter (last raw) (These are the results for data KING-283, red stimuli.)

Value of n	α [μV]	Tc [ms]	*f*o [KHz]	Φo [deg]	n	Std. Err
6	2.33	27.51	0.137	-31.08	*	0.846
7	2.39	28.17	0.135	-7.67	*	0.845
8	2.54	27.14	0.137	-37.68	*	0.846
9	2.59	27.02	0.137	-32.38	*	0.846
n	2.42	28.07	0.135	-6.577	7.33	0.842

It is seen from table 4.1 that fixing n to a value of 7 results in a reasonable close representation of the OP. So, our OP model becomes

$$OP(t) = \alpha \cdot (e \cdot \frac{t}{T_c})^7 \cdot e^{-7\frac{t}{T_c}} \cdot \sin(2\pi f_o t + \Phi_o) \qquad (4.14)$$

New established model of oscillatory potential gives the amplitude in μv, time delay in msec, frequency in KHz and phase angle in deg.

CHAPTER 5

PROTOCOLS AND PROCEDURE

The data used were obtained by Dr. King-Smith in 1984-85 [16]. All data obtained should fulfil the protocols set up by International Standards of Electro-retinogram [20], so that same ERGs can be evaluated and analyzed at any time and any where with different procedures. In order to separate the rod and cone system distribution, Dr. King-smith followed the International protocols and recorded the ERG responses for blue flash (λ = 475 nm), red flash (λ = 624 nm) orange flash (λ = 587 nm), white flashes and red on blue flashes.

5.1 Testing Procedure

Dr. King-Smith had used himself as a subject (normal subject) and performed the test on his left eye. The eye was dilated with two drops of 0.5 % tropicamide, and then dark adapted for nearly thirty minutes. Gold foil electrode (EL50, SC Electronics) were placed on the surface of the cornea and subjected to different stimuli. For scotopic condition, the ambient luminance of Ganzfeld was kept to a low level (about 2 X 10^{-6} scotopic cd/m^2). For photopic condition, for red flashes on blue background, luminance of Ganzfeld was kept at 7 scotopic cd/m^2 with blue light.

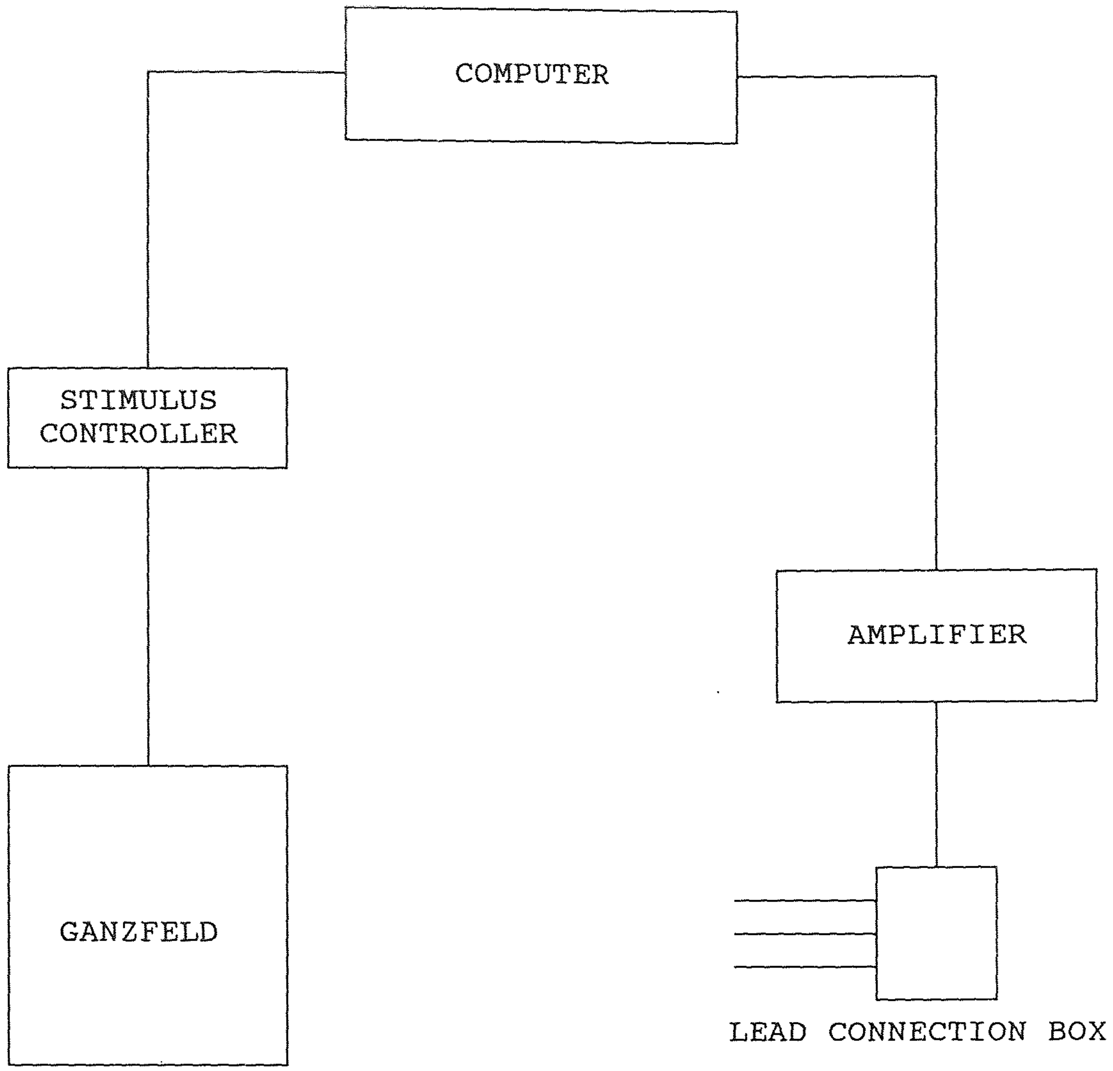


Figure 5.1 Major components of the ERG testing system [16].

5.2 Testing Setup

Figure 5.1 shows the test setup used by Dr. King-Smith, which consists of an amplifier, computer and a Grass photostimulator. The subject was subjected to different flashes in 50 cm Ganzfeld illuminated by a Grass PS22 Photostimulator. In the Ganzfeld, different colors were produced by controlling Grass colored filters (Model 5CF). Various intensities were obtained by photostimulator "Flash intensity" control and by Wratten neutral density filters. The intensities of white and colored flashes were calibrated by using a Pritchad 1980 photometer. The electrodes from the patients eye were connected to the amplifier (Data Inc. 2124, Fort Collins, CO) which had 0.2 and 500 Hz, low and high cut off frequency respectively. The amplifier was connected with the computer (Northstar Horizon). The amplified signal was fed to the computer where it was sampled with the interval of 1-msec. For every set of stimuli, the multiple responses were recorded at nearly 5 minutes interval. The final recorded responses were the average of 50 to 200 individual recorded responses. These responses were stored on the magnetic disk and computer.

5.3 Parameter Identification Program

The stored data, which was in Northstar Horizon computer format, was converted into DOS format, so that it can be processed in our laboratory at the Eye Institute, UMDNJ. The analysis of the data was performed with two major programs at our laboratory, Multi Function Testing System (MFTS3) and the parameter identification program (OPFIT2) [Appendix D].

MFTS3 is used to perform the actual test, to calibrate the testing system, and to read the data files. OPFIT2 is a parameter identification program, which is modified and divided into two sub-programs [Appendix D]. The first sub-program consists the mathematical equations, the model and initial parameters. It has an option to use different mathematical equations. In this revision of program, it has mathematical equations for the *a* & *b-waves*, for the oscillatory potential alone and for combination of the *a* & *b-waves*, and the oscillatory potential. The second sub-program performs the parameter identification by simplex method. The output presents the set of parameters, number of iterations, standard deviation error, estimated function error and the type of equation used. The output can be saved in formatted form so that it can be reviewed on spreadsheet.

CHAPTER 6

RESULTS

All the available data, with different stimuli and their various intensities, were tested by newly defined model (Equation 4.14). Set of parameters are obtained. Figure 6.1, 6.2, 6.3, 6.4 and 6.5[2] show the ERGs and oscillatory potentials obtained with bilateral (12th order Butterworth) band-pass filter of 65 to 170 Hz, for the stimuli of red, orange, blue, red on blue and white respectively. The corresponding tables are showing the set of parameters for various intensities for that individual stimuli. Figures 6.6 and 6.7 present plots of parameters α (envelope - amplitude), T_c (center-time, related to latency) f_o (frequency) respectively, each versus log intensity of the stimulus cd-sec/m^2. Curves are shown for all runs considered. The oscillatory potentials and their model representations are shown in Appendix-B.

The oscillatory potentials were also obtained by treating each ERG individually and filtered with 65-170 or 100-200 Hz, whose parameters are shown in Appendix-C.

[2] These figures are the representation of the first data of each stimuli. The oscillatory potential and its model representation for each data is shown in Appendix-B.

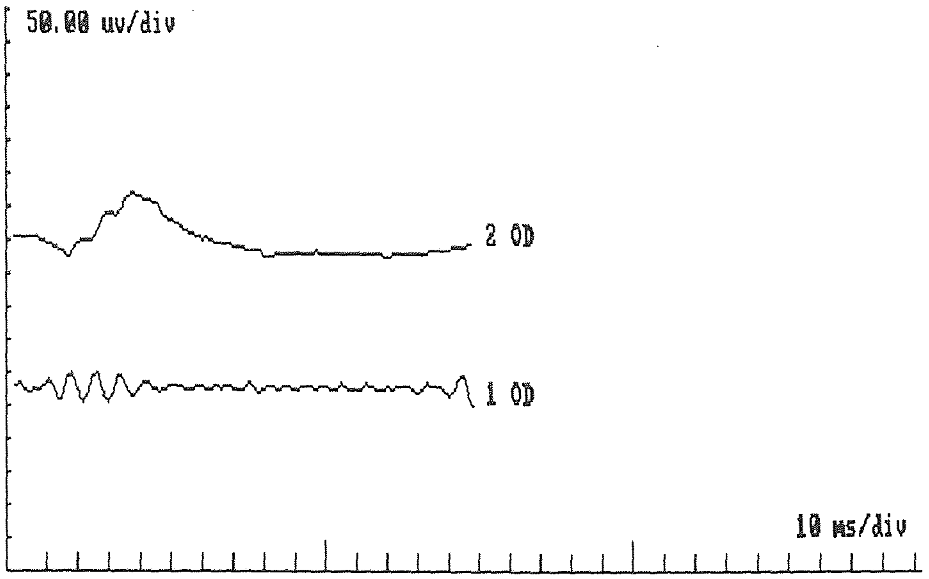


Figure 6.1 The ERG response and its oscillatory potential for red stimuli.

Table 6.1 Parameters of the oscillatory potential model for red stimuli.

Exp	Log Intensity	α [μV]	Tc [msec]	fo [KHz]	Std. Err
KING-281	-0.51	10.26301	26.06931	0.123735	0.094731
KING-282	-0.91	4.014149	19.07560	0.098099	0.75086
KING-283	-1.17	2.369313	27.34898	0.139117	0.93174
KING-284	-1.33	1.342331	32.87374	0.126735	0.50973
KING-285	-1.60	1.53329	22.04616	0.105868	0.41993

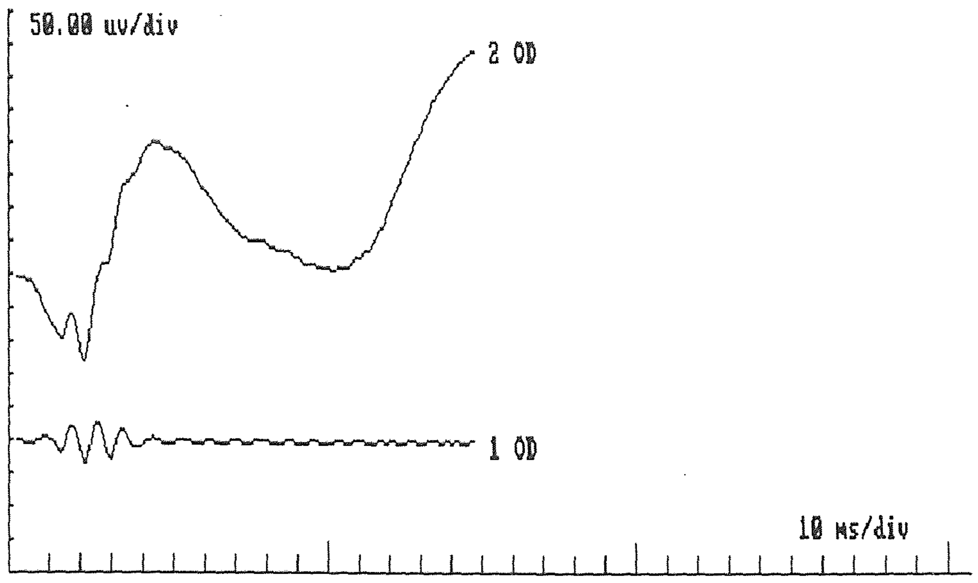


Figure 6.2 The ERG response and its oscillatory potential for orange stimuli.

Table 6.2 Parameters of the oscillatory potential model for orange stimuli.

Exp	Log Intensity	α [μV]	Tc [msec]	$f o$ [KHz]	Std. Err
KING-379	0.23	38.57754	20.83057	0.117046	3.03282
KING-380	-0.16	25.64651	18.64779	0.092706	3.79435
KING-381	-0.42	22.34930	23.78431	0.083447	2.20361
KING-382	-0.59	20.30886	25.40015	0.084641	2.45324
KING-383	-0.88	10.70566	30.61588	0.150388	3.84521
KING-384	-1.34	8.778239	34.19713	0.133354	2.64063
KING-385	-1.6	5.98896	35.00263	0.119559	1.5546
KING-386	-1.77	3.719862	37.40871	0.108865	1.31892
KING-387	-2.06	1.329832	59.28481	0.12259	1.07677
KING-388	-2.39	1.082916	78.97556	0.112728	0.7801
KING-389	-2.66	1.583515	84.60490	0.083066	0.26514

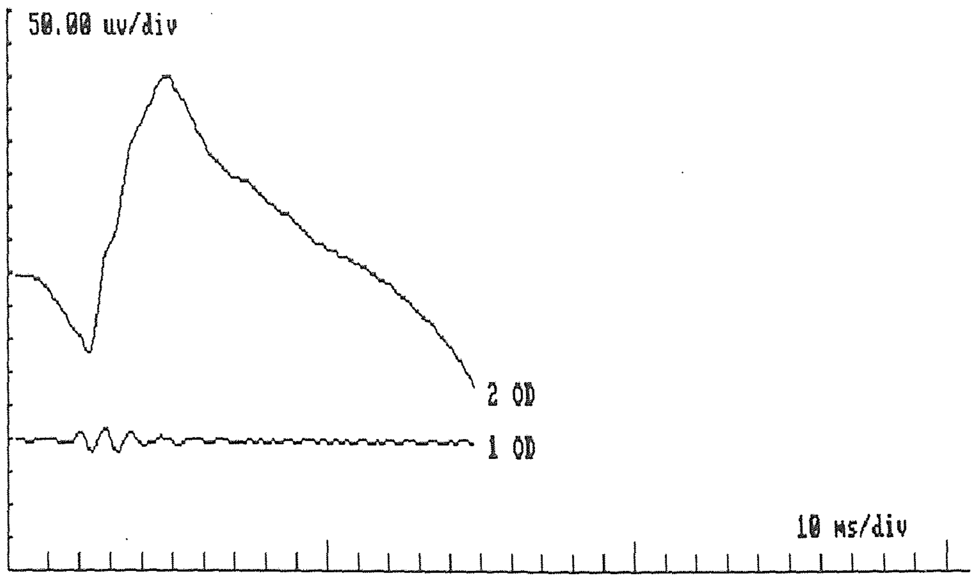


Figure 6.3 The ERG response and its oscillatory potential for blue stimuli.

Table 6.3 Parameter of the oscillatory potential model for blue stimuli.

Exp	Log Intensity	α [μV]	Tc [msec]	fo [KHz]	Std. Err
KING-393	-0.53	22.57278	28.53132	0.104093	2.39072
KING-394	-0.92	21.88391	27.17972	0.106076	2.74945
KING-395	-1.18	21.75148	26.46883	0.109572	4.42003
KING-396	-1.37	20.85893	28.50291	0.125943	5.0484
KING-397	-1.69	13.34942	29.78435	0.13621	4.92066
KING-398	-2.14	11.11591	34.88143	0.147321	3.65956
KING-399	-2.41	7.510646	39.7816	0.137961	2.72174
KING-400	-2.60	5.700554	37.46703	0.1175	2.82
KING-401	-2.92	4.480616	48.16443	0.135338	1.63113
KING-402	-3.26	3.480011	46.82095	0.076113	1.49662
KING-403	-3.52	1.723057	47.94317	0.137844	1.25056
KING-404	-3.71	2.12797	72.54427	0.091127	0.75427

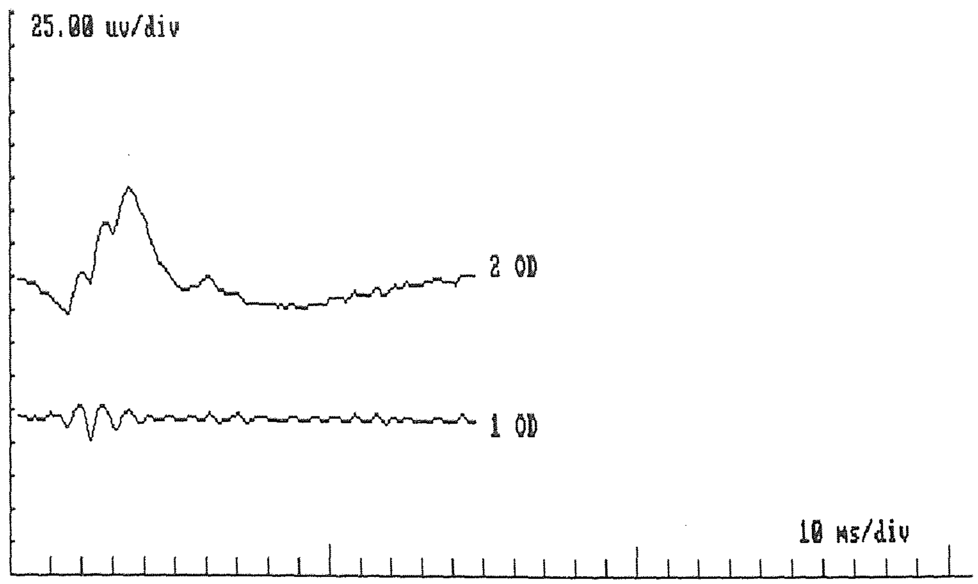


Figure 6.4 The ERG response and its oscillatory potential for red on blue stimuli.

Table 6.4 Parameters of the oscillatory potential model for red on blue Stimuli.

Exp	Log Intensity	α [μV]	Tc [msec]	fo [KHz]	Std. Err
KING-614	-0.51	12.36579	22.61574	0.123865	1.87389
KING-615	-0.91	4.689181	27.40737	0.077014	1.7288
KING-617	-1.17	2.39954	27.16956	0.137812	0.7967

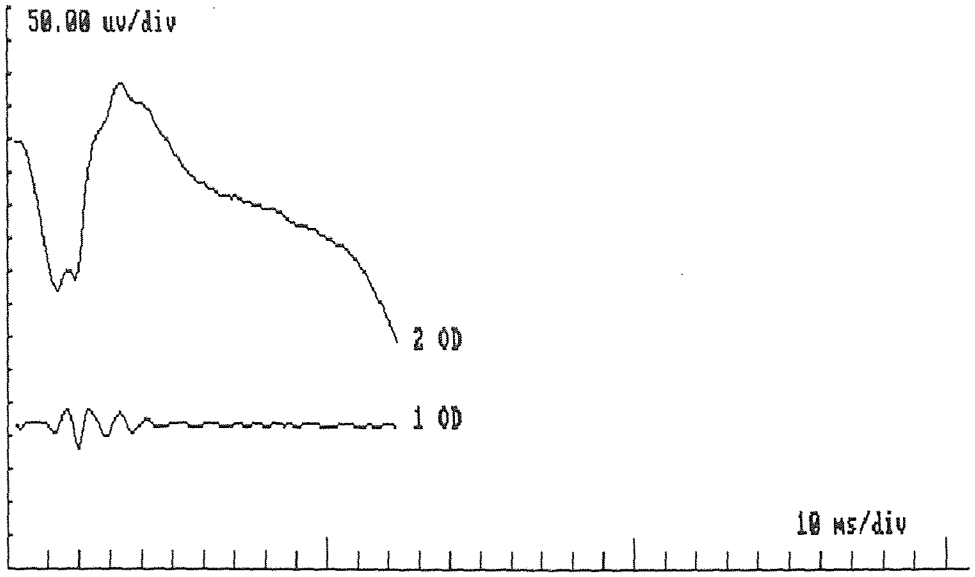


Figure 6.5 The ERG response and its oscillatory potential for white stimuli.

Table 6.5 Parameter of the oscillatory potential model for white stimuli.

Exp	Log Intensity	α [μV]	Tc [msec]	fo [KHz]	Std. Err
KING-714	0.54	34.52987	20.20775	0.112826	2.15208
KING-715	0.15	38.40489	23.27934	0.114867	3.24273
KING-716	-0.12	39.33766	23.55699	0.119383	3.46881
KING-717	-0.30	36.86165	23.82208	0.123271	3.95659
KING-718	-0.60	30.72739	24.52002	0.129505	4.06387
KING-719	-1.05	18.83768	28.23186	0.138661	3.39858
KING-720	-1.32	12.04443	31.13251	0.139758	2.66432
KING-721	-1.49	6.102283	30.08931	0.133634	2.14491
KING-722	-1.80	4.602764	42.56737	0.133123	1.41546

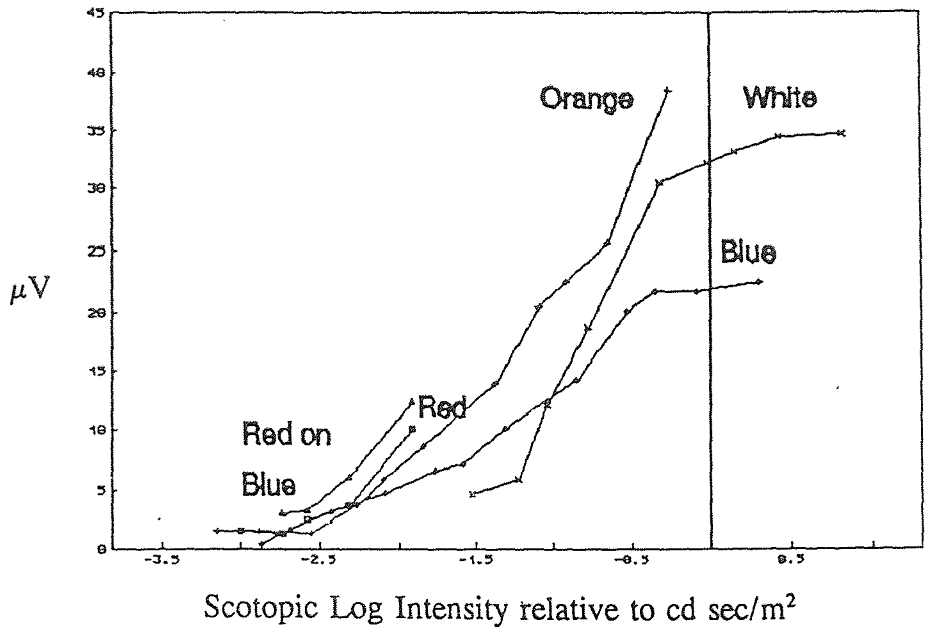


Figure 6.6 Representation of parameter "α" the amplitude.

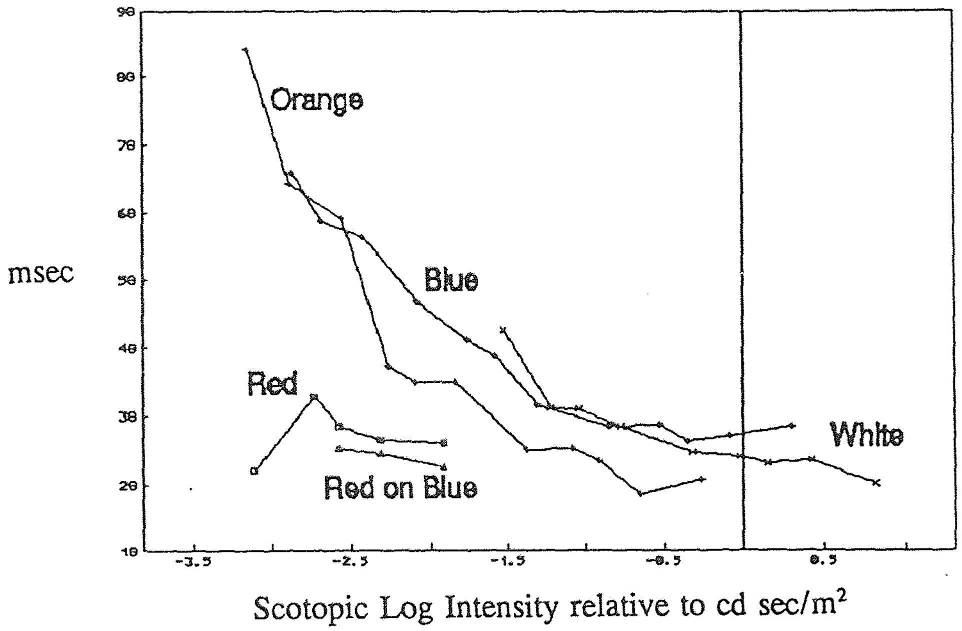


Figure 6.7 Representation of parameter "T_c" the center time.

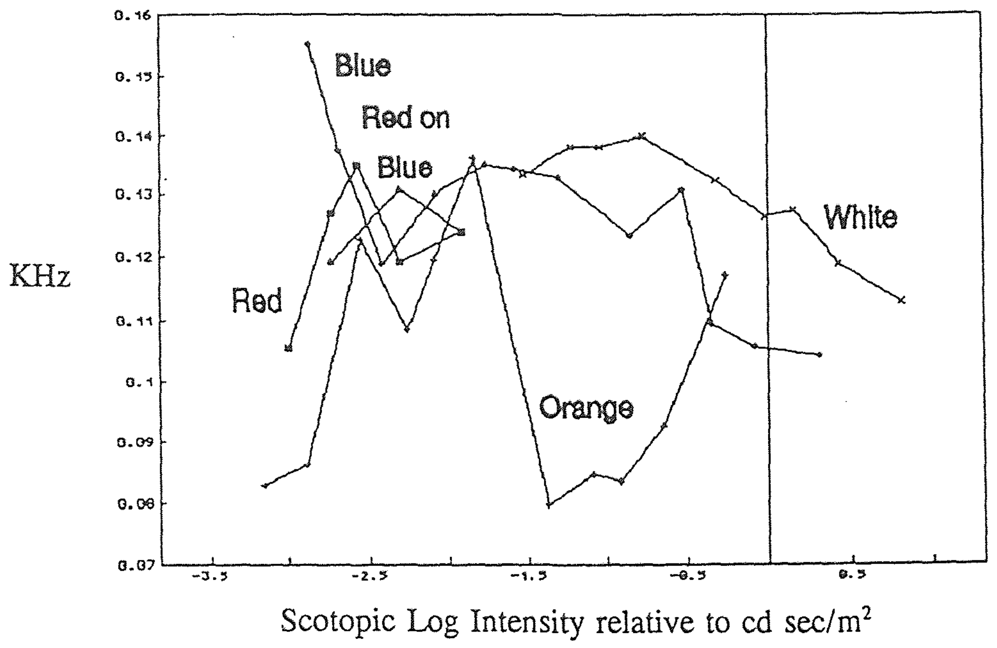


Figure 6.8 Representation of "f_o" the frequency.

CHAPTER 7

CONCLUSION

The new proposed mathematical model of the oscillatory potential provides the compressed information of the oscillatory potential obtained from different color stimuli and their various intensities. From Figure 6.6, we find that the amplitude for red flashes with or without a blue background shows similar increases at high stimulus intensities. White and orange flashes produce higher amplitudes at all intensities, thus demonstrating the presence of rod OP within the signal. In Figure 6.7, we note that parameter T_c, the center time of the OP envelope, changes relatively little for pure cone stimuli with increasing intensities. But T_c (latency) reduces sharply for responses to blue stimuli.

Although the model parameters contain most of the information of the oscillatory potential, their clinical and physiological importance needs to be investigated. This study provides the grounds for future study to determine the behavior of parameters for normal and abnormal subjects when subjected to different stimuli & their intensities.

APPENDIX A

REPRESENTATION OF OP(t) MODEL IN TERMS OF AMPLITUDE "α" AND TIME "T_c" OF ITS OCCURANCE

A generalized Equation of the oscillatory potential is:

$$OP(t) = k_o \cdot (\alpha_o \cdot t)^n \cdot e^{-\alpha_o t} \cdot \sin(2\pi f_o t + \Phi_o) \qquad \text{(A-1)}$$

At any instant given time, in above equation (A-1),

$$m(t) = k_o \cdot (\alpha_o \cdot t)^n \cdot e^{-\alpha_o t} \qquad \text{(A-2)}$$

represents the amplitude of the OP envelope. The maximum value of function m(t) can be found by derivating the equation with respect to time and equaling it to zero.

$$\frac{dm(t)}{dt} = \frac{d}{dt}\left[k_o \cdot (\alpha_o \cdot t)^n \cdot e^{-\alpha_o t} \right] = 0$$

$$k_o \cdot \alpha_o \cdot \left[n \cdot t^{n-1} - \alpha_o \cdot t^n \right] \cdot e^{-\alpha_o t} = 0$$

$$t = n/\alpha_o \qquad \text{and} \qquad \alpha_o = n/t$$

That means at time t = T_c, where $T_c = n/\alpha_o$, the value of envelope will be maximum. Substituting T_c in equation (A-2)

$$m(T_c) = M_{\max} = k_o \cdot \left(\alpha_o \cdot \frac{n}{\alpha_o}\right)^n \cdot e^{-\alpha_o \cdot \frac{n}{\alpha_o}} = \alpha$$

$$M_{\max} = \alpha = k_o \cdot (n / e)^n$$

Conversely,

$$k_o = M_{\max} \cdot (e / n)^n = \alpha \cdot (e / n)^n \tag{A-3}$$

Substituting the values of k_o and α_o in Equation (A-1), final OP model becomes

$$OP(t) = \alpha \cdot (e \cdot \frac{t}{T_c})^n \cdot e^{-n \frac{t}{T_c}} \cdot \sin (2 \pi f_o t + \Phi_o) \tag{A-4}$$

APPENDIX B

OSCILLATORY POTENTIAL RESPONSES AND THEIR MODEL REPRESENTATION

In this appendix, top wave (1) is the Oscillatory Potential and bottom wave (2) is its model representation, whose parameters are shown in Chapter 6.

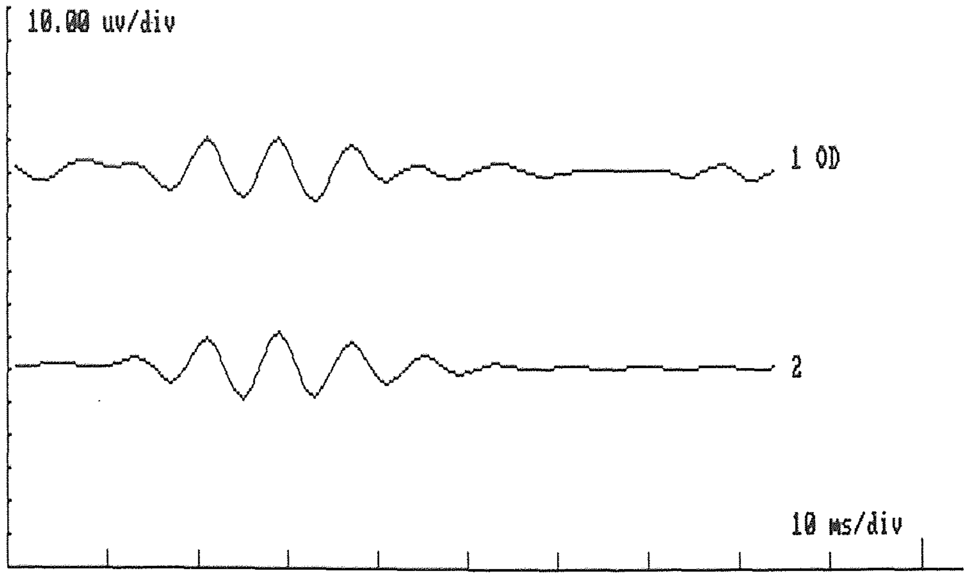


Figure B-1 OP and its model representation for data KING-281, red stimuli.

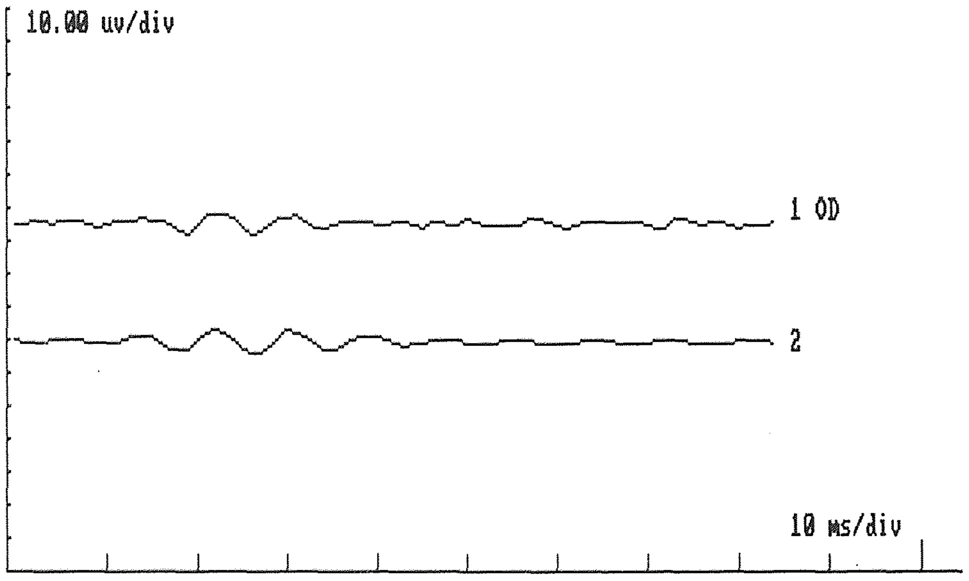


Figure B-2 OP and its model representation for data KING-282, red stimuli.

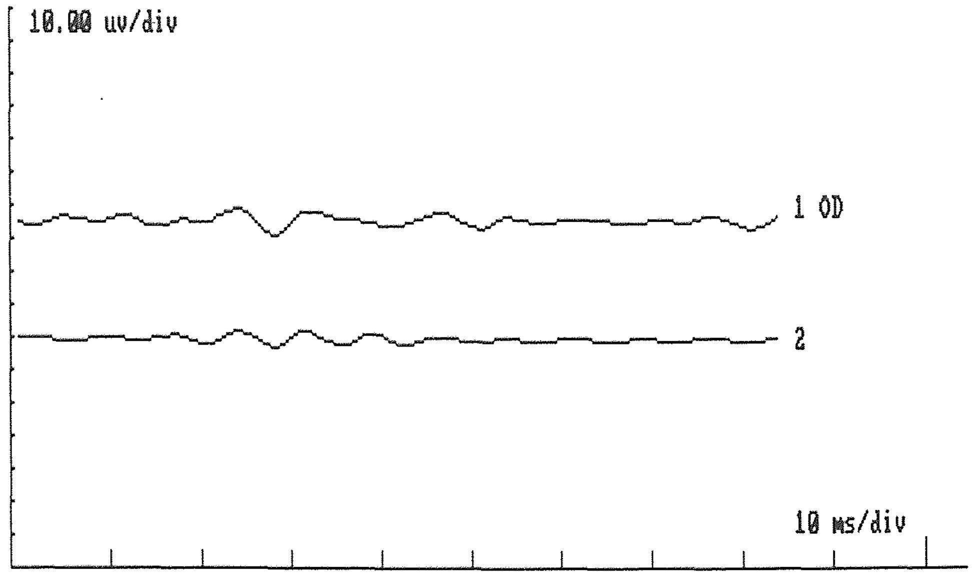


Figure B-3 OP and its model representation for data KING-283, red stimuli.

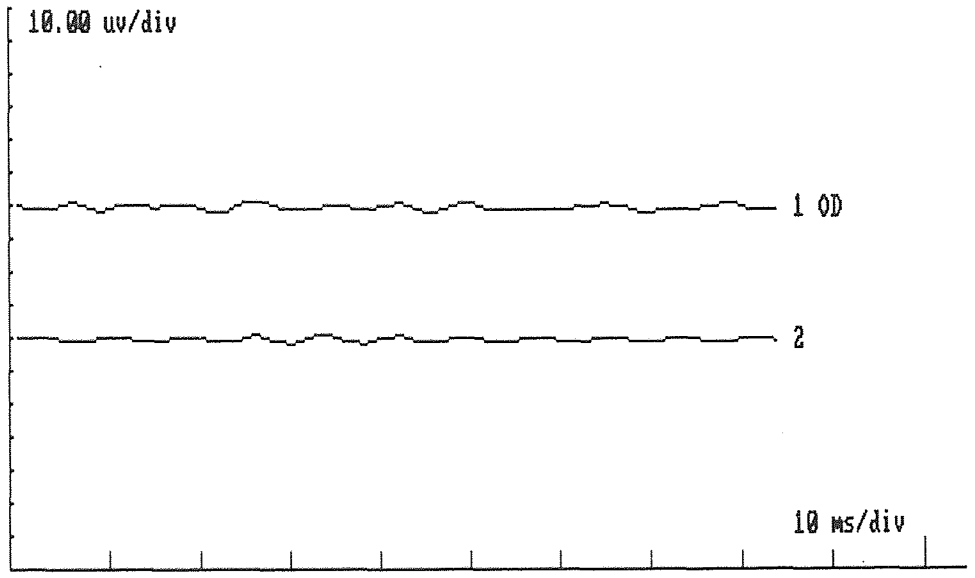


Figure B-4 OP and its model representation for data KING-284, red stimuli.

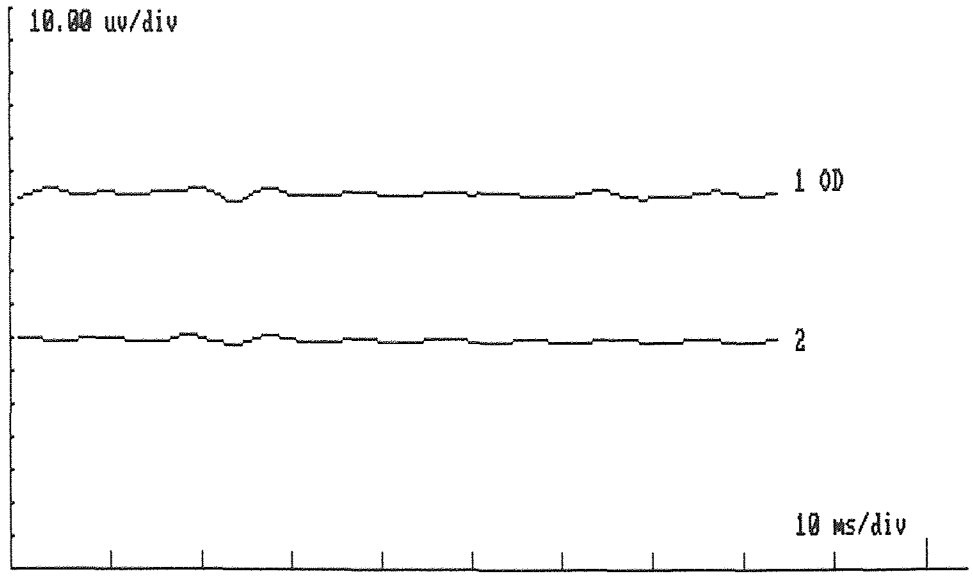


Figure B-5 OP and its model representation for data KING-285, red stimuli.

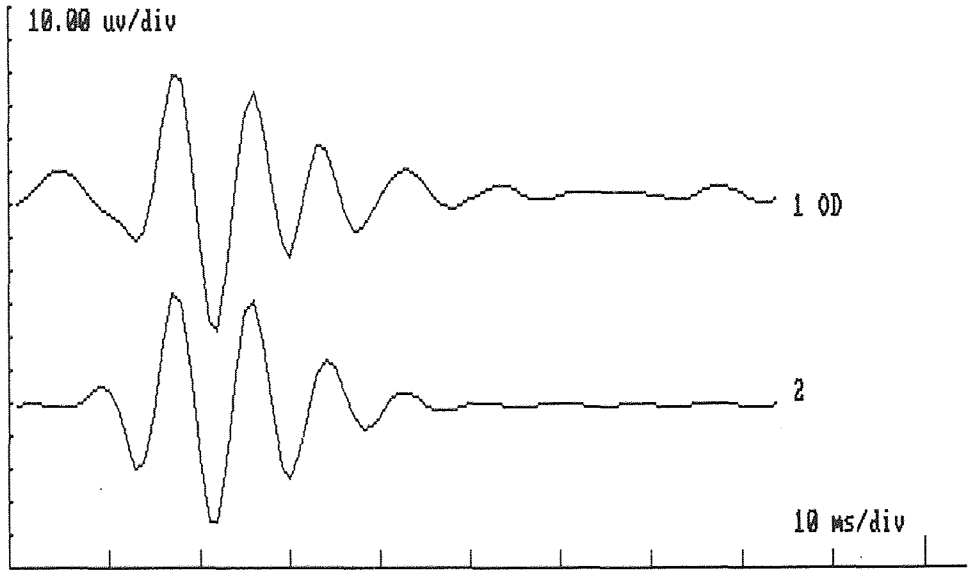


Figure B-6 OP and its model representation for data KING-379, orange stimuli.

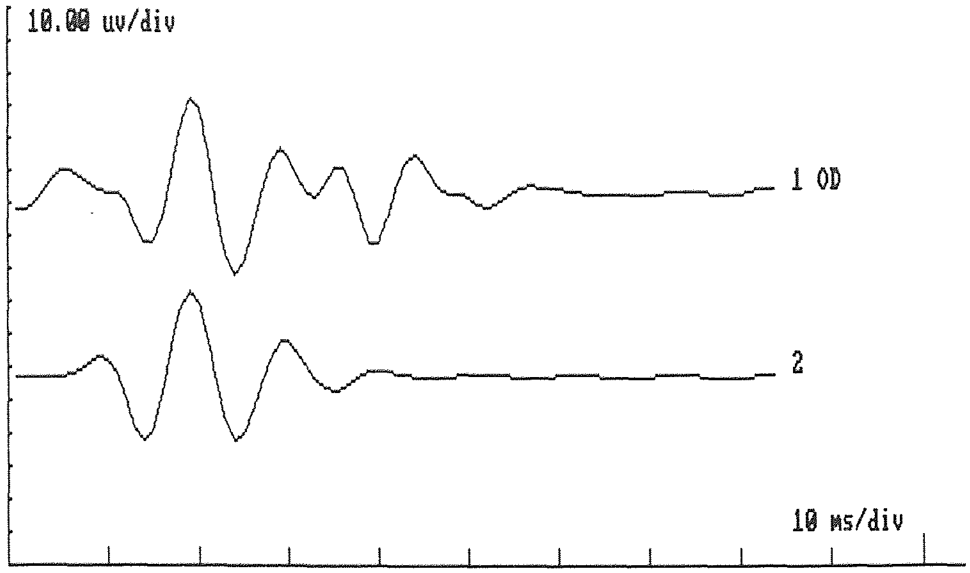


Figure B-7 OP and its model representation for data KING-380, orange stimuli.

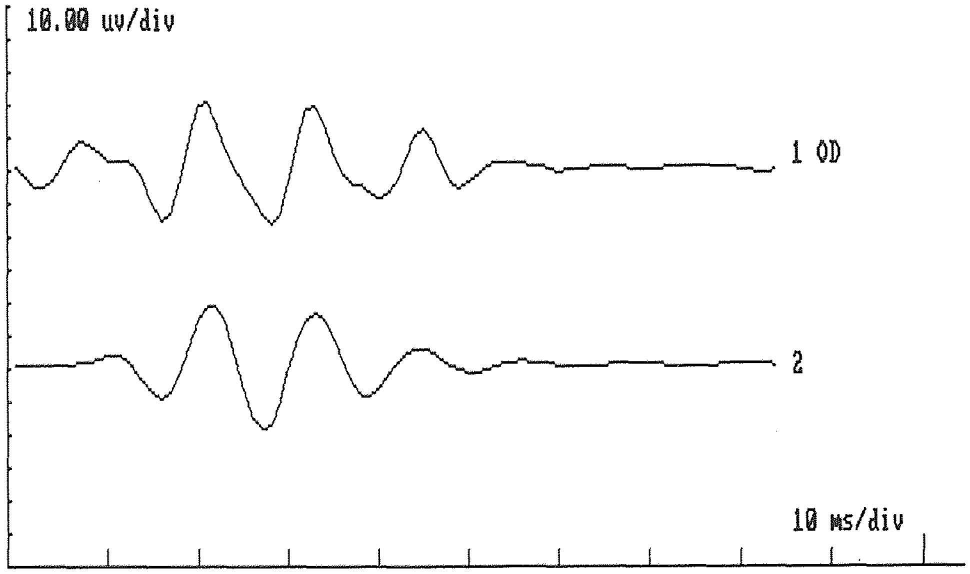


Figure B-8 OP and its model representation for data KING-381, orange stimuli.

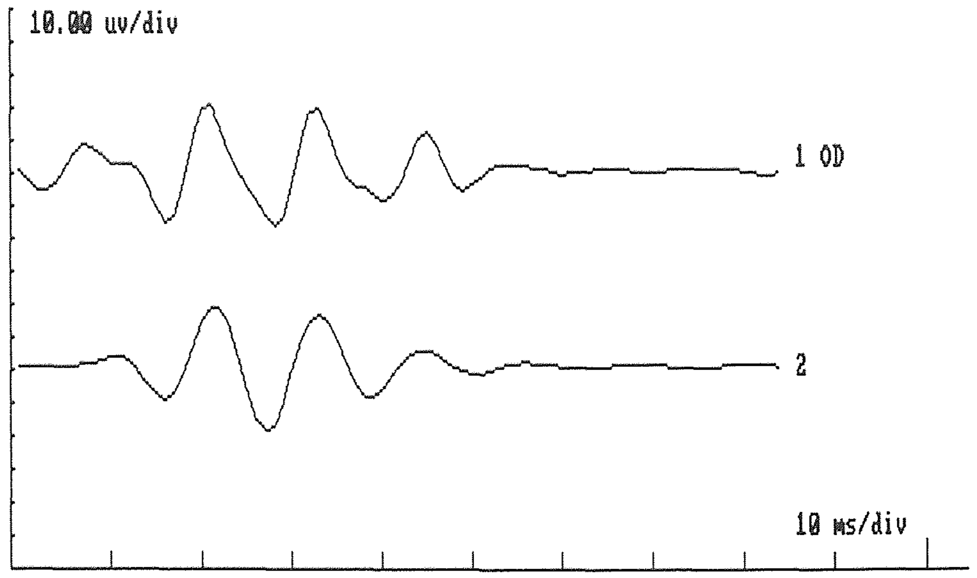


Figure B-9 OP and its model representation for data KING-382, orange stimuli.

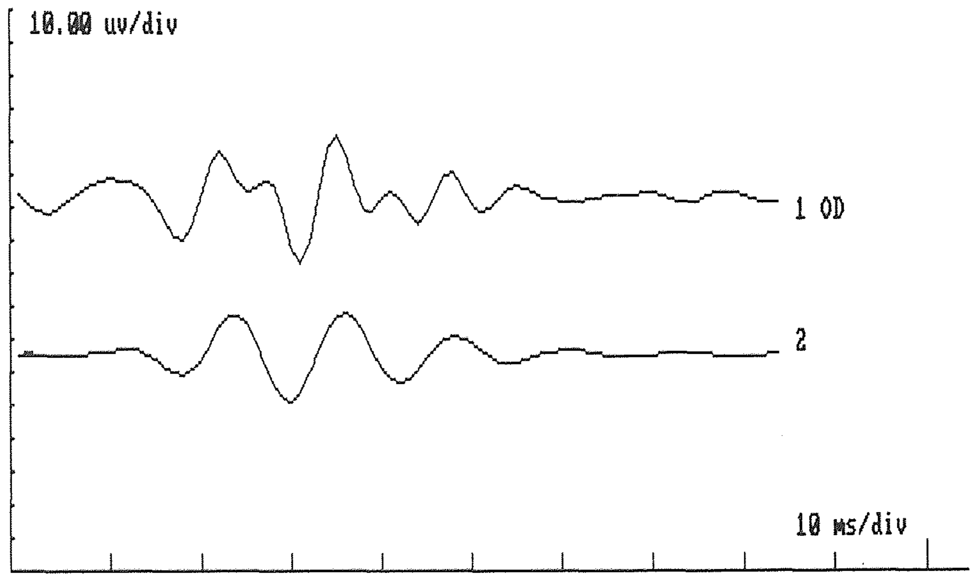


Figure B-10 OP and its model representation for data KING-383, orange stimuli.

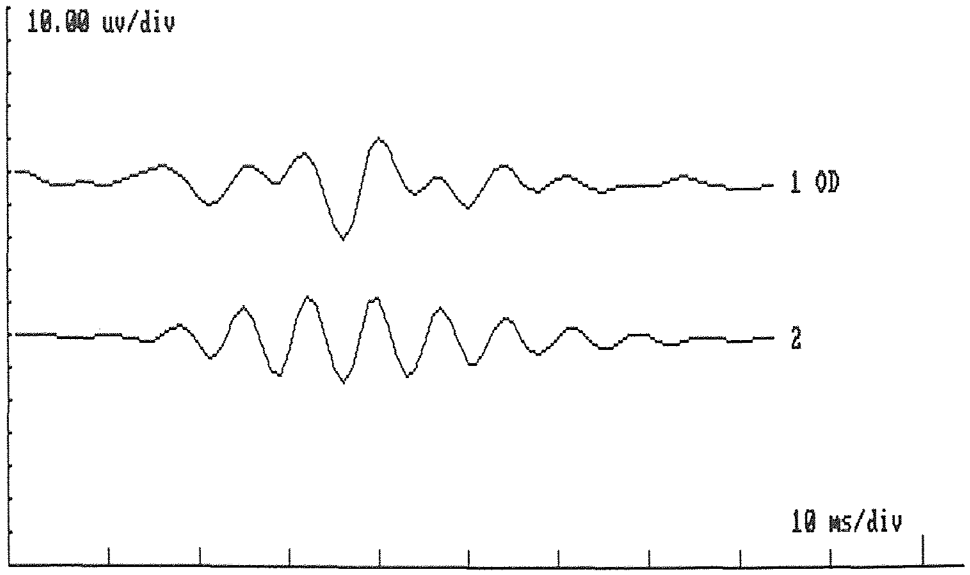


Figure B-11 OP and its model representation for data KING-384, orange stimuli.

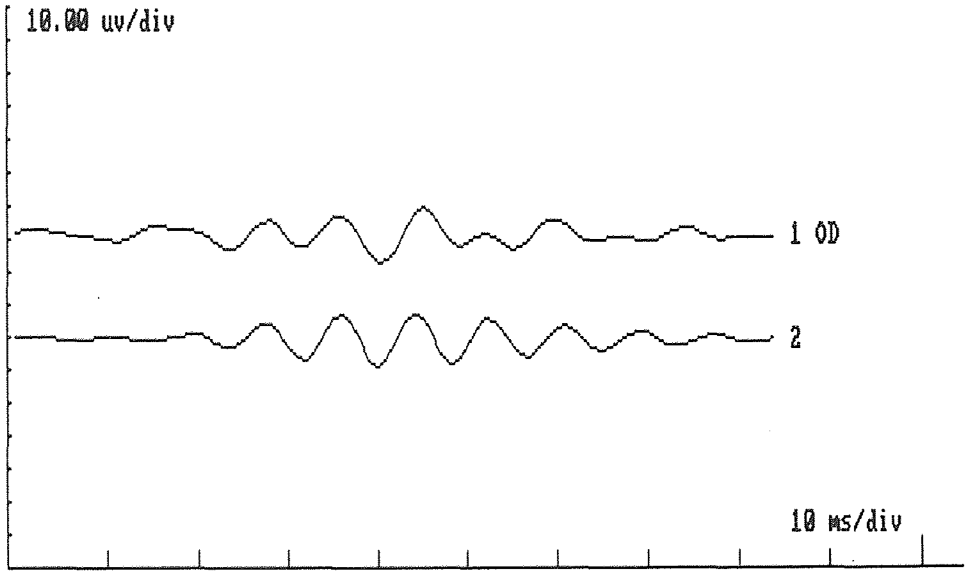


Figure B-12 OP and its model representation for data KING-385, orange stimuli.

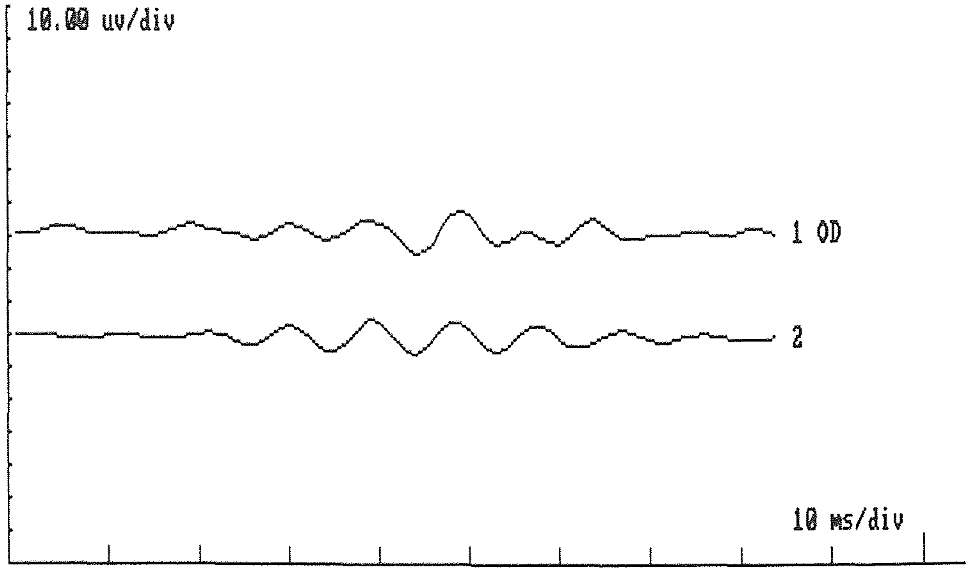


Figure B-13 OP and its model representation for data KING-386, orange stimuli.

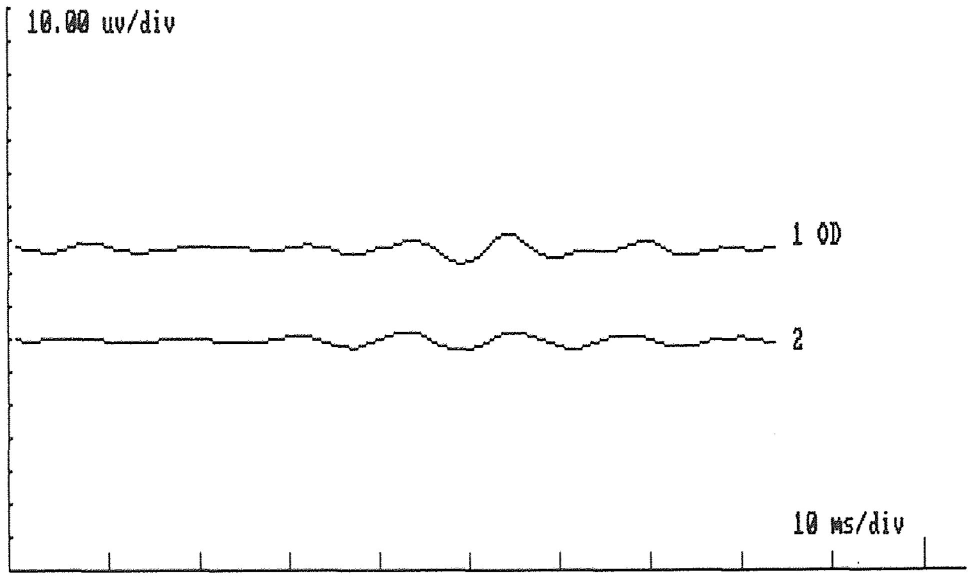


Figure B-14 OP and its model representation for data KING-387, orange stimuli.

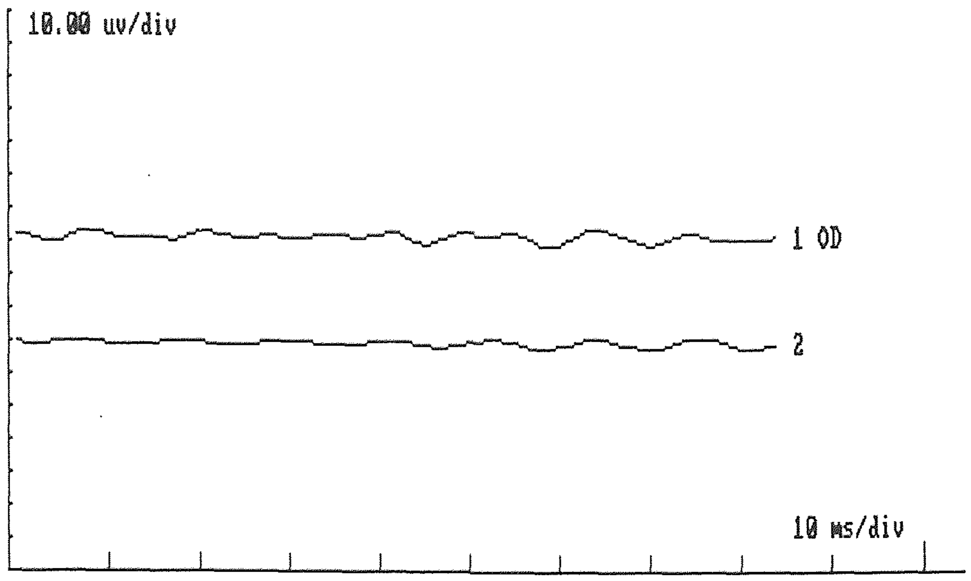


Figure B-15 OP and its model representation for data KING-388, orange stimuli.

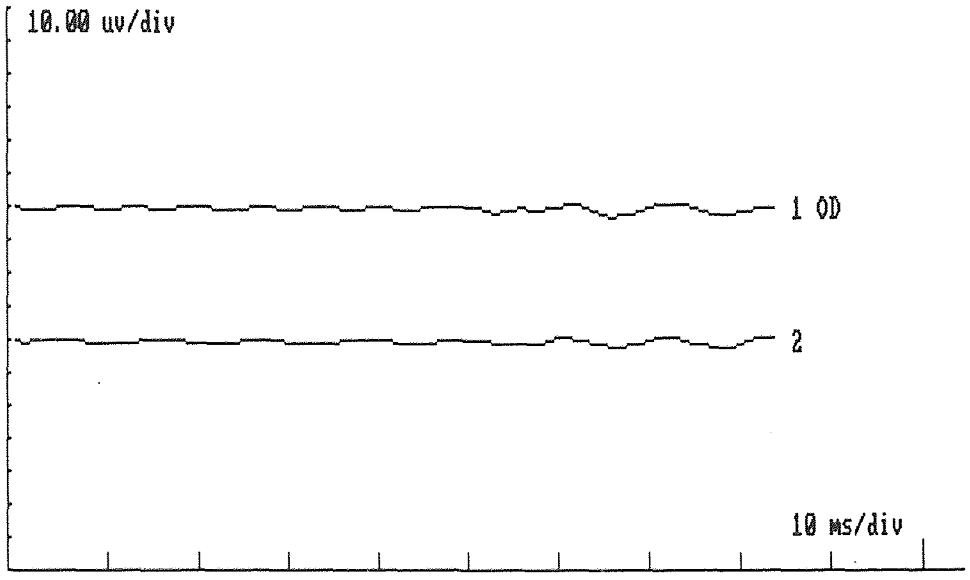


Figure B-16 OP and its model representation for data KING-389, orange stimuli.

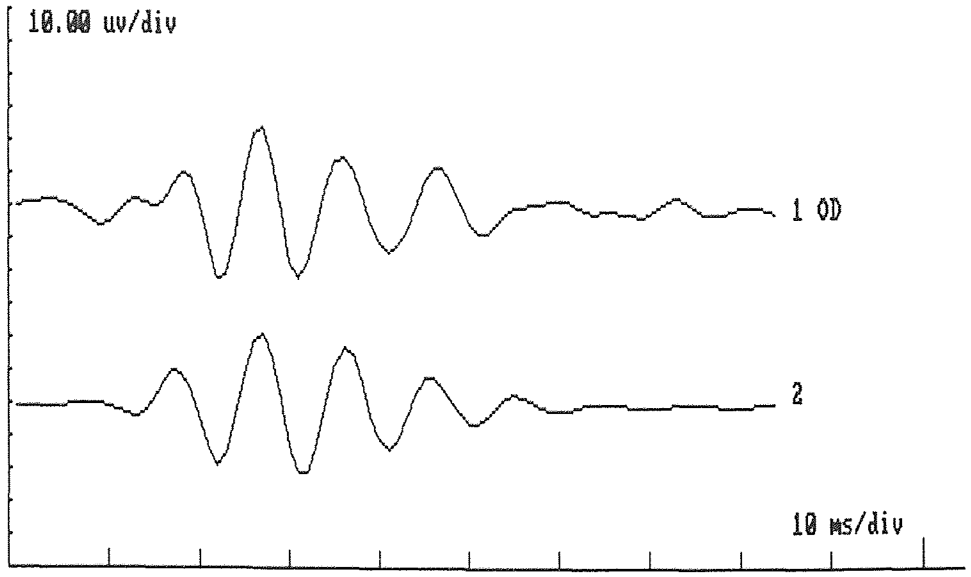


Figure B-17 OP and its model representation for data KING-393, blue stimuli.

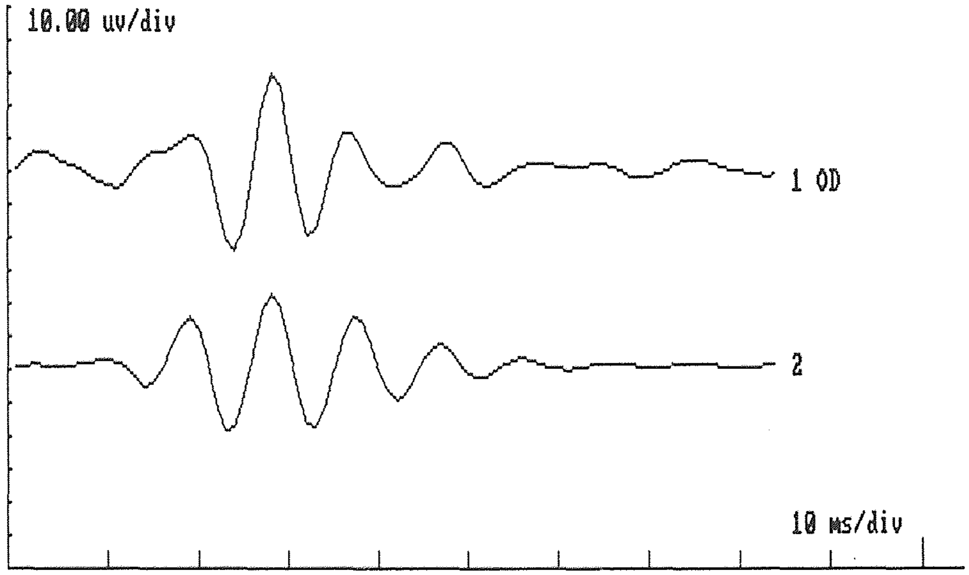


Figure B-18 OP and its model representation for data KING-394, blue stimuli.

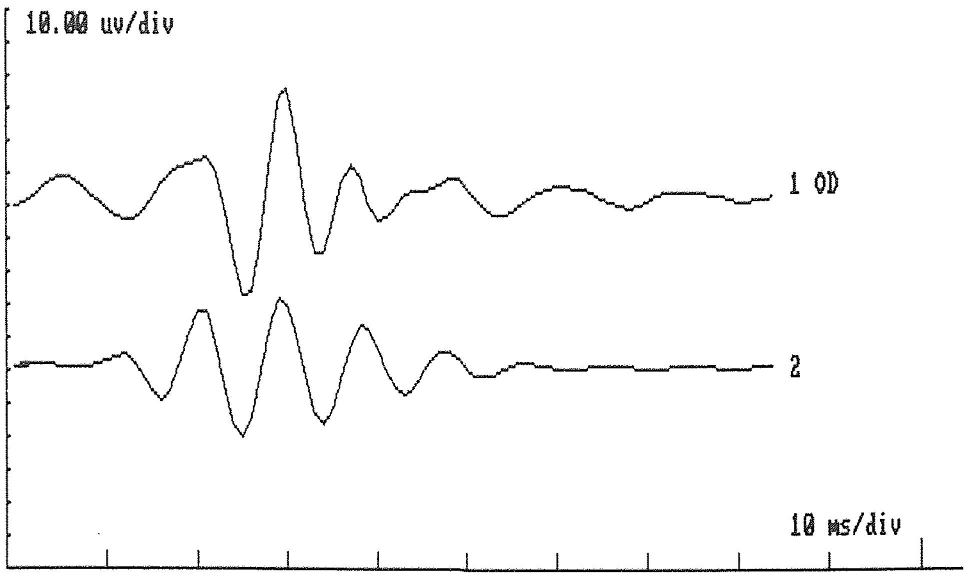


Figure B-19 OP and its model representation for data KING-395, blue stimuli.

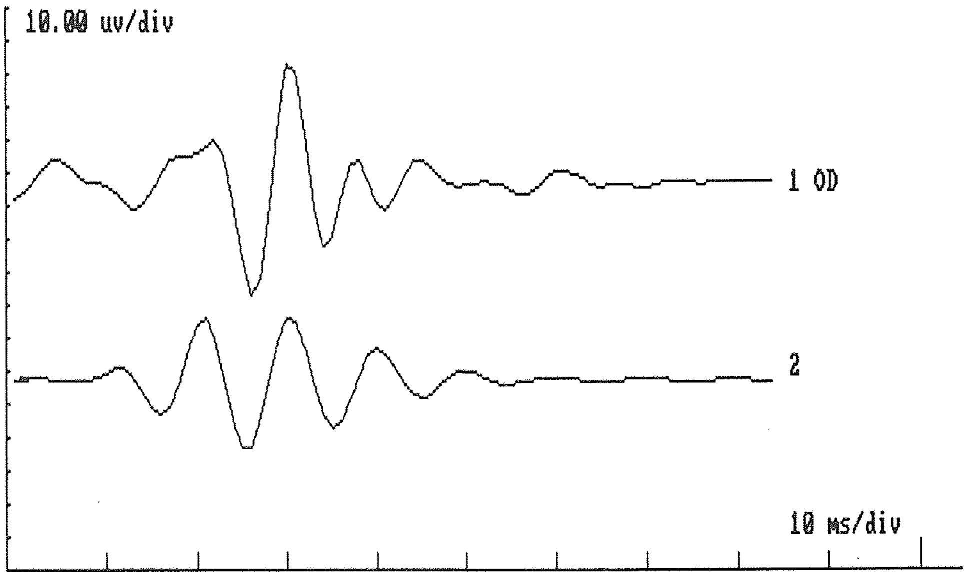


Figure B-20 OP and its model representation for data KING-396, blue stimuli.

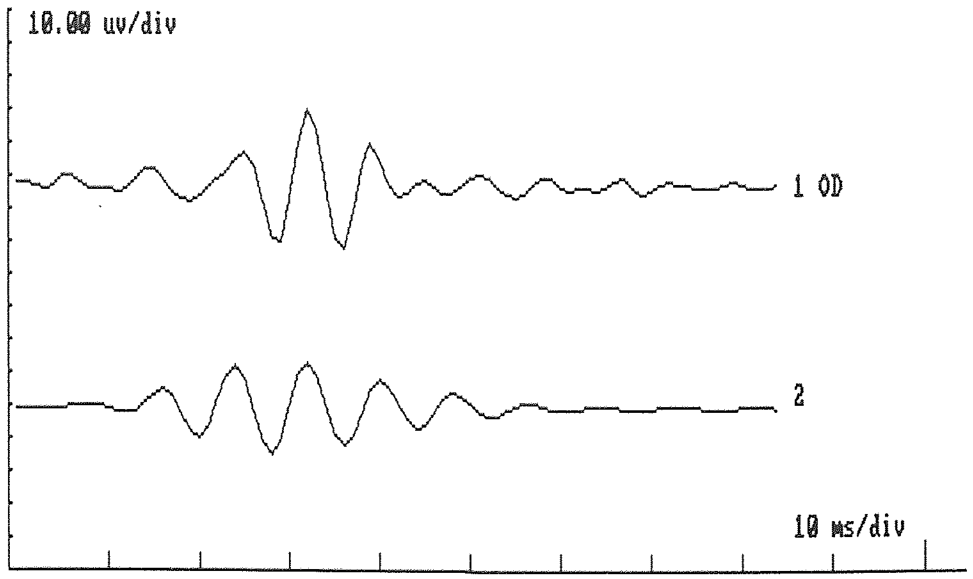


Figure B-21 OP and its model representation for data KING-397, blue stimuli.

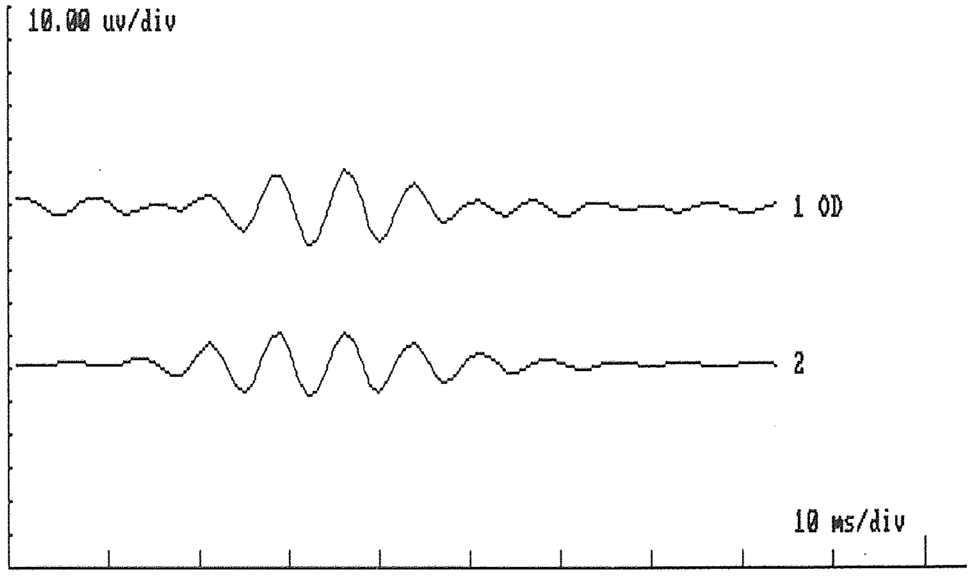


Figure B-22 OP and its model representation for data KING-398, blue stimuli.

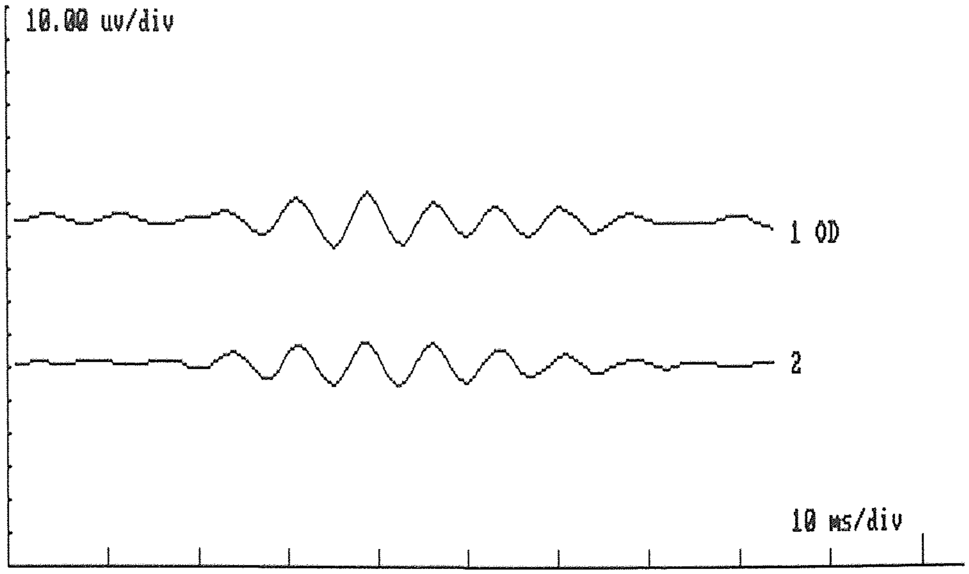


Figure B-23 OP and its model representation for data KING-399, blue stimuli.

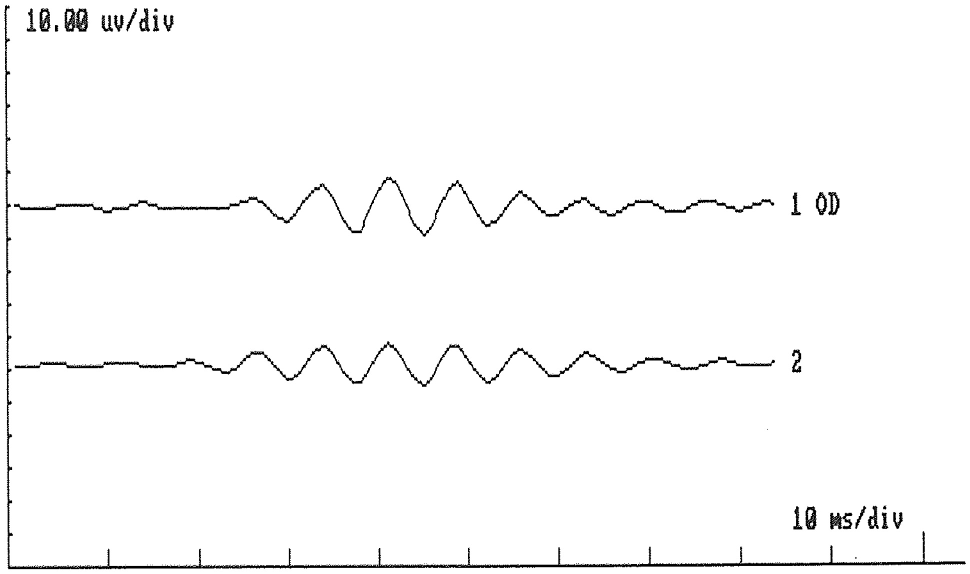


Figure B-24 OP and its model representation for data KING-400, blue stimuli.

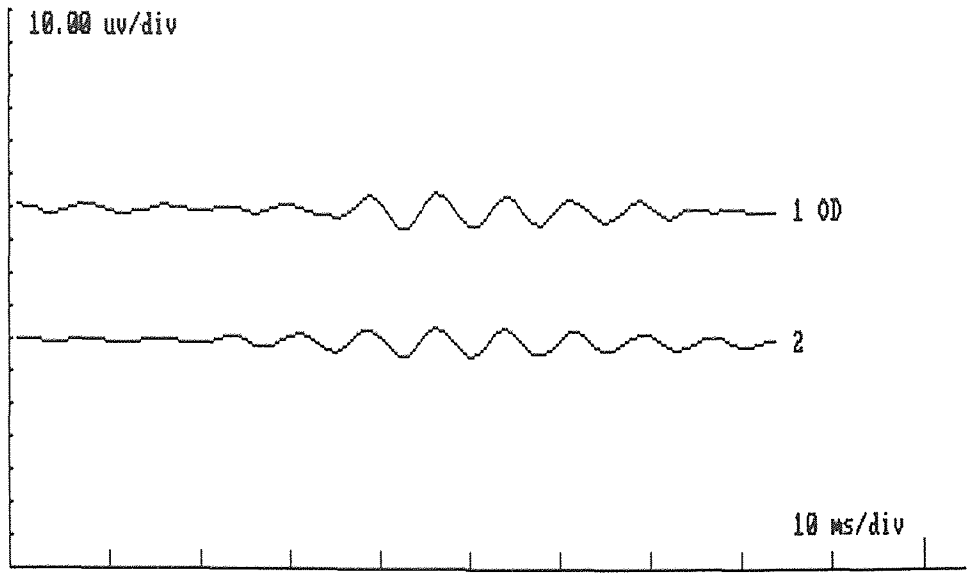


Figure B-25 OP and its model representation for data KING-401, blue stimuli.

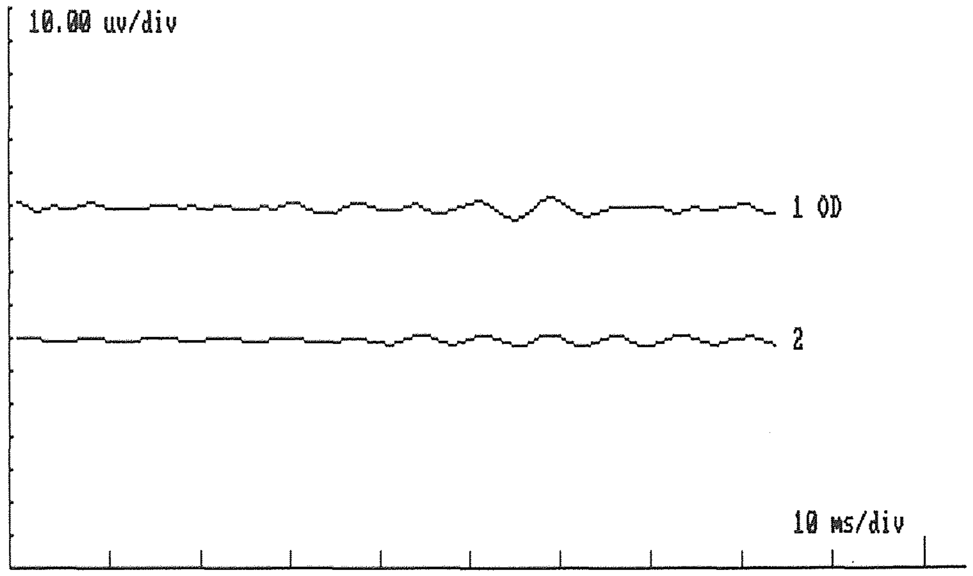


Figure B-26 OP and its model representation for data KING-402, blue stimuli.

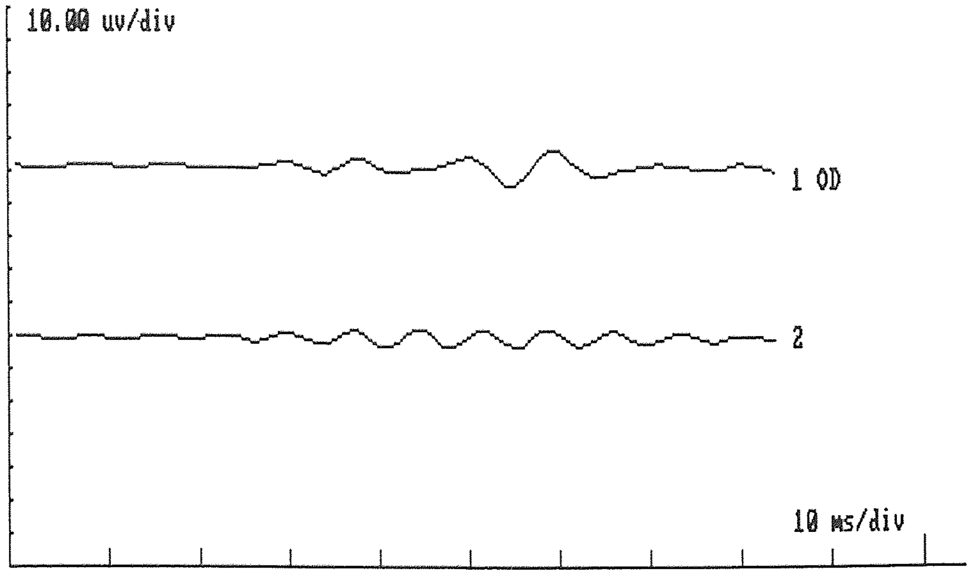


Figure B-27 OP and its model representation for data KING-403, blue stimuli.

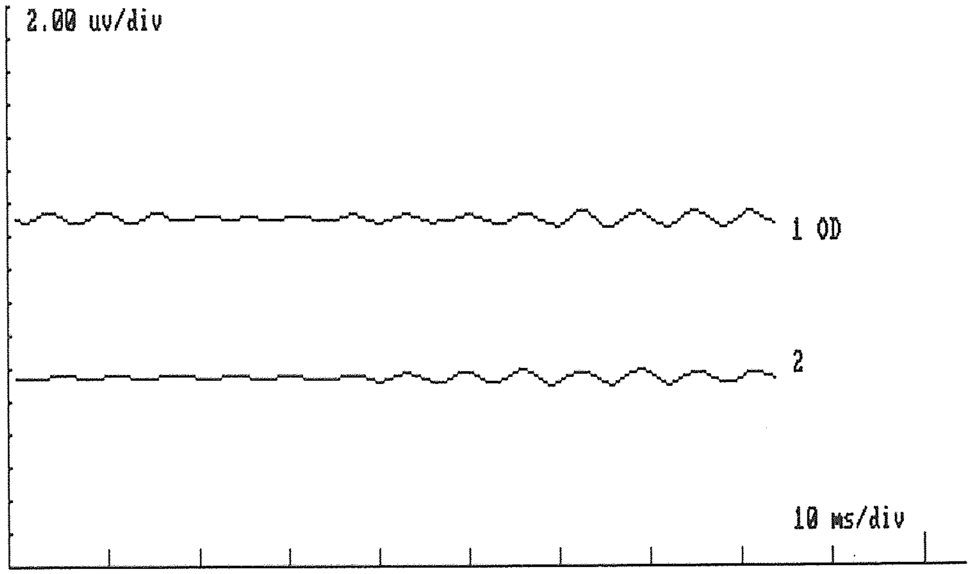


Figure B-28 OP and its model representation for data KING-404, blue stimuli.

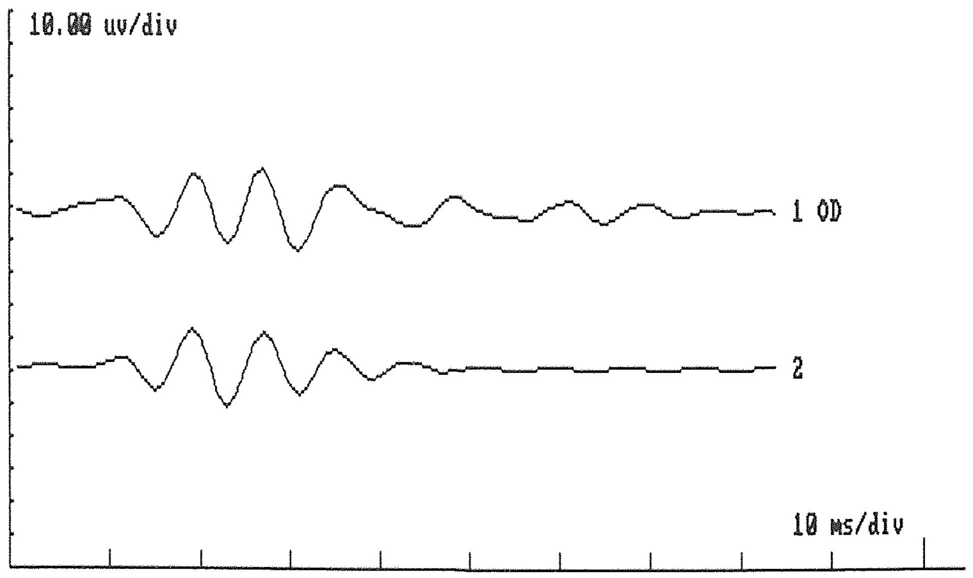


Figure B-29 OP and its model representation for data KING-614, red on blue stimuli.

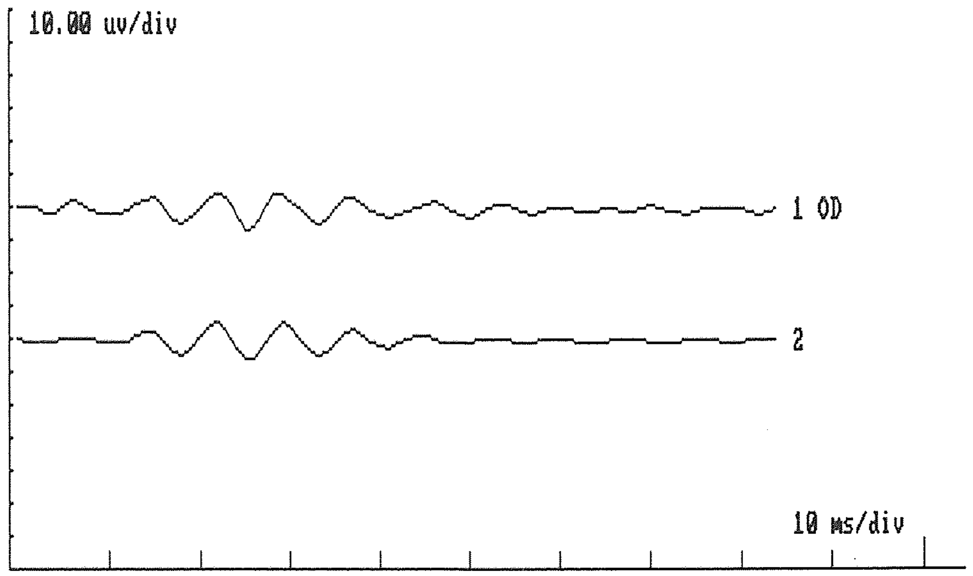


Figure B-30 OP and its model representation for data KING-615, red on blue stimuli.

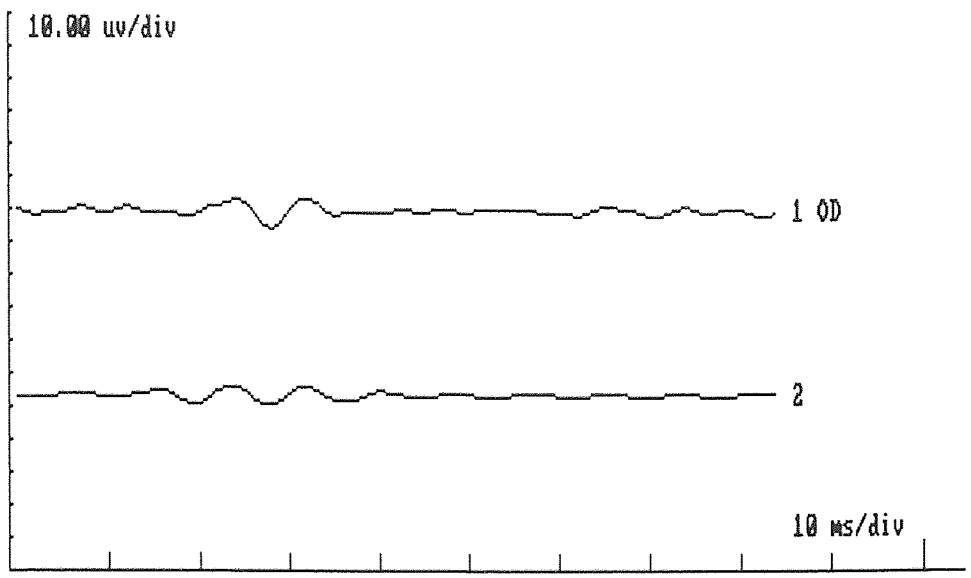


Figure B-31 OP and its model representation for data KING-617, red on blue stimuli.

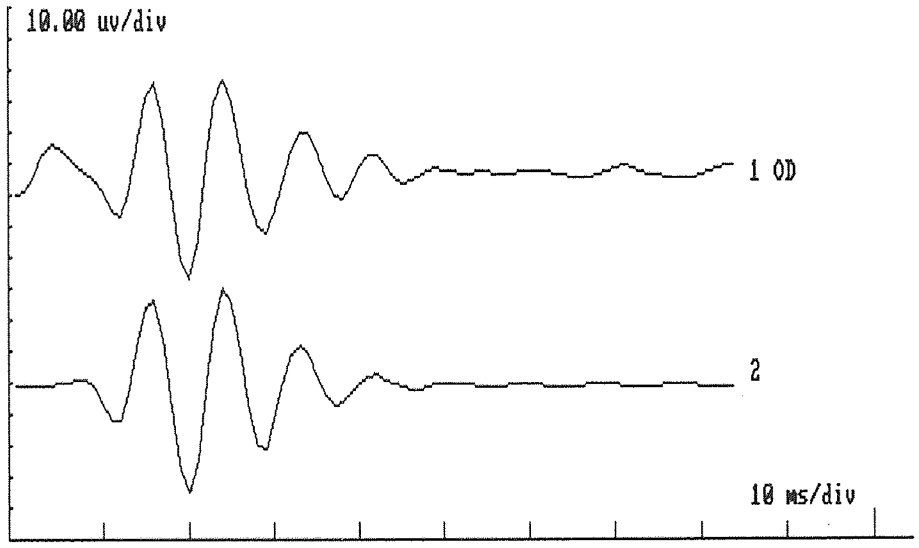


Figure B-32 OP and its model representation for data KING-714, white stimuli.

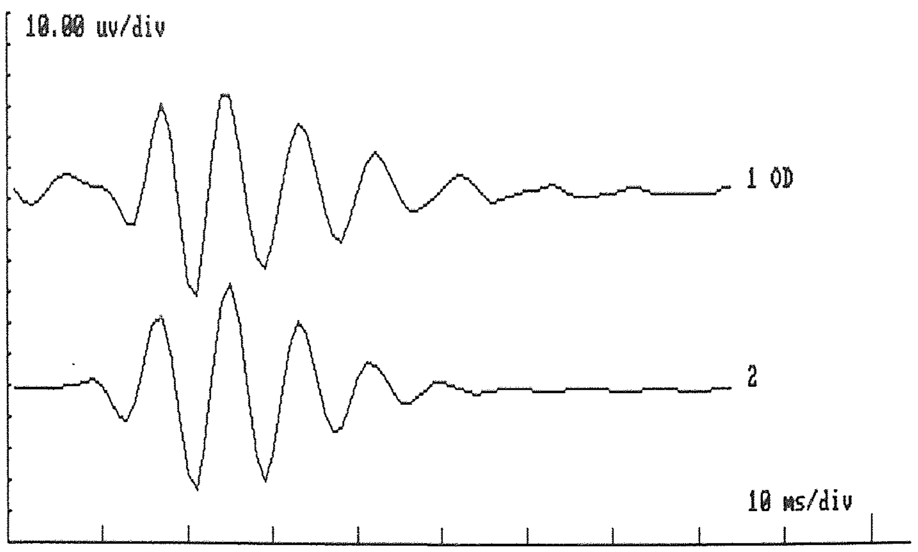


Figure B-33 OP and its model representation for data KING-715, white stimuli.

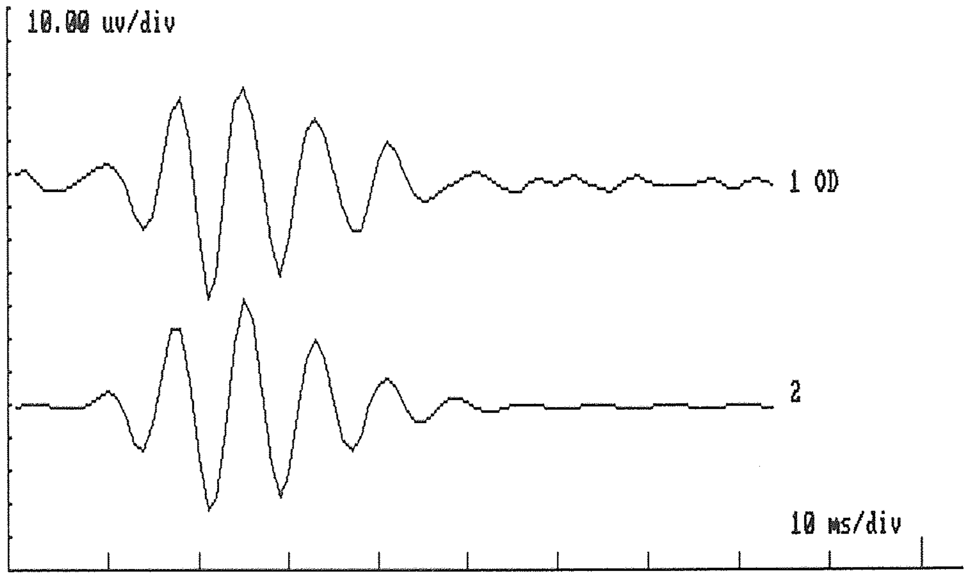


Figure B-34 OP and its model representation for data KING-716, white stimuli.

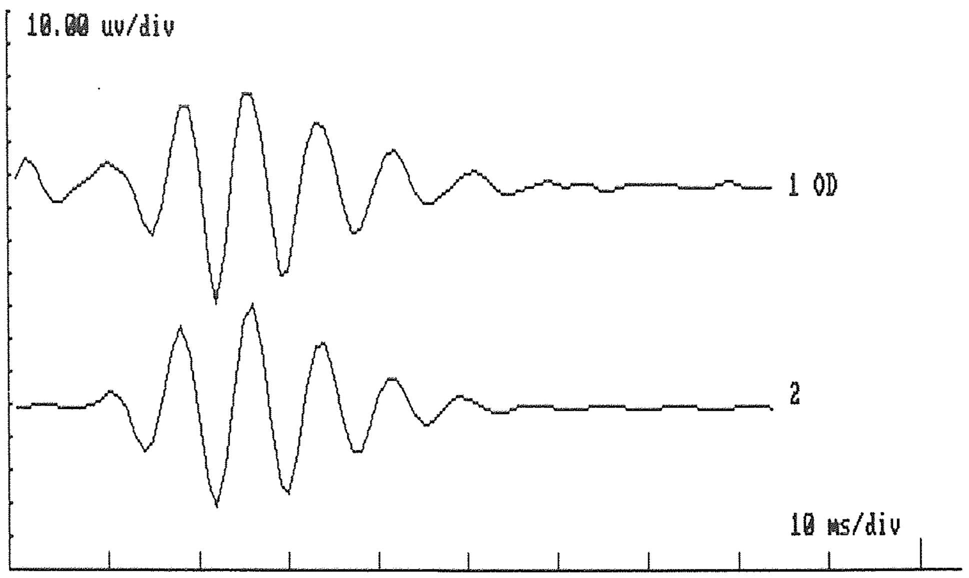


Figure B-35 OP and its model representation for data KING-717, white stimuli.

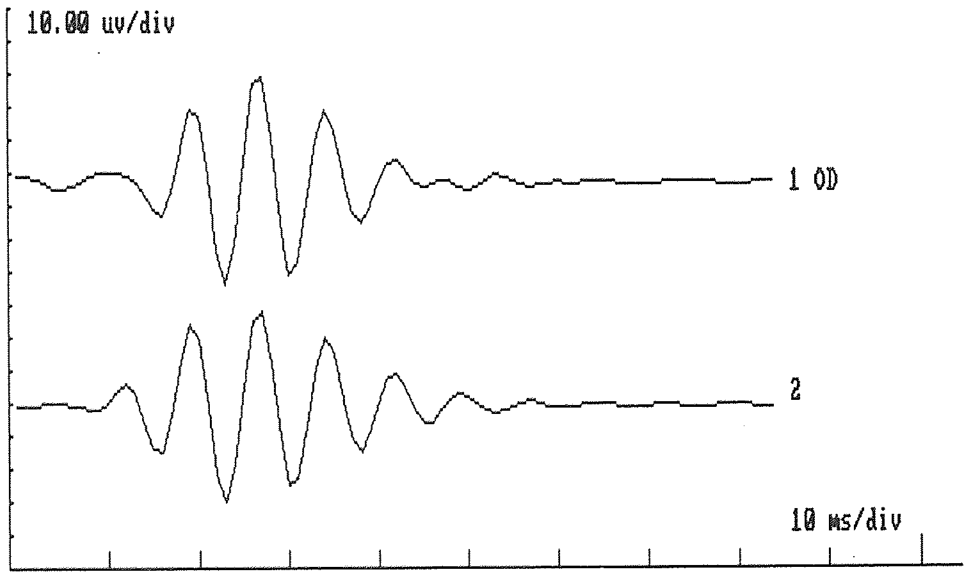


Figure B-36 OP and its model representation for data KING-718, white stimuli.

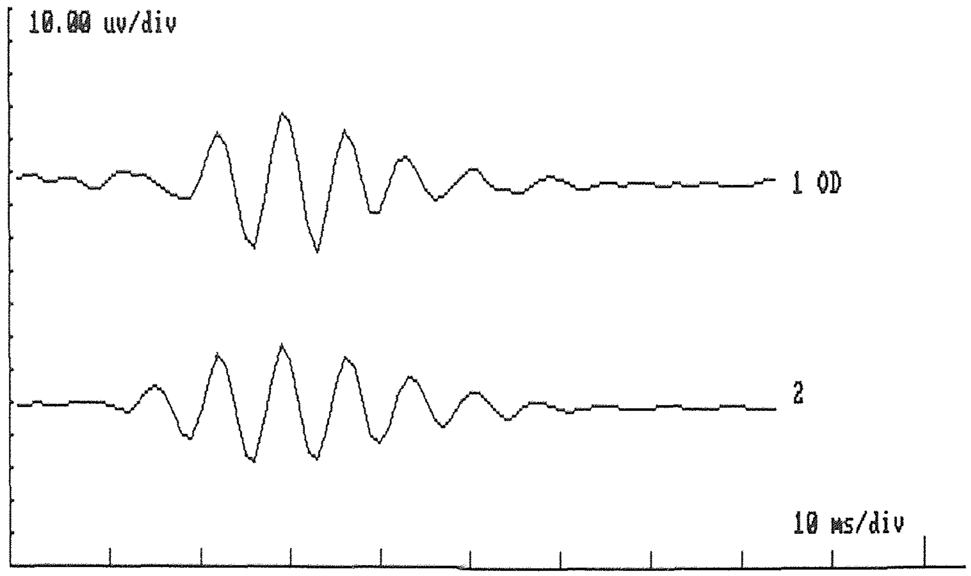


Figure B-37 OP and its model representation for data KING-719, white stimuli.

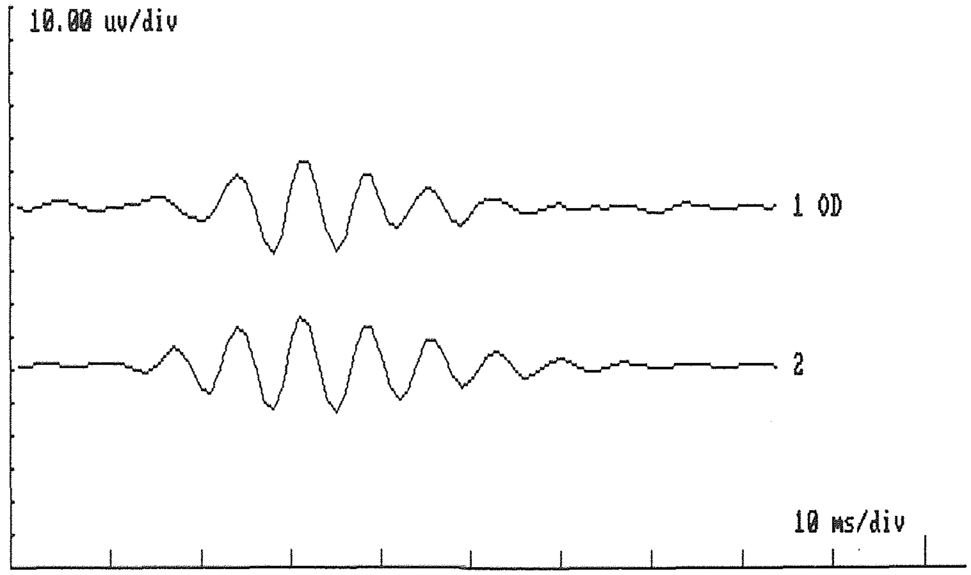


Figure B-38 OP and its model representation for data KING-720, white stimuli.

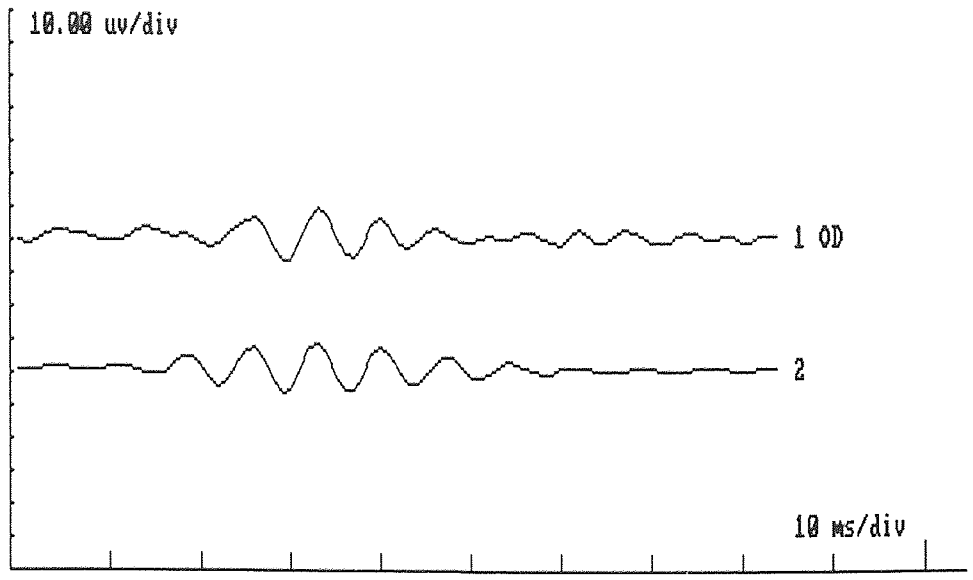


Figure B-39 OP and its model representation for data KING-721, white stimuli.

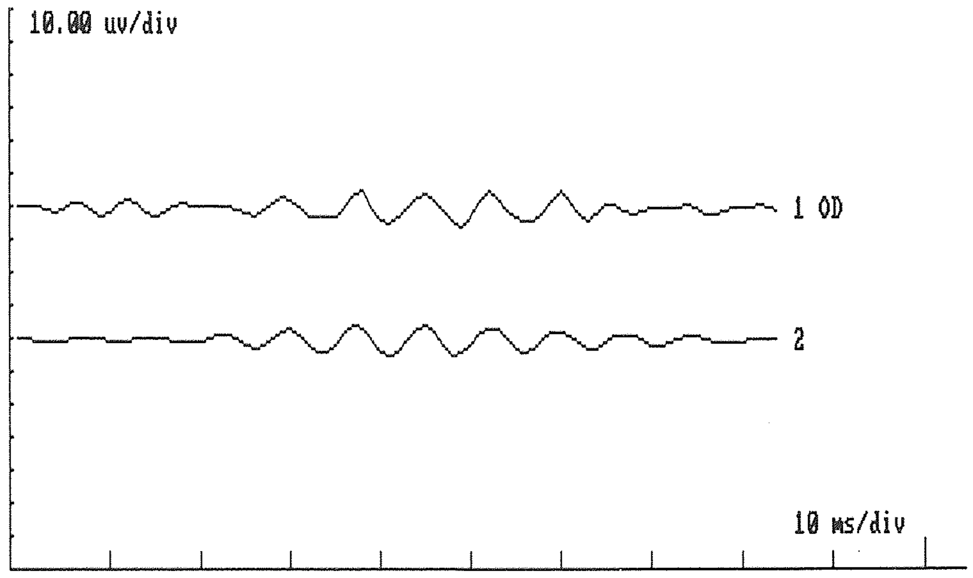


Figure B-40 OP and its model representation for data KING-722, white stimuli.

APPENDIX C

PARAMETERS OF THE OSCILLATORY POTENTIALS

Table C-1 Parameters of the OPs when the ERGs treated individually and filtered.

EXP	Log Intensity	Amplitude	Center Time	Freq	Std. Err
Red Stimuli	cd-s/m^2	α [μV]	Tc [msec]	fo [KHz]	
KING-281	-0.51	10.15662	26.23996	0.124023	1.05689
KING-282	-0.91	3.805809	26.66327	0.119227	0.72639
KING-283	-1.17	2.443827	28.62636	0.134688	0.92821
KING-284	-1.33	1.3459	32.75505	0.126775	0.50131
KING-285	-1.6	1.54852	22.1825	0.10563	0.42331
Orange	Stimuli				
KING-379	0.23	38.54199	20.87837	0.117155	2.98617
KING-380	-0.16	25.56456	18.67231	0.09262	3.80203
KING-381	-0.42	22.5413	23.64788	0.083607	2.45521
KING-382	-0.59	20.41151	25.35355	0.084757	2.56957
KING-383	-0.88	13.87424	25.2053	0.079743	3.30466
KING-384	-1.34	8.638502	35.09233	0.136063	2.61895
KING-385	-1.6	5.897897	35.08204	0.119436	1.58609
KING-386	-1.77	-3.71341	37.48687	0.108649	1.32157
KING-387	-2.06	-1.32855	59.27989	0.122791	1.07853
KING-388	-2.39	1.59034	64.23165	0.086315	0.66095
KING-389	-2.66	1.572536	84.16495	0.0829	0.32864
Blue	Stimuli				
KING-393	-0.53	22.52382	28.5035	0.104291	2.28919
KING-394	-0.92	21.67401	27.22725	0.10582	2.75106
KING-395	-1.18	21.73227	26.44661	0.109488	4.42903
KING-396	-1.37	20.15051	28.87631	0.1307	2.43543
KING-397	-1.69	14.21104	28.62064	0.123485	2.88002
KING-398	-2.14	10.14268	31.62804	0.132746	1.40307
KING-399	-2.41	7.209031	38.81053	0.134325	1.09234

Table C-1 (Continue...)

KING-400	-2.6	6.548103	41.1276	0.135075	0.9695
KING-401	-2.92	4.720441	46.71978	0.130042	0.85751
KING-402	-3.26	3.223296	56.41539	0.118822	0.90975
KING-403	-3.52	1.747989	58.82985	0.137389	0.75872
KING-404	-3.71	0.447804	65.83806	0.155304	0.26417
Red on	Blue	Stimuli			
KING-614	-0.51	12.36579	22.61574	0.123865	1.87272
KING-615	-0.91	6.092481	24.61112	0.131017	0.72552
KING-617	-1.17	3.056713	25.3675	0.119288	0.73454
White	Stimuli				
KING-714	0.54	34.71385	20.20144	0.112907	2.2004
KING-715	0.15	34.44234	23.54409	0.11895	2.72011
KING-716	-0.12	33.1679	23.29385	0.127197	2.37549
KING-717	-0.3	32.21531	24.18124	0.126304	2.64374
KING-718	-0.6	30.49593	24.75387	0.132277	1.89628
KING-719	-1.05	18.79569	28.50713	0.140091	1.95565
KING-720	-1.32	12.06275	31.06252	0.13818	1.37795
KING-721	-1.49	5.963615	31.25409	0.138209	1.45838
KING-722	-1.8	4.637379	42.6942	0.133193	1.08879

APPENDIX D

PARAMETER IDENTIFICATION PROGRAM

Main Program

```
#include <stdio.h>              /* ERG curve fitting    */
#include <math.h>                /* building simplex method      */
#include <conio.h>               /* Modified too many damn ways */
#include <stdlib.h>             /* for Banker master thesis      */
#include <string.h>
#include <io.h>
#include <dos.h>
#include <memory.h>

extern double pmtr[15];
extern double initpmtr[15];
extern unsigned char lbf[15][3];
extern unsigned char unt[15][8];
extern double dbint;
extern double pi;
extern int numprmts,NN;
extern int prnt();

int np;                    /* number of data point       */
int npm;        /* last number of data point to be filtered */
int ns=1;       /* first number of data point to be filtered */

int ntst,ntk,csr[2][2]={20,100,20,100};
int page, swch, rolloffp;
char pathn[100],datafile[15],filt_flag;
float lowf[5],highf[5];
float data[8][532],sen=30.;
float lpf[5]={500.,500.,500.,500.,500.0},hpf[5]={1.,1.,1.,1.,1.};
unsigned char chn=1,file_flag[5]={0,0,0,0,0},tdate[5][15],hauptdate[15];
unsigned char f_name[15],l_name[15],pfnm[40][15];
unsigned char ttype[5][5],bdate[10],sex[3],fflag[3],label[5][5];
unsigned char dgns[2][80],cmmts[3][80],tname[80][15],adrs[2][20];
unsigned char phone[15],filename[15],tst[15],eye[5][3];
struct STIMULI{
                        unsigned char frqcy[10];
                        unsigned char ptn[40];
                        unsigned char cntrst[10];
                        } sti[5];


main()
{
        void head(),position(),cls();
        char key;
        int n;

        head();
        cls(0,24);
        header();
        position(25,2);
        printf("Curve Fitting Program");
        report();
        cls(0,24);
}


report()
{
        char c;
        int s;
        void waitsec(),cls(),position();

         cls(10,24);
        position(0,10);
        printf("Insert data disk into drive A");
         strcpy(pathn,"a:");
        printf("\n\nPress [ENTER] key to continue.");
        while(getchar()!='\n')
                ;
b:       chn=1;
        if(list()==1)
                goto end;
n:       if(list1()==1){
                page=0;
                goto b;
                }
        chn=2; /*chn=4;*/
f:      fit();
        while((s=plot(2))>0){
                switch(s){
                        case 1:
                                fflag[0]=fflag[1]=fflag[2]=0;
                                goto b;
                                break;
                        case 2:
                                fflag[0]=fflag[1]=fflag[2]=0;
                                goto f;
                        break;
                case 3:
                        fflag[0]=fflag[1]=fflag[2]=0;
                        chn=1;
                        goto n;
                        break;
                }
        }
end:
        ntst=ntk=0;
        for(s=0;s<60;++s)
                strset(tname[s],'\0');
}



double a[2][5],a1,b1,tem[532];

          /*      Buterworth filter table       */

        float  b[8][4]={1.4142,0,0,0,1.,0,0,0,0.7653,1.8477,0,0,0.618,1.618,
               0,0,0.5176,1.4142,1.9318,0,0.445,1.247,1.8019,0,0.3902,1.1111,
               1.6629,1.9616,0.3473,1.,1.5321,1.8794};


filter()
{
        double          wc,B,k;
        char    key;
        int     n,m,i,j,c1=2000;

        g_clean(95,99);
        position(5,24);
        filt_flag = 'f';
        printf("Which waveform ?");
        while(((key=getch())>(chn+0x30))||(key<49));
        swch=key-49;
r:      g_clean(95,99);
        position(5,24);
        printf("Which order of filter? (enter EVEN number 0--18)");
        scanf("%d",&n);
        rolloffp=n;
/*      while(getchar()!='\n');                 */
        if(n&0x01!=0)
        goto r;               /*  get even order number  */
        k=1.0/n;
        n=n/2;
        if (n&0x01!=0){       /*   first order filter required    */
        m=1;
        n=(n-1)/2;
        }
        else{                 /*   first order filter not required   */
        m=0;
        n=n/2;
        }
/* Banker OP code insertion starts */
        g_clean(95,99);
        position(5,24);
er1:    printf("%d total datapoints. Last data point to filter?",np);
        scanf("%d",&npm);
        if(npm>np){
                goto er1; }
        if(npm<1){
                goto er1; }
        g_clean(95,99);
        position(5,24);
er2:    printf("%d total datapoints. First data point to filter?",np);
        scanf("%d",&ns);
        if(ns>npm){
                goto er2; }
        if(ns<1){
                goto er2; }
/*      while(getchar()!='\n');         */
/* Banker OP code insertion ends */
        g_clean(95,99);
        position(5,24);
        printf("Enter low cutoff frequency (0--500 Hz):");
        scanf("%f",&lowf[swch]);
        while(getchar()!='\n');
        g_clean(95,99);
        position(5,24);
        printf("Enter high cutoff frequency (0--500 Hz):");
        scanf("%f",&highf[swch]);
        while(getchar()!='\n');
        if(fflag[swch]==0) {
                for(i=ns-1;i<np+30;++i){
                        data[4+swch][i]=data[swch][i];
                }
        }
        else {
                for (i=ns-1;i<np+30;++i){
                        data[swch][i]=data[swch+4][i];
                }
        }


        a1=pi*highf[swch]/1000;
        a1=tan(a1)*c1;
        b1=pi*lowf[swch]/1000;
        b1=tan(b1)*c1;
        wc=a1*b1;
        B=a1-b1;

            /*    second order filter parameter determination    */

        a1=1.0/pow(0.4142,k);
        b1=a1*B/c1;
        i=2*n+m-2;

    a[0][4]=1.0;
    a[0][2]=(2*wc+a1*a1*B*B)/c1/c1;
    a[0][0]=wc/c1/c1;
    a[0][0]=a[0][0]*a[0][0];

    for (j=0;j<n;j++){
    a[0][3]=b[i][j]*b1;
    a[0][1]=a[0][3]*wc/c1/c1;
    a[1][4]=a[0][4]+a[0][3]+a[0][2]+a[0][1]+a[0][0];
    a[1][3]=2*(a[0][1]+2*a[0][0]-2*a[0][4]-a[0][3]);
    a[1][2]=2*(3*a[0][4]-a[0][2]+3*a[0][0]);
    a[1][1]=2*(a[0][3]+2*a[0][0]-2*a[0][4]-a[0][1]);
    a[1][0]=a[0][4]-a[0][3]+a[0][2]-a[0][1]+a[0][0];
        dpro();
}
    if (m==0)
    goto end;  /*   no first order filter required    */

         /*    filter order filter parameter determination    */
         a1=1/pow(0.4142,k);
         a[0][1]=b1=a1*B;
         a[0][0]=wc;
         a[1][2]=c1+a[0][1]+a[0][0]/c1;
        a[1][1]=2*(a[0][0]/c1-c1);
        a[1][0]=c1-a[0][1]+a[0][0]/c1;
        tem[0]=b1*data[swch][ns-1]/a[1][2];
        tem[1]=(b1*data[swch][ns]-a[1][1]*tem[0])/a[1][2];
        for(i=ns+1;i<npm+30;i++){
        tem[i]=a[1][1]*tem[i-1]+a[1][0]*tem[i-2];
        tem[i]=(b1*(data[swch][i]-data[swch][i-2])-tem[i])/a[1][2];
        }
        data[swch][npm+29]=b1*tem[npm+29]/a[1][2];
        data[swch][npm+28]=(b1*tem[npm+28]-a[1][1]*data[swch][npm+29])/a[1][2];
        for(i=ns+1;i<npm+30;i++){
data[swch][npm+29-i]=a[1][1]*data[swch][npm+30-i]+a[1][0]*data[swch][npm+31-i];
data[swch][npm+29-i]=(b1*(tem[npm+29-i]-tem[npm+31-i])-data[swch][npm+29-i]
)/a[1][2];
}
end:    fflag[swch]=1;

}



dpro()
{

     int  i;

    tem[0]=b1*b1*data[swch][ns-1]/a[1][4];
    tem[1]=(b1*b1*data[swch][ns]-a[1][3]*tem[0])/a[1][4];
    tem[2]=b1*b1*(data[swch][ns+1]-2*data[swch][ns-1]);
    tem[2]=(tem[2]-a[1][3]*tem[1]-a[1][2]*tem[0])/a[1][4];
    tem[3]=b1*b1*(data[swch][ns+2]-2*data[swch][ns]);
    tem[3]=(tem[3]-a[1][3]*tem[2]-a[1][2]*tem[1]-a[1][1]*tem[0])/a[1][4];
    for(i=ns+3;i<npm+30;i++){
    tem[i]=b1*b1*(data[swch][i]-2*data[swch][i-2]+data[swch][i-4]);
    a1=a[1][3]*tem[i-1]+a[1][2]*tem[i-2]+a[1][1]*tem[i-3]+a[1][0]*tem[i-4];
    tem[i]=(tem[i]-a1)/a[1][4];
    }
    data[swch][npm+29]=b1*b1*tem[npm+29]/a[1][4];
    data[swch][npm+28]=(b1*b1*tem[npm+28]-a[1][3]*data[swch][npm+29])/a[1][4];
    data[swch][npm+27]=b1*b1*(tem[npm+27]-2*tem[npm+29]);
    a1=a[1][3]*data[swch][npm+28]+a[1][2]*data[swch][npm+29];
    data[swch][npm+27]=(data[swch][npm+27]-a1)/a[1][4];
    data[swch][npm+26]=b1*b1*(tem[npm+26]-2*tem[npm+28]);
```

```
a1=a[1][1]*data[swch][npm+29]+a[1][2]*data[swch][npm+28]+a[1][3]*data[swch][
npm+27];
   data[swch][npm+26]=(data[swch][npm+26]-a1)/a[1][4];
   for(i=ns+3;i<npm+30;i++){
   data[swch][npm+29-i]=b1*b1*(tem[npm+33-i]-2*tem[npm+31-i]+tem[npm+29-i]);
   a1=a[1][2]*data[swch][npm+31-i]+a[1][3]*data[swch][npm+30-i];
   a1=a1+a[1][0]*data[swch][npm+33-i]+a[1][1]*data[swch][npm+32-i];
   data[swch][npm+29-i]=(data[swch][npm+29-i]-a1)/a[1][4];
   }
   }


void head()
{
        int i,j;
        void waitsec(),cls(),position();
        union REGS inregs,outregs;

        inregs.h.ah=1;
        inregs.h.ch=0x0f;
        inregs.h.cl=0x0f;
        int86(0x10,&inregs,&outregs);
        cls(0,24);
        for(j=0;j<2;++j){
                for(i=0;i<5;++i){
                        position(0+j,15+i);
                        w_char(219,0x0f);
                        }
                for(i=0;i<5;++i){
                        position(4+j,15+i);
                        w_char(219,0x0f);
                        }
                }
        position(2,19);
        w_char(219,0x0f);
        position(0,21);
        printf("Multi-Function Test System. Version 1.51 1988.");
        printf("\nNo COPYRIGHT (C) 1988. No TRADE MARK (R) 1988.");
        printf("\nNo Use (N) 1988.");
        waitsec(2.5);
        cls(0,24);
        inregs.h.ah=1;
        inregs.h.ch=0x07;
        inregs.h.cl=0x07;
        int86(0x10,&inregs,&outregs);
}


header()
{
        void position();
        int i,j;

        position(10,0);
        printf("%c",'\311');
        for(i=11;i<59;++i){
                for(j=0;j<8;j=j+7){
                        position(i,j);
                        printf("%c",'\315');
                        }
                }
        position(59,0);
        printf("%c",'\273');
        for(i=1;i<8;++i){
                for(j=10;j<60;j=j+49){
                        position(j,i);
                        printf("%c",'\272');
                        }
                }
        position(10,7);
        printf("%c",'\310');
        position(59,7);
        printf("%c",'\274');
}

menu1(n)
int n;
{
        void position();

        g_clean(95,99);
        position(5,24);
        cprintf("F1:Filter   F2:Move   F3:Scale   F4:Print   F5:Simpfit   F6:Continue
F7:Next");
}


savedata()
```

```
{
}

                                                                   /      *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */

fin(m)          /* Input patient's information  */
int m;          /* which one    */
{
        char name[15],c,tmp[6];
        FILE *fp,*fopen();

        strcpy(name,pathn);
        strcat(name,pfnm[m]);           /* get file name         */
        fp=fopen(name,"r");
        fgetstr(l_name,16,fp);
        fgetstr(f_name,16,fp);
        fgetstr(bdate,15,fp);
        fgetstr(sex,4,fp);
        fgetstr(adrs[0],30,fp);
        fgetstr(adrs[1],30,fp);
        fgetstr(phone,15,fp);
        fgetstr(dgns[0],80,fp);
        fgetstr(dgns[1],80,fp);
        fgetstr(cmmts[0],80,fp);
        fgetstr(cmmts[1],80,fp);
        fgetstr(cmmts[2],80,fp);
        fgetstr(tst,15,fp);
        fscanf(fp,"%d\n",&ntk);
        for(ntst=0;fgetstr(tname[ntst],15,fp)!=0;++ntst)
                ;
        fclose(fp);
}

                                                                   /      *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */

readdata(fname,n,m)             /* Read test data                    */
char fname[];
int n,m;        /* n is channel; m=0 only read test information */
{
```

```
        int k;
        char name[15];
        FILE *fp,*fopen();

        k=m?n-1:4;
        strcpy(name,pathn);
        strcat(name,fname);
        fp=fopen(name,"r");
        if(fp==0)
                strcpy(sti[k].ptn,"Data not available at this time");
        else{
                 strcpy(datafile,fname);
                fgetstr(tdate[k],16,fp);
                 strcpy(hauptdate,tdate[k]);
                fgetstr(ttype[k],6,fp);
                fgetstr(label[k],6,fp);
                fgetstr(eye[k],4,fp);
                fgetstr(sti[k].frqcy,11,fp);
                fgetstr(sti[k].ptn,40,fp);
                fgetstr(sti[k].cntrst,11,fp);
                fscanf(fp,"%f\n%f\n",&lpf[k],&hpf[k]);
                if(m>0)
                        for(np=0;fscanf(fp,"%f",&data[k][np])!=EOF;++np);
                fclose(fp);
                }
        if((np!=250)&&(np!=500))
                np-=30;
}


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/


fgetstr(s,n,fp)         /* read a string from file      */
char s[];
int n;
FILE *fp;                       /* get ride of the '\n' */
{
        char *ptr,d=0;

        if(fgets(s,n,fp)!=NULL)
                d=1;
```

```
        if((ptr=strchr(s,'\n'))!=0)
                *ptr='\0';
        return d;
}

                                                                        /           *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */

list1()         /* Test information list                  */
{
        int maxpage,i,j,ln,flag=0;
        unsigned char c,loop,not_done=1;
        void cls(),position();

        cls(0,24);                      /* patient's information        */
        position(0,0);
        printf(" PATIENT :  %s  %s ",f_name,l_name);
        position(30,0);
        printf("     BIRTHDATE: %-10s   SEX: %s",bdate,sex);
        printf("\n ADDRESS :  %-20s   TELEPHONE: %s",adrs[0],phone);
        printf("\n %-20s",adrs[1]);
        printf("\nDIAGNOSIS:\n%-s\n%-s",dgns[0],dgns[1]);
        printf("\nCOMMENTS :\n%-s\n%-s\n%-s",cmmts[0],cmmts[1],cmmts[2]);
        printf("\n\n   FILE      TEST    DATE             STIMULUS");
        printf("                          LABEL");
        for(j=12;j<25;j=j+11)
                for(i=0;i<78;++i){
                        position(i,j);
                        printf("%c",'\315');
                        }
        maxpage=(ntst-1)/10;
        while(not_done){
                cls(13,22);
                position(0,12);
                for(i=0;i<10 && i+page*10<ntst;++i){
                        readdata(tname[i+page*10],1,0); /* only read information        */
                        printf("\n%-12s %4s",tname[i+page*10],ttype[4]);
                        printf("%10s  %-35s",tdate[4],sti[4].ptn);      /* show it      */
                        printf("%5s  %3s",sti[4].frqcy,label[4]);
                        }               /* end for loop */

                ln=13;
                loop=1;
```

```
        lnrvs(ln,0,78);
        while(loop){
            switch(keyin()){
                case 200:           /* move one line up     */
                    if(ln>13){
                        lnrvs(ln,0,78);
                        --ln;
                        lnrvs(ln,0,78);
                        }
                    break;
                case 208:           /* down */
                        if(ln<22    &&    page<=maxpage    &&
ntst-page*10+12>ln ){
                        lnrvs(ln,0,78);
                        ++ln;
                        lnrvs(ln,0,78);
                        }
                    break;
                case 201:           /* previous page        */
                    if(page>0){
                        --page;
                        loop=0;
                        }
                    break;
                case 209:           /* next page    */
                    if(page<maxpage){
                        ++page;
                        loop=0;
                        }
                    break;
                case 13:            /* accepte selection    */
                    readdata(tname[ln+page*10-13],chn,1);
                    strcpy(filename,tname[ln+page*10-13]);
                    not_done=loop=0;
                    break;
                case 27:            /* exit */
                    not_done=loop=0;
                    flag=1;
                    break;
                default:
                    loop=1;
                    break;
                }       /* switch       */
            }       /* loop */
```

```
          }              /* not_done     */
     return(flag);
}


                                                                    /         *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~ */


list()         /* list all the patient's information data      */
{                   /* file name on the data disk                    */

     int maxpage,ln,i,j,page,nf,flag=0;
     unsigned char x,not_done=1,loop=1;
     void cls(),position();

     cls(0,24);
     nf=fsrch();              /* number of files with [.dat] extension         */
     maxpage=nf/20;
     page=0;
     for(j=1;j<23;j+=21)      /* two dividing line     */
          for(i=0;i<70;++i){
               position(i,j);
               printf("%c",'\315');
               }

     position(0,0);
     printf(" FILE               NAME                TEST");
     position(0,24);
     printf("              TOTAL FILES: %4d",nf);

     while(not_done){
          cls(2,21);
          position(0,1);
          for(i=0;i<20 && i+page*20<nf ; ++i){     /* display       */
               fin(i+20*page);
               printf("\n%4d:",i+20*page+1);
               printf("  %-12s,  %-12s",l_name,f_name);
               printf("  %-10s ",tst);
               }
          ln=2;
          lnrvs(ln,1,70);
          loop=1;
```

```
        while(loop){
            switch(keyin()){
                case 200:     /* one line up  */
                    if(ln>2){
                        lnrvs(ln,1,70); /* reverse old one      */
                        --ln;
                        lnrvs(ln,1,70); /* reverse new one      */
                        }
                    break;
                case 208:     /* one line down        */
                       if(ln<21    &&    page<=maxpage    &&
nf-page*20+1>ln){
                        lnrvs(ln,1,70);
                        ++ln;
                        lnrvs(ln,1,70);
                        }
                    break;
                case 201:     /* previouse page       */
                    if(page>0){
                        --page;
                        loop=0;
                        }
                    break;
                case 209:     /* next page    */
                    if(page<=maxpage){
                        ++page;
                        loop=0;
                        }
                    break;
                case 13:      /* find one     */
                    fin(ln+page*20-2);     /* read the information */
                    not_done=loop=0;       /* finish       */
                    break;
                case 27:      /* nothing, exit        */
                    flag=1;
                    not_done=loop=0;
                    break;
                }       /* end switch   */
            }       /* end loop     */
        }           /* end not_done */

    return(flag);
}
```

```
                                                                    /        *
==========================================
========= */

draw(a,b,c)                  /* plot waveform of test data               */
int a;                       /* which one  */
float b;                     /* vertical shift       */
unsigned char c;             /* write or erase       */
{
        int i,x1,x2,y1,y2;
        float step,ratio,xr;
        char c1;
        void position();

        x1=15;               /* x-axis starting point        */
        xr=15.0;
        y1=10*data[a-1][0]/sen+b*20+5;  /* data starting point */
/* Banker OP code change starts */
        if(npm!=0){
                ratio = 250.0/(npm-ns+1);}
        else{
                ns=1;
                npm=np;
                ratio=1.0;
                }
        step=npm>300?1.0:(2.0*ratio);    /* decide the graphic size      */
        for(i=ns;i<npm;++i){
                xr=xr+step;
                x2=xr+0.5;
                y2=10*data[a-1][i]/sen+b*20+5;  /* compute data position      */
                line(x1,x2,200-y1,200-y2,c);
                x1=x2;
                y1=y2;
                }
/* Banker OP code change ends */
        c1=c?a+48:0;         /* for label display    */
        x1=(x1+15)/8;        /* at the last data point        */
        y1=24-y1/8;
        position(x1,y1);
        w_char(c1,1);                /*  put it on    */
        position(x1+2,y1);
        c1=c?label[a-1][0]:0;
        w_char(c1,c);
        position(x1+3,y1);
        c1=c?label[a-1][1]:0;
        w_char(c1,c);

}


/*=========================================
=============*/

csrm(l)                 /* a pair of cursor measure the peak value      */
float l[];
{
        int i,j,k,x,y1,y2,key=49,swch,p=0;
        void position();
        float a,b,c;

        for(i=0;i<chn;++i)              /* the initial position */
                for(j=0;j<2;++j)
                        csrln(i,j,l,1);
        g_clean(95,99);
        position(5,24);
        swch=key-49;            /* channel 1 first      */
        cprintf("       Cursor of waveform %d",swch+1);
        while((key=keyin())!=27){       /* not finish   */
                switch(key){
                        case 203:               /* move left    */
                                csrln(swch,p,l,0);
                                --csr[swch][p];
                                csrln(swch,p,l,1);
                                break;
                        case 205:               /* move right   */
                                csrln(swch,p,l,0);
                                ++csr[swch][p];
                                csrln(swch,p,l,1);
                                break;
                        }               /* end of switch        */
                if(key==97||key==65)    /* A cursor             */
                        p=0;
                if(key==98||key==66)    /* B cursor             */
                        p=1;
                if(key>48&&key<chn+49){         /* change channel       */
                        g_clean(95,99);
                        position(5,24);
                        swch=key-49;
                        cprintf("      Cursor of waveform %d",swch+1);
                        }       /* end of if    */
                }       /* end of while */
}


/*=======================================
===========*/

csrln(i,j,l,s)  /* draw the little cursor line  */
int i,j;                /* which one    */
float l[];              /* vertical shift of the data   */
int s;                  /* on or off    */
{
        int x,y1,y2,k,step;

        step=np>300?1:2;
        x=15+step*csr[i][j];
        k=csr[i][j];
        y1=10*data[i][k]/sen+l[i]*20+6;
        y2=y1+10;
        line(x,x,200-y1,200-y2,s);
        if(s)
                for(k=0;k<chn;++k){             /* data */
                        position(40,2+k);
                        printf("#%1d: %3d<=> %7.3f",k+1,csr[k][0],data[k][csr[k][0]]);
                        printf("    %3d<=> %7.3f",csr[k][1],data[k][csr[k][1]]);
                        }

}


/*=======================================
=========*/


plot(n)
int n;
{
        int o_mode,mode=6,flag,i;
        char key;
        float offset[4];
        void position(),axis();
        offset[0]=4.0;
        offset[1]=4.0;
        offset[2]=6.0;
        offset[3]=6.0;
        o_mode=getmode();
        cmode(mode);
r:      g_clean(0,99);
        axis();
        for(i=1;i<=chn;++i)
                draw(i,offset[i-1],1);
m:       menu1(n);
k:       if((key=getch())==27)
                flag=0;
        else if(key==0){
                switch(getch()){
                        case 59:
                                filter();
                                goto r;
                                break;
                        case 60:
                                move(offset);
                                goto m;
                                break;
                        case 61:
                                g_clean(95,99);
                                position(5,24);
                                cprintf("Enter New Scale:");
                                scanf("%f",&sen);
                                while(getchar()!='\n')
                                        ;
                                goto r;
                                break;
                        case 62:
                                g_clean(95,99);
                                prnt();
                                goto r;
                                break;
                        case 63:
                                   if(numprmts<10)
                                   {
                                simpfit();
                                goto r;
                                break;
                                   }
                        else
                        {
                        simpsave();
                        goto r;
                        break;
                        }
                case 64:
                        flag=2;
                           npm=np;
                           ns=1;
                        break;
                case 65:
                        flag=3;
                           npm=np;
                           ns=1;
                        break;
                default:
                        goto k;
                        break;
                }
        }
        else
                goto k;
        cmode(o_mode);
        return(flag);
}


/*========================================
===========*/

move(shift)                   /* move waveforms                    */
float shift[];
{
        float l1=0,l2=0;
        char key,not_done=1;
        int swch;
        void position();

        g_clean(95,99);
        position(5,24);
        cprintf("Which one do you want to move :");
        while((key=keyin())>(chn+0x30)||key<49)
                ;
```

```
	swch=key-48;
	g_clean(95,99);
	position(5,24);
	cprintf("     MOVING  WAVEFORM  %d",swch);
	while(not_done)
		switch(keyin()){
			case 200:	/* up   */
				draw(swch,shift[swch-1],0);
				shift[swch-1]+=.2;
				draw(swch,shift[swch-1],1);
				break;
			case 208:	/* down */
				draw(swch,shift[swch-1],0);
				shift[swch-1]-=.2;
				draw(swch,shift[swch-1],1);
				break;
			case 27:	/* exit */
				not_done=0;
								/* redraw plots */

for(swch=1;swch<=chn;swch++)
					draw(swch,shift[swch-1],1);
				break;
			default:
				break;
			}
}


/*============================================
===============*/

void axis()
{
	int i,j,k;	/* Banker OP code added */
	float ratio;
	void position();

/* Banker OP code insertion starts */
	if(npm == 0)
		{
		npm=np;
		ns=1;
		}
```

```
        ratio = 250.0/(npm-ns+1)*5;
        line(10,630,185,185,1);          /* x-axis       */
        line(10,10,15,185,1);            /* y-axis       */
        j=150/ratio;
        for(i=1;i<=j;++i)                /* x scales     */
        {
                k=15.0+i*4*ratio+0.5;
                line(k,k,185,180,1);
        }
        for(i=0;i<17;++i)          /* y scales     */
                line(10,13,15+i*10,15+i*10,1);
        j=15/ratio;
        for(i=1;i<=j;++i)                /* x scales X 10 div. */
        {
                k=15.0+i*40*ratio+0.5;
                line(k,k,180,176,1);
        }
        position(65,21);
        cprintf("%2d ms/div",(npm-ns+2)>300?20:10);
/* Banker OP code insertion ends */
        position(3,2);
        printf("%4.2f uv/div",sen);
}

                                                                /           *
=========================================
======== */


#define GRSEG 0xb800              /* define video memory segment  */
                                            /* for small memory model         */

g_clean(m,n)                   /* graphics clean                      */
int m,n;                              /* start and finish line               */
{
        unsigned char rw[80];
        int i;
        struct SREGS reg;

        segread(&reg);           /* read DS value          */
        memset(rw,0,80);
        for(i=m;i<=n;++i){
              movedata(reg.ds,rw,GRSEG,i*80,80);
              movedata(reg.ds,rw,GRSEG,i*80+0x2000,80);
```

```
                }
}

void prtsc()                                    /* print graphics using BIOS    */

{       union REGS inregs,outregs;
        int86(0x05,&inregs,&outregs);
}

w_char(c,atr)                                   /* write a character on screen  */
char c,atr;                                        /* character and it's attribute
*/
{
        union REGS inregs,outregs;

        inregs.h.ah=9;                           /* write character
            */
        inregs.h.bh=0;
        inregs.h.al=c;
        inregs.h.bl=atr;
        inregs.x.cx=1;
        int86(0x10,&inregs,&outregs);      /* BIOS call Hex 10; video function    */

}

void getcurs(row,col)                        /* get cursor position                 */
int *row,*col;
{
        union REGS inregs,outregs;

        inregs.h.ah=3;
        inregs.h.bh=0;
        int86(0x10,&inregs,&outregs);
        *row=outregs.h.dh;
        *col=outregs.h.dl;

}

void position(c,r)      /* set the cursor position              */
int c,r;                     /* column and row       */
{
        union REGS inregs,outregs;

        inregs.h.ah=2;
        inregs.h.dh=r;
        inregs.h.dl=c;
        inregs.h.bh=0;
        int86(0x10,&inregs,&outregs);
}


void cls(r1,r2)                              /* clear screen in text mode    */
int r1,r2;                                       /* start and end line
*/
{
        union REGS inregs,outregs;

        inregs.h.ah=6;
        inregs.h.bh=0x07;
        inregs.h.al=0;
        inregs.h.ch=r1;
        inregs.h.cl=0;
        inregs.h.dh=r2;
        inregs.h.dl=79;
        int86(0x10,&inregs,&outregs);
}

void cursup()                                /* move cursor up one position  */
{
        int row,col;

        getcurs(&row,&col);
        if(row>0)
                position(col,row-1);
}

void cursdn()                                /* cursor down one position          */
{
        int row,col;

        getcurs(&row,&col);
        if(row<24)
                position(col,row+1);
}

r_char(c,atr)                                /* read character at current cursor     */
unsigned char *c,*atr;
{
        union REGS inregs,outregs;

        inregs.h.ah=8;
        inregs.h.bh=0;
        int86(0x10,&inregs,&outregs);
        *c=outregs.h.al;
        *atr=outregs.h.ah;
}

lnrvs(ln,cl1,cl2)       /* reverse a line on the screen */
int ln,cl1,cl2;         /* line position and columns    */
{
        int i;
        char c,a;
        void position();

        for (i=cl1;i<=cl2;++i){
                position(i,ln);
                r_char(&c,&a);  /* what's in there      */
                a=(~a)&0x77;    /* revers the attribute */
                w_char(c,a);
                }
}

getmode()       /* get video mode       */
{
        union REGS inregs, outregs;

        inregs.h.ah = 0x0F;                     /* Use BIOS call to get mode    */

        inregs.h.al = 0;
        int86(0x10,&inregs,&outregs);
                return(outregs.h.al);
}


cmode (num)                                             /* change mornitor mode
            */
int num;
{
        union REGS inregs, outregs;

        inregs.h.ah = 0;        /* Use BIOS call to set mode    */
        inregs.h.al = num;
        int86(0x10,&inregs,&outregs);
        return(num);
}


line(x,x1,y,y1,c)        /* draw line between two dots   */
int x,x1,y,y1;           /* corespondents        */
unsigned char c;         /* write or erase       */

{
        int p,q,e,deltax,deltay,hlfx,hlfy;
        int ix,iy,cont;

        deltax=abs(x1-x);
        deltay=abs(y1-y);
        hlfx=deltax/2;
        hlfy=deltay/2;
        ix=(x1<x)?-1:1;          /* going left or right  */
        iy=(y1<y)?-1:1;          /* going up or down             */
        if(abs(deltay)>=abs(deltax)){           /* slope greater than 1 */
                e=0;
                p=x;
                q=y;
                cont=deltay;
                while(cont>=0){
                                putdot(p,q,c);
                                q=q+iy;
                                e=e+deltax;
                                if(e>hlfy){             /* increment horizontally        */
                                        e=e-deltay;
                                        p=p+ix;
                                        }
                                cont=cont-1;
                        }
        }
        else{
                e=0;
                p=x;
                q=y;
                cont=deltax;
                while(cont>0){
                        putdot(p,q,c);
                        p=p+ix;
                        e=e+deltay;
                if(e>hlfx){            /* increment vertically */
                      e=e-deltax;
                      q=q+iy;
                      }
                cont=cont-1;
                }
      }

}


putdot(col,row,c)       /* write the dot in the video memory directly   */
int col,row;           /* position     */
unsigned char c;        /* set or clear */
{
      unsigned char far *scrptr;
      unsigned int offset,shift;

      scrptr=(unsigned char far *) (0xb800l<<16);
      shift=7-col&0x7;
      offset=80*(row>>1)+(col>>3);
      if(row&0x1)
            scrptr +=0x2000l;
      *(scrptr+offset) &=~(0x01<<shift);
      *(scrptr+offset) |=(c<<shift);

}

#define clockfreq 1193180L
#define spkrmode 0xb6
#define t_modeport 0x43
#define freqport 0x42
#define spkrport 0x61
#define spkron 0x03
#define freq0 0x12c
#define freq1 0x19f
#define div0 clockfreq/freq0
#define div1 clockfreq/freq1
#define click .40
void waitsec();

void sound()   /* sound using the speaker      */
{
      unsigned char port0;
```

```
        unsigned int d0=div0,d1=div1,ct=0;
        float delay=click;
        int i;

        outp(t_modeport,spkrmode);
        port0=inp(spkrport);
        outp(freqport,d0&0xff);
        outp(freqport,d0>>8);
        outp(spkrport,port0|spkron);
        waitsec(delay);
        outp(freqport,d1&0xff);
        outp(freqport,d1>>8);
        outp(spkrport,port0|spkron);
        waitsec(delay);
        outp(spkrport,port0);

}

#define t_per_s (18.2)
long t_counts();

void waitsec(secs)      /* timing function       */
double secs;
{
        unsigned long count0,count1;

        count0=t_counts();
        count1=count0+secs*18.2;
        while(t_counts()<count1)
                ;
}

#define int_time 0x1a
long t_counts()
{
        union REGS rin,rout;
        long tc;

        rin.h.ah=0;
        int86(int_time,&rin,&rout);
        tc=((long)rout.x.cx)<<16;
        tc+=rout.x.dx;
        return tc;
```

```
}

keyin()         /* keyboard input for IBM       */
{
        char key;

        if((key=getch())!=0)
                return(key);
        else
                return(getch()+128);    /* function keys        */
}

/*========================================
====================*/

 /*by Marco Caceci, with help from William Caceris.     1983*/
 /*Chem. Dept. Florida State University Tallahassee  FL32306*/
 /*no  copy-right.  SSSD  floppy  disk  copies  on  request*/

 /*see Nelder J.A. & R. Mead, Computer J. 7, 308 (1965) and */
 /*L.A. Yarbro & S.N. Deming,Anal. Chim. Acta 74, 391 (1974)*/


/*NOTE: INSTRUCTIONS FOR MODIFYING THE PROGRAM TO HANDLE
OTHER  */
/*FUNCTIONS, USE MORE (OR LESS) PARAMETERS, AS WELL AS THE      */
/*THEORY BEHIND IT, CAN BE FOUND IN THE MAY 1984 ISSUE OF BYTE  */
/*MAGAZINE, P.340) */

/*Modified and adapted to C by Zhongquan Li, Nov., 1988*/


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~*/

#define   date    " 8/28/88"
#define   memo     " Fit a ERG response with mathmatical functions "
/* #define   m        5 */    /*number of parameters to fit*/
extern int m;
#define   nvpp     2    /*total number of vars per data point*/
/* #define   n        6 */   /*m + 1*/ /*some compilers don't like this*/
extern int n;
/*#define   mnp      1000 */ /*maximum number of data points*/
int mnp = 532;
```

```
#define   alfa     1.0  /*reflection coefficient, >0*/
#define   beta     0.5  /*contraction coefficient, 0to1*/
#define   gamma    2.0  /*expansion coefficient, >1*/
#define   lw       7    /*width of line in data fields+1*/
#define   root2    1.414214


/*=========================================
===========*/

int       h[10],l[10];                 /*number high/low paramts*/
int       maxiter;               /*max number iterations*/
int       niter;                  /*number of iterations*/
double    next[10];               /*new vertex to be tested*/
double    center[10];             /*center of hyperplane described      */
                                   /* by all vertexes of the simplex excluding the worst*/
double    mean[10],error[10];
double    maxerr[10];              /*maximum error accepted*/
double    p[11],q[10];             /*to compute first simplex*/
double    step[10];               /* starting steps for this function       */

double    simp[10][10];                 /*the simplex*/
double    adata[532][2];                /*the data*/
double    sigma;                        /* error term */


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

double tanh(va)        /* hypabolic tangent function   */
double va;
{
        double  a,b;

        a=exp(va);
        b=exp(-va);
        return((a-b)/(a+b));

}               /* end of tanh */



/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```

```
enter()            /* enters data from disk file fname.
                          data in the order:
                             -maximum number iterations (integer),
                             -initial guesses of parameters)
                             -starting increments (e.g:0.1 1)
                             -maximum errors (e.g:1e-4 1e-4 1e-4)
                             -data (x,y
                                  x,y.....etc*/
{
        int i,j;

        for(i=0;i<m;++i)                       /* #m of initial guesses        */
                simp[0][i]=pmtr[i];
        for(i=0;i<m;++i)                       /* #m of first steps            */
                step[i]=fabs(pmtr[i]/10.0)==0?0.1:fabs(pmtr[i]/10.0);
        step[n]=0.01;
        for(i=0;i<n;++i)                       /* #n of maximum errors          */
                maxerr[i]=0.01;
        for(i=ns-1;i<npm;++i){
                adata[i][1]=data[0][i];
                adata[i][0]=i*1.0;
                }
}                                                        /* end of enter */


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

double sum_of_residuals(x) /*computes the sum of the squares of the residuals*/
double x[];                /*x(1..m) passes parameters. Result returned in x(n)*/
{
        int i;
        double u,v;
        double f();

        x[n-1]= 0.0;
        for(i=ns-1;i<npm;++i){
                u=f(x,adata[i][0])-adata[i][1];
                v=u*u;
                x[n-1]=x[n-1]+v;
                }
}                       /* end of sum_of_residuals */
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~*/
/*
rpt()
{
        double y,dy,xr;
        int j,i;
        char fitname[15];
        FILE *fp1,*fopen();
        double f();

        printf("\nEnter output file name:");
        scanf("%s",fitname);
        fp1=fopen(fitname,"a+");
              fprintf(fp1,"\n\n  =============  START
================ \n\n");
        fprintf(fp1,"Data from file----%s \n",filename);
        fprintf(fp1,"%5d data points",np);
        fprintf(fp1,"\n Program exited after %10d iterations.",niter);
        fprintf(fp1,"\nThe estimated best parameters are:\n\n");
         for(i=0;i<(n-2);++i)
                fprintf(fp1,"  x[%2d] ",i+1);
        fprintf(fp1,"\n");
        for(i=0;i<m;++i)
                fprintf(fp1,"%8.4f",mean[i]);
        fprintf(fp1,"\n\n The estimated fractional error is\n\n");
    for(i=0;i<n;++i)
                fprintf(fp1,"%7.4f",error[i]);
        fprintf(fp1,"\n--------------------------------\n");
        sigma= 0.0;

        for(i=ns-1;i<npm;++i){
                y= f(mean,adata[i][0]);
                dy= adata[i][1] - y;
                sigma = sigma + dy*dy;
                }

        xr = sqrt(sigma / (npm-ns+1));
        fprintf(fp1,"\nThe standard deviation is %10.5f",xr);
        xr = xr / sqrt((npm-ns+1)*1.0 - m);
        fprintf(fp1,"\nThe estimated error of the function is %10.5f",xr);
              fprintf(fp1,"\n\n  ~~~~~~~~~~~~~~  END
~~~~~~~~~~~~~~~\n\n\n");
```

```
        fclose(fp1);

}
*/


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

new_vertex()                    /*next in place of the worst vertex*/
{
        int i;


        for(i=0;i<n;++i)
                simp[h[n-1]][i]= next[i];

}                                         /* end of new_vertex */


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

order()                                 /*gives high/low in each parameter*/
                                            /*in simp.  caution:not initialized*/
{
        int i,j;

        for(j=0;j<n;++j){               /*all dimensions*/
                for(i=0;i<n;++i){       /*of all vertexes find best and worst*/
                        if(simp[i][j] < simp[l[j]][j])
                                l[j] = i;
                        if(simp[i][j] > simp[h[j]][j] )
                                h[j] = i;
                        }               /* end of i loop */
                }                       /* end of j loop */

}                                       /* end of order  */


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

simpfit()                       /* Curve fitting program        */
```

```
{
        char not_done=1,key;
        int i,j;
        double sum_of_residuals();
        void position(),cls();

        cmode(2);
        cls(0,24);
        printf("\n\nSimplex Fitting Procedure.\n");
        printf("\n\n\n Max Number of iterations:");
        scanf("%d",&maxiter);
        while(getchar()!='\n')
                ;
        enter();                                        /* read the data        */
        sum_of_residuals(simp[0]);      /*first vertex*/
        for(i=0;i<m;++i)                                /*compute offset of the vertexes*/
                {                                               /*of the starting simplex*/
                p[i] = step[i] * (sqrt(n*1.0) + m - 1.0) / (m * root2);
                q[i] = step[i] * (sqrt(n*1.0) - 1) / (m * root2);
                }
        for(i=1;i<n;++i){               /*all vertexes of the starting simplex */
                for(j=0;j<m;++j)
                        simp[i][j] = simp[0][j] + q[j];
                simp[i][i - 1]= simp[0][i - 1] + p[i - 1];
                sum_of_residuals(simp[i]);      /*and their residuals*/
                }
                for(i=0;i<n;++i)                                /*preset before calling*/
                l[i] = h[i] = 0;
        order();
        cmode(6);
        for(i=0;i<n;++i){                       /*average each parameter*/
                mean[i]= 0.0;
                for(j=0;j<n;++j)
                        mean[i]= mean[i] + simp[j][i];
                mean[i]= mean[i] / n;
                  pmtr[i] = mean[i];
                }
        view();
        niter = 0;                                      /*no iterations yet*/
        while(not_done&&(niter<maxiter)){       /*keep iterating*/
                not_done = 0;                           /*wish it were...*/
                ++niter;
                position(60,3);
                printf("%3d",niter);
```

```
position(30,24);
printf("fitting......");
 for(i=0;i<n;++i){                    /*average each parameter*/
        mean[i]= 0.0;
        for(j=0;j<n;++j)
                mean[i]= mean[i] + simp[j][i];
        mean[i]= mean[i] / n;
          pmtr[i] = mean[i];
        }
  if(niter%10==0)
        view();
for(i=0;i<n;++i)
        center[i] = 0.0;
for(i=0;i<n;++i)                      /*compute centroid*/
        if(i!=h[n-1] )             /*excluding the worst*/
for(j=0;j<m;++j)
        center[j]= center[j] + simp[i][j];
for(i=0;i<n;++i){                     /*first attempt to reflect*/
        center[i]=center[i] / m;
        next[i]=(1.0 + alfa) * center[i] - alfa * simp[h[n-1]][i];
        /*next vertex is the specular reflection of the worst*/
        }
sum_of_residuals(next);

if(next[n-1] <= simp[l[n-1]][n-1]){     /*better than the best ?*/
        new_vertex();                         /*accepted !*/
        for(i=0;i<m;++i)                         /*and expanded*/
                next[i]=gamma * simp[h[n-1]][i] + (1.0 - gamma) * center[i];
        sum_of_residuals(next);         /*still better ?*/
        if(next[n-1] <= simp[l[n-1]][n-1] )
                new_vertex();
        }                                               /*expansion accepted*/
else{                                            /*if not better than the best*/
        if(next[n-1] <= simp[h[n-1]][n-1])
                new_vertex();                 /*better than worst*/
        else{                          /*worse than worst then:  contract*/
                for(i=0;i<m;++i)
                        next[i]=beta*simp[h[n-1]][i]+(1.0-beta)*center[i];
                sum_of_residuals(next);
                if(next[n-1]<=simp[h[n-1]][n-1])
                        new_vertex();                  /*contraction accepted*/
                else                                          /*if still bad*/
                        {                                            /*shrink all bad
vertexes*/
```

```
                                        for(i=0;i<n;++i)
                                                for(j=0;j<m;++j){
                                                        simp[i][j]=(simp[i][j] + simp[l[n-1]][j]) *
beta;
                                sum_of_residuals(simp[i]);
                                                        }                       /* end of i loop*/
                                        }                                       /* end of else */
                                }                                               /* end of else*/
                        }                                                       /* end of else*/
                        order();
                        for(j=0;j<n;++j){                                       /*check for
convergence*/
                                error[j]=(simp[h[j]][j] - simp[l[j]][j]) / simp[h[j]][j];
                                if(not_done==0)
                                if(error[j] > maxerr[j])
                                        not_done=1;
                                }
                }

        cmode(2);
        finial();
        cmode(6);
}


view()
{
        int i,j;
        float offset[4];
        void position(),axis();


        for(i=ns-1;i<npm;++i)
                data[1][i]=f(mean,adata[i][0]);
        offset[0]=3.5;
        offset[1]=3.5;
        offset[2]=6.;
        offset[3]=6.;
        g_clean(0,99);
        axis();
        for(i=1;i<=chn;++i)
                draw(i,offset[i-1],1);
}
```

```
                                                                          /      *
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

fsrch()         /* search all the patient's information file name */
{                       /*      file with [.DAT] extension                      */
        int nf;         /* number of files      */
        unsigned char buffer[64],fcb[50];
        unsigned char *ptr;

         strcpy(fcb, pathn);
        strcat(fcb,"*.dat");
        ptr=buffer;                             /* pointer to buffer    */
        bdos(0x1a,ptr,0);               /* put file name in FCB */
        ptr=fcb;                                /* set pointer to FCB   */
        nf=0;
        if(bdos(0x4e,ptr,1)==0){        /* find the first one match     */
                strcpy(pfnm[nf],buffer+30);     /* get the file name from FCB   */
                ++nf;
                while(bdos(0x4f,ptr,1)==0){ /* continue to look */
                        strcpy(pfnm[nf],buffer+30);
                        ++nf;
                        }       /* end while    */
                }       /* end if       */
        return nf;
}

simpsave()
{
        int w;
        char filename[20];
        FILE *fpp;

        cmode(2);
        cls(0,24);
         printf("\nYou cannot simpfit more than nine (9) parameters.");
         printf("\nParameters and filtered waveform will be saved.");
         while(getchar()!='\n');
         printf("\nEnter path and output file name: ");
         gets(filename);
         if((fpp=fopen(filename,"a"))== 0)
                 printf("Can't open file -> %s\n",filename);
         else
         {
```

```
                fprintf(fpp,"\n%12s",datafile);
                fprintf(fpp,"\n%15s",hauptdate);
                fprintf(fpp,"\n%5d",npm);
                fprintf(fpp,"\n%5d",ns);
                fprintf(fpp,"\n%5d",NN);
                fprintf(fpp,"\n%10.5f",sen);
                for(w=1;w<numprmts+1;++w)
                        fprintf(fpp,"\n%17.6f",pmtr[w-1]);
                for(w=1;w<numprmts+1;++w)
                        fprintf(fpp,"\n%3s",lbf[w-1]);
                for(w=1;w<numprmts+1;++w)
                        fprintf(fpp,"\n%8s",unt[w-1]);
                fprintf(fpp,"\n");
                for(w=ns-1;w<npm+30;++w)
                        fprintf(fpp,"%10.5f",data[swch][w]);
        }
        cmode(6);
}
```

Sub-program for Mathematical Equations

```
#include <stdio.h>              /* ERG curve fitting    */
#include <math.h>                /* building simplex method      */
#include <conio.h>              /* Modified too many damn ways */
#include <stdlib.h>             /* for Banker Master Thesis     */
#include <string.h>
#include <io.h>
#include <dos.h>
#include <memory.h>

int NN = 7;  /* Order of OP eqtn term */

int numprmts;  /* NUMBER OF PARAMETERS USED BY SIMPLEX PROGRAM*/
int m;          /* SHOULD BE SAME AS numprmts */
int n;          /* SHOULD BE m + 1 */
char prmstr;   /* NUMBER OF PARAMETERS AS STRING */
char prmstr1;  /* NUMBER OF PARAMETERS + 1 AS STRING */

int tnumprmt = 10;    /*  TOTAL NUMBER OF PARAMETERS */
int tm = 10;          /* SHOULD BE SAME AS tnumprmt */
int tn =  11;         /* SHOULD BE m + 1 */
char tprmstr = 'A';   /* TOTAL NUMBER OF PARAMETERS AS STRING */
char tprmstr1 = 'B';  /* TOTAL NUMBER OF PARAMETERS + 1 AS STRING */
                      /* USE 'A' IF 10 PARAMETERS */
                      /* USE 'B' IF 11 PARAMETERS, ETC. */

int opnumprm = 4;  /* ENTER NUMBER OF OP PARAMETERS */
int opm = 4;          /* SHOULD BE SAME AS opnumprm */
int opn = 5;          /* SHOULD BE opm + 1 */
char opprmstr = '4';  /* ENTER NUMBER OF OP PARAMETERS AS STRING */
char opprmstr1 = '5';  /* ENTER NUMBER OF OP PARAMETERS + 1 AS STRING
*/

int abnumprm = 6;  /*  ENTER NUMBER OF A & B WAVE PARAMETERS */
int abm = 6;          /* SHOULD BE SAME AS abnumprm */
int abn = 7;          /* SHOULD BE abm + 1 */
char abprmstr = '6';  /* ENTER NUMBER OF A & B WAVE PARAMETERS AS
STRING */
char abprmstr1 = '7';  /* ENTER NUMBER OF A & B WAVE PARAMETERS + 1
AS STRING */

/* CHANGE HERE EQUATION THAT APPEARS ON SCREEN */
/* OP WAVE MODEL EQUATION */
```

```
char opeqtn[100] = "OP(t)=A*[exp(1)*t/Tc]^n * [exp(-t*n/Tc)] * sin(2*pi*t*Fo +
Po)";
/* A & B WAVE MODEL EQUATION */
char abeqtn[100] = "AB(t)=K1*(1 - exp(-a1((t/Tp1)exp(1-t/Tp1))^5))
+K2*(a2*t)^3*exp(-a2*t)*sin(Wt)";

double pmtr[15];
double initpmtr[15];
unsigned char lbf[15][3];
unsigned char unt[15][8];

/* ENTER STARTING VALUES FOR OP PARAMETER OPTIMIZATION */
double oppmtr[15]={38.0,25.0,0.130,-45.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

/* DO IT AGAIN */
d        o        u        b        l        e
opinitpm[15]={38.0,25.0,0.130,-45.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

/* ENTER SYMBOLS OF OP PARAMETERS */
unsigned char oplbf[15][3]={"A","Tc","Fo","Po"," "," "," "," "," "," "," "," "," ","
"," "};

/* ENTER UNITS OF OP PARAMETERS */
unsigned char opunt[15][8]={"uV","ms","kHz","deg."," "," "," "," "," "," "," "," ","
"," "," "};

/* ################################################## */

/* ENTER STARTING VALUES FOR A & B WAVE PARAMETER OPTIMIZATION
*/
d        o        u        b        l        e
abpmtr[15]={-350.0,200.0,125.0,1777.8,0.15,0.019,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0};

/* DO IT AGAIN */
d        o        u        b        l        e
abinitpm[15]={-350.0,200.0,125.0,1777.8,0.15,0.019,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0};

/* ENTER SYMBOLS FOR A & B WAVE PARAMETERS */
unsigned char ablbf[15][3]={"K1","Tp","a1","K2","a2","W"," "," "," "," "," "," ","
"," "," "};

/* ENTER UNITS FOR A & B WAVE PARAMETERS */
```

```
unsigned char abunt[15][5]={"uv","ms","none","uv","1/ms","rd/ms"," "," "," "," ","
"," "," "," "," "};



double dbint=0.0;

extern char opflag;

double pi = 3.1415926;



/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

/* PUT MODEL EQUATION HERE!!! */

double f(x,d)   /* fitting the data with the function */
double x[],d;   /* x(1..m) the parameters, d has the data*/
{
        double a,b,r,s,op,y,dum, tanh();

/* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
/* EQUATION RIGHT HERE!!! */


        if(opflag=='o')                 /* OP WAVE EQUATION */
        {
                if(x[1] < 0.0001)
                        x[1] = 0.0001;

y=x[0]*pow((exp(1)*d/x[1]),NN)*exp(-d/x[1]*NN)*sin(2*pi*x[2]*d+x[3]*pi/180);
        }

        if(opflag=='a')                 /* A & B WAVE EQUATION */
        {
                if(x[1]==0)
                        x[1]=.001;
                if(x[1]>300)
                        x[1]=300;
                a=pow((d/x[1]) * exp(1-(d/x[1])),5.0);
                b=(x[3]/100*pow(x[4]*d,3.0) * exp(-x[4]*d)*sin(x[5]*d));
                /* x[0] (a-wave amplitude) must be negative or zero*/
```

```
                if(x[0] >0)
                        x[0]=0;
                r=((1 - exp(-x[2]*a))*x[0]);
                y=r+b;
        }

        if(opflag=='b')                 /* OP WAVE PLUS A & B WAVE EQUATION
*/
        {
                if(x[1]==0)
                        x[1]=.001;
                if(x[1]>300)
                        x[1]=300;
                a=pow((d/x[1]) * exp(1-(d/x[1])),5.0);
                b=(x[3]/100*pow(x[4]*d,3.0) * exp(-x[4]*d)*sin(x[5]*d));
                /* x[0] (a-wave amplitude) must be negative or zero*/
                if(x[0] >0)
                        x[0]=0;
                r=((1 - exp(-x[2]*a))*x[0]);
                if(x[7] < 0.0001)
                        x[7] = 0.0001;

op=x[6]*pow((exp(1)*d/x[7]),NN)*exp(-d/x[7]*NN)*sin(2*pi*x[8]*d+x[9]*pi/180);
                y=r+b+op;
        }

/* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/

                 return(y);
}
```

REFERENCES

[1] Dewar, J. and McKendrick, J.: The physiological action of light, J. Anat. Physiol. **7:** 275, 1873.

[2] Einthoven, W. and Jolly, W.: The form and magnitude of the electrical response of the eye to stimulation by light at various intensities, Quart. J. Exp. Physiol. **1:**373, 1908.

[3] Granit, R.: The components of the retinal action potential and their relation to the discharge in optic nerve, J. Physiol. **77:** 207, 1933.

[4] Kahn, R. and Lowenstein, A.: The electroretinogram, Graefe's Arch. Ophthal. **114:** 302,1924.

[5] Hartline, H.: The electrical response to illumination of the eye in intect animals, including the human subject and in decerebrate preparaios, Am. J. Physiol. **73:** 600, 1925.

[6] Granit, R.: Sensory mechanism of the retina, Oxford University Press, 1947.

[7] Cobb, W. and Morton, H.: An new component of human electroretinogram, J. Physiol. **123:** 36, 1953.

[8] Brown, K. T.: The electroretinogram: its components and their origin, Vision Res. **8:** 633, 1968.

[9] Ogden, T. E.: The oscillatory waves of the primitive electroretinogram, Vision Res. **13:** 797, 1973.

[10] Wachtmeister, L. and Dowling, J.: The oscillatory potentials of the mudpuppy retina, Invest. Ophthalmol. Visual Sci. **17:** 1176, 1978.

[11] Yonemura, D. and Tsuzuki, K.: Clinical importance of the oscillatory potential in the human ERG, Acta. Ophthalmol. Suppl. **70:** 115, 1962.

[12] Pan, H.: Oscillatory potentials model and parameter identfication, Master Thesis, NJIT, Newark, NJ, 1991.

[13] Jang, S.: Model, parameter identification of the oscillatory potential in the ERG, Master Thesis, NJIT, Newark, NJ, 1993.

[14] Vander, A., Sherman, J. and Luciano, D.: Human physiology, Fifth Edition, McGraw-Hill Publishing Company, 1990.

[15] Berson, E., Gouras, P. and Hoff, M.: Temporal aspects of the electroretinogram, Arch. Ophthalmol. **81:** 207, 1969.

[16] King-Smith, P., Loffing, D. and Jones, R.: Rod and cone ERGs and their oscillatory potentials, Invest. Ophthalmol. Visual Sci. **27:** 270, 1986.

[17] Castro, J.: Electroretinogram modeling: parameter identification & intensity response function fits, Master Thesis, NJIT, Newark, NJ, 1991.

[18] Peachey, N. S., Alexander, K. R. and Fishman, G. A.: Rod and cone system contributions to oscillatory potentials: an explanation for the conditioning flash effect, Vision Res. **27:** 859, 1987.

[19] Robert A. Moses: Adler's physiology of the eye, The C V Mosby Company, 1970.

[20] Marmor, M. F., Arden, G. B., Nilsson, S. G., Zrenner, E.: Standard for clinical electroretinography, Arch. Ophthalmol. **107:** 816, 1989.