NASA Contractor Report 4140

Identification of Visual Evoked Response Parameters Sensitive to Pilot Mental State

G. L. Zacharias
Charles River Analytics Inc.
Cambridge, Massachusetts

Prepared for
Langley Research Center
under Contract NAS1-17816

National Aeronautics
and Space Administration

**Scientific and Technical
Information Division**

1988

IDENTIFICATION OF VISUAL EVOKED RESPONSE
PARAMETERS SENSITIVE TO PILOT MENTAL STATE

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The work reported here was performed under sponsorship by NASA Langley Research Center, under NASA Contract NAS1-17816. The Technical Representative of the Contracting Officer was Dr. Alan T. Pope of NASA LaRC.

The author wishes to thank the Technical Monitor, Alan Pope of the Crew-Vehicle Interface Research Branch, NASA LaRC, for his continuing interest, encouragement and direction on this project, and for many enjoyable technical discussions over the course of the program; Linda Haugh of Bionetics for her management of the experimental series conducted at LaRC, and for her spirited discussions throughout the effort; Andy Junker of USAF/AAMRL for generously sharing ssVER techniques and research results obtained in his own studies; Dan Burdett of PRC/Kentron for his able assistance with the experimental hardware and computer facility; Mary Loving of PRC/Kentron for her development of the ENSMBL program and general software assistance; Diana Othon of CRA for her assistance in VERSOS software development; and, finally, Pauline O'Donnell of CRA for her expertise in the creation and editing of this report.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In support of the overall goal of developing an objective and reliable cognitive loading indicator, considerable research has been directed at quantifying the relationship between internal mental workload and external physiological state. Several studies have focused on the physiological state reflected in the measured electroencephalograph (EEG), in particular, the visual evoked response (VER) which arises in connection with an experimentally-controlled evoking stimulus presented via the visual modality. Most studies have concentrated on the transient VER (tVER), which uses an impulse-like strobe stimulus to "excite" the VER. A smaller number of more recent studies have focused on the steady-state VER (ssVER), which uses continuously modulated light levels to evoke a measurable EEG response. Our focus in this effort is on the ssVER, and its potential for development into a reliable mental state indicator.

1.1 Study Objectives

The basic goal of the study reported here is to develop and validate analysis techniques to identify features of the ssVER that correlate with mental state, in particular, cognitive loading. The desire is to avoid the current proliferation of arbitrary features and metrics which characterize much of VER research, and focus on a rational characterization of the ssVER "system." This naturally leads to a systems approach to the problem, and the development and identification of a functional input/output model of the ssVER. The basic goal, then, is to develop such a model, and apply it in an analytic/experimental program aimed at developing an EEG-based cognitive loading indicator.

Supporting program objectives are the specification of the stimulus/response protocols to be used in experimental validation studies, the

specification of the analysis techniques to be used for response modeling, their implementation in a software package fully compatible with the sponsoring LaRC facility, and their application to ssVER data generated in one or more controlled mental loading experiments.

1.2 General Technical Approach

Our overall technical approach is a three-stage process: 1) a review of the basic systems identification background relevant to the problem of identifying the functional characteristics of the VER; 2) the development and implementation of a software package for experiment control and data analysis; and 3) the demonstration of the proposed identification and modeling techniques, via controlled mental loading experiments.

To provide some background for the type of system identification techniques required for a rational analysis of the VER, we review basic input/output functional modeling. The description is done in mathematical terms to provide a basis for the interpretation of conventional (and unconventional) VER analysis techniques, in a more formally defined functional modeling context.

The review attempts to formalize the basic input/output structure of the VER, with the proposal of a quasi-linear model structure for the response: a response comprised of a "signal" portion which is linearly dependent on the input, and a "noise" portion which is independent of the input. The implications for identification of the impulse response and the steady-state response are then discussed, and related to the conventional tVER and ssVER techniques already in use. The review then discusses the implications of different ssVER identification stimuli, focusing on periodic impulsive flashing and sinusoidal continuous modulation. The review is concluded with a discussion of quasi-linear modeling issues, which center on sum-of-sines

stimulation of the ssVER, and subsequent identification of model structure and parametric dependence on imposed loading.

The second stage of our technical approach focuses on the development and implementation of a software package for experiment control, data analysis, and model identification. The package consists of four main programs:

1) VERRUN: This provides for pre-run setup of the experimental parameters, generation of the run-time sum-of-sines (SOS) stimulus, recording of the resulting response, and calculation of simple post-run statistics.

2) VERNAL: This supports time- and frequency-domain analysis of the time histories recorded by VERRUN, and provides for computation of the EEG RMS signal levels, and ssVER transfer function and remnant spectra.

3) ENSMBL: This calculates ensemble average statistics across individual subject runs, for the time- and frequency-domain analysis data generated by VERNAL.

4) MODLER: This supports the fitting of specified analytic transfer functions to the ensemble average transfer function data generated by ENSMBL, and supports the generation of an optimized model parameter set.

These programs provide for a full capability from data generation to model analysis, and can support, via direct expansion, an advanced modeling program under future research efforts.

The third stage of our technical approach consists of a demonstration of the proposed identification and modeling techniques, via direct experimentation and analysis. This effort begins with a brief review of past ssVER identification efforts, to uncover dominant frequency domain trends, and to specify the desired identification bandwidth. A pre-experimental design effort then focuses on a specification of appropriate time base parameters (sample rates, run times, etc.), a specification of the required sum-of-sines stimulus parameters (amplitudes, phases, etc.), and a calibration protocol to determine stimulus intensity and response amplification levels.

The demonstration experiment is conducted at NASA LaRC, using the delivered ssVER package and pre-experimental design parameters. Subjects are given three tasks: a null task comprised of viewing a blank display (baseline); a task testing visual perceptual processing; and a task testing encoding of working memory.

The VERRUN package is used to control the run-time ssVER stimulus generation, response recording, and data file generation. Following the experimental runs, the VERNAL package is used to generate corresponding single-run RMS level and frequency-domain files. These files are then grouped according to task loading conditions imposed, and the ENSMBL package is used to compute single-subject across-replication ensemble files, comprised of the average RMS level and frequency-domain metrics. The MODLER program is then used to fit very simple transfer function models to the observed data, to demonstrate the method's descriptive simplicity.

1.3 Study Results

The primary result of this study is the development and demonstration of systems analysis techniques for modeling the ssVER and relating it to cognitive loading. The major study findings supporting this development and demonstration effort can be summarized as follows.

The review on input/output functional modeling conducted under this effort provides a general framework for relating conventional transient VER (tVER) and steady-state VER (ssVER) techniques already in use; it also provides a basis for the development of advanced VER identification methods, and corresponding stimulus/response models. As noted in the review, the tVER technique has a number of short-comings, including: the potential for amplitude response saturation, which hampers any linear modeling effort; a lack of generated remnant statistics, which may eventually provide important

clues as to VER function and workload correlation; and, an overdependence on ad hoc time-domain features, which directs attention away from the essential transfer characteristics of the VER system. The conventional ssVER technique using repetitive strobe stimulation extends the tVER approach into the frequency domain, but brings with it its own set of problems, including: potential confounding of responses due to harmonic distortion; inflexibility in stimulus amplitude and frequency; and, stimulus predictability on the part of the presumably "causal" subject.

Many of these problems are avoided or ameliorated when using sum-of-sines (SOS) stimulation of the ssVER, in conjunction with quasi-linear model analysis. The basic identification procedure focuses on the transfer features of the VER system itself (structural form and parametric values), and serves to separate the response into input-related and system-generated components. The approach also allows for the quantification of remnant response which is uncorrelated with the input, and which may reflect cognitive loading effects, such as seen in alpha-suppression. The SOS-based ssVER also provides, via appropriate adjustment of the stimulus parameters, means for: minimizing the effects of amplitude saturation, by distributing the stimulus power across a wide frequency band; avoiding the confounding effects of harmonic distortion by appropriate probe frequency selection; maximizing reliability in the transfer estimates by selective "shaping" of the SOS spectrum; and ameliorating the effects of stimulus predictability, by random phasing of the SOS components.

The second major finding of the study concerns the development and demonstration of an integrated software package for the control and analysis of ssVER cognitive loading experiments. Four packages were developed under the study: VERRUN, VERNAL, ENSMBL, and MODLER. Operation of the VERRUN package at the LaRC facility demonstrated the flexibility of the software in

pre-run set-up tasks, and its ease of operation during run-time control. Post-run execution of the VERNAL package demonstrated the generation of a variety of single-run time- and frequency-domain ssVER measures, under both interactive control and batch mode operation. Subsequent processing of the single-run data by the ENSMBL package provided a direct means for generating across-replication and across-subject ensemble response statistics. Finally, operation of the MODLER package demonstrated how interactive software can support the quasi-linear ssVER development effort, and provide the analyst with a direct means of evaluating candidate response models.

The third major set of study findings were obtained from a pilot ssVER experiment conducted at LaRC during the course of this effort. Subjects were tested across three task loading conditions: a null task, a Probability Monitoring Task (PMT), and a Continuous Recall Task (CRT). A number of replications were made under each task, to yield several single-subject across-replication ensemble-average frequency-domain measures of the ssVER. Subsequent model fits of the ensemble data means were based on a simple variable-gain delay transfer function model.

The results presented here show that the transfer functions in all three task loading conditions are reasonably well-modeled by a simple two-parameter gain/delay model. The model accounts for the measured flat gain and linear phase trends with frequency, across the bandwidth of interest and the task triplet. On the basis of the preliminary data set and analysis results presented here, one may conclude the following. First, there does appear to be gain enhancement over baseline, when the subject is loaded by either of the two tasks (PMT or CRT). A corresponding time delay difference is not to be seen, however, remaining fixed at about 70 msec across all three conditions. Second, there does not appear to be any significant difference between the two loading tasks, in terms of mean measurement values. There are, however,

apparent differences in data variability with task and frequency. Finally, the data trends, across the three conditions, can be quite well modeled with an exceptionally simple analytic transfer function: a variable-gain delay. The use of more complex models would not appear to be warranted by the data at least over the frequency range studied.

In short, we have demonstrated how a systems approach to functional modeling of the ssVER can be the basis for the eventual development of a rational and reliable ssVER-based cognitive loading indicator. The review we conducted shows how both tVER and ssVER research is related at the functional stimulus/response level, and the corresponding software development effort demonstrates how basic identification techniques can be applied directly to ssVER characterization. The pilot experiment conducted under this study provided a test of this overall procedure, and the results show that a very simple model can indeed capture the basic dynamic response of the ssVER, under different cognitive loading tasks. It remains to be seen, however, whether the observed loading sensitivity holds up over a larger subject base and a wider range of tasks, or if individual subject differences will tend to dominate the ssVER. It should be clear, however, that the general methodology developed and evaluated here can serve as a starting point for a more concerted effort aimed at developing a sensitive ssVER-based workload metric.

1.4 Report Outline

This report summarizes and documents the results of our effort to develop and validate analysis techniques for modeling the ssVER and relating it to cognitive loading.

Chapter 2 provides a review of functional modeling and its relation to the ssVER. In the chapter, we define a variable-parameter quasi-linear model of the ssVER (section 2.1), describes how such a model is related to the VER

impulse response (section 2.2) and to conventional tVER identification (section 2.3), and discuss the basis for ssVER identification, using impulse trains and sums-of-sines (section 2.4).

Chapter 3 discusses the implications for sums-of-sines (SOS) stimulation and quasi-linear model identification of the ssVER. In the chapter, we specify how the SOS stimulus is used to estimate the model transfer function and remnant spectrum (section 3.1), describe how model structure and parameter values can be inferred from the frequency-domain data (section 3.2), and outline basic requirements for ssVER stimulus protocol and model analysis (section 3.3).

Chapter 4 summarizes the results of the development and demonstration effort, in going from experimental data to analytic models. In the chapter, we describe the VERSOS software package developed for experimentation, analysis, and modeling (section 4.1), outline the pre-experimental stimulus design effort (section 4.2), and summarize the experimental data trends and model results generated under a pilot ssVER study conducted at LaRC (section 4.3).

Chapter 5 summarizes the study results (section 5.1), presents conclusions (section 5.2), and outlines recommended areas for further research (section 5.3).

2. IDENTIFICATION OF THE VISUAL EVOKED RESPONSE

An earlier literature review covering the physiological assessment of mental state found that EEG evoked response (ER) measures show a fair amount of promise for the eventual development of sensitive and reliable workload correlates (Zacharias (1980)). However, the review also found a proliferation of arbitrary metrics and a wide variety of "key features" advocated by different researchers. On the basis of this review, it became clear that what was needed was a more rational approach to the central problems of EEG signal processing and feature analysis.

Two factors contributed to this situation: a lack of signal processing expertise; and a fundamental lack of understanding concerning the characteristics of the EEG "black box." The former leads to an often capricious selection of signal metrics, while the latter promotes an unending search for a single "key" feature to be correlated with mental state. This is not meant to suggest, however, that a detailed neural model of the brain must be constructed before reliable EEG-based mental state correlates can be developed. On the contrary, we believe that significant progress can be made by largely ignoring the detailed structure, and simply concentrating on the functional characteristics which relate input (evoking stimulus) to output (evoked response).

By taking a "black box" systems approach to the problem, and setting as a goal the development of a functional input/output model of the VER, one naturally sets the stage for the rational development of an EEG-based mental state correlate. This development can be most conveniently visualized as a two-stage process: a model identification stage, in which an input/output VER model is inferred from the data; and a feature identification stage, in which one searches for feature correlates of the internal mental state.

This systems identification approach is a standard tool in the engineering analysis and modeling of dynamic systems, but there have been many applications in non-engineering systems. The work we are most familiar with concerns the modeling of the human pilot or operator. Here, early workers recognized the advantages of a frequency-domain description of the operator (see, for example, Tustin (1947)), which related input (operator display) to output (operator control) in a fashion which was relatively independent of the input. This first-order decoupling of stimulus from response led to operator models characterized by a small number of parameters, which, with further research, were found to vary in a relatively systematic fashion with task factors. Subsequent pilot modeling work has yielded a progressive series of functional input/output models, embodying ever greater levels of abstraction from the basic stimulus/response data supporting their development.

We believe that this approach is directly applicable to the current effort aimed at developing a sensitive VER metric. The success of such an approach naturally requires some demonstrated correlation of VER features with internal state, but this has already begun to be shown in the literature. For example, in transient VER studies involving a single stimulus pulse, the amplitude and latency of the P_{300} peak shows a definite correlation with imposed task workload (Wickens et al (1977), O'Donnell and Spicuzza (1977)); likewise P_{200} latency has been shown to relate to the visual information processing workload imposed on the subject (O'Donnell and Spicuzza (1977)). For steady state VER (ssVER) studies involving sinusoidal modulation of stimulus intensity, it has been shown that response phase lags correlate with task difficulty (Wilson (1979)), over some limited frequency band of interest.

The features used in these studies were chosen fairly arbitrarily, but there has been a recent trend towards the more rational approach of model-based feature extraction and evaluation. For example, after researchers at

USAF/AAMRL began investigating the ssVER for workload assessment, they found that a frequency-domain description of the ssVER can efficiently capture a number of significant features in the recorded responses (Junker and Peio (1983, 1984)). Preliminary results also indicated distinctive changes in the describing function characteristics with cognitive loading. More recent results have shown that subjects with high alpha-band response also show, under cognitive loading conditions, significant reductions in alpha-band gain and remnant (Junker (1986)). Subsequent quasi-linear model analysis of such data demonstrate how this data can be compressed to a small number of significant model parameters (Zacharias (1982), Levison and Zacharias (1984), Junker (1986)).

To provide some background for the type of system identification techniques we require for VER analysis, we present in the remainder of this chapter a summary account of basic input/output dynamic functional modeling. Although the description is done in mathematical terms, it should not be taken as a rigorous derivation; rather, it should be viewed as a heuristic argument which will allow us to interpret conventional (and unconventional) VER analysis techniques in terms of more precisely defined functional modeling concepts. In this way, we can bring to bear modern system identification techniques to the problem of VER analysis, and justify an appropriate approach to the development of a reliable and accurate VER model.

2.1 General Functional Model

To begin the discussion, consider the situation in which the VER is being recorded from a single electrode site, in response to a strobe light stimulus. Suppose we characterize this stimulus as having an intensity profile x(t) which varies with time; as an example, it could be the simple pulse illustrated in figure 2.1a. Suppose we also characterize the resulting evoked

response, by associating with it a recorded voltage time history y(t); an example waveform is illustrated in figure 2.1b. Finally, suppose we had means of specifying the pilot's mental state m, during the recording interval. We might then conjecture that an appropriate functional model, relating stimulus to response, could be specified in the following manner:

$$y(t) = f[x(t), \underline{p}_m(m), \underline{p}_c, t] \qquad (2.1)$$

Here, we assume we have some specified functional form, defined by f[ ], which relates the stimulus x(t) to the response y(t), and which may vary with time (hence the inclusion of t in the argument of f[ ]). Associated with the function f[ ] are two parameter vectors, $\underline{p}_m$ and $\underline{p}_c$; together with the form of f[ ], they specify in detail just how y(t) relates to x(t). We assume that the first parameter vector, $\underline{p}_m$, is comprised of all those parameters which vary with mental state (hence the argument m); the second, $\underline{p}_c$, is comprised of all remaining parameters, and provides a convenient means of accounting for all factors unrelated to internal mental state (such as color of the stimulus, recording site, electrode impedance, etc.)

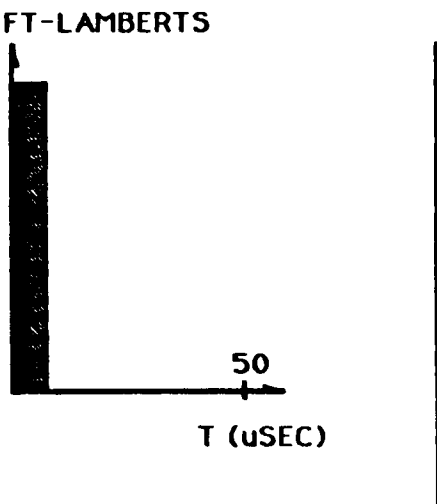

Figure 2.1a: Stimulus Pulse

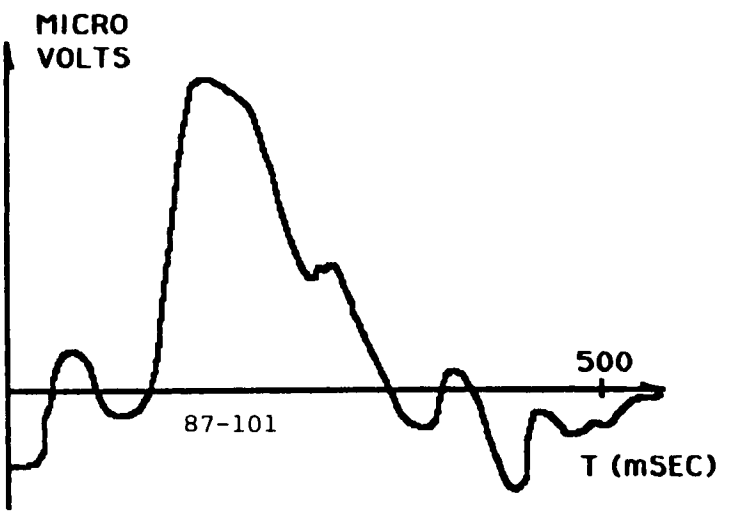

Figure 2.1b: Visual Evoked Response

Note that we have, in effect, specified a model of the VER, albeit in a very general form. To be able to suggest appropriate ways of identifying both the

form and parametric values of this model, we need to consider some simplifying assumptions, and how they might be justified on the basis of experimental protocol and used to advantage in the system identification problem.

If the experimental design provides for "good" control over extrinsic experimental conditions, we might assume that the non-state-related parameter vector $\underline{p}_c$ remains constant throughout a recording session. If we now also assume, for this initial identification exercise, that the mental state somehow stays constant, then the mental state parametric vector $\underline{p}_m$ might also be reasonably assumed to remain a constant throughout the session.* If we now combine the two parameter vectors into one (constant) vector $\underline{p}$, we can rewrite (2.1) as:

$$y(t) = f[x(t),\underline{p},t] \qquad (2.2)$$

where it is understood that certain components of $\underline{p}$ explicitly depend on the (fixed) mental state m.

A further simplification can result by making the very basic assumption that the response is fundamentally time-invariant. The relation above says that if we stimulate or "probe" the system with a signal x(t), we will measure a response y(t); time-invariance, in effect, says that if we delay (or advance) in time the stimulus signal x(t), to obtain x(t+T), then we should observe a correspondingly delayed (or advanced) response signal, y(t+T), where T is some arbitrary time shift. Obviously no physical system (with a finite lifetime) is time-invariant, but if we consider sufficiently small time shifts (so that T is limited to some finite range), many systems can be modeled quite

---

*This simplification obviously does not take into account potential factors such as adaptation, habituation, or fatigue, but, for now, we assume that the experiment has been somehow designed so as to ensure that these will be insignificant contributors to the overall response.

accurately by attributing time-invariance to them. For this discussion, we will make the assumption that the system we are attempting to identify can likewise be assumed to be time-invariant, so that (2.2) becomes:

$$y(t,\underline{p}) = f[x(t),\underline{p}] \tag{2.3}$$

where we have removed the explicit dependence of f[ ] on the time t.*

We now make one final assumption concerning system structure, one which is perhaps the most critical to our eventual choice of an appropriate identification technique. Specifically, we assume that the response function f[ ] can be effectively "decomposed" into a linear component g[ ] and a noise component n, so that (2.3) becomes:

$$y(t,\underline{p}) = g[x(t),\underline{p}] + n(t,\underline{p}) \tag{2.4}$$

where we have allowed for the eventuality that both components may depend on the parameter vector $\underline{p}$.

This equation implies that if we have a stimulus x(t), then one component of the response, g(t), should be linearly related to the stimulus; that is, if we were to scale the stimulus so that we were driving the system with the signal ax(t), then we should obtain a correspondingly scaled linear response component ag(t), where a is some arbitrary scale factor. Again, as with time-invariance, no physical system is truly linear, but if we consider a sufficiently small range of scale factors (so that a is limited to some finite range), many systems can be modeled quite accurately in a linear fashion. We will assume here, that over some appropriate range of stimulus magnitude, the system we are attempting to identify can likewise be modeled as having a linear response component.

One immediate advantage in separating out a linear response component is that we can make full use of linear systems theory. Specifically, it can be

---

*and made explicit the parametric dependence of y on $\underline{p}$.

shown that if g[ ] represents a linear time-invariant (and causal) input/output function, it can be represented in the following integral form (Truxal (1955)):

$$g[x(t),\underline{p}] = \int_0^\infty h(\lambda,\underline{p})x(t-\lambda)d\lambda \qquad (2.5)$$

where h( ) is the parameter-dependent "impulse response function" defining the linear transformation from input to output. Integration is performed over the dummy time variable λ. This allows us to restate the response equation (2.4) in an equivalent form, as follows:

$$y(t,\underline{p}) = \int_0^\infty h(\lambda,\underline{p})x(t-\lambda)d\lambda + n(t,\underline{p}) \qquad (2.6)$$

The usefulness of this form of the response equation will be demonstrated shortly.

The form of the input/output relationship specified by (2.4) also implies a "noise" contribution n to the overall measured response y. We assume that the noise statistics do not vary with time (in line with our earlier assumption of time-invariance), so that standard measures such as noise mean and variance can be represented by fixed constants. We further assume, without loss of generality, that the noise mean is zero.* Finally, we assume that the noise component n is uncorrelated with the stimulus x, so that the portion of the response which is correlated with the stimulus is due entirely to the linear component g of (2.4). In effect, the "magnitude" of the noise provides a measure of the goodness-of-fit of a purely linear model of the VER.

---

*If it were non-zero, we could simply define a new noise signal n', where n' = n-$\bar{n}$ (where the overbar denotes a mean value), and a new response signal y', where y' = y-$\bar{n}$, and reformulate (2.4) and (2.6) in terms of y' and n'.

This "quasi-linear" model has been used extensively and with considerable success in past system modeling efforts. The advantage of using such a model lies not only in the relative strength of analytic techniques which can be used in defining its detailed structure, but also in the fact that such a model provides an inherent measure of system linearity: the ratio of correlated response power to uncorrelated power, appearing in the response output signal y. Clearly, if most of the response signal is uncorrelated with the input (i.e., the noise component is dominant), then primary attention must be given to an appropriate characterization of the noise statistics, and only secondary attention need be given to that of the linear response dynamics. This model structure can provide us with a self-contained indicator of the emphasis to be placed on the detailed modeling of the two response components.

We now turn to some of the implications for system identification, imposed by the input/output model of (2.6).

2.2 Identification of System Impulse Response

Consider the situation in which we measure the transient VER (tVER), by stimulating with a short pulse of light. Here we inject an input pulse having a pulse width typically several orders of magnitude smaller than the fastest time constant of the VER. We can thus model the pulse x(t) as effectively an impulse $\delta(t)$, so that, from (2.6) the corresponding response on the ith stimulus trial will be given by:

$$y_i(t,\underline{p}) = \int_0^\infty h(\lambda,\underline{p})\ \delta(t-\lambda)d\lambda + n_i(t,\underline{p}) \qquad (2.7)$$

so that, from the properties of the impulse function (see, for example Truxal (1955)):

$$y_i(t,\underline{p}) = h(t,\underline{p}) + n_i(t,\underline{p}) \qquad (2.8)$$

so that the response should be simply the "impulse response" plus a specific noise signal associated with the ith trial. If we now ensemble average across N such input-output measurement trials, as is commonly done for signal enhancement in VER processing, we obtain an estimate for the impulse response $\hat{h}$, given by:

$$\hat{h}(t,\underline{p}) \equiv \frac{1}{N} \sum_{i=1}^{N} y_i(t) = \frac{1}{N} \sum_{i=1}^{N} h(t,\underline{p}) + \frac{1}{N} \sum_{i=1}^{N} n_i(t,\underline{p}) \tag{2.9}$$

or, since the time-invariant impulse response h is unchanged over individual trials,

$$\hat{h}(t,\underline{p}) = h(t,\underline{p}) + \frac{1}{N} \sum_{i=1}^{N} n_i(t,\underline{p}) \tag{2.10}$$

As the number of trials N increases, the second term above approaches the expectation of the noise $\bar{n}$, which is zero, so that

$$\overline{\hat{h}(t,\underline{p})} \equiv \lim_{N=>\infty} \hat{h}(t,\underline{p}) = h(t,\underline{p}) \tag{2.11}$$

so that we obtain an unbiased estimate of the impulse response. The convergence speed of the above relation will depend on the noise statistics characterizing the n_i of (2.10).

We can estimate the noise statistics, once $\hat{h}$ is obtained, by first estimating the individual trial noise sequences on the basis of (2.8), via:

$$\hat{n}_i(t,\underline{p}) \equiv y_i(t,\underline{p}) - \hat{h}(t,\underline{p}) = n_i(t,\underline{p}) + \epsilon_N(t,\underline{p}) \tag{2.12}$$

where, from (2.8) and (2.10)

$$\epsilon_N(t,\underline{p}) = -\frac{1}{N} \sum_{k=1}^{N} n_k(t,\underline{p}) \tag{2.13}$$

The estimate from (2.12) can then be used directly in computing the estimated noise autocorrelation function $\hat{\phi}_{nn}(\tau,\underline{p})$, whose statistics in turn, will depend on the statistics of ϵ_N above. The autocorrelation function estimate $\hat{\phi}_{nn}$ can then be used to formally define the noise power spectral density Φ_{nn}, which in turn, can serve as the basis for defining a shaping filter for simulated or modeled VER noise generation. Additional details of this process can be found

in classical texts on dynamic random processes, such as Lanning and Battin (1956).

This type of noise sequence generator, in conjunction with the impulse response model estimate $\hat{h}$, can then serve as the basis for a quasi-linear time-domain impulse response description of the system being identified.

2.3 Identification of the Transient VER

From our review of the literature, it would appear that most researchers use the basic approach of ensemble averaging, defined by (2.10), when working with the transient VER (tVER). It requires a minimum of signal processing (off-the-shelf hardwired averaging computers now being commonplace), it provides a direct means of viewing signal enhancement with trial replication (by observation of successive ensemble-average time histories), and it results in a relatively easy-to-comprehend processed output: a single time history of the EEG voltage fluctuations. Two points are worth noting, however.

First, most researchers seem to ignore the noise component of the response n, and concentrate solely on the impulse response function h. If (2.6) is to be accepted as a basic model of the VER, then clearly both terms contribute to the response. We are not suggesting that individual noise sequences are likely to provide any significant insight into modeling the VER; however, the statistical properties of the noise may very well provide some important clues as to VER function. In particular, we might conjecture that the noise statistics (including any relevant spectral characteristics) may be strongly related to some of the experimental factors being investigated via the VER, such as internal mental state. To keep this possibility open, we have deliberately specified, in (2.6), that the noise term be a function of the experimental parameter vector $\underline{p}$, either explicitly, or implicitly, via its impact on the statistics associated with the "noise generator" responsible for

$n(t,\underline{p})$. Unfortunately, from an experimental standpoint, it would appear that most of this "noise data" is being discarded by current researchers, in their use of ensemble averaging, and their concentration on identifying the system's impulse response function $h(t,\underline{p})$.

A second point to be made regarding tVER research has to do with the type of "signal processing" used once a VER is obtained. Because it is usually a well-defined time-history which can provide a visual record of voltage fluctuations versus time, many researchers have taken the "direct" approach of trying to correlate specific features of this history with particular experimental factors they have manipulated during the recording sessions. Since a continuous time history has, in theory, an infinite number of points associated with it, it would seem that an infinite number of "features" (and combinations of same) could be associated with any single time history. We suspect that this basic characteristic of the continuous signal is the fundamental cause of the numerous and seemingly-arbitrary tVER "features" reported on in the literature (e.g., P_{300} correlation studies of workload).

The point we wish to make here is that such exercises may fail to uncover the fundamental relationship (if one exists) between the tVER and internal mental state. From a systems analysis standpoint, the fundamental "features" of the impulse response $h(t,\underline{p})$ are <u>not</u> the time-domain features; rather, they are the parameters which comprise the parameter vector $\underline{p}$, specifically, those associated with our hypothesized mental state parameter vector $\underline{p}_m$.

2.4 Identification of the Steady-State VER

To this point, we have discussed the transient VER and its relation to time-domain modeling of the impulse response function; we now turn our attention to the steady-state VER (ssVER), and corresponding frequency-domain modeling. Before discussing specific stimulus protocols, however, it is

appropriate to provide some general background concerning frequency-domain measurements made on a noisy linear time-invariant system.

If we have the time history of some arbitrary (real) signal z(t), we can define the associated Fourier transform according to:*

$$Z(\omega) \equiv \{z(t)\} = \int_{-\infty}^{\infty} z(t)e^{-j\omega t}dt \tag{2.14}$$

where ω is the frequency, and j is the square root of negative one. Since $Z(\omega)$ will, in general, be complex, we can represent it as follows:

$$Z(\omega) = a(\omega) + jb(\omega) \tag{2.15}$$

where the real and imaginary components $a(\omega)$ and $b(\omega)$ will be dependent on the particulars of z(t). The frequency-dependent amplitude A and phase ϕ associated with $Z(\omega)$ are then defined by

$$A(\omega) = [a^2(\omega) + b^2(\omega)]^{1/2} \tag{2.16a}$$

$$\phi(\omega) = \tan^{-1}[b(\omega)/a(\omega)] \tag{2.16b}$$

so that, by progressing through relations (2.14) through (2.16), we can define amplitude and phase characteristics of any reasonably well-behaved time-domain function.

In particular, we can consider the impulse response function $h(t,\underline{p})$. From (2.14) and (2.15) we can define the corresponding frequency-domain "transfer function" according to:

$$H(\omega,\underline{p}) = \int_{-\infty}^{\infty} h(t,\underline{p})e^{-j\omega t}dt = \int_{0}^{\infty} h(t,\underline{p})e^{-j\omega t}dt \tag{2.17}$$

where we have changed the lower limit of integration since we assume that h is zero for negative time, reflecting the fact that h is a "causal" system and does not respond before the impulse is applied at time zero. This allows us

---

*assuming z(t) is piecewise differntiable and that $\int_{-\infty}^{\infty} |z(t)|dt$ exists.

to express H in terms of the Laplace transform, defined for an arbitrary "well-behaved" function z(t), by

$$\{z(t)\} \equiv \int_{-\infty}^{\infty} z(t)e^{-st}dt \qquad (2.18)$$

From (2.17), then, we see that we can obtain H by substituting $j\omega$ for s, in the Laplace transform:

$$H(\omega,\underline{p}) = \{h(t,\underline{p}\}\Big|_{s=j\omega} \qquad (2.19)$$

which provides us with a convenient means for calculating H, given an analytic impulse response function h. In this manner, we can specify the gain and phase characteristics (with frequency) of the linear portion of the system being measured.

There are several ways to obtain this ssVER transfer function from measurements made on the noisy system. One way, which is immediately suggested from the form of (2.17), involves three basic steps:

1) Conduct a tVER experiment and use conventional ensemble averaging to estimate the time history of the impulse response function $h(t,\underline{p})$.

2) Identify a model structure and parameter vector which, in some sense, best "fits" the empirical time history, so that $h(t,\underline{p})$ can be analytically defined in terms of t and $\underline{p}$.

3) Perform the analytic transform indicated by (2.18) and (2.19), to obtain the corresponding transfer function $H(\omega,\underline{p})$.

This approach, however, is very sensitive to the convergence rate of the ensemble average to the true impulse response, and is also strongly affected by the choice and accuracy of the analytic approximation to the impulse response. The approach also fails to provide a convenient direct means of characterizing the additive noise component.

An alternative to this transfer function specification problem involves the use of direct frequency-domain measures. We begin by defining the cross-correlation function for two arbitrary signals, x(t) and y(t), as follows (for background, for example, see Lanning and Battin (1956)):

$$\phi_{xy}(\tau) = \overline{x(t)y(t+\tau)} \tag{2.20}$$

where the overbar denotes an expectation or average, and the time variable τ denotes the relative time shift between the two signals being correlated. If, consistent with our early notation, we take x(t) to be the VER stimulus, and y(t) to be the VER response, and make use of the response equation introduced earlier by (2.6) it then follows directly that the stimulus/response cross-correlation function will be given by:

$$\phi_{xy}(\tau,\underline{p}) = \int_0^\infty h(\lambda,\underline{p})\phi_{xx}(\tau-\lambda)d\lambda \; + \; \phi_{xn}(\tau,\underline{p}) \tag{2.21}$$

where, as before, λ is a dummy variable of integration, and where we have introduced the auto-correlation function for the input, ϕ_{xx}, and the cross-correlation function between the input and the noise, ϕ_{xn}. Both are defined in accordance with (2.20).

The quasi-linear model defined by (2.6) specifies, however, that the noise is to be uncorrelated with the input. This means that the second term of (2.21) above is identically zero, so that

$$\phi_{xy}(\tau,\underline{p}) = \int_0^\infty h(\lambda,\underline{p})\ \phi_{xx}(\tau-\lambda)d\lambda \tag{2.22}$$

We are now in a position to specify the cross PSD's (cross PSD), Φ_{xy}, which, in accordance with the definition given by (2.14) is simply the Fourier transform of the corresponding cross-correlation function:

$$\Phi_{xy}(\omega,\underline{p}) \equiv \{\phi_{xy}(\tau,\underline{p})\} = \int_{-\infty}^{\infty} \phi_{xy}(\tau,\underline{p})e^{-j\omega\tau}d\tau \tag{2.23}$$

Taking the Fourier transform of (2.22), making some assumptions regarding the interchangeability of order of integration, and using the definition given by (2.23) above, it can then be shown that

$$\Phi_{xy}(\omega,\underline{p}) = H(\omega,\underline{p})\Phi_{xx}(\omega) \tag{2.24}$$

where Φ_{xx} is the auto PSD of the input signal, and, in accordance with (2.17), H is the system transfer function. This relation specifies the stimulus/response cross PSD, but a totally analogous derivation can be used to find a corresponding relation for the response auto PSD, so that:

$$\Phi_{yy}(\omega,\underline{p}) = |H(\omega,\underline{p})|^2\Phi_{xx}(\omega) + \Phi_{nn}(\omega,\underline{p}) \qquad (2.25)$$

Here, the response auto PSD is determined both by the magnitude squared of H, and the auto PSD's of x and n.

These general relations of (2.24) and (2.25) underlie a number of candidate frequency-domain identification techniques. For example, if one has the capability of computing auto and cross PSD's from measured data, one could then use (2.24) directly to obtain the transfer function, by formal division:

$$H(\omega,\underline{p}) = \Phi_{xy}(\omega,\underline{p})/\Phi_{xx}(\omega) \qquad (2.26)$$

In implementing this identification algorithm, one would replace the right-hand side PSD's by their estimates, calculated from the data. Once an estimate for H were obtained in this manner, one could then use (2.25) as the basis for calculating the "remnant" or noise spectrum, by calculating what was "left over" after one takes into consideration the linear response component:

$$\Phi_{nn}(\omega,\underline{p}) = \Phi_{yy}(\omega,\underline{p}) - |H(\omega,\underline{p})|^2\Phi_{xx}(\omega) \qquad (2.27)$$

Here, one would replace the right-hand side PSD's by their estimates (as before), and use the value for H obtained in (2.26). Thus, the above equation pair could be used for a frequency-domair specification of both the transfer function and the noise portion of the quasi-linear system model.

Note that these relations make no assumptions regarding the type of input signal x(t) used for system identification. Once it is specified, however, its auto PSD, Φ_{xx}, can be calculated and substituted into the equations as appropriate, thus requiring measurement-based calculations only for the auto and cross PSD's associated with the response signal y(t).

For example, if we chose to stimulate the VER using "white noise," then the associated PSD Φ_{xx} would be a simple constant over all frequencies. This constant, say Φ_o, could then be used to scale the cross PSD Φ_{xy}, to obtain H directly, on the basis of (2.26). Likewise, Φ_o could be used to scale the magnitude squared value of H in (2.27), for subsequent differencing with the response PSD, Φ_{yy}, and estimation of the noise spectrum, Φ_{nn}.

2.4.1 Identification of the ssVER via Periodic Flashing

If one were conducting a "standard" ssVER experiment, by stimulating with a periodic train of (impulsive) light flashes, the stimulus could be represented by the following series:

$$x(t) = \sum_{k=-\infty}^{\infty} \delta(t-KT) \tag{2.28}$$

where T is the flash period, and where, for analytic convenience, we have assumed an infinite series of impulses. The autocorrelation function of this signal is obtained by applying (2.20), to yield (see, for example Kuo (1963)):

$$\phi_{xx}(\tau) = \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta(\tau-kT) \tag{2.29}$$

so that the corresponding auto PSD, in accordance with (2.23), can be shown to be specified by

$$\Phi_{xx}(\omega) \equiv \{\phi_{xx}(\tau)\} = \frac{\omega_o}{T} \sum_{k=-\infty}^{\infty} \delta(\omega-k\omega_o) \tag{2.30}$$

where the base frequency ω_o is simply $2\pi/T$. Thus, the input auto PSD is an impulse train in the frequency domain, with impulses spaced every ω_o. Combined with the basic input/output relation specified by (2.24), this then allows us to specify the expected cross PSD:

$$\Phi_{xy}(\omega,\underline{p}) = \frac{\omega_o}{T} \sum_{k=-\infty}^{\infty} H(\omega,\underline{p})\ \delta(\omega-k\omega_o) \tag{2.31}$$

In effect, this means that the cross power spectrum of Φ_{xy} can provide us with information about H only at the discrete frequencies $k\omega_o$; that is, by using

this type of impulse train stimulus, we can only infer H at the harmonics of ω_o.

In practical terms, this is not a lot of information. Suppose, for argument's sake, that the bandwidth of the VER transfer function H were 100 Hz. Suppose also that we decided to stimulate with a steady-state impulse train, via flashes every 20 msec, yielding a base frequency ω_o of 50 Hz. From (2.31), we would then obtain only three significant measurements of H: one at zero frequency (ω=0), one at 50 Hz (ω=ω_o), and one at 100 Hz (ω=2ω_o). Higher frequency responses would not register, because of the bandwidth of H, and intermediate frequency measurements would not be available, because of the discrete form of (2.31).

This situation is further aggravated in current ssVER work by the apparently common practice of only measuring the frequency response at the stimulation frequency. Thus, only one measurement of H is obtained, even when more would appear to be available.

If, instead of the relatively high frequencies now being used, one chose a stimulation frequency, say, of 1 Hz, then one could theoretically obtain a spectral measurement every 1 Hz, from zero Hz to the bandwidth frequency of the transfer function -- in our 100 Hz bandwidth example, this would mean about 100 gain/phase measurements with which to specify $H(\omega,\underline{p})$. Such a situation would clearly afford a much richer data environment than the single point measurements now being made.

This approach, while appealing because of its potential for measurement efficiency, suffers some drawbacks, however.

The first problem concerns the regularity, in frequency, of the stimulus signal PSD, and the potential confounding of fundamental responses with harmonic responses, associated with possible system nonlinearities. If significant VER nonlinearities existed, they might very well go undetected by

the researcher, and continued use of a quasi-linear system representation might eventually lead to significant errors in modeling system response.

The second problem with this protocol concerns the uniformity, in amplitude, of the stimulus signal PSD. From (2.30) it is seen that the input PSD provides equal (area) impulses for driving the system. Since the VER is known to drop off considerably with frequency (Regan (1977)), then one would expect a considerable disparity between the relatively high response power measured at low frequencies (say, less than 10 Hz), and the relatively low power measured at mid and high frequencies. If a reasonable signal-to-noise ratio is obtained at the low frequencies, then one might expect comparatively poor ratios at the upper frequencies. Of course, one can "boost" the overall input signal level to ensure sufficiently high signal-to-noise ratios everywhere in the response spectrum, but this may lead to amplitude saturation of the VER system, thus aggravating the linear modeling problem.

The final problem with this stimulus protocol, and perhaps the most significant, concerns the use of an impulse-like flash for system stimulation. Clearly, with a short-duration large-amplitude flash, we run the risk of incurring amplitude-dependent non-linearities which increases the likelihood that our quasi-linear model will be in error. Naturally, this potential short-coming also applies to the "single-shot" stimulus/response protocol used in obtaining transient VER's.

2.4.2 Identification of the ssVER via Sum-of-Sines

We now consider a steady-state stimulus protocol which, instead of using a periodic pulse train, uses a "sum-of-sines" signal to modulate stimulus intensity. By this, we mean a signal of the form

$$x(t) = \sum_{i=1}^{N} a_i \sin(h_i \omega_o t + \phi_i) \qquad (2.32)$$

which is a summation over N sinewaves, and in which the ith wave has associated with it an amplitude a_i, a relative phase ϕ_i, and an integer harmonic multiplier h_i. By specifying a desired period T for x(t), so that x(t+T)=x(t), we can then specify the base frequency ω_o, according to

$$\omega_o = 2\pi/T \tag{2.33}$$

By choosing the harmonics as positive integers, we can ensure, for each sinewave component of the signal, that an integral number of cycles appear in one period of the stimulus x(t).

The auto-correlation function associated with this stimulus can be found from (2.20):

$$\phi_{xx}(\tau) = \sum_{i=1}^{N} \frac{a_i^2}{2} \cos(h_i \omega_o \tau) \tag{2.34}$$

Here, the relative phase information has been lost, but the function retains the basic period T. The corresponding auto PSD can be found from (2.23):

$$\Phi_{xx}(\omega) = \sum_{i=1}^{N} \frac{\pi a_i^2}{2} \{\delta(\omega - h_i\omega_o) + \delta(\omega + h_i\omega_o)\} \tag{2.35}$$

Substitution into the general stimulus/response equation given by (2.24) then results in the following expression for the cross PSD, Φ_{xy}:

$$\Phi_{xy}(\omega) = \sum_{i=-N}^{N} \frac{\pi a_i^2}{2} H(\omega,\underline{p})\ \delta(\omega - h_i\omega_o) \tag{2.36}$$

where H is the VER transfer function being measured, and where, for notational convenience, we have allowed the index i to take on negative values and defined $h_{-i}=-h_i$. This result shows that, as in the case of the impulse train stimulus, the cross power spectrum of Φ_{xy} can provide us with information about H only at a finite number of discrete frequencies,* in this case, at $\omega=h_i\omega_o$. Superficially, then, it would appear that no significant measurement

*In contrast to the infinite number of frequency measurements potentially available with the impulse train stimulus.

advantages are to be had with a sum-of-sines input. However, a direct comparison of the response equations associated with the two stimulus protocols, (2.31) and (2.36), shows significant differences.

First, as we noted earlier, the impulse train provides driving power at all harmonics of the base frequency associated with the input signal, a situation which can lead to confounding of linear system fundamental response with non-linear system harmonic response. In contrast, the sum-of-sines input provides the researcher with the capability for directly specifying all of the drive frequencies, in an independent fashion, via the selection of the harmonic multiplier set, $\{h_i\}$. The confounding problem can thus be simply avoided by merely ensuring that each h_i is a distinct prime greater than one, so that a mutually prime set of multipliers results. This ensures that a response measurement at any one input frequency will not contain harmonic power associated with any other input frequency, so that misleading trends based on quasi-linear model assumptions will be avoided.*

Second, this freedom to specify the stimulus frequencies also provides for operational flexibility in the identification process. For example, if one wishes to concentrate measurements in some small region of the spectrum (say, because one anticipates rapid variations in H with frequency in this region), then one could choose a cluster of h_i's to ensure adequate resolution. Or, one could select the h_i's on the basis of eventual plotting or curve-fitting requirements; for example, if semi-log Bode plots are anticipated, then it is appropriate to choose the h_i's to approximate a geometric series in frequency.

---

*Conversely, the presence of significant power at a harmonic of one of the input frequencies can serve to alert the researcher to the possible presence of significant non-linearities.

Third, we note that the impulse train limits the researcher to an application of power which is uniform with frequency; in contrast, the sum-of-sines input gives the researcher the capability to "shape" the input spectrum, via the specification of the sinewave amplitudes, $\{a_i\}$. Thus, if one anticipates significant differences in VER attenuation with frequency, one can minimize the problems associated with low signal-to-noise ratios by selectively "boosting" the power at the appropriate measurement frequencies, while also minimizing the potential for driving the system into a non-linear operating regime.

Finally, it is appropriate to comment on the apparent degree of "randomness" associated with the two stimulus signals. The impulse train is, of course, perfectly periodic in the short term, and therefore highly predictable. Although we are not aware of such a finding in the EEG literature, this type of stimulation could lead to "anticipatory" or "predictive" response in the VER, in which the measured response precedes the associated stimulus response. This situation would clearly violate our earlier assumption concerning "causality" of the impulse response function, and add another dimension of difficulty to the modeling problem, specifically, that of modeling the system's predictive capabilities.

In contrast, the sum-of-sines signal can be made to appear quite random and unpredictable by simply:

1) choosing a sufficiently long base period T
2) choosing a sufficiently large number of sinusoidal components N
3) randomizing the choice of phases, $\{\phi_i\}$

The first ensures that, even though the signal is periodic, its period T will far exceed the prediction time capabilities of the system being measured. The second ensures the presence of a sufficient number of unpredictable "wiggles" in the waveform, due to the contributions of the different

sinusoidal components. The last, phase randomization, ensures that, even if the system "learns" the stimulus over the course of a single trial, an entirely new stimulus can be generated for the next, by rerandomizing the phases. A comparison of the time history expression of (2.32), which includes the phase information, with the autocorrelation and PSD functions of (2.34) and (2.35), in which the phase information is effectively "lost," shows that we still stimulate the VER system with the same basic identification signal, although its realization in time can appear quite different (and unpredictable) from trial to trial. Thus, the use of a sum-of-sines signal allows us to choose stimulation parameters which ensure a random-appearing input, which, in turn, supports the development of a causal model of the VER.

3. QUASI-LINEAR VER MODEL ANALYSIS

The use of a quasi-linear VER model of the form described in the previous chapter, in conjunction with a sum-of-sines stimulation protocol, leads to significant simplifications in subsequent model analysis. In this chapter, we describe the implications for both signal processing and model development, and outline the resulting functional requirements for implementation in an integrated software package.

3.1 SOS Stimulation and the Quasi-Linear Model

The use of a sum-of-sines (SOS) stimulus provides certain simplifications in subsequent data processing and frequency analysis. To see this, we can recast the basic quasi-linear response equation of (2.6) as follows:

$$y(t,\underline{p}) = z(t,\underline{p}) + n(t,\underline{p}) \tag{3.1}$$

where, as before, n represents the noise, and where we have introduced z to represent the correlated (linear) portion of the response. A sketch of this quasi-linear model is given in figure 3.1. Since the noise is uncorrelated with the linear response, the total response PSD Φ_{yy} is given by

$$\Phi_{yy}(\omega,\underline{p}) = \Phi_{zz}(\omega,\underline{p}) + \Phi_{nn}(\omega,\underline{p}) \tag{3.2}$$

which is simply a restatement of (2.25), where

$$\Phi_{zz}(\omega,\underline{p}) = |H(\omega,\underline{p})|^2 \Phi_{xx}(\omega) \tag{3.3}$$

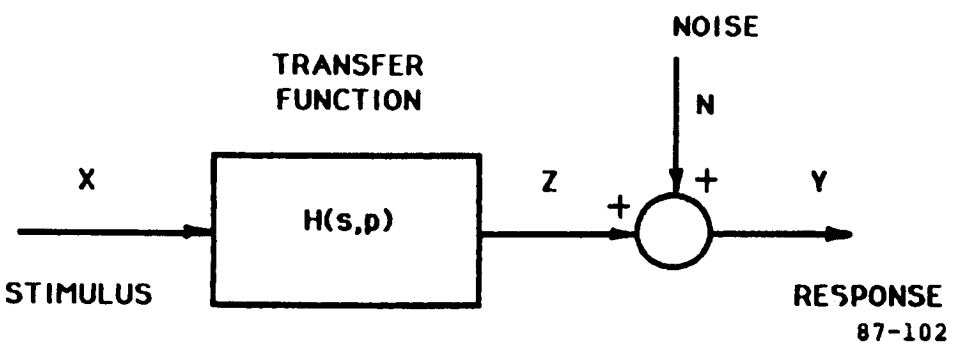

Figure 3.1: Quasi-Linear Model

Now, the use of an SOS stimulus ensures that the input signal will contain power only at a selected set of "input" frequencies (the $h_i\omega_o$ of (2.32)). Since there will be no stimulus power at non-input frequencies, any measurement of response power at non-input frequencies must be attributed to VER noise.* In terms of the response power equation of (3.2) above, this means that

$$\Phi_{yy}(\omega,\underline{p}) = \Phi_{nn}(\omega,\underline{p}) \qquad \text{for } \omega \neq h_i\omega_o \tag{3.4}$$

Thus, an estimate of the noise PSD, Φ_{nn}, in the neighborhood of a nominal input frequency, can be obtained by averaging the response power measurements over a frequency band on either side (but not including) the input frequency.

To see how this might be implemented, we assume the use of a discrete Fourier transform (DFT; see Oppenheim and Schafer (1975)) for the basic transformation of time-domain data into frequency-domain data. The transform effectively provides us with signal amplitude a and phase ϕ, at each frequency ω, to a resolution $\Delta\omega$ (the spectral "bin" size). If we chose the recording "window" size to equal the period T of the sum-of-sines signal, then the resolution $\Delta\omega$ will equal the base frequency ω_o. If this is sufficiently small (T sufficiently large), then the output power relation of (3.4), can be obtained via simple multiplication by the "bin" size ω_o:

$$\frac{1}{2}a_y^2(\omega) \simeq \Phi_{yy}(\omega,\underline{p})\omega_o = \Phi_{nn}(\omega,\underline{p})\omega_o \simeq \frac{1}{2}a_n^2(\omega) \tag{3.5}$$

where, as before, ω excludes the input frequencies $h_i\omega_o$, and where we have introduced the (sinusoid) amplitudes associated with the response and noise signals, a_y and a_n, respectively. We can then obtain an estimate for the average noise power at an input frequency by averaging over the uncorrelated

---

*and measurement system noise, which, for our purposes, we assume to be negligibly small.

response power, in the neighborhood of that frequency. Assuming discrete frequency measurements, we could average as follows:

$$\overline{a_n^2}\,(h_i\omega_o) = \frac{1}{M} \sum_{k=k_{lo}}^{k_{hi}} a_y^2(k\omega_o) \qquad (k \neq h_i) \tag{3.6}$$

where M is the number of measurements being averaged (equal to $k_{hi}-k_{lo}$) and where k_{lo} and k_{hi} are the transform indices defining the "low" and "high" frequency bins delineating the averaging neighborhood surrounding the input frequency $h_i\omega_o$. Note that the averaging specifically excludes the correlated response power at $h_i\omega_o$.

With this estimate for the noise power, we can then directly estimate its fractional contribution to the total response power, at each measurement frequency, by simply calculating the ratio $\overline{a_n^2}/a_y^2$. If this is large (say, greater than 25%), then we are alerted to the fact that our correlated response measurement may not be particularly reliable. If the converse situation holds, however, we can obtain an estimate of the correlated response power by use of (3.2):

$$\begin{aligned} \tfrac{1}{2}a_z^2(\omega) \simeq \Phi_{zz}(\omega,\underline{p})\omega_o &= [\Phi_{yy}(\omega,\underline{p}) - \Phi_{nn}(\omega,\underline{p})]\omega_o \\ &\simeq \tfrac{1}{2}a_y^2(\omega) - \tfrac{1}{2}a_n^2(\omega) \end{aligned} \tag{3.7}$$

for ω equal to $h_i\omega_o$. This, in turn, allows us to estimate the transfer function gain g at the input frequencies. From the relation given by (3.3),

$$g_H(\omega) \equiv |H(\omega,\underline{p})| = [\Phi_{zz}(\omega,\underline{p})/\Phi_{xx}(\omega)]^{1/2} \simeq a_z(\omega)/a_x(\omega) \tag{3.8}$$

where a_z is obtained from (3.7), a_x is the corresponding input sinewave amplitude a_i, and the relation is restricted for ω equal to $h_i\omega_o$. Thus, we have a relatively straight-forward method for estimating the gain g_H of the linear transfer function H.

To find the phase, ϕ_H, we assume that the noise phase is uniformly (randomly) distributed with frequency, with zero mean over a range from $-\pi$ to

$+\pi$ radians. Since the Fourier transform of the response equation (3.1) is given by:

$$Y(\omega,\underline{p}) = H(\omega,\underline{p})X(\omega) + N(\omega,\underline{p}) \tag{3.9}$$

the corresponding signal phases must be given by:

$$\phi_Y(\omega) = \phi_H(\omega) + \phi_X(\omega) + \phi_N(\omega) \tag{3.10}$$

Taking the average value of this relation over the course of a measurement run results in the following expression for estimating the transfer function phase:

$$\phi_H(\omega) = \phi_Y(\omega) - \phi_X(\omega) \tag{3.11}$$

assuming zero mean noise phase ($\bar{\phi}_N=0$), and assuming that ϕ_Y is measurable from the Fourier transform of the response signal. The phase ϕ_X is simply the (run-fixed) sinewave phase (ϕ_i) used in specifying the SOS signal.

Thus, (3.8) and (3.11) provide us with the means of calculating the transfer function gain and phase, at all of the input frequencies specified in the sum-of-sines stimulus. Use of (3.7) allows us to estimate the noise power at these frequencies, and, by dividing by the base frequency ω_o, the noise PSD Φ_{nn} over the measurement spectrum. These relations, which take advantage of the special properties of the SOS input, allow us to efficiently and completely specify the quasi-linear model of the VER.

3.2 Identification of Quasi-Linear Model Structure and Parameters

For a complete description of the quasi-linear model, we need to specify H and Φ_{nn}, for the frequency range of interest to us. Assuming N measurement frequencies form the SOS stimulus, the frequency domain data will consist of N "triplets" of the form $\{(g_i, \phi_i, r_i) \quad , \quad (i=1,....N)\}$ where the ith triplet is associated with the ith measurement frequency ω_i. The problem is then one of fitting or matching these triplet data points by an appropriate choice of continuous model functions $H(\omega)$ and $\Phi_{nn}(\omega)$.

The problem breaks down to two simpler problems if we assume that we can fit the gain/phase data with $H(\omega)$ independently of fitting the remnant data with $\Phi_{nn}(\omega)$.

Identification of the noise PSD is straightforward, given that we have available an estimate of remnant power density for each measurement frequency ω_i. If a fixed-form remnant PSD model $\Phi_{nn}(\omega)$ is specified, then it is a direct matter to use standard non-linear least-squares model fitting procedures (e.g., Dixon (1973)) to chose the parameters of $\Phi_{nn}(\omega)$ to provide a "best-fit" to the required PSD data. Iteration over a number of different forms of Φ_{nn} can then be used to identify both the form and the parameter values which best fit the data.

Identification of the transfer function H is not quite as straightforward. The conventional approach is to use "Bode-Plot" fitting. An example is given by Zacharias (1982), where a second-order transfer function with delay of the form:

$$H(s) = e^{-\tau_d s}[K\omega_o^2/(s^2 + 2\zeta_o\omega_o s + \omega_o^2)] \qquad (3.12)$$

is fit to actual ssVER gain/phase data. The data is shown in Bode-Plot form in figure 3.2; five-run ensemble means and standard deviations (SD) are indicated by the circles and bars, respectively. The smooth gain/phase curves are those of fitted model transfer functions: the solid curve represents the second-order model of (3.12) and the dotted curve corresponds to a similar fourth-order model.

The fitting procedure was accomplished by minimizing a model-matching error function, of the form

$$J = \sum_{i=1}^{N} \left[\frac{g_i - g(\omega_i)}{\sigma_{gi}}\right]^2 + \sum_{i=1}^{N} \left[\frac{\phi_i - \phi(\omega_i)}{\sigma_{\phi i}}\right]^2 \qquad (3.13)$$

where (g_i, σ_{gi}) is the ith gain mean and SD, $(\phi_i, \sigma_{\phi i})$ is the ith phase mean and SD, and $g(\omega_i)$ and $\phi(\omega_i)$ are the gain and phase predicted at the ith

measurement frequency ω_i, on the basis of the model (3.12). The actual fitting itself involved a numerical search for the best set of model parameters $(K, \tau_d, \omega_o, \zeta_o)$ defining H.

Levison and Zacharias (1984) note two advantages of this approach:

1) By normalizing each data-model error component of (3.13) by the data's associated SD, we obtain error weighting which is inversely proportional to the data's SD, and hence directly proportional to the data's reliability.

2) By this normalization, we also solve the "apples/oranges" problem due to the different dimensions used for gain and phase (typically dB and deg); both are normalized to dimensionless units of SD.

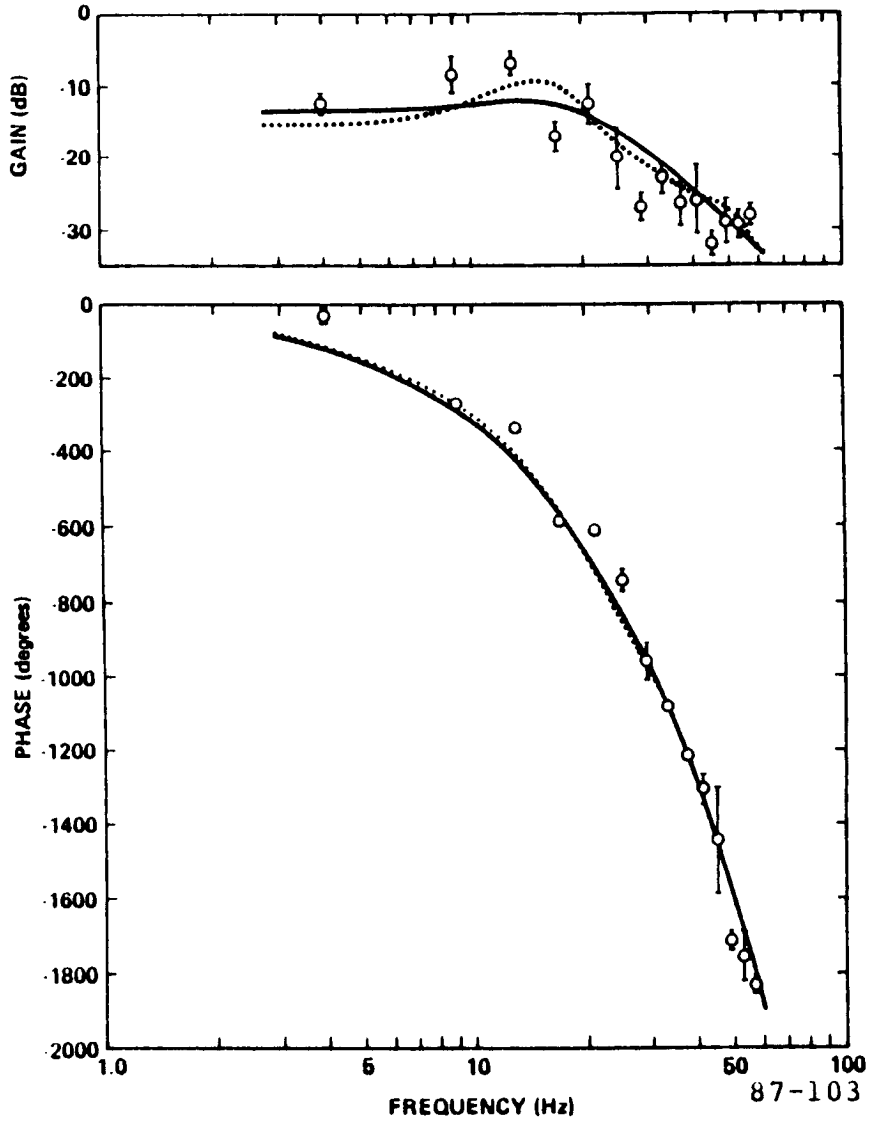

Figure 3.2: ssVER Gain and Phase vs. Frequency (data from Junker (1982))

These advantages, coupled with past investments made in software packages implementing this approach, have served to make this type of Bode-Plot model identification now an almost traditional approach with many researchers. The approach, however, has some significant shortcomings. They can be summarized as follows:

1) The Bode-Plot format of data presentation requires a detailed knowledge of the phase behavior. As noted by Levison and Zacharias (1984), this knowledge is not available when measurement frequencies are sparsely spaced over frequency regions in which phase changes rapidly (such as in figure 3.2), leading to situations where the measured phase can be off by an integer multiple of 360°. A method of "unwrapping" the phase shift may be required to obtain a true picture of the frequency-dependency of the phase response.

2) Error normalization by the SD gives excessive weighting to data points having unusually low variability. Resort must be made to ad hoc procedures to accommodate these situations, such as limiting the maximum weighting value, or assigning arbitrary-value minimum SD`s.

3) The Bode-Plot ensemble-average fitting procedure generates a set of model parameter means but no corresponding set of parameter SD's. Model-based hypothesis testing across experimental conditions is thus considerably more difficult, because of a lack of parameter reliability information.

4) Because the error normalization is based on ensemble-average SD's, the approach, of necessity, requires an ensemble of data replications, so that single-run model fits cannot be accommodated without recourse to some alternate means of generating SD information.

These are distinct shortcomings for a method that is to be applied to a state-of-the-art investigation of the ssVER. An alternate approach, having the potential of avoiding these problems, can be based on the Nyquist-Plot representation of frequency response data, given in figure 3.3. It combines the two gain and phase Bode plots into one plot, with frequency now a parameter along the curve. Every point on this curve can be defined in terms of its polar coordinates, specified by a radius, given by the transfer function gain $g(\omega)$, and an angle, given by the transfer function phase $\phi(\omega)$.

Also shown on the sketch is a transfer function data point associated with the ith measurement frequency ω_i. It is similarly specified in polar coordinates by gain g_i and phase ϕ_i.

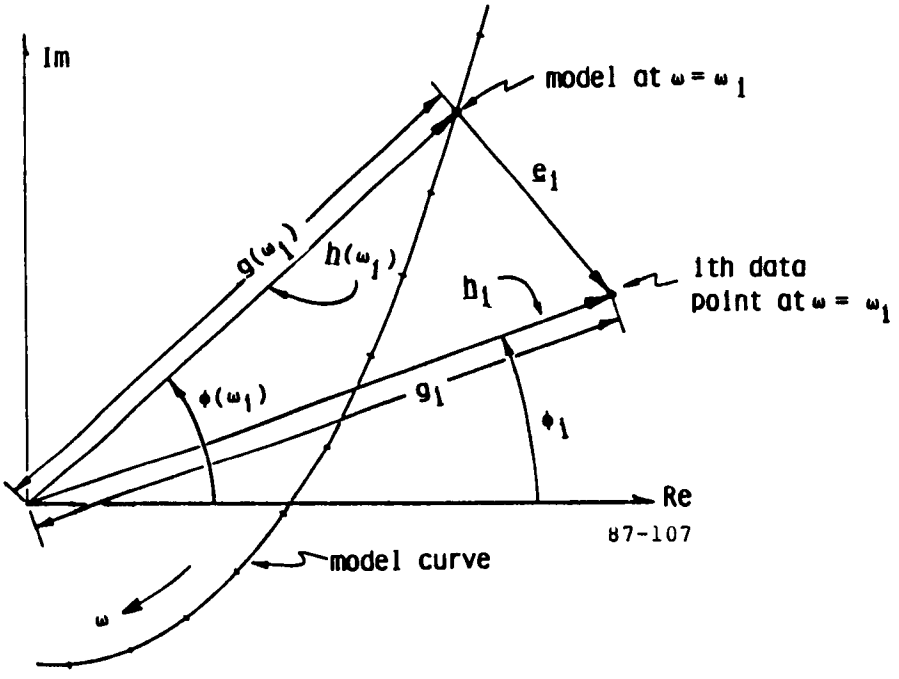

Figure 3.3: Nyquist-Plot Representation of Frequency Response Data

We now associate with each point a vector. With the ith point on the model curve, we define the vector $\underline{h}(\omega_i)$, having a length $g(\omega_i)$ and orientation $\phi(\omega_i)$, or,

$$\underline{h}(\omega_i) = g(\omega_i) \angle \phi(\omega_i) \qquad (3.14a)$$

Likewise, with the ith data point, we define the corresponding vector $\underline{h}_i$ given by

$$\underline{h}_i = g_i \angle \phi_i \qquad (3.14b)$$

Clearly, the error between model and data is a vector quantity, which we define as:

$$\underline{e}_i = \underline{h}_i - \underline{h}(\omega_i) \qquad (3.14c)$$

For N data points, we will have N such error vectors. Fitting the model to the data implies that we minimize these error vectors to assure "closeness"

between the model curve and the data points. The most direct means of doing this is by defining the following model-matching cost function:

$$J = \sum_{i=1}^{N} |\underline{e}_i|^2 = \sum_{i=1}^{N} \underline{e}_i \cdot \underline{e}_i \qquad (3.15)$$

which allows us to minimize the sum of the squares of error vector lengths and thus force the model transfer function H to fit the data.

This approach has a number of advantages over conventional Bode-Plot modeling:

1) Because the matching function of (3.15) is based on the Euclidean distance metric dimensioned in the natural units of the transfer function, there is no need for normalizing individual data-model error components (as there is in the case with the Bode-Plot gain and phase error metrics). In particular, we need not normalize by data SD's, as the conventional approach requires.

2) By not requiring error normalization based on data SD's, we avoid the problem of how to treat data points having unusually low variability; we need not resort to ad hoc procedures. More importantly, however, by not requiring SD error normalization, we totally eliminate the need for working with ensemble-averaged data. Thus, single-run model fits can be made and used to investigate factors whose effects are expected to show up at a single-run level.

3) In addition to allowing single-run model fits, the method also provides us with a means of computing the statistical reliability of the parameter estimates, and thus enabling model-based hypothesis testing of different experimental factor effects. This follows since the method produces model parameters for each individual set of frequency data. Thus, we can calculate, for each condition or factor, not only the mean parameter values (for example, the parameters of the model of (3.12)), but also the SD's associated with those values.

4) The Nyquist-Plot format of data presentation avoids the "phase wrapping" problems associated with the Bode-Plot format. This is because the polar coordinates used in constructing the Nyquist-Plot require only modulo-360° phase angle values; multiple turns or "wraps" have no effect on the plotted location of a data point. Thus, the Nyquist-Plot approach places no special demands on developing an "unwrapping logic," nor does it require the use of excessively close measurement frequencies in regions of rapid phase changes.

3.3 Requirements for Stimulus Protocol and Model-Based Analysis

On the basis of the above discussion concerning various approaches to identifying and modeling the VER, we recommend a basic approach which is

centered on a quasi-linear representation of the VER "system," and which utilizes steady-state sum-of-sines stimulation for empirical identification of system structure and parametric values. The model identification effort is aimed at a specification of both the linear (transfer function) portion of the VER and the uncorrelated noise (remnant) portion.

3.3.1 Stimulus Protocol

The recommended stimulus protocol uses a sum-of-sines signal to directly modulate stimulus intensity. This requires a light source which can be continuously modulated (in contrast to conventional strobe sources), and a real-time sum-of-sines generator to provide the modulation signal. For the generator, a general-purpose digital computer can implement the calculation of the sum-of-sines signal in accordance with (2.32). Associated (pre-run-time) software allows for convenient specification and/or modification of the sum-of-sines signal parameters: the base frequency ω_o, the number of sinusoidal components N, the set of harmonics, $\{h_i\}$, and the set of desired signal amplitudes and phases, $\{a_i, \phi_i\}$.

The stimulus generator can also provide the option for generating a different stimulus sequence for each stimulus/recording trial, to minimize potential "learning" effects. As discussed earlier, this is implemented by generating a new set of randomized signal phases, prior to the start of a data run.

3.3.2 Data Recording

For convenience in later analysis, the run-time computer is used for recording both the stimulus and the evoked response. The software implementing this recording function also supports necessary housekeeping functions associated with data file maintenance, such as providing for

inclusion of data headers (identifying the subject, the run number, date, etc.) and important stimulus/response parameters (e.g., sample period, sum-of-sines parameters, calibration voltages, etc.).

3.3.3 Frequency-Domain Analysis

Frequency-domain analysis of the stimulus/response data is conducted to support development of the quasi-linear model. In particular, we recommend the general approach outlined in chapter 2 earlier, which is specific to the sum-of-sines stimulus signal, and which provides for efficient and reliable estimates of both the linear and noise components of the quasi-linear VER model.

For flexibility in processing, we recommend that, at a minimum, the analysis package support the following basic functions:

1) Opening and reading the raw data files, and providing access to both the file management information and the data itself.

2) Calculating relevant time-domain signal parameters, such as signal mean, variance, etc.

3) Calculating the discrete Fourier transforms (DFT's) of both stimulus and response, and their corresponding power spectral density (PSD) functions.

4) Estimating the linear transfer gain and phase, at the measurement frequencies, and providing an indication of correlated response measurement reliability.

5) Estimating the uncorrelated response noise PSD, at the measurement frequencies.

6) Writing and closing reduced data files, containing the relevant frequency- and time-domain measures, along with necessary file management information.

7) Supporting appropriate hard-copy printouts, operating mode options, and other user-oriented features which will facilitate ease and flexibility of use.

This analysis package will be devoted to the transformation of individual raw data (time-domain) files into corresponding reduced data (frequency-domain) files, on a one-for-one basis. Thus, for every replication, subject, and experimental condition for which there exists a valid raw data file, we would expect to generate a corresponding frequency-domain file.

3.3.4 Ensemble Averaging

To reduce the number of frequency-domain data files, and to provide a means for reducing frequency-domain estimation errors, we recommend the use of ensemble averaging of the estimates, across appropriate data files. For example, to minimize sensitivity to such effects as habituation and fatigue, one might average across replications, for a single subject and test condition. If no significant inter-subject differences were found at this level, one might then average across subjects, for a fixed test condition. The potential for both data reduction and error minimization is considerable, and we see such a capability as a highly useful adjunct to the basic frequency-domain analysis package.

We recommend that, at a minimum, the averaging package support the following basic functions:

1) Opening and reading the frequency-domain data files requested by the user, and providing access to both the file management information and the data itself.

2) Providing options to the user regarding data sets to be averaged, checking for invalid or missing data points, and calculating the appropriate ensemble averages and corresponding standard deviations.

3) Writing and closing ensemble-average data files, containing the relevant reduced data, along with necessary file management information.

4) Supporting miscellaneous user-oriented functions, as required.

This averaging package will be devoted to the transformation of numerous individual frequency-domain data files into a small number of averaged data

files, corresponding to specific experimental conditions (and possibly subjects). We expect that, if significant inter-subject variations are not found, then only a few files will result, one associated with each experimental condition imposed on the subject population.

3.3.5 Model Development

To reduce this data still further, and provide an understanding of ssVER dependence on imposed workload, we recommend the development of analytic quasi-linear models to account for the averaged data trends. In its most basic form, this model development effort consists of the following tasks:

1) On the basis of the linear gain and phase trends with frequency, propose a linear fixed-form model for the transfer function. Likewise, on the basis of the noise trends with frequency, propose a linear fixed-form "shaping filter" for the noise PSD.

2) For each model so obtained, optimize the model parameter set $\underline{p}$ to minimize the residual fit error, between data and model.

3) Identify condition-dependent (i.e., state-sensitive) parameters (comprising the set $\underline{p}_m$) by comparing derived parameter sets across conditions.

4) Iterate on the choice of fixed-form models, and reoptimize the parameter set, to globally minimize data-fitting errors and maximize parametric sensitivity across conditions.

5) Summarize parametric dependence on or correlation with internal mental state either analytically or graphically, and propose a model-based mental state metric.

This model development exercise should culminate in an analytic quasi-linear model of the VER, which succinctly summarizes the experimental data. If the subject mental state can be successfully manipulated and controlled, and the VER is found to reflect these manipulations with sufficient sensitivity, then the derived analytic model, with its optimized parameter set, should provide a parametric indicator of internal mental state. This indicator can then serve as the basis for a preliminary specification of a model-based metric.

4. SOFTWARE DESCRIPTION, EXPERIMENT DESIGN, AND PRELIMINARY RESULTS

We now describe the VERSOS software package developed for ssVER experimentation and analysis, outline the experimental design effort, and summarize the results of a pilot ssVER study conducted at LaRC.

4.1 VERSOS Package for ssVER Research

The VERSOS software package is designed for use in ssVER experimentation and analysis. The package consists of four main programs.

1) VERRUN: This provides for pre-run setup of the experimental parameters, generation of the run-time sum-of-sines (SOS) stimulus, recording of the resulting response, and calculation of simple post-run statistics.

2) VERNAL: This supports time- and frequency-domain analysis of the time histories recorded by VERRUN, and provides for computation of the EEG RMS signal levels, and ssVER transfer function and remnant spectra.

3) ENSMBL: This calculates ensemble average statistics across individual subject runs, for the time- and frequency-domain analysis data generated by VERNAL.

4) MODLER: This supports the fitting of specified analytic transfer functions to the ensemble average transfer function data generated by ENSMBL, and supports the generation of an optimized model parameter set.

These programs will be described in more detail in the remainder of this section. Program listings are given in the Appendices.

Figure 4.1 is an overall block diagram showing the data flow between programs. The VERRUN program generates the *.dat files, which are comprised of a header describing the run itself, and data arrays containing the recorded stimulus/response data. The VERNAL program processes the *.dat files to generate corresponding *.anl files, containing the reduced time- and frequency-domain data describing each run. The ENSMBL program processes several of these *.anl files, to generate a single corresponding *.ens file, containing the ensemble average statistics. Finally, the MODLER program

processes each *.ens file, to generate a corresponding *.mod file, containing the fitted model function and parameters.

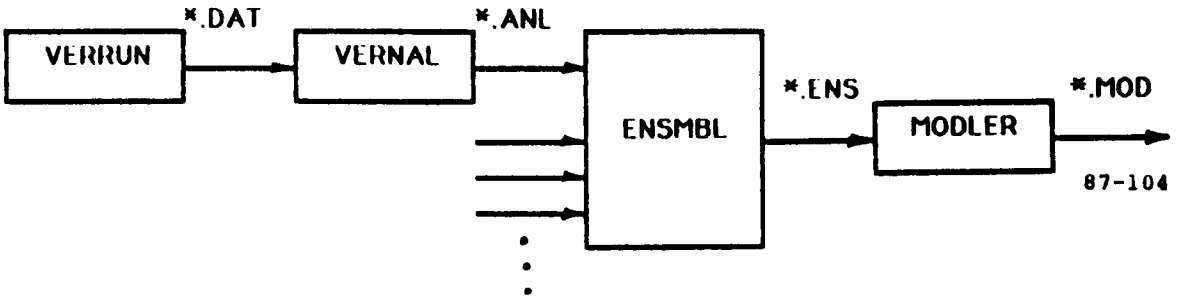

Figure 4.1: Data Flow with VERSOS Package

We now describe the functions of the four programs in greater detail.

4.1.1 VERRUN: Run-Time Control

VERRUN is intended for use in the closed-loop stimulus/response environment sketched in figure 4.2. The stimulus generator is driven by the software, through a digital-to-analog (D/A) converter, via a commanded SOS signal I_c. The generator, in turn, provides an intensity-modulated visual stimulus I for driving the human subject's ssVER. The resulting scalp voltages (E_1 through E_N) are transduced and amplified by the EEG recording hardware, and the measured voltages ($\tilde{E}_1$ through $\tilde{E}_N$) are sampled through a multi-channel analog-to-digital (A/D) converter. A stimulus intensity signal ($\tilde{I}$) is likewise transduced and sampled, through an additional A/D channel. VERRUN implements four major functions as diagrammed in figure 4.3:

1) Initial setup and parameter specification
2) Pre-trial initialization
3) Real-time SOS generation and data recording
4) Post-trial file maintenance.

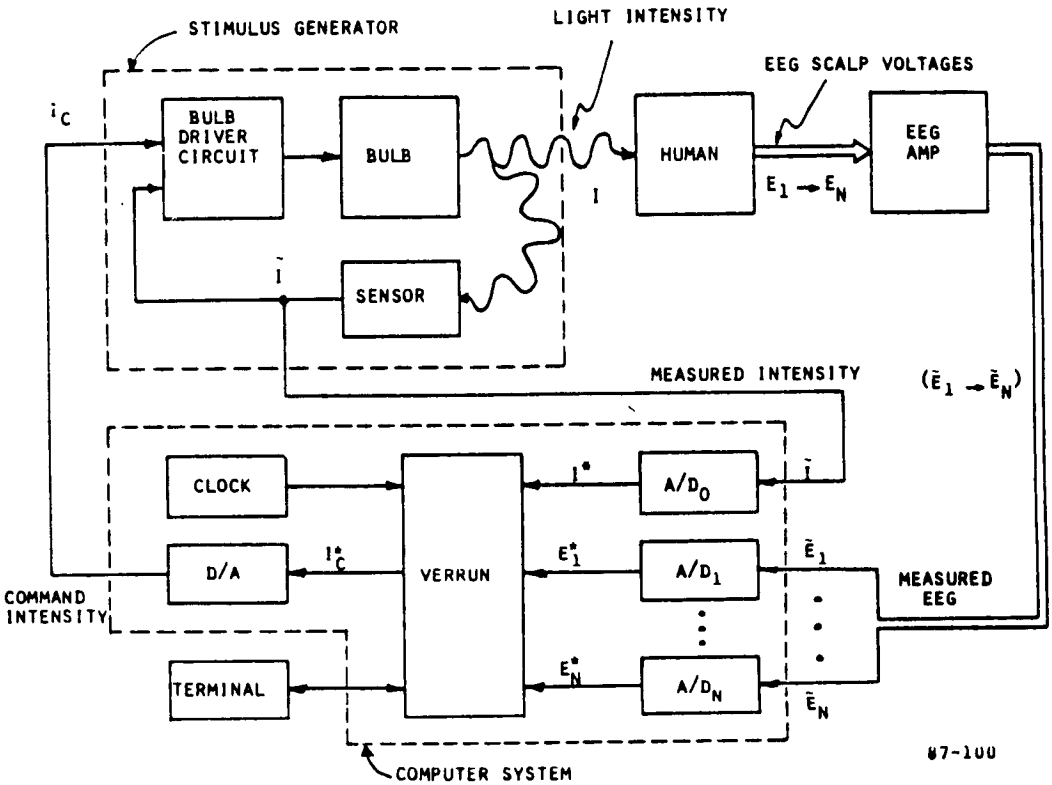

Figure 4.2: Closed-Loop Stimulus/Response Environment

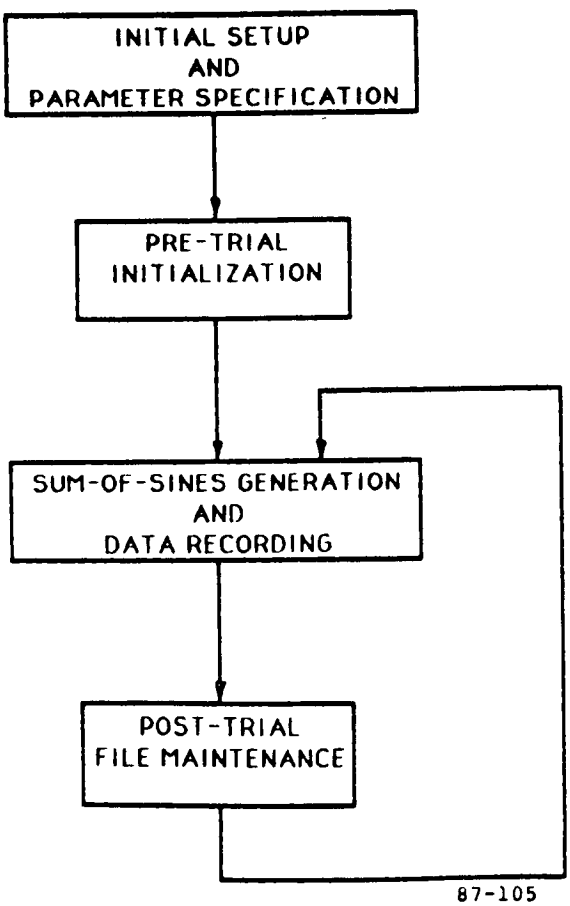

Figure 4.3: Major VERRUN Functions

Typically, initial setup and parameter specification is performed only at the start of a multi-trial experimental session, and the remaining three functions are performed in order during each experimental trial.

A user will generally want to use the same time-base and SOS parameters throughout an entire experiment (except for re-randomization of the SOS phases). Since VERRUN can be initialized with values stored on a previously-created data file, an entire experiment can be run with a minimum of user interaction with the program.

Initial Setup and Parameter Specification:

Both time-base and SOS parameters are specified during this initialization phase. Parameters may be specified in one of four ways:

1) Read all parameter values from a previously-created file.

2) Request "nominal" (pre-stored) values for all parameters.

3) Specify all parameters interactively.

4) Request nominal values for time-base (or SOS) parameters and specify SOS (or time-base) parameters interactively.

If parameters are specified interactively, or if nominal values are requested individually for the time-base and SOS parameter sets, the user is provided an opportunity to review and modify parameter values before continuing on. This review/modification option is omitted if the parameters are read from a file, or if nominal values have been requested for all parameters.

The user is then asked if he wishes to perform a run. If so, VERRUN executes pre-trial initialization. If not, the parameters are stored on a file specified by the user, and VERRUN provides the options of: 1) specifying another parameter set; 2) performing an experimental trial; and 3) terminating the program.

With direct user specification of the time-base parameters, the user is prompted to enter the sample interval in milliseconds, I_S, and the overall run length in seconds T_R, defining the duration of an experimental trial. Both entries are checked against minimum and maximum limits; nominal as well as limiting values are shown in Table 4.1. VERRUN then computes the sample interval in seconds, T_S, and the number of samples per trials, N_R, as follows:

$$T_S = I_S/1000 \qquad (4.1)$$

$$N_R = T_R/T_S + 1 \qquad (4.2)$$

Values specified for I_S and T_R are checked again to make sure that N_R does not exceed the system's preset upper storage limits; nominal values are given in table 4.1.

Table 4.1: Time-Base Parameter Values and Limits

Parameter	Units	Nominal	Minimum	Maximum
I_S	msec	10	1	100
T_R	sec	50	0	100
N_R	--	5000	0	10,000

Given the total number of sample points N_R comprising a run, VERRUN specifies the total number of sample points N_O for one period of the SOS signal. For compatibility with the VERNAL FFT routine used for signal analysis, the value for N_O is chosen to be the largest power of 2 less than or equal to N_R. VERRUN also computes the overall SOS period in seconds T_O, the base frequency in Hz f_O, and the base phase in degrees ϕ_O, as described in section 2.4. Time-base parameters are then listed for user verification and respecification if not satisfactory.

With direct user specification of the SOS parameters, the user is first prompted to specify the number of sinewave components, N_C. This may be done

by specifying the "nominal" value option, or by direct entry, in which case limit checks are provided. Limiting and nominal values are given in table 4.2.

The user is then prompted to specify a desired SOS frequency set, f'_j, where j ranges from 1 to N_c, f'_j is in Hz. This can be done by specifying the "nominal" frequency set option (if N_c is nominally specified), in which case the first N_c components of the nominal frequency set are selected. If the user chooses instead to specify the N_c frequencies directly, VERRUN allows for corrections to be made during data entry, and provides checks to ensure that the chosen frequencies are consistent with the previously-chosen sample and run times. Limiting and nominal values are given in table 4.2.

Table 4.2: SOS Parameter Values and Limits

Parameter	Units	Nominal	Minimum	Maximum	Note
N_c	--	6	1	15	
f_j	Hz	5,10,...	f_o	$f_s/2$	(1)
$\tilde{a}_j$	--	1,1,1,...	0	100	
ϕ_j	deg	--	0	360	(2)
I_{RMS}	volts	1	0	5	

Notes: (1) $f_o = 1/T_o$ and $f_s = 1/T_S$

(2) nominal values set by random number generator

Once the desired frequency set has been specified, VERRUN then computes, for each component, the nearest corresponding integer multiplier according to:

$$h_j = [f'_j/f_o] \quad (j=1,\ldots,N_c) \tag{4.3}$$

This then yields the harmonically related SOS frequencies f_j, where

$$f_j = h_j f_o \quad (j=1,\ldots,N_c) \tag{4.4}$$

Naturally, progressively smaller values of f_o allow for progressively closer matches between the desired drive frequency sets f'_j, and the actual harmonically derived set, f_j. Smaller values of f_o can, in turn, be obtained by increasing T_o.

Once the SOS frequency set has been specified in this fashion, the user is provided the opportunity of listing both desired and actual frequencies, along with the corresponding harmonics. If not satisfactory, VERRUN allows for respecification.

Following SOS frequency specification, VERRUN prompts the user for the distribution of SOS amplitudes with frequency. This is done by specifying normalized (dimensionless) amplitudes $\tilde{a}_j$, which are related to the SOS (dimensioned) amplitudes a_j, by a scale factor r, or:

$$a_j = r\tilde{a}_j \qquad (j=1,\ldots,N_c) \tag{4.5}$$

so that, with r free, the user can specify the <u>shape</u> of the a_j distribution, independent of the signal RMS <u>level</u>.

The normalized amplitudes $\tilde{a}_j$ may be set by specifying the "nominal" amplitude set option (if N_c is nominally specified), or by direct entry of the N_c normalized amplitudes. If the user chooses the latter, VERRUN allows for corrections to be made during data entry, and provides checks to ensure that the chosen amplitudes are within prespecified limits. Limiting and nominal values are given in table 4.2.

Once the normalized amplitude set has been specified, VERSOS then prompts the user for the desired RMS signal level of the SOS signal, I_{RMS}. This may be done by specifying the "nominal" value option, or by direct entry, in which case limit checks are provided (limiting and nominal values are given in table 4.2). VERRUN then computes the amplitude scale factor r according to:

$$r = \sqrt{2}\ I_{RMS} \left[\sum_{j=1}^{N_c} \tilde{a}_j^2 \right]^{-1/2} \tag{4.6}$$

By then computing the SOS amplitudes according to (4.5), VERRUN ensures that the SOS signal I(t) will have the desired RMS level, since

$$I^2(t) = \sum_{j=1}^{N_c} \frac{1}{2} a_j^2 = \frac{1}{2} r^2 \sum_{j=1}^{N_c} \tilde{a}_j^2 = I_{RMS}^2 \qquad (4.7)$$

Following SOS amplitude specification, VERRUN prompts the user for a desired SOS phase set, ϕ_j, where j ranges from 1 to N_c. Phases can be selected in one of three ways:

1) The "nominal" selection procedure

2) Specification of a "seed" for picking a set of random phases

3) Direct specification of phases.

If the user chooses the nominal option, VERRUN uses a random number generator to select uniformly distributed values between 0 and 360 deg; the "seed" of the random number generator is automatically changed from run to run to allow for a consistent means of randomizing the phase sets each run (and thus the SOS time history). If the user specifies the seed for phase randomization, or specifies phases directly, VERRUN allows for corrections to be made during data entry, and provides checks to ensure that the chosen phases are within prespecified limits (given in table 4.2).

Once the desired phase set has been specified, VERRUN then computes, for each component, the nearest corresponding integer phase multiplier, p_i, according to:

$$p_j = [\phi_j'/\phi_o] \qquad (j=1,\ldots,N_c) \qquad (4.8a)$$

The SOS phases can then be computed as integral multiples of the base phase as

$$\phi_j = p_j \cdot \phi_o \qquad (4.8b)$$

This, of course, quantizes the phase choices, but progressively smaller values of ϕ_o allow for progressively closer matches between the desired phase set ϕ_j' and the actual set ϕ_j. Smaller values for ϕ_o can, in turn, be obtained by reducing the ratio of T_S/T_o.

Pre-Trial Initialization:

Pre-trial initialization consists of four basic steps. First, if an experimental trial has just been completed, and the user has requested another run, the user is provided the option to change all, some, or none of the time-base and SOS parameters. If the user requests no changes, SOS component phases are automatically re-randomized. This step is omitted on the first trial following initial setup and parameter specification.

Next VERRUN displays the date, time, and run number selected for the upcoming trial. The run number is set to 1 during initial setup and is automatically incremented by 1 for successive trials. The user either accepts or modifies the run number and then specifies up to 6 lines of commentary.

VERRUN then prompts the user for a filename for parameter and data storage. After some simple legality checks on the entered name, VERSOS opens a file and writes out the header: that portion of the data file comprised of the (previously-defined) run parameter values, along with miscellaneous housekeeping parameters and tags to aid in later data file maintenance.

Finally, VERRUN generates a pre-stored version of the entire SOS signal to be used. This is done by first generating and storing a "quarterwave" sine table associated with the sample and base periods, T_S and T_O, of the SOS signal. With this table, the sampled-time version of the SOS signals is then computed for all N_R samples which comprise a complete run. Each sample value is then scaled for eventual conversion by the D/A hardware, and then stored in a linear data array. With the SOS signal generated and stored, VERSOS prompts the user for a "run start" signal, and waits for the user's response.

Real-Time Control:

Once a start signal is received from the user, VERRUN zeros the D/A channels and starts the digital clock "ticking" at a pre-specified rate

(nominal clock rate is 100 kHz). After the clock has counted down the number of ticks corresponding to the desired sample interval T_S, D/A and A/D conversions are performed. This cycle is repeated N_R times to generate an experimental trial of the desired length T_R seconds, after which the clock is stopped and the D/A channels zeroed.

Two signals are generated each sample interval: 1) a square wave alternating between maximum positive and negative values on D/A channel 0, to be used for test purposes; and 2) the SOS signal on channel 1, obtained by table lookup.

Three signals are recorded by A/D channels 1-3 and are stored in the same linear array containing the SOS stimulus signal. The data sequences recorded from the three A/D channels are interleaved with each other and with the SOS stimulus. That is, the first element of the linear data array contains the first SOS sample, the second through fourth elements contain the first samples recorded from A/D channels 1-3, respectively, the fifth element contains the second SOS sample, and so forth. The linear data array will therefore contain $4*N_R$ samples at the end of the experimental trial.

Post-Run File Maintenance and Multi-Run Control:

VERRUN "closes-out" a run by first writing the recorded data strings onto the file opened at the beginning of the run, thus appending the data to the parameter set used to specify the run. The file is then closed, and the user is provided the options of: 1) performing another run; b) setting up a parameter file; or 3) terminating the program. If another run is requested, the run number is incremented, and VERRUN proceeds with pre-trial initialization as described above. Request for a new parameter file returns the program to the initialization mode described earlier.

4.1.2 VERNAL: Post-Run Analysis

VERNAL performs the five major operations shown in figure 4.4. This program is menu-driven in that the user specifies interactively, via a "part" number, the operation VERNAL is to perform. Upon completion of a given operation, the user specifies the next operation to be performed. A part number of 0 displays the options shown in figure 4.4, and a part number of -1 terminates the program.

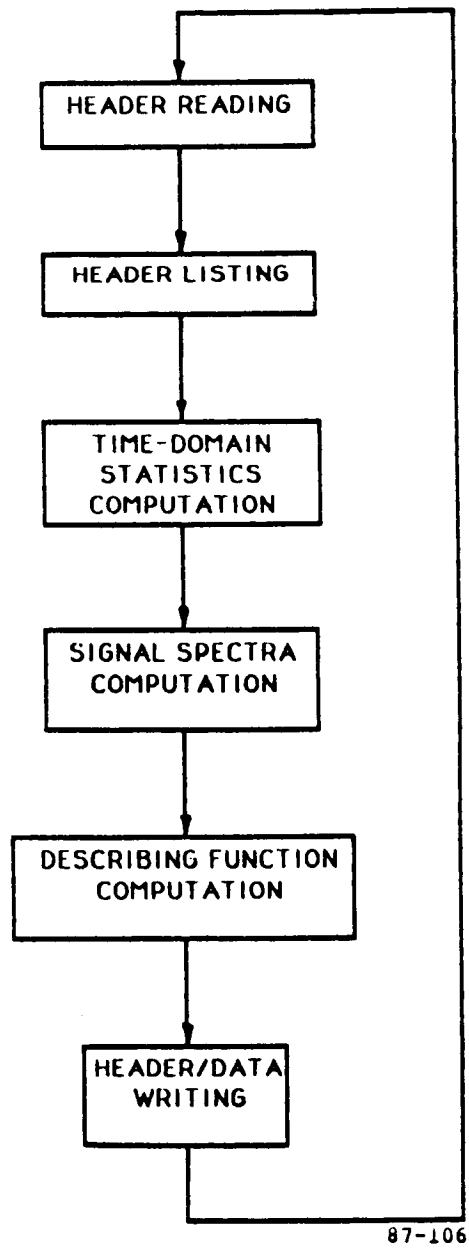

Figure 4.4: Major VERNAL Functions

Part 1 (read header) must be performed first; otherwise, program parts may be executed in any order. Figure 4.4 shows the typical order in which program functions are executed. These functions are described individually below.

Part 1: Read Header:

Once the user has specified the name of the data file, VERNAL opens the file, reads the header information, and leaves the file open for subsequent reading of the experimental data.

Part 2: List Header:

Header information consisting of run identification, problem parameters, and user commentary, is displayed on the user's terminal. If the user then discovers he has not requested a file of interest, he may next request re-execution of Part 1, in which case the current file is closed, and a new file is requested and opened.

Part 3: Time-Domain Statistics:

When a statistical computation (either time- or frequency-domain) is first requested for a given data file, VERNAL reads the experimental data from the current file, stores the data in a linear array, and closes the file. The user is informed of the currently specified starting point for calculations, and is given the option to change the start point, which must lie within the range of 1 to $(N_R - N_O + 1)$, where N_R is the number of samples/channel in the experimental trial, and N_O is the number of samples in the SOS period. This restriction guarantees that N_O samples will be available for computation. The user will typically request a start point greater than 1 to minimize the

influence of the transients that most likely followed the onset of the SOS stimulus.

Before computing time-domain statistics, VERNAL provides the option to list the entire data base stored in the integer scaled array IDATA, or to list an array XDATA of data from a single channel of the user's choosing. Unless the user is debugging the program, or suspects unusual response behavior, this option will typically not be exercised.

The primary function of this part is to compute mean, standard deviation, and RMS signal amplitudes. These quantities are computed for all data channels and displayed on the user's terminal.

Part 4: Spectra:

Part 4 computes the spectra of one or more signals of the user's choosing, using fast-Fourier transform (FFT) techniques. Once the spectrum has been computed for a specified data channel, the user has the option of listing either the entire spectrum (i.e., at all FFT frequencies) or the spectral components at only the SOS frequencies. Again, unless the user is debugging the program or looking for some specific spectral feature (say, evidence of significant nonlinear response behavior), the full spectrum option will typically not be exercised.

Whether or not this option is exercised, VERNAL will list, for each input frequency:

1) Correlated power per measurement bin
2) Remnant power per bin
3) Ratio of the correlated to remnant power
4) Correlated power per rad/sec
5) Remnant power per rad/sec
6) Ratio for correlated power to remnant power (rad/sec)
7) Number of frequency bins included in the remnant averaging window.

These spectral quantities are given in dB. The following overall statistics (in problem units) are then listed:

1) Correlated power summed over all input frequencies
2) Rate of correlated to total signal power
3) Remnant power summed over all non-input frequencies
4) Rate of remnant to total power
5) Total signal power (i.e., sum of all spectral computations over all frequencies).

The user is then given the option to perform another spectral analysis or to specify another program part.

Part 5: Describing Functions:

Part 5 performs a describing function analysis to obtain transfer functions relating stimulus input to EEG response. When execution is begun, VERNAL prompts the user for indices corresponding to the desired numerator and denominator channels. After the requested describing function h has been computed, gain (in dB) and phase (in degrees) are printed out at each SOS frequency, except that computations failing a 6 dB signal/noise ratio test are flagged by a printout of the string (****). The user then has the option of computing another describing function or specifying another program part.

Part 6: Histogram Plots:

Part 6 provides the user with a plot of the amplitude histogram for a chosen signal channel. When this section is chosen, VERNAL prompts the user for the desired channel number, and for the desired amplitude bin magnitude. VERNAL then computes the frequency distribution of the sampled signal amplitudes, and generates a rudimentary sketch of the amplitude histogram on

the user's console, for immediate review by the user. The user then has the option of computing another histogram or specifying another program part.

Part 7: Summary and File Writing:

Part 7 generates a summary set of analysis data and supports the generation of output *.anl files. The signal time-domain statistics for all channels are first computed. This is followed by a computation of the spectra for two channels: one containing the stimulus recording and the other containing one of the two recorded EEG channels. VERNAL then computes the corresponding ssVER transfer function. The time-domain statistics, signal spectra, and describing function are then all written to the specified *.anl file, which is then closed. The user then has the option of generating another summary file or specifying another program part.

4.1.3 ENSMBL: Multi-Run Statistics

ENSMBL takes in the experimental time history data for each subject and calculates the across-replication and/or across-subject mean μ and standard deviation σ for:

1) RMS signal levels (for the stimulus and response on each channel)
2) ssVER transfer function gains at each test frequency
3) ssVER transfer function gains at each test frequency
4) Uncorrelated ssVER response, or remnant

Ensemble statistics, for the ith measurement at the kth run, are calculated in the conventional manner, according to:

$$\mu_{ik} = \frac{1}{N_g} \sum_{j=1}^{N} \Lambda_j d_{ijk} \qquad (4.9a)$$

$$\sigma^2_{ik} = \frac{1}{N_g - 1} \sum_{j=1}^{N} \Lambda_j (d_{ijk} - \mu_{ik})^2 \qquad (4.9b)$$

where the summations are over all N subjects, where d_{ijk} denotes the ith measurement for the jth subject at the kth trial (covering RMS score, gain, phase, or remnant measurement), and where

$$N_g = \sum_{j=1}^{N} \Lambda_j \tag{4.10}$$

where the flag Λ_j denotes whether a measurement is "good" or "bad," in accordance with

$$\Lambda_j = \begin{matrix} 0 & \text{"bad"} \\ 1 & \text{"good"} \end{matrix} \tag{4.11}$$

Denoting whether a data point is "good" or "bad" generally applies only to gain/phase data. If the uncorrelated response power, at a given measurement frequency, exceeds approximately 20% of the total power, then the gain/phase measurement pair is flagged as "bad," and the corresponding Λ_j set to zero. The RMS signal level and remnant measurements are not flagged in this fashion, but any missing data point is so-flagged (i.e., when there are "holes" in the data base).

As noted, across-subject mean and standard deviation calculations are done in accordance with (4.9), for each RMS level, gain, phase, and remnant measurement. Since we generate a mean and standard deviation for each set of M measurements, we effect, in general, a reduction by a factor of M/2 in overall data set size, assuming all measurements are "good." It should also be noted that a measurement will occasionally be "bad" across all subjects or "good" for only one subject. In this case, the standard across-subject deviation is undefined, so that the corresponding ensemble mean is flagged as a "bad" point.

Finally, it is appropriate to note that ENSMBL can be applied for both across-replication and across-subject averaging. Typically, across-replication averaging is conducted first, to minimize adaptation or habituation effects. This results in a new set of average data, one for each

subject. This is then reprocessed by ENSMBL to generate an across-subject average, describing the population response for that experimental loading condition. Similar processing would be conducted for other tested experimental conditions.

4.1.4 MODLER: Model-Based Identification

MODLER supports the fitting of analytic transfer functions to the ensemble-average transfer function data generated by ENSMBL. It reads the transfer function data, accepts a user-specified transfer function, and computes a goodness-of-fit metric as a function of chosen transfer function parameter. It may be used directly by the analyst, in a manual search for optimum model parameter values, or in conjunction with a numeric search scheme, to support an automatic search for parameter values.

MODLER assures a model transfer function of the following form:

$$H(s,\underline{p}) = (K/s^N)\, e^{-sT_D}\, \frac{\Pi(s+a_{zi})\Pi(s^2+2\zeta_{zi}\omega_{zi}s+\omega_{zi}^2)}{\Pi(s+a_{pi})\Pi(s^2+2\zeta_{pi}\omega_{pi}s+\omega_{pi}^2)} \qquad (4.12)$$

where N specifies the order of integration, T_D sets the time delay, $\{a_{zi}\}$ and $\{\zeta_{zi},\omega_{zi}\}$ set the first- and second-order zeros, and $\{a_{pi}\}$ and $\{\zeta_{pi},\omega_{pi}\}$ set the first- and second-order poles. Also explicitly specified are the number of first- and second-order zeroes and poles. Although this is a fixed-form transfer function, a wide range of frequency response characteristics can be closely matched.

Once a model form and parameter set is specified, and the desired data file is read in, MODLER computes a model-matching error function, of the form:

$$J = \sum_{i=1}^{N} \left[\frac{g_i - g(\omega_i)}{\sigma_{gi}}\right]^2 + \sum_{i=1}^{N} \left[\frac{\phi_i - \phi(\omega_i)}{\sigma_{\phi i}}\right]^2 \qquad (4.13)$$

which, for convenient reference, we have repeated from (3.13) earlier. In this error function (g_i, σ_{gi}) is the ith gain data mean and SD couple,

$(\phi_i, \sigma_{\phi i})$ is the ith phase data mean and SD couple, and $g(\omega_i)$ and $\phi(\omega_i)$ are the corresponding gain and phase predicted at the ith measurement frequency, ω_i, on the basis of the model (4.12). As described earlier, this form of the matching function provides for component error weighting in direct proportion to data reliability.

Currently, MODLER is used in a manual search mode, where the user specifies the model form and parameter set to minimize the matching cost of (4.13). Direct linkage with a numerical search scheme can provide for automatic parameter optimization, for a given model form. A discrete search over forms of the family specified by (4.12) would support an automatic search for the best-fit model and parameter set, over the full family of models.

Once a solution is found, MODLER then generates the model gain and phase curves against frequency, for later plotting and direct comparison with the trends in the input data set.

4.2 Experiment Design

We outline here the experiment design, in terms of the proposed stimulus/recording parameters for the initial ssVER series.

4.2.1 Time Base Parameters

Based on our earlier review of the literature given in chapter 2, we assume that our maximum frequency of interest will be about 25 Hz. If we sample at twice the minimum Nyquist rate of 50 Hz, we have:

$$f_s = 100 \text{ Hz} \qquad (4.14a)$$

and

$$T_s = 1/f_s = 10 \text{ msec} \qquad (4.14b)$$

The VERRUN software constrains us to a maximum data string length of 2^{12}, or 4096, points. We choose the maximum and thus obtain an SOS period and frequency resolution of:

$$T_o = 2^{12} \cdot T_s = 40.96s$$

$$f_o = 1/T_o = 0.0244 \text{ Hz} \qquad (4.15)$$

Thus, using VERRUN, we simply set the sample period T_s to 10 msec, and set the run time T_R to a value slightly greater than T_o, say, 50 sec.

4.2.2 Frequency Distribution

A sufficiently large number of SOS frequencies should be chosen, so as to reduce potential phase ambiguities (±N*360°) with increasing frequencies. From the response illustrated earlier in figure 3.2, we see roughly an 1800 deg phase change over 50 Hz, or 180 deg every 5 Hz. To stay within half a cycle of the true phase, it would thus appear that we should measure the response every 5 Hz. Note that we are proposing a uniform arithmetic series of frequencies, of the form

$$f_{j+1} = f_j + \Delta f \qquad (4.16)$$

as used by Junker (1982) in his VER work, rather than a uniform geometric series, of the form

$$f_{j+1} = f_j^K \quad (K>1) \qquad (4.17)$$

as is usually used in human operator work.

4.2.3 Harmonic Characteristics

Consideration should be given to the set of harmonics used to specify the desired SOS frequencies. In our problem, this means the choice of the integer harmonic h_j, which, for the jth frequency, best approximates f_j according to

$$f_j \simeq h_j f_o \qquad (4.18)$$

It can simply be chosen as the integer nearest (f_j/f_o), and if no significant

non-linearities are expected in the response, this choice would appear entirely reasonable. If we anticipate non-linearities, however, then it would be better to choose h_j as the prime integer nearest (f_j/f_o). This ensures that measured response at any given input frequency will not be confounded by the (non-linear) response associated with multiples or sub-multiples of any of the other input frequencies.

4.2.4 SOS Stimulus Frequencies

The above considerations have led to the generation of table 4.3, showing the proposed SOS stimulus frequencies. A 10 component signal is proposed, with frequencies ranging from 5 to 27.5 Hz, in 2.5 Hz increments (f_j^{des}). Non-prime harmonics are shown in column 3, and the two integer prime harmonics (lo, hi), which bracket this non-prime harmonic, are given in column 4. The closest bracketing prime harmonic (h_j) is then given in column 5. The resulting SOS frequency f_j is given in column 6. VERRUN currently does not force selection of prime harmonics. To ensure that the prime harmonics h_j are generated by the program, we should specify the frequencies (f_j'), as indicated in the last column. Note only components 2, and 6 thru 9 differ from the resulting frequencies f_j, of column 6.

Table 4.3: Proposed SOS Stimulus Frequencies

j	f_j^{des}	f_j^{des}/f_o	(lo, hi)	h_j	f_j	f_j'
1	5	204.8	(199,211)	199	4.86	4.86
2	7.5	307.2	307	307	7.49	7.50
3	10	409.6	(409,419)	409	9.98	9.98
4	12.5	512	(509,521)	509	12.42	12.42
5	15	614.4	(613,617)	613	14.96	14.96
6	17.5	716.8	(709,719)	719	17.54	17.55
7	20	819.2	(811,821)	821	20.03	20.04
8	22.5	921.6	(919,929)	919	22.42	22.43
9	25	1024	(1021,1031)	1021	24.91	24.92
10	27.5	1126.4	(1123,1129)	1129	27.55	27.55

4.2.5 SOS Stimulus Amplitudes

For this preliminary SOS stimulus design, it is proposed that the signal amplitudes a_i not vary with frequency, but simply all take on a constant value. The value can be chosen on the basis of the desired RMS intensity level eventually chosen for the experimental effort.

We want to specify the amplitudes in terms of percent modulation depth. To do this we need to know the ambient light level I_o, and the drive gain K relating D/C volts (v) to light-box ft-lamberts (fL), in accordance with

$$I = Kv + I_o \tag{4.19}$$

On the basis of earlier system measurements made at the LaRC facility, we assume a v_{max} D/A command of +5v yields an I_{max} of 87.5 fL, while a v_{min} D/A command of -5v yields an I_{min} of 12.5 fL, so that, from above:

$$K = (I_{max}-I_{min})/(v_{max}-v_{min}) = 7.5 \text{ fL/v} \tag{4.20a}$$

and

$$I_o = [(I_{max}+I_{min})-K(v_{max}+v_{min})]/2 = 50 \text{ fL} \tag{4.20b}$$

We can now specify the jth SOS amplitude a_j, given in volts, in terms of a modulation depth m_j, according to:

$$a_j = m_j I_o/K = mI_o/K \tag{4.21}$$

where we assume all the m_j, and hence the a_j, are equal. The RMS voltage coming out of the D/C is then given by:

$$\begin{aligned} V_{RMS} &= [1/2 \sum a_j^2]^{1/2} \\ &= (N/2)^{1/2}(mI_o/K) \end{aligned} \tag{4.22a}$$

so that

$$m = (2/N)^{1/2}(K/I_o)V_{RMS} \tag{4.22b}$$

To specify V_{RMS}, we require that the D/A voltage hit the 5v limits less than 1% of the time ($P<0.01$). Assuming the SOS signal amplitude is approximately normally distributed, we use the standard tables to find a required z-value of 2.58, so that

V_{RMS}= 5v/2.58 = 1.94v

With N = 10, we then use (4.22b) to find m:

$m = (2/10)^{1/2}(7.5/50)\ (1.94) = 0.13 = 13\%$

so that, from (2),

$a_j = (0.13)\ (50/7.5) = 0.8667v$

which, via (4.19), corresponds to a 6.5 fL SOS amplitude about the ambient level. To set the amplitudes using VERRUN, we simply set all the relative amplitudes to 1, and specify an RMS level of 1.94v.

4.2.6 SOS Stimulus Phases

For this preliminary SOS stimulus design, and for subsequent designs, it is proposed that the phases ϕ_i be chosen as uniform random variables, over the unit circle. On-line re-randomization by VERRUN between each run will ensure a different ("unlearnable") SOS time history for each presentation, while maintaining a fixed stimulus spectrum.

4.3 Preliminary Experimental Results

We now briefly describe an experiment designed to evaluate the overall experimental protocol, and test the utility of the VERSOS software package.

A pilot ssVER experiment was conducted at NASA LaRC, using the delivered ssVER package ported onto the LaRC DEC MNC-11 computer system. This was integrated with general purpose EEG recording amplifiers and signal conditioners, and with a custom modulated fluorescent stimulus, and photodiode feedback circuit.

The fluorescent stimulus was modeled after that used by Junker (1986). It provided for continuous modulation of the stimulus level, and was driven by the MNC D/A hardware, in turn driven by the sum-of-sines signal generated by the VERRUN software. Overall average intensity was 50 fL, with a diffuse

presentation via diffuser screens and a beam splitter (see below). Six sinusoidal components (rather than the originally proposed ten) comprised the SOS signal, and each component amplitude was chosen to yield a 13% modulation depth, or 6.5 fL per component. SOS phases were randomized between runs.

EEG signals were recorded from occipital leads, amplified, and filtered via standard recording modules. They were then sampled by the MNC A/D hardware, and stored by the VERRUN software. Also recorded was the stimulus light level, as recorded by a monitor photodiode.

Subjects were given three tasks presented on a computer generated display, which was viewed through a half-silvered splitter window, to allow for the superposition of the the SOS illumination on the display. The three tasks were: a null task comprised of viewing a blank display (baseline); a task testing visual perceptual processing, called the Probability Monitoring Task (PMT); and a task testing encoding of working memory, called the Continuous Recall Task (CRT). Both the PMT and the CRT are part of a larger set designed to assess human performance, called the Criterion Task Set (CTS). These are described in detail in Shingledecker (1984).

Two undergraduate male subjects with normal vision were run through the pilot experiment. After adapting to a steady ambient light level of the average stimulus level of 50 fL, the subjects were presented a task of approximately three minutes duration. During the course of the task, two 50-sec ssVER recordings were made. Several replication three-minute runs were made in this fashion, to support ensemble averaging later on.

The VERRUN package controlled the run-time ssVER stimulus generation, response recording, and data file generation. Following the experimental runs, the VERNAL package was used to generate corresponding single-run performance score and frequency-domain files. These files were then grouped according to task loading conditions imposed, and the ENSMBL package was used

to compute single-subject across-replication ensemble files, comprised of the average RMS level and frequency-domain metrics. This overall sequence of data generation and analysis using the VERRUN, VERNAL, and ENSMBL packages was performed by the LaRC staff, at the experimental facility.

Figures 4.5 through 4.7 present some of the results of this processing, showing single-subject across-replication ensemble-average transfer functions measured under the baseline task loading condition (figure 4.5), the PMT condition (figure 4.6), and CRT condition (figure 4.7). The transfer functions are specified by the gain and phase measurements obtained at the input SOS frequencies; at each frequency, the measurement mean is denoted by a circle, and plus-or-minus one standard deviation is denoted by an error bar. Superimposed on the data are continuous curves depicting simple model matches to the data, which we describe shortly.

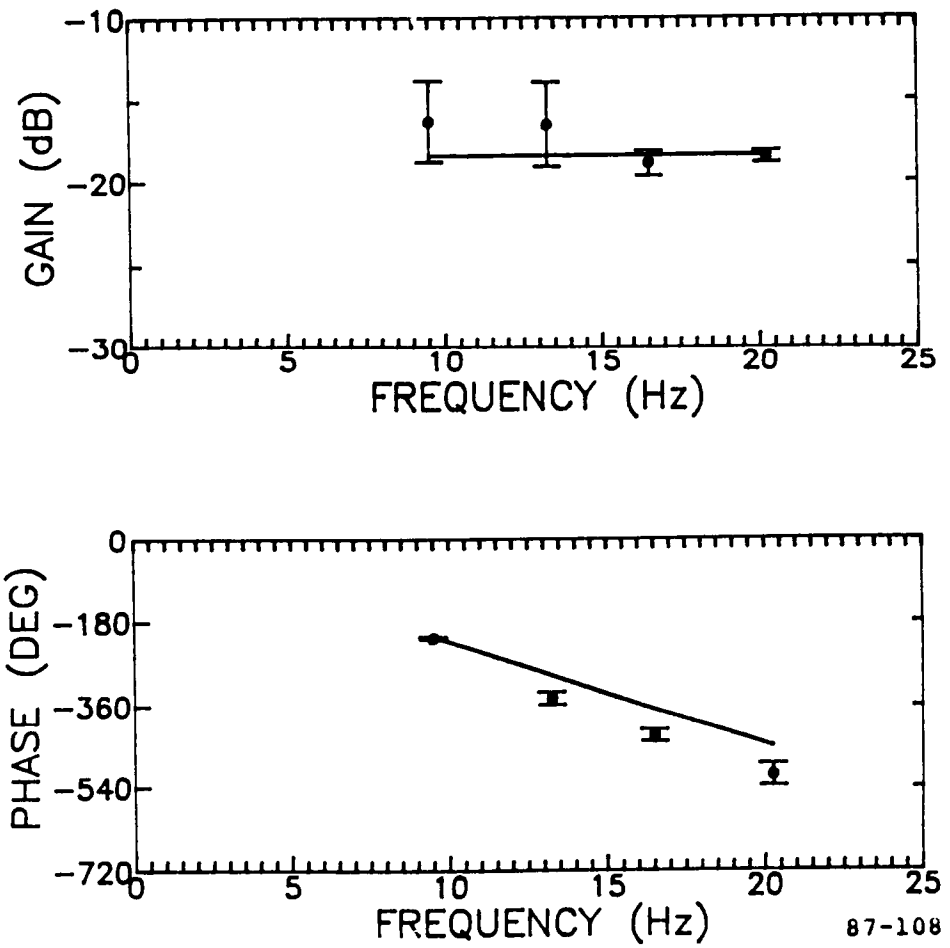

Figure 4.5: Single-Subject Transfer Function Data for Baseline Task Loading

Comparison of the figures shows that the gain/phase trends of all three task loading conditions are similar. The gain means are all relatively flat with frequency and the phase means all follow a fairly consistent linear trend with frequency. The major differences are to be found in the across-replication variances. Here, one can characterize the baseline task (figure 4.5) as one with large low-frequency gain variance, the CRT task (figure 4.7) as one with large high-frequency gain variance, and the PMT task (figure 4.6) as one with large overall gain variance, and large high-frequency phase variance.

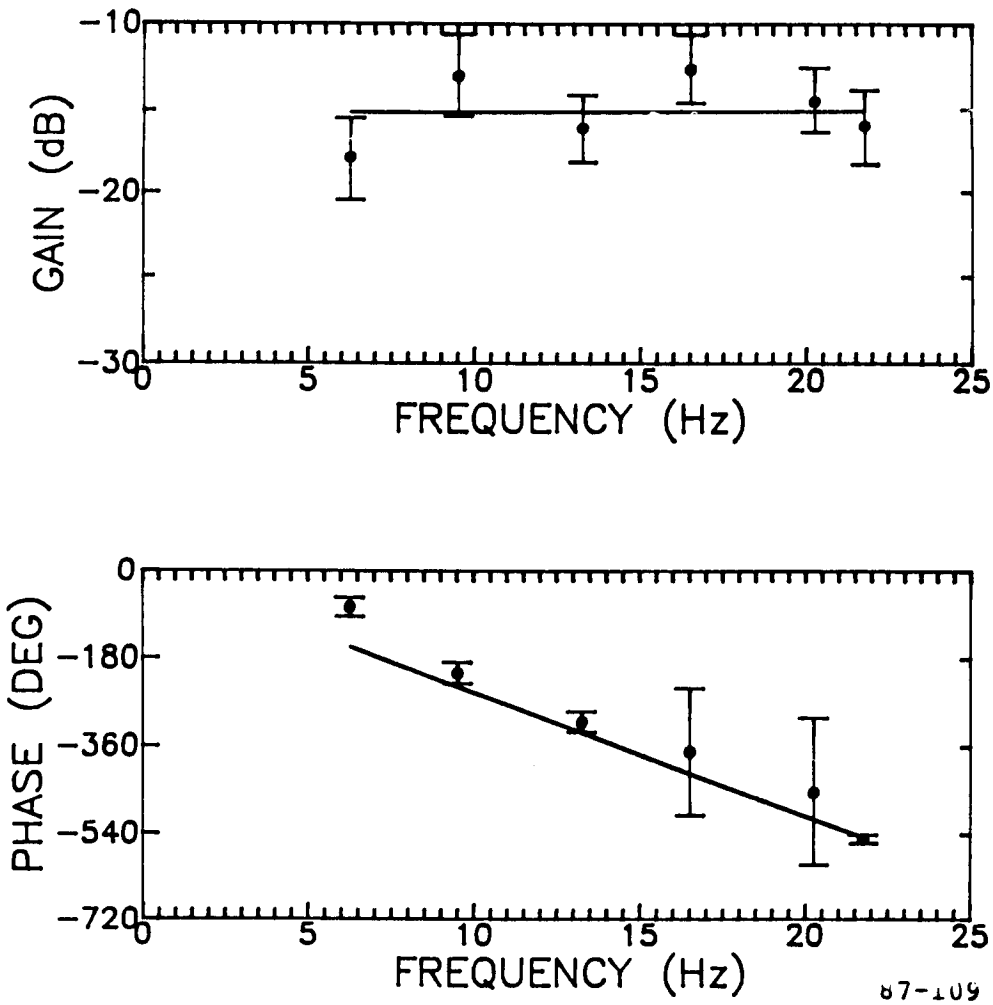

Figure 4.6: Single-Subject Transfer Function Data for Probability Monitoring Task

The model parameters generated by this procedure are given in table 4.4, along with the corresponding number of replications N, and the root mean matching error ($=\sqrt{J/N}$), to provide some indication as to relative goodness-of-fit.

Table 4.4: Gain/Delay Model Parameters for Three Task Loading Conditions (Single Subject)

Task	Gain K (dB)	Delay T_D (msec)	Replications N	Match Error $\sqrt{J/N}$
Baseline	-18.4	65	4	2.8
PMT	-15.1	70	6	2.0
CRT	-15.9	70	5	1.0

Two general trends are evident. First, the gain in the baseline case is lower than that found under the two loading conditions; this may be seen directly from the differences in the gain trends of figures 4.5 through 4.7. Second, the root mean match error is worst in the baseline case, improves with the PMT task, and is best in the CRT task. Note also that no significant differences in time delay are to be seen across conditions; this null result is reflected in the similarity of the model phase curves of figures 4.5 through 4.7.

On the basis of this preliminary data set and analysis results, one may conclude the following. First, there does appear to be gain enhancement over baseline, when the subject is loaded by either of the two tasks (PMT or CRT). A corresponding time delay difference is not to be seen, however. Second, there does not appear to be any significant difference between the two loading tasks, in terms of mean measurement values. There are, however, apparent differences in data variability with task and frequency. Finally, the data trends, across the three conditions, can be quite well modeled with an exceptionally simple analytic transfer function: a variable-gain delay. The

use of more complex models would not appear to be warranted by the data, at least over the frequency range studied.

It is clear that these conclusions would be strengthened if they held up in the face of additional measurement frequencies, further modeling efforts, and a wider range of loading tasks. Additional measurement frequencies would serve to pin down both the low- and high-frequency response trends, and help identify fundamental model order in the overall transfer function. Further modeling efforts could then determine if the simple model of (4.22) is an adequate description over the frequency range of interests, given the trends and variability of the data, or whether a more complex model is required to account for response trends with frequency, and across loading conditions. Finally, a wider range of loading tasks would serve to identify candidate tasks that may be more effective modifiers of ssVER characteristics. This would support a wider-scope identification and classification of differential efforts on the ssVER, to build a basis for ssVER-based differential task loading analysis.

We should recognize that the conclusions presented here are very tentative, based on the data from one subject. A similar gain/delay model analysis of comparable data by Junker (1986), over a 7-subject pool, failed to identify a significant and consistent across-subject pattern of gain/delay trends with task loading, although loading tasks different from those reported here were used in the Junker study. Clearly, a larger subject pool is called for if we are to extend the current findings and attach statistical significance to the results presented here.

5. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

5.1 Summary

The basic goal of the study reported here was to develop and validate analytic and experimental techniques to identify features of the ssVER that correlate with mental state, in particular, cognitive loading. Supporting program objectives were the specification of the stimulus/response protocols to be used in experimental validation studies, the specification of the analysis techniques to be used for response modeling, their implementation in a software package fully compatible with the sponsoring LaRC facility, and their application to ssVER data generated in one or more controlled mental loading experiments.

Our overall technical approach was a three-stage process: 1) a review of the basic systems identification background relevant to the problem of identifying the functional characteristics of the VER; 2) the development and implementation of a software package for experiment control and data analysis; and 3) the demonstration of the proposed identification and modeling techniques, via controlled mental loading experiments.

To provide some background for the type of system identification techniques required for a rational analysis of the VER, we reviewed basic input/output functional modeling. The description was done in mathematical terms to provide a basis for the interpretation of conventional (and unconventional) VER analysis techniques, in a more formally defined functional modeling context.

The review attempted to formalize the basic input/output structure of the VER, with the proposal of a quasi-linear model structure for the response: a response comprised of a "signal" portion which is linearly dependent on the input, and a "noise" portion which is independent of the input. The

implications for identification of the impulse response and the steady-state response were then discussed, and related to the conventional tVER and ssVER techniques already in use. The review then discussed the implications of different ssVER identification stimuli, focusing on periodic impulsive flashing and sinusoidal continuous modulation. The review concluded with a discussion of quasi-linear modeling issues, which center on sum-of-sines stimulation of the ssVER, and subsequent identification of model structure and parametric dependence on imposed loading.

The second stage of our technical approach focused on the development and implementation of a software package for experiment control, data analysis, and model identification. The package consists of four main programs:

1) VERRUN: This provides for pre-run setup of the experimental parameters, generation of the run-time sum-of-sines (SOS) stimulus, recording of the resulting response, and calculation of simple post-run statistics.

2) VERNAL: This supports time- and frequency-domain analysis of the time histories recorded by VERRUN, and provides for computation of the EEG RMS signal levels, and ssVER transfer function and remnant spectra.

3) ENSMBL: This calculates ensemble average statistics across individual subject runs, for the time- and frequency-domain analysis data generated by VERNAL.

4) MODLER: This supports the fitting of specified analytic transfer functions to the ensemble average transfer function data generated by ENSMBL, and supports the generation of an optimized model parameter set.

These programs provide for a full capability from data generation to model analysis, and can support, via direct expansion, an advanced modeling program under future research efforts.

The third stage of our technical approach consisted of a demonstration of the proposed identification and modeling techniques, via direct experimentation and analysis. This effort began with a brief review of past ssVER identification efforts, to uncover dominant frequency domain trends, and

to specify the desired identification bandwidth. A pre-experimental design effort then focused on a specification of appropriate time base parameters (sample rates, run times, etc.), a specification of the required sum-of-sines stimulus parameters (amplitudes, phases, etc.), and a calibration protocol to determine stimulus intensity and response amplification levels.

The demonstration experiment was conducted at NASA LaRC, using the delivered ssVER package and pre-experimental design parameters. Subjects were given three tasks: a null task comprised of viewing a blank display (baseline); a task testing visual perceptual processing; and a task testing encoding of working memory.

The VERRUN package was used to control the run-time ssVER stimulus generation, response recording, and data file generation. Following the experimental runs, the VERNAL package was used to generate corresponding single-run RMS level and frequency-domain files. These files were then grouped according to task loading conditions imposed, and the ENSMBL package was used to compute single-subject across-replication ensemble files, comprised of the average RMS level and frequency-domain metrics. The MODLER program was then used to fit very simple transfer function models to the observed data, to demonstrate the method's descriptive simplicity.

5.2 Conclusions

The primary result of this study has been the development and demonstration of systems analysis techniques for modeling the ssVER and relating it to cognitive loading. The major study conclusions supporting this development and demonstration effort can be summarized as follows.

The review on input/output functional modeling conducted under this effort provides a general framework for relating conventional transient VER (tVER) and steady-state VER (ssVER) techniques already in use; it also

provides a basis for the development of advanced VER identification methods, and corresponding stimulus/response models. As noted in the review, the tVER technique has a number of short-comings, including: the potential for amplitude response saturation, which hampers any linear modeling effort; a lack of generated remnant statistics, which may eventually provide important clues as to VER function and workload correlation; and, an overdependence on ad hoc time-domain features, which directs attention away from the essential transfer characteristics of the VER system. The conventional ssVER technique using repetitive strobe stimulation extends the tVER approach into the frequency domain, but brings with it its own set of problems, including: potential confounding of responses due to harmonic distortion; inflexibility in stimulus amplitude and frequency; and, stimulus predictability on the part of the presumably "causal" subject.

Many of these problems are avoided or ameliorated when using sum-of-sines (SOS) stimulation of the ssVER, in conjunction with quasi-linear model analysis. The basic identification procedure focuses on the transfer features of the VER system itself (structural form and parametric values), and serves to separate the response into input-related and system-generated components. The approach also allows for the quantification of remnant response which is uncorrelated with the input, and which may reflect cognitive loading effects, such as seen in alpha-suppression. The SOS-based ssVER also provides, via appropriate adjustment of the stimulus parameters, means for: minimizing the effects of amplitude saturation, by distributing the stimulus power across a wide frequency band; avoiding the confounding effects of harmonic distortion by appropriate probe frequency selection; maximizing reliability in the transfer estimates by selective "shaping" of the SOS spectrum; and ameliorating the effects of stimulus predictability, by random phasing of the SOS components.

The second major conclusion of the study concerns the development and demonstration of an integrated software package for the control and analysis of ssVER cognitive loading experiments. Four packages were developed under the study: VERRUN, VERNAL, ENSMBL, and MODLER. Operation of the VERRUN package at the LaRC facility demonstrated the flexibility of the software in pre-run set-up tasks, and its ease of operation during run-time control. Post-run execution of the VERNAL package demonstrated the generation of a variety of single-run time- and frequency-domain ssVER measures, under both interactive control and batch mode operation. Subsequent processing of the single-run data by the ENSMBL package provided a direct means for generating across-replication and across-subject ensemble response statistics. Finally, operation of the MODLER package demonstrated how interactive software can support the quasi-linear ssVER development effort, and provide the analyst with a direct means of evaluating candidate response models.

The third major set of study conclusions were obtained from a pilot ssVER experiment conducted at LaRC during the course of this effort. Subjects were tested across three task loading conditions: a null task, a Probability Monitoring Task (PMT), and a Continuous Recall Task (CRT). A number of replications were made under each task, to yield several single-subject across-replication ensemble-average frequency-domain measures of the ssVER. Subsequent model fits of the ensemble data means were based on a simple variable-gain delay transfer function model.

The results presented here show that the transfer functions in all three task loading conditions are reasonably well-modeled by a simple two-parameter gain/delay model. The model accounts for the measured flat gain and linear phase trends with frequency, across the bandwidth of interest and the task triplet. On the basis of the preliminary data set and analysis results presented here, one may conclude the following. First, there does appear to

be gain enhancement over baseline, when the subject is loaded by either of the two tasks (PMT or CRT). A corresponding time delay difference is not to be seen, however, remaining fixed at about 70 msec across all three conditions. Second, there does not appear to be any significant difference between the two loading tasks, in terms of mean measurement values. There are, however, apparent differences in data variability with task and frequency. Finally, the data trends, across the three conditions, can be quite well modeled with an exceptionally simple analytic transfer function: a variable-gain delay. The use of more complex models would not appear to be warranted by the data at least over the frequency range studied.

In short, we have demonstrated how a systems approach to functional modeling of the ssVER can be the basis for the eventual development of a rational and reliable ssVER-based cognitive loading indicator. The review we conducted shows how both tVER and ssVER research is related at the functional stimulus/response level, and the corresponding software development effort demonstrates how basic identification techniques can be applied directly to ssVER characterization. The pilot experiment conducted under this study provided a test of this overall procedure, and the results show that a very simple model can indeed capture the basic dynamic response of the ssVER, under different cognitive loading tasks. It remains to be seen, however, whether the observed loading sensitivity holds up over a larger subject base and a wider range of tasks, or if individual subject differences, or other uncontrollable factors, will tend to dominate the ssVER. It should be clear, however, that the general methodology developed and evaluated here can serve as a starting point for a more concerted effort aimed at developing a sensitive ssVER-based workload metric.

5.3 Recommendations

We recommend that this research effort be continued, to support the longer-term goal of developing a sensitive, reliable, and discriminating indicator of internal mental state. We recommend that additional effort be focused in three areas: 1) an expansion of the experimental effort, to broaden the data base; 2) an advancement of the modeling effort, to extend beyond current simple transfer function models; and 3) an application of the developed methodology to more realistic workload situations, to evaluate operational feasibility. We discuss these briefly below.

First, we recommend expansion of the scope of the experimental effort, to extend the current findings and attach statistical significance to the results presented here. At a minimum, we see a need for the use of a larger subject pool, and a wider range of loading tasks. The larger subject pool would obviously allow us to assess the validity of the current, tentative findings. It would also allow us to assess the magnitude of individual subject variations, and evaluate the method's potential utility for working with normative standards of ssVER-based workload levels. A wider range of loading tasks is also called for, if we are to evaluate the full potential of an ssVER-based metric. It would not be unreasonable to expect that some task loads will be more effective than others at modifying ssVER characteristics, and a wider range of tasks would support the development of a sensitive metric with the potential for discriminating among task types and/or load levels.

Second, we recommend an advancement of the modeling effort to extend the current findings, and to evaluate the implications of alternate modeling concepts on the development of an ssVER-based metric. Extension of the current findings could be achieved fairly directly in three areas: 1) functional modeling of the ssVER remnant spectrum, to identify potential sensitivity to task loading; 2) stimulus/response measurement over a wider

bandwidth, to help identify fundamental model order in the ssVER dynamic response; and 3) advanced ensemble-average processing to maximize the contributions of "noisy" individual subject runs, and minimize the amount of stimulus/response testing needed for reliable modeling. Evaluation of alternative modeling concepts could be initiated with a new experimental protocol that combined both ssVER and tVER techniques. Both could be used in a complementary fashion to directly support a quasi-linear modeling effort, but they could also individually serve to uncover any significant response non-linearities. For example, the ssVER stimulus could be used to search for multiplicative harmonic effects, while the tVER stimulus could probe for amplitude limiting. Subsequent non-linear functional modeling would then provide a structure for relating these effects, and evaluating their sensitivity to imposed loading, and ultimately, their potential utility as a component of a sensitive VER-based loading metric.

Finally, we recommend an evaluation of the developed methodology in a more realistic workload environment, to assess operational feasibility as early as practicable. Effort should be directed at evaluating the potential intrusiveness of the visual stimulus and the EEG recording hardware, and any potential procedural interference caused by the methodology itself. In addition, an evaluation should also be made of the metric's sensitivity or immunity to normal operational environmental factors, such as electrical interference, subject motion, alternate-modality stimulation, etc. We would recommend a three-stage evaluation process, conducted in: 1) a laboratory environment, where operational task elements and environmental factors are introduced and controlled separately; 2) a moderately realistic flight simulator environment, to evaluate combined effects under controlled conditions; and 3) a flight test environment, to demonstrate the methodology in a realistic operational scenario. Such evaluations could then lead to

modifications in the methodology to improve operational performance and to identify guidelines for proper usage of the technique in realistic workload environments.

6. REFERENCES

Dixon, W.J., BMD Biomedical Computer Programs, University of California Press, January, 1973

Junker, A.M., and Peio, K.J. "In Search of a Visual-Cortical Describing Function," presented at 20th Annual Conference on Manual Control, San Francisco, CA, 1984.

Junker, A.M., A Systems Engineering Based Methodology for Analyzing Human Brain Function, Doctoral Thesis, University of Connecticut, 1986.

Kuo, B.C., Analysis and Synthesis of Sampled-Data Control Systems, Prentice Hall, Inc., Englewood Cliffs, NJ, 1963.

Laning, Jr., J.H. and Battin, R.H., Random Processes in Automatic Control, McGraw-Hill Book Company, Inc., 1956.

Levison, W.H., and Zacharias, G.L., "The VERRUN and VERNAL Software Systems for Steady-State Visual Evoked Response Experimentation," NASA CR-172311, March 1984.

Moise, Jr., S.L., "Development of Neurophysiological and Behavioral Metrics of Human Performance," Final Report Sept. 1976 to Dec. 1979, AFAMRL-TR-80-39, May 1980.

O'Donnell, R., and Spicuzza, R., "Visually Evoked Brain Potentials as Aids in Display Design," Frontiers in Medical Signal Processing, MIDCON 77, November, 1977.

Oppenheim, A.V. and Schafer, R.W., Digital Signal Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1975.

Peio, K.J., and Junker, A.M., "Visually Evoked Response from Sum of Sines Stimulation," Proceedings of NAECON, Dayton, OH, May 1983.

Regan, D., "Steady-State Evoked Potentials," J. Opt. Soc. Am., Vol. 67, 1977.

Shingledecker, C., "A Task Battery for Applied Human Performance Assessment Research," AFAMRL Technical Report AFAMRL-TR-84-071, AFAMRL, WPAFB, Ohio, November 1984.

Truxal, J.G., Automatic Feedback Control System Synthesis, McGraw-Hill Book Company, Inc., 1955

Tustin, A., "The Nature of the Operator's Response in Manual Control and Its Implications for Controller Design," J. IEEE, Vol. 94, 1947.

Wickens, C.D., Israel, J., and Donchin, E. "The Event Related Cortical Potential as an Index of Task Workload," Proc. of Human Factors Society, 21 st Annual Meeting, 1977.

Wilson, G.F., "Steady State Evoked Responses as a Measure of Tracking Difficulty," AFOSR Final Report, Contract No. F49620-79-C-0156, Air Force Office of Scientific Research, Bolling AFB, D.C., November 1979

Wilson, G.F. and O'Donnell, R.D., "Human Sensitivity to High Frequency Sine Wave and Pulsed Light Stimulation as Measured by the Steady-State Cortical Evoked Response," AFAMRL-TR-80-133, AMRL, Wright-Patterson AFB, OH, Feb. 1981.

Zacharias, G.L., "Physiological Correlates of Mental Workload," NASA CR-166054, Feb. 1980.

Zacharias, G.L., "SOS Stimulus Design for EEG VER Experiments," BBN Technical Memorandum, Bolt Beranek and Newman Inc., Cambridge, MA, November, 1982.

Zacharias, G.L., and Ho, A., "VERSOS: A System for Sum-of-Sines Stimulation of the Electroencephalographic Visual Evoked Response," BBN Report No. 5214, Bolt Beranek and Newman Inc., Cambridge, MA, November 1982.

APPENDIX A: LISTING FOR PROGRAM VERRUN

```
      PROGRAM VERRUN
C
      COMMON /TTLCM1/ FNAME, IDATE, ITIME, TITLE
      COMMON /LENGTH/NRMAX
      COMMON /TMPCOM/TMPVEC
      COMMON /FILCOM/ NUMFIL, IFILE
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK, LANS, ICHNGE, MODE, DUMMY
      CHARACTER*8 ITIME
      CHARACTER*9 IDATE
      CHARACTER*10 FNAME(10)
      CHARACTER*255 TITLE
      CHARACTER*4 CHNAME(4)
      INTEGER HARM(15), PMUL(15), DUM(4)
      DIMENSION TMPVEC(20), AMP(15)
C
      DIMENSION IDATA(16400)
C
C     * DIMENSION OF IDATA
      DATA IDIM/16400/
C     * LUN FOR DATA FILE
      DATA LUNFIL/3/
C     * LUN FOR TTY
      DATA LUNTTY /0/
      DATA NCHAN /4/
C     * START WITH UNDEFINED MODE
      DATA MODE /'U'/
C
C     SET UP PARAMETERS...
C
 100  NRMAX = IDIM/NCHAN
      ICHNGE = 'Y'
      IRUN = 1
      IFILE = 0
      NUMFIL = 1
      DO 101 I=1,4
 101  DUM(I)=0
C
 110  CALL PARSET (ICHNGE,LUNFIL,NCHAN,IRUN,ISAMP,NPER,
     1             CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
C
C     * SET MODE TO P OR R
      IF (MODE .NE. 'U') GO TO 120
      MODE = 'P'
      IF (LASK ('DOING A RUN NOW? ') .EQ. 'Y')  MODE = 'R'
C     * GO SET PARAMETERS
 120  IF (MODE .EQ. 'P') GOTO 300
C
C             NORMAL RUN MODE
C
C     CHECK TO SEE IF USER WANTS MULTIPLE RUNS
      IF (LASK('MULTIPLE RUNS? ') .EQ. 'N') GOTO 140
         CALL TTYOUT ('ENTER NUMBER OF RUNS: $')
         NUMFIL = IANS(1,10)
         IFILE = 0
```

```
C
 140   DO 10 J=1,NUMFIL
C
C      * ZERO OUT IDATA
       DO 150 I = 1,IDIM
 150   IDATA(I) = 0
C
C      * READ TITLE FROM TTY
 200   IRW = 1
       CALL TITLER (IRW, LUNTTY, MODE, IRUN)
C      * WRITE HEADER ONTO FILE
       IRW = 2
C      * AND LEAVE OPEN
       ICLOSE = 2
       CALL RWHEAD (IRW,LUNFIL,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1              CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
       CALL TTYOUT ('GENERATING SOS SIGNAL NOW...')
       CALL SOSGEN (NPER, NRUN, NCHAN, NCOMP, HARM, AMP, PMUL, IDATA)
       CALL TTYOUT ('TYPE S TO START: $')
       DUMMY = LANS ('S', 'S')
       CALL LOOP (ISAMP, NRUN, NCHAN, IDATA)
       CALL TTYOUT ('STORING DATA NOW...')
C      * WRITE DATA TO FILE & CLOSE IT
       IRW = 2
       CALL RWDATA (IRW, LUNFIL, NRUN, NCHAN, IDATA)
C
       CALL STATUS (DUM,2,FNAME(J),NCHAN,CHNAME)
C      * INCREMENT RUN NUMBER
       IRUN = IRUN + 1
       ICHNGE = 'N'
       CALL SOSPHS (NOMSOS, ICHNGE, IRUN, NCOMP, PMUL, ISEED)
 10    CONTINUE
C
       CALL TTYOUT (' ')
       IF (LASK ('DOING ANOTHER RUN? ') .EQ. 'N') GOTO 210
       ICHNGE = LASK ('ANY CHANGES? ')
       IFILE = 0
       NUMFIL = 1
       GOTO 110
C
 210   IF (LASK ('SET UP A PARAMETER FILE? ') .EQ. 'N')  STOP
C
       MODE = 'P'
       GOTO 100
C
C              PARAMETER FILE SET UP MODE
C
C      * READ TITLE FROM TTY
 300   IRW = 1
       CALL TITLER (IRW, LUNTTY, MODE, IRUN)
C      * WRITE HEADER ONTO FILE
       IRW = 2
C      * AND CLOSE IT
       ICLOSE = 1
       CALL RWHEAD (IRW,LUNFIL,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1              CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
```

```
C
C              USER SPECIFIES WHAT'S NEXT
C
      CALL STATUS (DUM,2,FNAME(J),NCHAN,CHNAME)
C
      CALL TTYOUT (' ')
      IF (LASK ('ANOTHER PARAMETER FILE? ') .EQ. 'Y')  GOTO 110
      IF (LASK ('DOING A RUN NOW? ') .EQ. 'N')  STOP
C
      MODE = 'R'
      GOTO 100
      END
```

```
C
      SUBROUTINE LOOP (ISAMP, NRUN, NCHAN, IDATA)
C
C             INPUTS: (VIA ARGLST)    ISAMP, NRUN, NCHAN, IDATA
C             OUTPUTS:(VIA ARGLST)    IDATA
C
      LOGICAL CLWAIT
C
      CHARACTER*4 DUMMY(4)
      DIMENSION ITEMP(4)
      DIMENSION IDATA(1)
C
C     * SET CLOCK 100KHZ
      DATA IRATE /2/
      DATA IZERO /2048/
C     * TEST CODE
      DATA IFLIP,ITEST/1,0/
C
C     * GET TICK COUNT
      NTEMP = 10.**(4-IRATE) + 0.1
      NTICKS = NTEMP * ISAMP
C     * SET IDATA INDEX
      I = 1
C
C     * CREATE DUMMY NAME ARRAY
      DO 5 I=1,NCHAN
 5    DUMMY(I)='    '
C
C     * STOP CLOCK & ZERO D/A'S
      CALL CLSTOP
      CALL DTOA (0, IZERO)
      CALL DTOA (1, IZERO)
C
C     * THEN START CLOCK
      CALL CLSTRT (IRATE, NTICKS)
C
      INIT = 0
C
      DO 100 IFRAME = 1, NRUN
      IF (CLWAIT()) GOTO 10
      CALL TTYOUT ('*****LOOP: BAD TIME INTERVAL*****')
      STOP
 10   CONTINUE
C
      ITEMP(1)=IDATA(I)
C
C     * TEST CODE
      CALL DTOA (0, ITEST)
      CALL DTOA (1, ITEMP (1))
      CALL ATOD (1, ITEMP (2))
      CALL ATOD (2, ITEMP (3))
      CALL ATOD (3, ITEMP (4))
C
      CALL STATUS (ITEMP,INIT,'DUMMY',NCHAN,DUMMY)
C
      DO 30 ICHAN=2,NCHAN
```

```
 30     IDATA(I+ICHAN-1)=ITEMP(ICHAN)
C
       I = I + NCHAN
C
       IFLIP = -IFLIP
       IF (IFLIP .EQ. 1) ITEST=0
       IF (IFLIP .EQ. -1) ITEST=4095
C
 100   CONTINUE
C
C      * STOP CLOCK & ZERO D/A'S
       CALL CLSTOP
       CALL DTOA (0, IZERO)
       CALL DTOA (1, IZERO)
       RETURN
       END
```

```
C
      SUBROUTINE NAMPAR (NOMPAR, NCHAN, CHNAME)
C
      COMMON /LUNCOM/ LUNTTY
      CHARACTER*1 LASK, NOMPAR
      CHARACTER*4 CHNAME(4)
C
      CHNAME(1) = 'SOS '
      CHNAME(2) = 'LITE'
      CHNAME(3) = 'EEG1'
      CHNAME(4) = 'EEG2'
C
      IF (NOMPAR .EQ. 'Y') RETURN
      CALL TTYOUT ('**********CHANNEL NAME PARAMETERS**********')
      IF (LASK ('NOMINAL CHANNEL NAMES? ') .EQ. 'Y') GOTO 13
      DO 11 I=1,NCHAN
         WRITE (LUNTTY,14) I, CHNAME(I)
 14      FORMAT ('NAME FOR CHANNEL ',I1,' : ',A4,', ',$)
         IF (LASK ('WANT TO CHANGE NAME? ') .EQ. 'N') GOTO 11
         WRITE (LUNTTY,30) I
 15      READ (LUNTTY,40,ERR=101) CHNAME(I)
         GOTO 11
 101     WRITE (LUNTTY,70)
         GOTO 15
 11   CONTINUE
 13   IF (LASK ('WANT NAMES LISTED? ') .EQ. 'N') GOTO 90
 60   DO 10 I=1,NCHAN
         WRITE (LUNTTY,20) I,CHNAME(I)
 20      FORMAT ('NAME FOR CHANNEL ',I1,' : ',A4)
 10   CONTINUE
      IF (LASK ('ANY CHANGES? ') .EQ. 'N') GOTO 90
 50   CALL TTYOUT ('ENTER INDEX FOR NAME TO BE CHANGED: $')
      IDUM = IANS(1,4)
      WRITE (LUNTTY,30) IDUM
 30   FORMAT ('ENTER NEWNAME FOR CHANNEL ',I1,': ',$)
 80   READ (LUNTTY,40,ERR=100) CHNAME(IDUM)
 40   FORMAT (A4)
      IF (LASK ('ANOTHER CHANGE? ') .EQ. 'Y') GOTO 50
      GOTO 13
 90   RETURN
 100  WRITE (LUNTTY,70)
 70   FORMAT ('ERROR - INPUT MUST BE STRING NO MORE THAN 4 ',
     1        'CHARACTERS IN LENGTH.  TRY AGAIN: ',$)
      GOTO 80
      END
```

```
C
      SUBROUTINE PARSET(ICHNGE,LUNFIL,NCHAN,IRUN,ISAMP,NPER,
     1                  CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
C
C     SETS THE PROBLEM PARAMETERS BY USER-SPECIFIED INPUTS, OR...
C                               BY READING FROM AN OLD FILE
C
C     INPUTS:  (VIA ARGLST)   ICHNGE, LUNFIL, IRUN
C     OUTPUTS: (VIA ARGLST)   ISAMP
C              (     "    )   NPER, NRUN, NCOMP
C              (     "    )   HARM, AMP, PMUL, ISEED
C
C
      CHARACTER*1 LASK, NOMPAR, ICHNGE
      CHARACTER*4 CHNAME(4)
      INTEGER HARM(1), PMUL(1)
      DIMENSION AMP(1)
C
      IF (ICHNGE .EQ. 'N') GOTO 200
      CALL TTYOUT (' ')
      IF (LASK ('PARAMETERS FROM A FILE? ') .EQ. 'Y') GOTO 300
C
C     GET PARAMETERS DIRECTLY FROM USER
C
 100  NOMPAR = LASK('NOMINAL PARAMETERS? ')
      CALL NAMPAR (NOMPAR, NCHAN, CHNAME)
      CALL TIMPAR(NOMPAR, ISAMP, NPER, NRUN)
 200  CALL SOSPAR (NOMPAR,ICHNGE, IRUN, NPER, NCOMP,
     1    HARM, AMP, PMUL, ISEED)
      RETURN
C
C     GET PARAMETERS FROM AN OLD FILE
C
C     * READ HEADER FROM FILE
 300  IRW = 1
C     * AND CLOSE IT
      ICLOSE = 1
      CALL RWHEAD(IRW,LUNFIL,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1            CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
      CALL TTYOUT (' ')
      RETURN
      END
C
      CHARACTER FUNCTION PRIME (NUMBER)
C
      ISQRT = SQRT (REAL(NUMBER))
      IF (MOD (NUMBER,2) .EQ. 0) GOTO 20
      DO 10 J=3,ISQRT,2
 10      IF (MOD (NUMBER,J) .EQ. 0) GOTO 20
      PRIME = 'Y'
      RETURN
 20   PRIME = 'N'
      RETURN
      END
```

```
C
      SUBROUTINE RWDATA (IRW, LUNIT, NFRAME, NCHAN, IDATA)
C
C     RWDATA READS/WRITES THE DATA ARRAY IDATA FROM/TO FILE
C
C             INPUTS: (VIA ARGLST)    IRW (1 = READ, 2 = WRITE)
C                                     LUNIT, NFRAME, NCHAN, IDATA
C             OUTPUT: (VIA ARGLST)    IDATA
C
      COMMON /LUNCOM/ LUNTTY
C
      DIMENSION IDATA(1)
C
      DATA NCMAX/4/
C
      IF(NCHAN .LE. NCMAX) GOTO 10
      CALL TTYOUT('******RWDATA: NCHAN .GT. NCMAX******')
      STOP
C
 10   GOTO (100,200) IRW
C
C     READ DATA FROM FILE & LOAD IDATA
 100  IF (LUNIT .NE. LUNTTY) GOTO 105
      CALL TTYOUT ('******RWDATA: TRYING TO READ FROM TTY******')
      STOP
C
 105  READ (LUNIT, 999)
 999  FORMAT(/,/)
C
      INDEX = 0
      DO 120 I = 1, NFRAME
      READ (LUNIT,1000) IDUMMY, (IDATA(INDEX+J), J=1,NCHAN)
 1000 FORMAT (1X,5I5)
 120  INDEX = INDEX + NCHAN
      GOTO 300
C
C     WRITE ALL CHANNELS OF DATA FROM IDATA TO FILE (OR TTY)
 200  WRITE (LUNIT, 2000) NCHAN
 2000 FORMAT (/, 1X, '***RECORDED DATA OF ', I3, ' CHANNELS***')
      WRITE (LUNIT, 2001)
 2001 FORMAT (2X, 'IFRM','  C1 ','  C2 ','  C3 ','  C4 ')
      INDEX = 0
      DO 220 I=1,NFRAME
      WRITE (LUNIT,1000) I, (IDATA(INDEX+J),J=1,NCHAN)
 220  INDEX = INDEX + NCHAN
      IF (LUNIT .EQ. LUNTTY) RETURN
C
 300  CLOSE (UNIT=LUNIT,STATUS='KEEP')
      RETURN
      END
```

```
C
      SUBROUTINE RWHEAD (IRW,LUNIT,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1                   CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
C
C     READS/WRITES HEADER FROM/TO A DATA FILE
C     ALSO WRITES HEADER TO TTY
C
C     INPUTS: (VIA ARGLST)    IRW    (1=READ HEADER,2=WRITE HEADER)
C             (     "    )    LUNIT
C             (VIA ARGLST)    ICLOSE (1=CLOSE FILE, 2=LEAVE FILE OPEN)
C     I/O:    (VIA ARGLST)    IRUN, ISAMP
C             (     "    )    NPER, NRUN, NCOMP
C             (     "    )    HARM, AMP, PMUL, ISEED
C             (VIA TIMCOM)    PZERO, FZERO, TSAMP, TRUN
C
      COMMON /TIMCOM/ PZERO, FZERO, TSAMP, TRUN
      COMMON /TTLCM1/ FNAME, IDATE, ITIME, TITLE
      COMMON /TTLCM2/ NLINE
      COMMON /FILCOM/ NUMFIL, IFILE
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 IOPEN,MDUMY
      CHARACTER*8 ITIME
      CHARACTER*9 IDATE
      CHARACTER*10 FNAME(10)
      CHARACTER*255 TITLE
      CHARACTER*4 CHNAME(4)
      INTEGER HARM(1), PMUL(1), HOURS, SECONS
      DIMENSION AMP(1)
C
      DATA NVERS /2/
      DATA IOPEN /'N'/
      DATA MDUMY /'P'/
C
      IF (LUNIT .EQ. LUNTTY) GOTO 201
C
C     * GET FILE NAME
      IF (IFILE .GT. 0) GOTO 7
C
   4  DO 6 J=1,NUMFIL
   6  CALL FILNAM (IRW, FNAME(J), NCHAR)
C
  7   IFILE = IFILE + 1
      IF (IOPEN .EQ. 'N') GOTO 10
C     * AND CLOSE IT IF OPEN
      CLOSE (UNIT=LUNIT,STATUS='KEEP')
C     * AND INDICATE IT'S CLOSED
      IOPEN = 'N'
 10   GOTO (100, 200) IRW
C
C             READ FROM FILE
C
 100  CONTINUE
      OPEN (UNIT=LUNIT,FILE=FNAME(IFILE),
     1         STATUS='OLD')
      IOPEN = 'Y'
```

```
      READ (LUNIT, 105) NVERS
 105  FORMAT (17X, I1,/,/)
      CALL TITLER (IRW, LUNIT, MDUMY, IRUN)
      READ (LUNIT, 107) (CHNAME(I),I=1,NCHAN)
 107  FORMAT (/,/,2X,4(2X,A4))
      READ (LUNIT, 110) ISAMP
 110  FORMAT (/, /, 25X, I4)
      READ (LUNIT, 115) FZERO, PZERO
 115  FORMAT (17X, 1PE12.3, 21X, 1PE12.3)
      READ (LUNIT, 120) TEMP, NPER
 120  FORMAT (17X, 1PE12.3, 28X, I5)
      READ (LUNIT, 125) TRUN, NRUN
 125  FORMAT (17X, 1PE12.3, 28X, I5)
      READ (LUNIT, 130) NCOMP, ISEED
 130  FORMAT (/,/, 22X, I4, 27X, I5, /)
      READ (LUNIT, 135)  (HARM(I), AMP(I), PMUL(I), I=1,NCOMP)
 135  FORMAT (10X, I5, 16X, F6.3, 5X, I6)
      GOTO 300
C
C             WRITE TO FILE (OR TTY)
C
 200  CONTINUE
      OPEN (UNIT=LUNIT,FILE=FNAME(IFILE),
     1         STATUS='NEW')
      IOPEN = 'Y'
 201  WRITE (LUNIT, 205) NVERS
 205  FORMAT (1X, 'VERSION NUMBER: ', I1)
      WRITE (LUNIT, 206)
 206  FORMAT (/, 1X, '***RUN IDENTIFICATION***')
      CALL TITLER (IRW, LUNIT, MDUMY, IRUN)
      WRITE (LUNIT, 207)
 207  FORMAT (/, 1X, '***CHANNEL NAMES***')
      WRITE (LUNIT, 208) (CHNAME(I),I=1,NCHAN)
 208  FORMAT (2X,4(2X,A4))
      WRITE (LUNIT, 209)
 209  FORMAT (/, 1X, '***TIME BASE PARAMETERS***')
      WRITE (LUNIT, 210) ISAMP
 210  FORMAT (1X, 'SAMPLE PERIOD:  ', 8X, I4,' MSEC')
      WRITE (LUNIT, 215) FZERO, PZERO
 215  FORMAT (1X, 'BASE FREQUENCY: ',1PE12.3, '   HZ',
     1         4X, 'BASE PHASE: ',    1PE12.3, ' DEG')
      WRITE (LUNIT, 220) NPER*(ISAMP/1000.), NPER
 220  FORMAT (1X, 'SOS PERIOD:     ',1PE12.3, '  SEC',
     1         4X ,'WITH: ', 13X, I5, ' PTS')
      WRITE (LUNIT, 225) TRUN, NRUN
 225  FORMAT (1X, 'RUN LENGTH:     ',1PE12.3, '  SEC',
     1         4X, 'WITH: ', 13X, I5, ' PTS')
      WRITE (LUNIT, 229)
 229  FORMAT (/, 1X, '***SOS SIGNAL PARAMETERS***')
      WRITE (LUNIT, 230) NCOMP, ISEED
 230  FORMAT (1X, '# OF SOS COMPONENTS: ', I4,
     1         8X, 'RANDOM PHASE SEED: ', I5)
      WRITE (LUNIT, 234)
 234  FORMAT (2X, 'COMP', 5X, 'HARM', 7X, 'FREQ', 7X, 'AMP',
     1         8X, 'PMUL', 7X, 'PHS')
      WRITE (LUNIT, 235)  (J,HARM(J),FZERO*HARM(J),AMP(J),
```

```
     1                          PMUL(J),PZERO*PMUL(J), J=1,NCOMP)
 235   FORMAT (I5,5X,I5,5X,F6.2,5X,F6.3,5X,I6,5X,F6.1)
C
C      * RETURN IF JUST DONE TTY WRITE
       IF (LUNIT .EQ. LUNTTY) RETURN
C
 300   IF (ICLOSE .NE. 1)  RETURN
C      * CLOSE FILE
       CLOSE (UNIT=LUNIT,STATUS='KEEP')
C      * AND INDICATE CLOSED
       IOPEN = 'N'
       RETURN
       END
```

```
C
      SUBROUTINE SOSAMP (NOMSOS, NCOMP, AMP)
C
C     INPUTS: (VIA ARGLST)    NOMSOS, NCOMP
C     OUTPUTS:(VIA ARGLST)    AMP
C
      COMMON /TMPCOM/AMPTMP
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK, NOMSOS
      DIMENSION AMP(1), AMPNOM(15), AMPTMP(15)
C
      DATA AMPNOM /15 * 1./
      DATA RMSMIN, RMSNOM, RMSMAX /0., 1., 5./
      DATA AMIN, AMAX /0., 100./
C
      IF (NOMSOS .EQ. 'Y') GOTO 120
 100  IF (LASK ('NOMINAL RELATIVE AMPLITUDES? ') .EQ. 'Y') GOTO 120
C
 110  CALL TTYOUT ('ENTER (RELATIVE) AMPLITUDES: ')
      CALL VECTIN (1, 'AMP', NCOMP, AMPTMP, AMIN, AMAX)
      GOTO 140
C
 120  DO 130 J = 1, NCOMP
 130  AMPTMP(J) = AMPNOM(J)
C
 140  RMSLVL = RMSNOM
      IF (NOMSOS .EQ. 'Y') GOTO 150
      IF (LASK ('NOMINAL RMS LEVEL? ') .EQ. 'Y') GOTO 150
      CALL TTYOUT ('RMS LEVEL (VOLT) = $')
      RMSLVL = RANS(RMSMIN, RMSMAX)
C
 150  SUMSQ = 0.0
      DO 160 J = 1, NCOMP
 160  SUMSQ = SUMSQ + AMPTMP(J) * AMPTMP(J)
C
      SCALE = RMSLVL * SQRT(2.0/SUMSQ)
C
      DO 170 J = 1, NCOMP
 170  AMP(J) = SCALE * AMPTMP(J)
C
      IF (NOMSOS .EQ. 'Y') RETURN
      CALL TTYOUT ('$ ')
      IF (LASK ('LIST AMPLITUDES? ') .EQ. 'N') RETURN
C
      WRITE (LUNTTY, 200)
 200  FORMAT (1X, 'COMP', 7X, 'AMP', 7X, 'AMP (REL)', /)
      WRITE (LUNTTY, 201) (J, AMP (J), AMPTMP (J), J = 1, NCOMP)
 201  FORMAT (I4, 5X, F7.2, 5X, F7.2)
      CALL TTYOUT (' ')
      IF (LASK ('OK? ') .EQ. 'N') GOTO 100
      RETURN
      END
```

```
C
      SUBROUTINE SOSGEN(NPER,NRUN,NCHAN,NCOMP,HARM,AMP,PMUL,IDATA)
C
C     SOSGEN GENERATES SOS SIGNAL & LOADS IT INTO FIRST CHANNEL OF
IDATA
C
C             INPUTS:  (VIA ARGLST)
NPER,NRUN,NCHAN,NCOMP,HARM,AMP,PMUL
C             OUTPUTS: (VIA ARGLST) PMUL,IDATA
C
C     NOTES:  1)SOSGEN KEEPS HARMONIC COUNTER IN PMUL,OVERWRITING IT
C             2)SOS SCALING ASSUMES PLUS/MINUS 5 VOLT 12-BIT D/A
C
      INTEGER HARM(1),PMUL(1)
      DIMENSION AMP(1)
      DIMENSION IDATA(1)
C
      I=1
      CALL TABGEN(NPER)
      DO 10 IFRAME = 1, NRUN
      CALL SOSVAL(NPER,NCOMP,HARM,AMP,PMUL,SOS)
      CALL CONVRT (0,ITEMP,SOS)
      IDATA(I) = ITEMP
 10   I = I + NCHAN
      RETURN
      END
C
C
      SUBROUTINE TABGEN(NPER)
C
C     TABGEN CALCULATES HALF & QUARTER WAVE INDICES NHALF & NQUART
C            AND SETS UP QUARTER WAVE SINE TABLE SINTAB
C
C             WHERE SINTAB (N+1) = SIN (2 * PI * (N / NPER))
C             FOR 0 .LE. N .LE. (NPER / 4)
C      INPUT: (VIA ARGLST) NPER
C     OUTPUT: (VIA TABCOM) NHALF,NQUART,SINTAB
C
C     NOTE:   CURRENTLY ASSUMES NPER .LE. 4096
C
      DIMENSION SINTAB (1025)
C
      COMMON/TABCOM/NHALF,NQUART,SINTAB
C
      IF (NPER.LE.4096) GOTO 10
      CALL TTYOUT('******TABGEN: NPER TOO BIG')
      STOP
C
10    IF (NPER .NE. 0) GOTO 15
      STOP '**********TABGEN ZERO DIVIDE**********'
15    TWOPI=2.*3.14159
      NHALF=NPER/2
      NQUART=NHALF/2
      TEMP=TWOPI/NPER
      DO 20 N=0,NQUART
 20   SINTAB(N+1)=SIN(N*TEMP)
      RETURN
      END
```

```
C
      SUBROUTINE SOSVAL(NPER,NCOMP,HARM,AMP,PMUL,SOS)
C
C     CALCULATES NEW SOS VALUE FOR EACH CALL
C     AND INCREMENTS PMUL BY HARM
C
C      INPUT: (VIA ARGLST) NPER,NCOMP,HARM,AMP,PMUL
C     OUTPUT: (VIA ARGLST) PMUL,SOS
C
      INTEGER HARM(1),PMUL(1)
C
      DIMENSION AMP(1)
C
      SOS=0.
C
      DO 10 J=1,NCOMP
      N=PMUL(J)
      IF(N.GE.NPER)N=MOD(N,NPER)
      SOS=SOS + AMP(J)*SINFCN(N,NPER)
      N=N +HARM(J)
      PMUL(J)=N
10    CONTINUE
C
      RETURN
      END
C
      FUNCTION SINFCN(N,NPER)
C
C     CALCULATES SINFCN (N) = SIN (2 * PI (N / NPER))
C             FOR 0 .LE. N .LE. (NPER-1)
C     USES QUARTER WAVE SINE TABLE SINTAB
C
C      INPUT: (VIA ARGLST) N,NPER
C             (VIA TABCOM) NHALF,NQUART,SINTAB
C
C     OUTPUT:              SINFCN
C
      DIMENSION SINTAB (1025)
C
      COMMON/TABCOM/NHALF,NQUART,SINTAB
C
      NTEMP=N
      IF(NTEMP.GT. NHALF) NTEMP=NPER-NTEMP
      IF(NTEMP.GT.NQUART) NTEMP=NHALF-NTEMP
      SINFCN=SINTAB(NTEMP+1)
      IF(N.GT.NHALF) SINFCN=-SINFCN
      RETURN
      END
```

```
C
      SUBROUTINE SOSHMC (NOMSOS, NCOMP, NPER, HARM)
C
C     INPUTS  (VIA ARGLST)    NOMSOS, NCOMP, NPER
C             (VIA TIMCOM)    FZERO, TSAMP
C
C     OUPUTS  (VIA ARGLST)    HARM
C
      COMMON /TMPCOM/ FRQTMP
      COMMON /TIMCOM/ PZERO, FZERO, TSAMP
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LANS, LASK, NOMSOS
      INTEGER HARM (1)
      DIMENSION FRQNOM (15), FRQTMP (15)
C
      DATA FRQNOM /5., 10., 15., 20., 25., 30., 35., 40., 45.,
     1     50., 55., 60., 65., 70., 75./
C
      FMIN = FZERO
      FMAX = 1.0/(2.0 * TSAMP)
 100  IF (NOMSOS .EQ. 'Y') GOTO 120
      IF (LASK ('NOMINAL FREQUENCIES? ') .EQ. 'Y')  GOTO 120
C
 110  CALL TTYOUT ('ENTER DESIRED FREQUENCIES (HZ): ')
      CALL VECTIN (1, 'FREQ', NCOMP, FRQTMP, FMIN, FMAX)
      GOTO 140
C
 120  DO 130 J = 1, NCOMP
 130  FRQTMP (J) = FRQNOM (J)
C      * CHECK FOR LIMIT EXCEEDANCE
 140  IERR = 0
      DO 150 J = 1, NCOMP
      FTEMP = FRQTMP (J)
      IF ((FTEMP .LT. FMIN) .OR. (FTEMP .GT. FMAX)) IERR = 1
 150  CONTINUE
C     * SKIP BELOW IF WITHIN LIMITS
      IF (IERR .EQ. 0) GOTO 160
      CALL TTYOUT ('ONE OR MORE FREQUNCIES EXCEED LIMITS')
      WRITE (LUNTTY, 151) FMIN, FMAX
 151  FORMAT (' FMIN=', F7.2, 3X, 'FMAX=', F7.2)
      CALL TTYOUT ('$ ')
      IF (LASK ('WANT FREQUENCIES LISTED? ') .EQ. 'N') GOTO 153
      WRITE (LUNTTY, 152) (J, FRQTMP (J), J = 1, NCOMP)
 152  FORMAT (1X, I4, 5X, F7.2)
      CALL TTYOUT ('$ ')
 153  CALL TTYOUT ('CHANGE FREQENCIES OR TIME BASE? (F/T) $')
      IF (LANS ('F', 'T') .EQ. 'F') GOTO 110
      CALL TTYOUT ('TIME BASE CHANGE OPTION NOT IMPLEMENTED')
      GOTO 153
C
 160  DO 170 J = 1, NCOMP
 170  HARM (J) = FRQTMP (J)/FZERO + 0.5
C
      IF (NOMSOS .EQ. 'Y') RETURN
      IF (LASK ('WANT FREQUENCIES LISTED? ') .EQ. 'N') RETURN
```

```
C
      SUBROUTINE SOSPAR (NOMPAR,ICHNGE,IRUN,NPER,NCOMP,
     1                   HARM,AMP,PMUL,ISEED)
C
C     SOSPAR SETS UP THE SOS PARAMETERS FOR SOSGEN
C     SOS PARAMETERS ARE EITHER USER SPECIFIED, OR
C                          SET TO NOMINAL VALUES
C
C     INPUTS: (VIA ARGLST)    NOMPAR, ICHNGE, IRUN, NPER
C     OUTPUTS:(VIA ARGLST)    NCOMP, HARM, AMP, PMUL, ISEED
C
      COMMON /TIMCOM/ PZERO, FZERO, TSAMP
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK, NOMPAR, NOMSOS, ICHNGE
      INTEGER HARM (1), PMUL (1)
      DIMENSION AMP (1)
C
      IF (ICHNGE .EQ. 'N') GOTO 300
      NOMSOS = 'Y'
      IF (NOMPAR .EQ. 'Y') GOTO 110
      CALL TTYOUT ('************SOS PARAMETERS************')
 100  NOMSOS = LASK ('NOMINAL SOS? ')
C
 110  CALL SOSNCP (NOMSOS, NCOMP)
      CALL SOSHMC (NOMSOS, NCOMP, NPER, HARM)
      CALL SOSAMP (NOMSOS, NCOMP, AMP)
 200  CALL SOSPHS (NOMSOS, ICHNGE, IRUN, NCOMP, PMUL, ISEED)
      IF (NOMPAR .EQ. 'Y') RETURN
      IF (LASK ('LIST SOS PARAMETERS? ') .EQ. 'N') RETURN
      WRITE (LUNTTY, 1000)
 1000 FORMAT (1X,'COMP',5X,'HARM',7X,'FREQ',7X,'AMP',8X,'PHASE',/)
      WRITE (LUNTTY, 1001) (J, HARM(J), FZERO * HARM(J), AMP(J),
     1     PZERO * PMUL(J), J = 1, NCOMP)
 1001 FORMAT (I5, 5X, I4, 5X, F6.2, 5X, F6.2, 5X, F8.2)
      CALL TTYOUT (' ')
      IF (LASK ('OK? ') .EQ. 'N') GOTO 100
      RETURN
C
 300  CALL SOSPHS(NOMSOS, ICHNGE, IRUN, NCOMP, PMUL, ISEED)
      RETURN
      END
```

```
C
      SUBROUTINE SOSPHS (NOMSOS, ICHNGE, IRUN, NCOMP, PMUL, ISEED)
C
C     INPUTS: (VIA ARGLST)    NOMSOS, ICHNGE
C             (VIA ARGLST)    IRUN, NCOMP
C             (VIA TIMCOM)    PZERO
C
C     OUTPUTS:(VIA ARGLST)    PMUL, ISEED
C
      COMMON /TMPCOM/PHSTMP
      COMMON /TIMCOM/ PZERO
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK, NOMSOS, ICHNGE
      INTEGER PMUL (1)
      DIMENSION PHSTMP(15)
C
      DATA PMIN, PMAX /0., 360./
      DATA IMAX,TMAX /32767,32767./
C
      IF (ICHNGE .EQ. 'N') GOTO 130
 100  IF (NOMSOS .EQ. 'Y') GOTO 130
      IF (LASK ('NOMINAL PHASES? ') .EQ. 'Y') GOTO 130
C
 110  IF (LASK ('RANDOM PHASES? ') .EQ. 'Y') GOTO 120
C
      CALL TTYOUT ('ENTER (DESIRED) PHASES (DEG): ')
      CALL VECTIN (1, 'PHASE', NCOMP, PHSTMP, PMIN, PMAX)
      GOTO 160
C
 120  CALL TTYOUT ('RANDOM PHASE SEED (POS INT) = $')
      ISEED = IANS (0, IMAX)
      CALL TTYOUT (' ')
      GOTO 140
C
C      * NORMAL SEED = RUN #  +  1
 130  ISEED = IRUN + 1
C     ------------------------------------------------------------

C      * SET GENERATOR
 140  ISEED1 = ISEED
      CALL RNSEED (0, ISEED1)
C
C     * WARM UP GENERATOR
      DO 145 I = 1, 100
 145  CALL RNUM (ITEMP,1)
C
      DO 150 J = 1, NCOMP
      CALL RNUM (ITEMP, 1)
      TEMP = ITEMP
      TEMP = (TEMP + TMAX)/(2.*TMAX)
 150  PHSTMP (J) = PMAX * TEMP
C     ------------------------------------------------------------
C     THE PREVIOUS CODE CAN BE WRITTEN IN ONE LINE:
C     PHSTMP (J) = PMAX * RNDM1 (ISEED)
C     ------------------------------------------------------------
```

```
 160   DO 170 J = 1, NCOMP
 170   PMUL (J) = (PHSTMP (J) / PZERO) + 0.5
C
       IF (ICHNGE .EQ. 'N') RETURN
       IF (NOMSOS .EQ. 'Y') RETURN
       IF (LASK ('LIST PHASES? ') .EQ. 'N') RETURN
C
       WRITE (LUNTTY, 200)
 200   FORMAT (1X, 'COMP', 6X, 'PMUL', 8X, 'PHS', 8X,
     1    'PHS(DES)', /)
       WRITE (LUNTTY, 201) (J, PMUL(J), PZERO*PMUL(J),
     1    PHSTMP(J),J=1,NCOMP)
 201   FORMAT (I4, 5X, I6, 6X, F7.2, 6X, F7.2)
       CALL TTYOUT (' ')
       IF (LASK ('OK? ') .EQ. 'N') GOTO 100
       RETURN
       END
```

```
C
      SUBROUTINE STATUS (ITEMP, INIT, FILNAM, NCHAN, CHNAME)
C
      COMMON /STSCOM/ NUM, SUM, SUMSQR, IOVRLD
      COMMON /LUNCOM/ LUNTTY
C
      DIMENSION NUM(4), SUM(4), SUMSQR(4), IOVRLD(4), ITEMP(4), CH(4)
      CHARACTER*10 FILNAM
      CHARACTER*4 CHNAME(4)
C
      DATA IMIN, IMAX /0,4095/
C
C     INITIALIZE VARIABLES
C
      IF (INIT .EQ. 1) GOTO 30
      IF (INIT .EQ. 2) GOTO 40
      DO 20 I=1,NCHAN
         NUM(I) = 0
         SUM(I) = 0.0
         SUMSQR(I) = 0.0
         IOVRLD(I) = 0
 20   CONTINUE
C
 30   DO 16 I=1,NCHAN
 16   CALL CONVRT (1,ITEMP(I),CH(I))
C
      DO 10 I=1,NCHAN
         NUM(I) = NUM(I) + 1
         SUM(I) = SUM(I) + CH(I)
         SQUARE = CH(I) * CH(I)
         SUMSQR(I) = SUMSQR(I) + SQUARE
         IF ((ITEMP(I) .GE. IMAX) .OR. (ITEMP(I) .LE. IMIN))
     1        IOVRLD(I)=IOVRLD(I)+1
 10   CONTINUE
      INIT = 1
      RETURN
C
C     CALCULATE AND WRITE OUT STATISTICS
C
 40   WRITE (LUNTTY,90)
 90   FORMAT ()
      WRITE (LUNTTY,110) FILNAM
 110  FORMAT (1X,'STATISTICS FOR FILE ',A10)
      WRITE (LUNTTY,120)
 120  FORMAT ()
      WRITE (LUNTTY,50)
 50   FORMAT (5X,'CHANNEL #',3X,'NAME',6X,'MEAN',6X,
     1    'STND. DEV.',3X,'ROOT MEAN SQR.',3X,'OVERLOAD')
      DO 60 I=1,NCHAN
         WRITE (LUNTTY,80)
 80      FORMAT ()
         AMEAN = SUM(I)/FLOAT(NUM(I))
         SQMEAN = SUMSQR(I)/FLOAT(NUM(I))
         RMS = SQRT(SQMEAN)
         STNDEV = SQRT(ABS(SQMEAN - (AMEAN * AMEAN)))
         OVER = 100.0*FLOAT(IOVRLD(I))/FLOAT(NUM(I))
```

```
         WRITE (LUNTTY,70) I,CHNAME(I),AMEAN,STNDEV,RMS,OVER
70       FORMAT (8X,I1,8X,A4,2X,F10.2,4X,F10.2,
     1            4X,F10.2,4X,F8.1)
60     CONTINUE
       WRITE (LUNTTY,100)
100    FORMAT ()
       RETURN
       END
```

```
C
      SUBROUTINE TIMPAR (NOMPAR, ISAMP, NPER, NRUN)
C
C     TIMPAR SETS UP THE TIME BASE PARAMETERS FOR VERRUN
C     TIME PARAMETERS ARE EITHER USER SPECIFIED, OR
C                              SET TO NOMINAL VALUES
C     INPUTS: (VIA ARGLST)    NOMPAR
C             (VIA LENGTH)    NRMAX
C
C     OUTPUTS:(VIA ARGLST)    ISAMP, NPER, NRUN
C             (VIA TIMCOM)    PZERO, FZERO, TSAMP, TRUN
C
      COMMON /LENGTH/NRMAX
      COMMON /TIMCOM/PZERO, FZERO, TSAMP, TRUN
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK, NOMPAR
C
      DATA ISMIN, ISNOM, ISMAX /1, 5, 100/
      DATA IMAX /32767/
      DATA TRNOM, TRMAX /5.2, 100.0/
C
      IF (NOMPAR .EQ. 'Y') GOTO 110
      CALL TTYOUT ('************TIME BASE PARAMETERS************')
 100  IF (LASK ('NOMINAL TIME BASE? ') .EQ. 'Y') GOTO 110
C
      CALL TTYOUT ('SAMPLE PERIOD (MSEC) = $')
      ISAMP = IANS (ISMIN, ISMAX)
      TSAMP = ISAMP/1000.0
      CALL TTYOUT ('RUN TIME (SEC) = $')
      TRUN = RANS (TSAMP, TRMAX)
      CALL TTYOUT (' ')
      GOTO 120
 110  ISAMP = ISNOM
      TSAMP = ISAMP/1000.0
      TRUN = TRNOM
 120  TEMP = TRUN/TSAMP + 1.5
      IF (TEMP .LE. IMAX) GOTO 125
      WRITE (LUNTTY, 200) IMAX
 200  FORMAT (' SAMPLE COUNT EXCEEDS INTEGER LIMIT OF ',I6,
     1     '; TRY AGAIN')
      GOTO 126
 125  NRUN = TEMP
      IF (NRUN .LE. NRMAX) GOTO 130
      WRITE (LUNTTY, 201) NRUN, NRMAX
 201  FORMAT(' SAMPLE COUNT',I6,' EXCEEDS FRAME LIMIT OF',I6,
     1     '; TRY AGAIN')
 126  TNEED = (NRMAX - 1) * TSAMP
      WRITE (LUNTTY, 202) ISAMP, TNEED
 202  FORMAT (' WITH SAMPLE PERIOD =', I4,
     1         ' (MSEC), NEED RUN TIME .LE. ',F6.3,' (SEC)',/)
      GOTO 100
C
 130  NPER = 1
      DO 140 J = 1, 20
      NPER = 2 * NPER
```

```
 140   IF (NPER .GT. NRUN) GOTO 150
 150   NPER = NPER/2
C
       TPER = NPER * TSAMP
       TRUN = (NRUN - 1) * TSAMP
C
       PZERO = 360.0/NPER
       FZERO = 1.0/TPER
C
       IF (NOMPAR .EQ. 'Y') RETURN
       IF (LASK ('LIST TIME BASE PARAMETERS? ') .EQ. 'N') RETURN
       WRITE (LUNTTY, 205) ISAMP
 205   FORMAT (' SAMPLE PERIOD =', I4, '   (MSEC)')
       WRITE (LUNTTY, 206) TRUN, NRUN
 206   FORMAT (' RUN LENGTH =', F10.2, ' (SEC) WITH', I5, ' SAMPLES')
       WRITE (LUNTTY, 207) TPER, NPER
 207   FORMAT (' SOS PERIOD =', F10.2, ' (SEC) WITH', I5, ' SAMPLES')
       WRITE(LUNTTY, 208) FZERO,PZERO
 208   FORMAT (' BASE FREQ = ', F10.2, ' (HZ), BASE PHASE =',
     1           F10.2,' (DEG)',/)
       IF (LASK ('OK? ') .EQ. 'N') GOTO 100
       RETURN
       END
```

```
C
      SUBROUTINE TITLER (IRW, LUNIT, MODE, IRUN)
C
C     TITLER READS/WRITES THE TITLE FROM/TO A FILE OR TTY
C     THE TITLE INCLUDES FILE NAME, DATE, TIME, AND COMMENTS
C
C             INPUTS  (VIA ARGLST)    IRW (1 = READ, 2 = WRITE)
C                     (VIA ARGLST)    LUNIT,MODE
C             OUTPUTS:(VIA ARGLST)    IRUN
C                     (VIA TTLCOM)    IDATE, ITIME, NLINE, TITLE
C
      COMMON /TTLCM1/ FNAME, IDATE, ITIME, TITLE
      COMMON /TTLCM2/ NLINE
      COMMON /FILCOM/ NUMFIL, IFILE
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LASK,MODE
      CHARACTER*8 ITIME
      CHARACTER*9 IDATE
      CHARACTER*10 FNAME(10)
      CHARACTER*255 TITLE
      INTEGER HOURS, SECONS
C
      DATA NDIM /255/
C
      GOTO (100,200) IRW
C
C             READ-IN SECTION
 100  IF (LUNIT .NE. LUNTTY)  GOTO 130
C
C             READ IN FROM TTY
 110  CALL DATE (IDATE)
      CALL TIME (ITIME)
      WRITE (LUNIT,3005)
 3005 FORMAT ()
      WRITE (LUNIT, 3000) IRUN, IDATE, ITIME
 3000 FORMAT (1X,'RUN NUMBER: ',I4,4X,'DATE: ',A9,4X,'TIME: ',A8,/)
      IF (IFILE .GT. 0) RETURN
      IF (MODE .EQ. 'P') GOTO 120
      CALL TTYOUT(' ')
      IF (LASK('CHANGING THE RUN NUMBER? ') .EQ. 'N') GOTO 120
      CALL TTYOUT ('NEW RUN NUMBER: $')
      IRUN = IANS (0, 100)
      GOTO 110
 120  CALL TTYOUT ('NUMBER OF COMMENT LINES: $')
      NLINE = IANS (0, 6)
      IF (NLINE .EQ. 0) RETURN
      CALL TTYOUT ('ENTER COMMENTS:')
      CALL TTYIN (NLINE, NDIM, TITLE)
      RETURN
C
C             READ IN FROM FILE
 130  READ (LUNIT, 1000) FNAME(IFILE), IRUN, IDATE, ITIME
 1000 FORMAT (7X,A10,15X,I4,11X,A9,10X,A8)
      READ (LUNIT, 1010) NLINE
 1010 FORMAT (19X,I4)
```

```
      IF (NLINE .EQ. 0) RETURN
      CALL FILIN (NLINE, NDIM, TITLE, LUNIT)
      RETURN
C
C             WRITE-OUT SECTION
 200  IF (LUNIT .NE. LUNTTY) GOTO 210
C
C             WRITE OUT ONTO TTY
      WRITE (LUNIT, 2000) FNAME(IFILE), IRUN, IDATE, ITIME
 2000 FORMAT (1X,'FILE: ',A10,6X,' RUN NO: ',I4,4X,' DATE: ',
     1     A9,3X,' TIME: ',A8)
C     * SUPPRESS TITLE WRITEOUT
      IF (MODE .EQ. 'S') RETURN
      IF (NLINE .NE. 0) CALL TTYOUT (TITLE)
      RETURN
C
C             WRITE OUT ONTO FILE
 210  WRITE (LUNIT, 2000) FNAME(IFILE), IRUN, IDATE, ITIME
      WRITE (LUNIT, 2010) NLINE
 2010 FORMAT (1X,'TITLE LINE COUNT: ',I4)
      IF (NLINE .NE. 0) CALL FILOUT (TITLE, LUNIT)
      RETURN
      END
```

APPENDIX B: LISTING FOR PROGRAM VERNAL

```
      PROGRAM VERNAL
C
      COMMON /TIMCOM/ PZERO, FZERO, TSAMP, TRUN
      COMMON /TTLCM1/ FNAME, IDATE, ITIME, TITLE
      COMMON /TTLCM2/ NLINE
      COMMON /SPCCOM/ AMPCOR, PHSCOR, CDIVR, PWRCOR, PWRREM,
     1                TOTCOR, TOTREM, NREM
      COMMON /FILCOM/ IFILE, NUMFIL
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 LANS, LASK, LSOS, MODE, INFILE
      CHARACTER*8 ITIME
      CHARACTER*9 IDATE
      CHARACTER*10 FNAME
      CHARACTER*255 TITLE
      CHARACTER*4 CHNAME(4)
      CHARACTER*4 CRFLAG(15)
      CHARACTER*4 CHR(15,4)
      CHARACTER*10 FNAME2
      INTEGER HARM (15), PMUL (15), HOURS, SECONS
      DIMENSION AMP(15)
      DIMENSION AVG(4),SIG(4),RMS(4),OVER(4)
      DIMENSION AMPCOR(15,4),PHSCOR(15,4),CDIVR(15,4),PWRCOR(15,4),
     1          PWRREM(15,4),TOTCOR(4),TOTREM(4),NREM(15)
      DIMENSION JDFCN(2),GAIN(15),PHASE(15)
      DIMENSION TMP1(15,4),TMP2(15,4),TMP3(15,4)
      DIMENSION IQUAN(50)
C
      DIMENSION XDATA(5000)
      DIMENSION IDATA(16400)
C
      DATA LUNTTY /0/
      DATA LUNFIL /3/
      DATA INFILE /'N'/
      DATA RTD /57.296/
C     * TEMP CODE
      DATA NSTART/1/
C     * TEMP CODE
      DATA NCHAN/4/
      IFILE=1
C
      LUNIN=21
      IRUN=1
      IFLAG=0
C
 10   CALL PART (INFILE, IPART)
      IF (IPART .LT. 0) STOP
      IF (INFILE .EQ. 'N') GOTO 100
C
  20  GOTO (100, 200, 300, 400, 500, 600, 700) IPART
C
C             PART1:  READ HEADER FROM DATA FILE
C
 100  CONTINUE
C     * READ HEADER FROM FILE
      IRW = 1
```

```
C       * AND LEAVE OPEN
        ICLOSE = 2
        CALL RWHEAD (IRW,LUNFIL,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1               CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
C       * INDICATE WE HAVE AN INPUT FILE
        INFILE = 'Y'
C       * IDATA NOT LOADED
        LIDATA = 0
C       * XDATA NOT COMPUTED
        LSIGNL = 0
C       * STATS NOT COMPUTED
        LSTATS = 0
C       * SPECTRA NOT COMPUTED
        LSPECT = 0
        IF (IPART .EQ. 1) GOTO 10
        GOTO 20
C
C               PART2:  LIST HEADER
C
 200    CONTINUE
C       * WRITE HEADER TO TTY
        IRW = 2
        CALL RWHEAD (IRW,LUNTTY,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1               CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
        GOTO 10
C
C               PART3:  COMPUTE SIGNAL STATISTICS
C
 300    CONTINUE
        IF (LIDATA .EQ. 0) GOTO 800
C
C**********************START TEST CODE*******************************
C
        CALL TTYOUT(' ')
        IF(LASK('**TEST CODE: WANT IDATA LISTED?').EQ.'N')GOTO 301
C       * WRITE DATA ONTO TTY
        IRW=2
        CALL RWDATA (IRW,LUNTTY,NRUN,NCHAN,IDATA)
C
 301    CALL TTYOUT(' ')
        IF(LASK('**TEST CODE: WANT XDATA LISTED? ').EQ.'N')GOTO 302
        CALL TTYOUT('ENTER JCHAN $')
        JCHAN=IANS(1,NCHAN)
        CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
        WRITE(LUNTTY,1010)(I,XDATA(I),I=1,NRUN)
1010    FORMAT(I5,1PE12.4)
 302    CONTINUE
C
C***********************END TEST CODE*********************************
C
 303    IF (LSTATS .EQ. 1) GOTO 320
C
        CALL TTYOUT('DOING STATS NOW...')
        DO 310 JCHAN = 1, NCHAN
        CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
        CALL STATS (JCHAN,NPER,XDATA,AVG,SIG,RMS,OVER)
```

```
 310   CONTINUE
C      * INDICATE STATS COMPUTED
       LSTATS = 1
C
C      * WRITE TITLE ONTO TTY
 320   IRW = 2
       IF (IFLAG .EQ. 1) GOTO 740
C      * BUT SUPPRESS COMMENTS
       MODE = 'S'
       CALL TITLER (IRW, LUNTTY, MODE, IRUN)
       WRITE (LUNTTY,3000)
 3000  FORMAT(/,5X,'CHAN',4X,'CHAN NAME',5X,'AVG',10X,'S.D.',10X,'RMS',
     1    8X,'OVERLOAD')
       WRITE (LUNTTY,3005)
 3005  FORMAT (25X,'(VOLTS)',6X,'(VOLTS)',7X,'(VOLTS)',8X,'(%)')
       WRITE (LUNTTY,3010) (J,CHNAME(J),AVG(J),SIG(J),RMS(J),
     1    OVER(J),J=1,NCHAN)
 3010  FORMAT(2X,I5,8X,A4,2X,F10.2,3X,F10.2,4X,F10.2,4X,F8.1)
       WRITE (LUNTTY,3020)
 3020  FORMAT(//)
       GOTO 10
C
C              PART4:  COMPUTE SIGNAL SPECTRA
C
 400   CONTINUE
       IF (LIDATA .EQ. 0) GOTO 800
C
 401   CALL TTYOUT ('SPECTRUM FOR CHANNEL #: $')
       JCHAN = IANS (1,NCHAN)
C
       CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
       CALL SPECT  (JCHAN,NCOMP,HARM,NPER,LSPECT,XDATA)
C
C**********START TEST CODE**************************
C
       IF(LASK('**TEST CODE: WANT SPECTRUM LISTOUT? ').EQ.'N') GOTO 405
       LSOS = LASK('**TEST CODE: ALL FREQS? ')
C
       NHALF = NPER/2
       DO 404 K=0,NHALF
       IF (LSOS .EQ. 'Y') GOTO 403
       DO 402 L = 1,NCOMP
 402   IF (K .EQ. HARM(L)) GOTO 403
       GOTO 404
C
 403   INDEX = 2*K + 1
C
       WRITE(LUNTTY,4000) K,XDATA(INDEX),RTD*XDATA(INDEX+1)
 4000  FORMAT(I5,2F10.3)
 404   CONTINUE
 405   CONTINUE
C
C**********END TEST CODE********************************
C
C      WRITE SPECTRUM TO PLOT FILE
C
```

```
      LUNIT=13
      OPEN (UNIT=LUNIT,FILE='VERTMP.PLT',STATUS='NEW')
      WRITE (13,4004) NPER/2,FZERO
C
      DO 408 K=1,NPER/2
         INDEX=2*K+1
 408     WRITE (13,4004) K,XDATA(INDEX)
 4004    FORMAT (I5,F12.5)
C
      CLOSE (UNIT=13)
C
C
      CALL TTYOUT(' ')
C     * WRITE TITLE ONTO TTY
      IRW = 2
C     * BUT SUPPRESS COMMENTS
      MODE = 'S'
      CALL TITLER (IRW,LUNTTY,MODE,IRUN)
C
      WRITE (LUNTTY,4010) JCHAN
 4010 FORMAT (/,32X, 'SPECTRUM FOR CHANNEL # ', I2, /)
      WRITE (LUNTTY,4011)
 4011 FORMAT (27X, 'PWR/BIN', 22X, 'PWR/HZ')
      WRITE (LUNTTY,4012)
 4012 FORMAT (/,1X,'COMP   FREQ *    COR      REM      C/R',
     1                        '  *    COR      REM      C/R',
     1                        '  * NREM')
      WRITE (LUNTTY,4013)
 4013 FORMAT(13X,'*',27X,'*',27X,'*')
 4014 BINLOG = 10.0 * ALOG10(FZERO)
      A0=0
      DO 420 J = 1,NCOMP
      AFREQ = FLOAT (HARM(J))
      IF ( J .EQ. NCOMP ) GOTO 410
      A1 = SQRT (AFREQ * FLOAT(HARM (J+1)))
      GOTO 415
 410  A1 = (AFREQ**2)/A0
 415  WIDTH = 10.0 * ALOG10(A1-A0)
      A0 = A1
      AFREQ = AFREQ * FZERO
      TEMP1 = PWRCOR(J,JCHAN) - WIDTH - BINLOG
      TEMP2 = PWRREM(J,JCHAN) - BINLOG
      TEMP3 = TEMP1 - TEMP2
      IF (IFLAG .NE. 1) GOTO 419
         TMP1(J,JCHAN) = TEMP1
         TMP2(J,JCHAN) = TEMP2
         TMP3(J,JCHAN) = TEMP3
         GOTO 420
 419  WRITE (LUNTTY,4020)  J,FZERO*HARM(J),PWRCOR(J,JCHAN),
     1                 PWRREM(J,JCHAN),CDIVR(J,JCHAN),
     1                 TEMP1,TEMP2,TEMP3,NREM(J)
 4020 FORMAT (I4,F8.2,' *',F8.2,2F9.2,' *',F8.2,2F9.2,' *',I4)
 420  CONTINUE
      IF (IFLAG .EQ. 1) GOTO 772
C
      TOTPWR = TOTCOR(JCHAN) + TOTREM(JCHAN)
```

```
      WRITE (LUNTTY,4031) TOTCOR(JCHAN), TOTCOR(JCHAN)/TOTPWR
      WRITE (LUNTTY,4032) TOTREM(JCHAN), TOTREM(JCHAN)/TOTPWR
      WRITE (LUNTTY,4033) TOTPWR
 4031 FORMAT(//,5X, 'COR PWR = ', F9.2, 5X, 'COR/TOT PWR = ', F9.2)
 4032 FORMAT(   5X, 'REM PWR = ', F9.2, 5X, 'REM/TOT PWR = ', F9.2,/)
 4033 FORMAT(   5X, 'TOT PWR = ', F9.2,//)
C
      CALL TTYOUT(' ')
      IF (LASK ('ANOTHER SPECTRUM? ') .EQ. 'Y') GOTO 401
C
      GOTO 10
C
C             PART5:  COMPUTE TRANSFER FUNCTIONS
C
 500  CONTINUE
      IF (LIDATA .EQ. 0) GOTO 800
C
 510  CALL TTYOUT ('CHANNEL # FOR DFCN   NUM: $')
      JDFCN(1) = IANS(1,NCHAN)
      CALL TTYOUT ('CHANNEL # FOR DFCN DENOM: $')
      JDFCN(2) = IANS(1,NCHAN)
C
C     * GET SPECTRA FOR NUM & DENOM
 512  DO 520 I = 1,2
      JCHAN = JDFCN(I)
      CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
      CALL SPECT  (JCHAN,NCOMP,HARM,NPER,LSPECT,XDATA)
 520  CONTINUE
C
      CALL DFCN (JDFCN, NCOMP, GAIN, PHASE, CRFLAG)
C
      IF (IFLAG .NE. 1) GOTO 513
         DO 522 J=1,NCOMP
            TMP1(J,K) = GAIN(J)
            TMP2(J,K) = RTD * PHASE(J)
            CHR(J,K) = CRFLAG(J)
 522     CONTINUE
         GOTO 779
C
C     * WRITE TITLE ONTO TTY
 513  IRW = 2
C     * BUT SUPPRESS COMMENTS
      MODE = 'S'
      CALL TITLER (IRW, LUNTTY, MODE, IRUN)
      WRITE (LUNTTY, 5010) JDFCN
 5010 FORMAT (/,25X,'DFCN FOR (CHAN ',I1,')/(CHAN ',I1,')',//)
      WRITE (LUNTTY,5015)
 5015 FORMAT (20X,'COMP     FREQ      GAIN      PHASE')
      DO 530 J = 1, NCOMP
 530  WRITE (LUNTTY, 5020) J, FZERO*HARM(J), GAIN(J), RTD*PHASE(J),
     1                     CRFLAG(J)
 5020 FORMAT (20X, I4, F9.2, F10.1, F10.1, A4)
      CALL TTYOUT(' ')
C
      IF (LASK ('ANOTHER DFCN? ') .EQ. 'Y') GOTO 510
C
```

```
      GOTO 10
C
C              PART6:  HISTOGRAM
C
 600  CONTINUE
      IF (LIDATA .EQ. 0) GOTO 800
C
      CALL TTYOUT ('ENTER CHANNEL # FOR HISTOGRAM ANALYSIS: $')
      JCHAN = IANS(1,4)
      CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
      IF (LASK('NOMINAL INCREMENTS FOR GRAPH? ') .EQ. 'N') GOTO 610
         VALUE = .50
         GOTO 620
 610  CALL TTYOUT ('ENTER INCREMENT VALUE: $')
         VALUE = RANS(.33,1.00)
 620  NVAR = 10.0/VALUE+.5
      IF (NVAR*VALUE .EQ. 10.0) GOTO 625
         NUMINC = 10.0/VALUE+1.0
         GOTO 628
 625  NUMINC = 10.0/VALUE+0.5
C
C     ZERO OUT DATA
C
 628  DO 630 I=1,50
 630     IQUAN(I)=0
      IUNDER=0
      IOVER=0
C
      XHIGH = XDATA(1)
      XLOW = XDATA(1)
C
      DO 650 I=1,NPER
         IF (XDATA(I) .LT. XLOW) XLOW=XDATA(I)
         IF (XDATA(I) .GT. XHIGH) XHIGH=XDATA(I)
         CALL CONVRT (0, IVALUE, XDATA(I))
         IF (IVALUE .GT. 0) GOTO 631
            IUNDER = IUNDER + 1
            GOTO 650
 631     IF (IVALUE .LT. 4095) GOTO 632
            IOVER = IOVER + 1
            GOTO 650
 632     DO 660 J=1,NUMINC
            VAL = FLOAT(J)*VALUE-5.0
            CALL CONVRT (0, NVAR, VAL)
            IF (IVALUE .GT. NVAR) GOTO 660
               IQUAN(J) = IQUAN(J) + 1
               GOTO 650
 660     CONTINUE
 650  CONTINUE
C
      IQHIGH=IQUAN(1)
      IF (IOVER .GT. IQHIGH) IQHIGH=IOVER
      IF (IUNDER .GT. IQHIGH) IQHIGH=IUNDER
      DO 6070 I=1,50
         IF (IQUAN(I) .GT. IQHIGH) IQHIGH=IQUAN(I)
 6070 CONTINUE
```

```
C
      WRITE (LUNTTY,661)
 661  FORMAT ()
      WRITE (LUNTTY,6040) VALUE
 6040 FORMAT (31X,'BINWIDTH=',F4.2)
      WRITE (LUNTTY,6041) XHIGH
 6041 FORMAT (31X,'HIGHEST VOLTAGE=',F7.2)
      WRITE (LUNTTY,6042) XLOW
 6042 FORMAT (1X,'FREQ.',25X,'LOWEST VOLTAGE=',F7.2)
      WRITE (LUNTTY,6043) IQHIGH
 6043 FORMAT (31X,'HIGHEST FREQUENCY=',I5)
      DO 665 I=1,20
         J=20-I+1
         IF (MOD(J,2) .NE. 0) GOTO 671
            WRITE (LUNTTY,695) J*10
 695        FORMAT (2X,I4,' ! ',$)
            GOTO 664
 671     WRITE (LUNTTY,696)
 696     FORMAT (6X,' ! ',$)
 664     IF (IUNDER .LE. (20-I)*10) GOTO 662
            WRITE (LUNTTY,690)
            GOTO 663
 662     WRITE (LUNTTY,691)
 663     DO 670 J=1,NUMINC
            IF (IQUAN(J) .LE. (20-I)*10) GOTO 666
               WRITE (LUNTTY,690)
               GOTO 670
 666        WRITE (LUNTTY,691)
 670     CONTINUE
         IF (IOVER .LE. (20-I)*10) GOTO 667
            WRITE (LUNTTY,692)
            GOTO 665
 667     WRITE (LUNTTY,693)
 665  CONTINUE
C
 690  FORMAT (' *',$)
 691  FORMAT ('  ',$)
 692  FORMAT (' *')
 693  FORMAT ('  ')
      WRITE (LUNTTY,696)
      WRITE (LUNTTY,699)
 699  FORMAT (1X,'!',$)
      DO 6035 I=2,NUMINC,2
         WRITE (LUNTTY,6036)
 6036    FORMAT (3X,'!',$)
 6035 CONTINUE
      IF (MOD(NUMINC,2) .EQ. 0) GOTO 6037
         WRITE (LUNTTY,6036)
         GOTO 6038
 6037 WRITE (LUNTTY,6036)
 6038 WRITE (LUNTTY,693)
C
      WRITE (LUNTTY,698)
 698  FORMAT (6X,' !',$)
      WRITE (LUNTTY,6030) IUNDER
 6030 FORMAT (I3,$)
```

```
      DO 6031 I=2,NUMINC,2
         WRITE (LUNTTY,6032) IQUAN(I)
 6032    FORMAT (1X,I3,$)
 6031 CONTINUE
      WRITE (LUNTTY,6032) IOVER
      WRITE (LUNTTY,693)
      WRITE (LUNTTY,685)
 685  FORMAT ('-----------------------------------------',
     1        '------------------------VOLTS')
C
      WRITE (LUNTTY,697)
 697  FORMAT (10X,'!',$)
      JFLAG = 1
      DO 6900 J=1,10
      DO 681 I=JFLAG,NUMINC
         EXP1 = -5.00+REAL(I)*VALUE
         EXP2 = -5.00+REAL(J)+VALUE-.1
         EXP3 = -5.00+REAL(J)-VALUE+.1
         IF ((EXP1 .LT. EXP2) .AND. (EXP1 .GT. EXP3)) GOTO 6901
            WRITE (LUNTTY,682)
 682        FORMAT ('  ',$)
            GOTO 681
 6901       WRITE (LUNTTY,6903)
 6903       FORMAT (' !',$)
            JFLAG = I+1
            GOTO 6900
 681  CONTINUE
 6900 CONTINUE
      WRITE (LUNTTY,693)
C
      WRITE (LUNTTY,6904)
 6904 FORMAT (9X,'-5',$)
      JFLAG = 1
      DO 6905 J=1,10
      DO 6906 1=JFLAG,NUMINC
         EXP1 = -5.00+REAL(I)*VALUE
         EXP2 = -5.00+REAL(J)+VALUE-.1
         EXP3 = -5.00+REAL(J)-VALUE+.1
         IF ((EXP1 .LT. EXP2) .AND. (EXP1 .GT. EXP3)) GOTO 6907
            WRITE (LUNTTY,682)
            GOTO 6906
 6907       WRITE (LUNTTY,6908) -5+J
 6908       FORMAT (I2,$)
            JFLAG = I+1
            GOTO 6905
 6906 CONTINUE
 6905 CONTINUE
      WRITE (LUNTTY,693)
      WRITE (LUNTTY,693)
C
      IF (LASK('ANOTHER HISTOGRAM? ') .EQ. 'Y') GOTO 600
C
      GOTO 10
C
C             PART7: WRITE RESULTS TO OUTPUT FILE
C
```

```
 700   CONTINUE
       IF (LIDATA .EQ. 0) GOTO 800
C
       IFLAG=1
       CALL FILNAM (2,FNAME2,NCHAR)
       OPEN (UNIT=LUNIN,FILE=FNAME2,STATUS='NEW')
       WRITE (LUNIN,710)
 710   FORMAT ('ANALYSIS OUTPUT')
       WRITE (LUNIN,720)
 720   FORMAT ()
       CALL TITLER (2,LUNIN,'X',IRUN)
       WRITE (LUNIN,730) NCOMP
 730   FORMAT (/,I3,22X,'****SIGNAL STATISTICS****')
       GOTO 303
 740   WRITE (LUNIN,3000)
       WRITE (LUNIN,3005)
       WRITE (LUNIN,3010) (J,CHNAME(J),AVG(J),SIG(J),RMS(J),
     1    OVER(J),J=1,NCHAN)
       WRITE (LUNIN,3020)
C
       WRITE (LUNIN,750)
 750   FORMAT (25X,'****SIGNAL SPECTRA****')
       WRITE (LUNIN,760) CHNAME(3), CHNAME(4)
 760   FORMAT (/,28X,A4,25X,A4)
       WRITE (LUNIN,770)
 770   FORMAT (/,26X,'(PWR/HZ)',21X,'(PWR/HZ)')
       WRITE (LUNIN,4012)
       WRITE (LUNIN,4013)
       JCHAN=3
 771      CALL SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
          CALL SPECT (JCHAN,NCOMP,HARM,NPER,LSPECT,XDATA)
          GOTO 4014
 772      JCHAN=JCHAN+1
          IF (JCHAN .EQ. 4) GOTO 771
       DO 773 J=1,NCOMP
          WRITE (LUNIN,4020) J,FZERO*HARM(J),TMP1(J,3),TMP2(J,3),
     1       TMP3(J,3),TMP1(J,4),TMP2(J,4),TMP3(J,4),NREM(J)
 773   CONTINUE
C
       WRITE (LUNIN,3020)
       WRITE (LUNIN,774)
 774   FORMAT (25X,'****DESCRIBING FUNCTION****')
       WRITE (LUNIN,775) CHNAME(3), CHNAME(2), CHNAME(4), CHNAME(2)
 775   FORMAT (/,25X,A4,'/',A4,19X,A4,'/',A4)
       WRITE (LUNIN,776)
 776   FORMAT (/,1X,'COMP   FREQ *    GAIN    PHASE   VALID',
     1                        '  *    GAIN    PHASE   VALID  *')
       WRITE (LUNIN,777)
 777   FORMAT (8X,'(HZ) *    (DB)    (DEG)        ',
     1               '  *    (DB)    (DEG)          *')
       WRITE (LUNIN,4013)
       K=3
 778      JDFCN(1) = K
          JDFCN(2) = 2
          GOTO 512
 779      K=K+1
```

```
         IF (K .EQ. 4) GOTO 778
       DO 780 J=1,NCOMP
         WRITE (LUNIN,781) J,FZERO*HARM(J),TMP1(J,3),TMP2(J,3),
     1      CHR(J,3),TMP1(J,4),TMP2(J,4),CHR(J,4)
 781     FORMAT (I4,F8.2,' *',F9.1,F9.1,3X,A4,1X,' *',F9.1,
     1      F9.1,3X,A4,1X,' *')
 780   CONTINUE
       WRITE (LUNIN,3020)
       CLOSE (UNIT=LUNIN)
       IFLAG = 0
       IF (LASK('ANOTHER ANALYSIS OUTPUT FILE? ') .EQ. 'Y') GOTO 100
C
       GOTO 10
C
C              PART8:  READ DATA FROM FILE; SET START POINT
C
 800   CONTINUE
       CALL TTYOUT ('READING IN DATA NOW....')
C      * READ DATA FROM FILE & CLOSE IT
       IRW = 1
       CALL RWDATA (IRW,LUNFIL,NRUN,NCHAN,IDATA)
C      * INDICATE IDATA IS LOADED
       LIDATA = 1
C
C      * SET UP START POINT
       NSTART=1
       CALL TTYOUT ('ANALYSIS STARTS AT POINT $')
       WRITE (LUNTTY,105) NSTART
 105   FORMAT (1X,I5$)
       IF (LASK('  WANT TO CHANGE? ') .EQ. 'N') GOTO 20
       NTEMP = NRUN - NPER + 1
       CALL TTYOUT ('ENTER START POINT IN RANGE 1 THRU $')
       WRITE (LUNTTY,105) NTEMP
       CALL TTYOUT (': $')
       NSTART = IANS(1,NTEMP)
       GOTO 20
C
       END
```

```
C
      SUBROUTINE DFCN (JDFCN, NCOMP, GAIN, PHASE, CRFLAG)
C
C     DFCN COMPUTES DESCRIBING FUNCTION FOR TWO CHANNELS
C     CHANNEL NUMBERS FOR (NUM,DENOM) ARE (JDFCN(1),JDFCN(2))
C     GAIN/PHASE IS DIFFERENCE IN DB/RAD OF CORRELATED SIGNAL
C     AMPS/PHASES
C     PHASE CHANGE WITH FREQUENCY IS LIMITED , AND A FLAG IS
C     SET WHEN THE C/R RATIO IS LOW, FOR EITHER CHANNEL
C
C             INPUTS: (VIA ARGLST)    JDFCN, NCOMP
C                     (VIA SPCCOM)    AMPCOR,PHSCOR,CDIVR
C             OUTPUTS:(VIA ARGLST)    GAIN,PHASE,CRFLAG
C
      COMMON /SPCCOM/ AMPCOR,PHSCOR,CDIVR
C
      CHARACTER*4 CRFLAG(15)
      CHARACTER*4 BLANK
      CHARACTER*4 STARS
      DIMENSION JDFCN(1), GAIN(1), PHASE(1),
     1          AMPCOR(15,4), PHSCOR(15,4), CDIVR(15,4)
C
      DATA BLANK, STARS /'    ', '****'/
      DATA SIXDB /6./
      DATA PI, TWOPI /3.14159, 6.28318/
C
      PHSOLD =0.
C
      DO 40 J= 1,NCOMP
      JNUM = JDFCN(1)
      JDENOM =JDFCN(2)
C     * GET GAIN
      GAIN(J) = AMPCOR(J,JNUM) - AMPCOR(J,JDENOM)
C
C     * GET PHASE
      PHSTMP =  PHSCOR(J,JNUM) - PHSCOR(J,JDENOM)
      PHSDIF = PHSTMP - PHSOLD
 10   IF (PHSDIF .LE. PI) GOTO 20
      PHSDIF = PHSDIF -TWOPI
      GOTO 10
 20   IF (PHSDIF .GE. -PI) GOTO 30
      PHSDIF = PHSDIF + TWOPI
      GOTO 20
 30   PHSTMP = PHSOLD + PHSDIF
      PHASE(J) = PHSTMP
C
C     * SET C/R FLAG IF C/R LOW
      CRFLAG(J) = STARS
      IF (CDIVR(J,  JNUM) .LT. SIXDB) GOTO 40
      IF (CDIVR(J,JDENOM) .LT. SIXDB) GOTO 40
      CRFLAG(J) = BLANK
      PHSOLD = PHSTMP
 40   CONTINUE
C
      RETURN
      END
```

```
C
      SUBROUTINE FFT (N,XDATA)
C
C     RETURNS N-POINT FFT OF XDATA, IN XDATA, WHERE N IS A PWR OF 2
C     (AMP,PHS) FOR JTH HARMONIC STORED IN (XDATA(2J+1),XDATA(2J+2)),
C                                    FOR J = 1 THRU N/2-1
C     (AMP,PHS) FOR   0TH HARMONIC STORED IN (XDATA(1),  XDATA(2))
C                  N/2TH                     (XDATA(N+1),XDATA(N+2))
C
C             INPUTS: (VIA ARGLST)    N,XDATA
C             OUTPUTS:(VIA ARGLST)    XDATA
C
      DIMENSION XDATA(1)
C
      DATA PI /3.14159/
C
      CALL FAST (N,XDATA)
C
      NHALF = N/2
      TWODN = 1./NHALF
C
C     * DO ZEROTH HARMONIC (DC)
      TEMP = XDATA(1)/N
       XDATA(1) = ABS(TEMP)
       XDATA(2) = 0.
      IF (TEMP .LT. 0.) XDATA(2) = PI
C
C     * DO HARMONICS FROM 1 TO (NHALF-1)
      DO 100 I = 1, (NHALF-1)
      JODD  = 2*I + 1
      JEVEN = JODD + 1
      TEMP1 = XDATA( JODD)
      TEMP2 = XDATA(JEVEN)
       XDATA( JODD) = TWODN*SQRT (TEMP1*TEMP1 + TEMP2*TEMP2)
       XDATA(JEVEN) = ATAN2 (TEMP1,-TEMP2)
 100  CONTINUE
C
C     * DO N/2 HARMONIC (NYQUIST)
      TEMP = XDATA(N+1)
       XDATA(N+1) = TWODN*ABS (TEMP)
       XDATA(N+2) = 0.
C
      RETURN
      END
```

```
C
C SUBROUTINE:  FAST
C REPLACES THE REAL VECTOR B(K), FOR K=1,2,...,N,
C WITH ITS FINITE DISCRETE FOURIER TRANSFORM
C-----------------------------------------------------------------------
C
      SUBROUTINE FAST(N,B)
C
C THE DC TERM IS RETURNED IN LOCATION B(1) WITH B(2) SET TO 0.
C THEREAFTER THE JTH HARMONIC IS RETURNED AS A COMPLEX
C NUMBER STORED AS  B(2*J+1) + I B(2*J+2).
C THE N/2 HARMONIC IS RETURNED IN B(N+1) WITH B(N+2) SET TO 0.
C HENCE, B MUST BE DIMENSIONED TO SIZE N+2.
C THE SUBROUTINE IS CALLED AS  FAST(N,B) WHERE N=2**M AND
C B IS THE REAL ARRAY DESCRIBED ABOVE.
C
      DIMENSION B(2)
      COMMON /CONS/ PII, P7, P7TWO, C22, S22, PI2
C
C IW IS A MACHINE DEPENDENT WRITE DEVICE NUMBER
C
      IW = 0
C
      PII = 4.*ATAN(1.)
      PI8 = PII/8.
      P7 = 1./SQRT(2.)
      P7TWO = 2.*P7
      C22 = COS(PI8)
      S22 = SIN(PI8)
      PI2 = 2.*PII
      DO 10 I=1,15
        M = I
        NT = 2**I
        IF (N.EQ.NT) GO TO 20
  10  CONTINUE
      WRITE (IW,9999)
9999  FORMAT (33H N IS NOT A POWER OF TWO FOR FAST)
      STOP
  20  N4POW = M/2
C
C DO A RADIX 2 ITERATION FIRST IF ONE IS REQUIRED.
C
      IF (M-N4POW*2) 40, 40, 30
  30  NN = 2
      INT = N/NN
C
      CALL FR2TR(INT, B(1), B(INT+1))
C
      GO TO 50
  40  NN = 1
C
C PERFORM RADIX 4 ITERATIONS.
C
  50  IF (N4POW.EQ.0) GO TO 70
      DO 60 IT=1,N4POW
        NN = NN*4
```

```
        INT = N/NN
C
        CALL FR4TR(INT, NN, B(1), B(INT+1), B(2*INT+1), B(3*INT+1),
     *      B(1), B(INT+1), B(2*INT+1), B(3*INT+1))
C
  60  CONTINUE
C
C PERFORM IN-PLACE REORDERING.
C
  70  CALL FORD1(M, B)
      CALL FORD2(M, B)
      T = B(2)
      B(2) = 0.
      B(N+1) = T
      B(N+2) = 0.
      DO 80 IT=4,N,2
        B(IT) = -B(IT)
  80  CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C SUBROUTINE:  FR2TR
C RADIX 2 ITERATION SUBROUTINE
C-----------------------------------------------------------------------
C
      SUBROUTINE FR2TR(INT, B0, B1)
      DIMENSION B0(2), B1(2)
      DO 10 K=1,INT
        T = B0(K) + B1(K)
        B1(K) = B0(K) - B1(K)
        B0(K) = T
  10  CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE:  FR4TR
C RADIX 4 ITERATION SUBROUTINE
C-----------------------------------------------------------------------
C
      SUBROUTINE FR4TR(INT, NN, B0, B1, B2, B3, B4, B5, B6, B7)
      DIMENSION L(15), B0(2), B1(2), B2(2), B3(2), B4(2), B5(2), B6(2),
     *    B7(2)
      COMMON /CONS/ PII, P7, P7TWO, C22, S22, PI2
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))
C
C JTHET IS A REVERSED BINARY COUNTER, JR STEPS TWO AT A TIME TO
C LOCATE THE REAL PARTS OF INTERMEDIATE RESULTS, AND JI LOCATES
C THE IMAGINARY PART CORRESPONDING TO JR.
C
      L(1) = NN/4
      DO 40 K=2,15
        IF (L(K-1)-2) 10, 20, 30
```

```
  10    L(K-1) = 2
  20    L(K) = 2
        GO TO 40
  30    L(K) = L(K-1)/2
  40  CONTINUE
C
      PIOVN = PII/FLOAT(NN)
      JI = 3
      JL = 2
      JR = 2
C
      DO 120 J1=2,L1,2
      DO 120 J2=J1,L2,L1
      DO 120 J3=J2,L3,L2
      DO 120 J4=J3,L4,L3
      DO 120 J5=J4,L5,L4
      DO 120 J6=J5,L6,L5
      DO 120 J7=J6,L7,L6
      DO 120 J8=J7,L8,L7
      DO 120 J9=J8,L9,L8
      DO 120 J10=J9,L10,L9
      DO 120 J11=J10,L11,L10
      DO 120 J12=J11,L12,L11
      DO 120 J13=J12,L13,L12
      DO 120 J14=J13,L14,L13
      DO 120 JTHET=J14,L15,L14
        TH2 = JTHET - 2
        IF (TH2) 50, 50, 90
  50    DO 60 K=1,INT
          T0 = B0(K) + B2(K)
          T1 = B1(K) + B3(K)
          B2(K) = B0(K) - B2(K)
          B3(K) = B1(K) - B3(K)
          B0(K) = T0 + T1
          B1(K) = T0 - T1
  60    CONTINUE
C
        IF (NN-4) 120, 120, 70
  70    K0 = INT*4 + 1
        KL = K0 + INT - 1
        DO 80 K=K0,KL
          PR = P7*(B1(K)-B3(K))
          PI = P7*(B1(K)+B3(K))
          B3(K) = B2(K) + PI
          B1(K) = PI - B2(K)
          B2(K) = B0(K) - PR
          B0(K) = B0(K) + PR
  80    CONTINUE
        GO TO 120
C
  90    ARG = TH2*PIOVN
        C1 = COS(ARG)
        S1 = SIN(ARG)
        C2 = C1**2 - S1**2
        S2 = C1*S1 + C1*S1
        C3 = C1*C2 - S1*S2
```

```
        S3 = C2*S1 + S2*C1
C
        INT4 = INT*4
        J0 = JR*INT4 + 1
        K0 = JI*INT4 + 1
        JLAST = J0 + INT - 1
        DO 100 J=J0,JLAST
          K = K0 + J - J0
          R1 = B1(J)*C1 - B5(K)*S1
          R5 = B1(J)*S1 + B5(K)*C1
          T2 = B2(J)*C2 - B6(K)*S2
          T6 = B2(J)*S2 + B6(K)*C2
          T3 = B3(J)*C3 - B7(K)*S3
          T7 = B3(J)*S3 + B7(K)*C3
          T0 = B0(J) + T2
          T4 = B4(K) + T6
          T2 = B0(J) - T2
          T6 = B4(K) - T6
          T1 = R1 + T3
          T5 = R5 + T7
          T3 = R1 - T3
          T7 = R5 - T7
          B0(J) = T0 + T1
          B7(K) = T4 + T5
          B6(K) = T0 - T1
          B1(J) = T5 - T4
          B2(J) = T2 - T7
          B5(K) = T6 + T3
          B4(K) = T2 + T7
          B3(J) = T3 - T6
 100    CONTINUE
C
        JR = JR + 2
        JI = JI - 2
        IF (JI-JL) 110, 110, 120
 110    JI = 2*JR - 1
        JL = JR
 120  CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE:  FORD1
C IN-PLACE REORDERING SUBROUTINE
C-----------------------------------------------------------------------
C
      SUBROUTINE FORD1(M, B)
      DIMENSION B(2)
C
      K = 4
      KL = 2
      N = 2**M
      DO 40 J=4,N,2
        IF (K-J) 20, 20, 10
  10    T = B(J)
        B(J) = B(K)
```

```
        B(K) = T
  20    K = K - 2
        IF (K-KL) 30, 30, 40
  30    K = 2*J
        KL = J
  40  CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE:  FORD2
C IN-PLACE REORDERING SUBROUTINE
C-----------------------------------------------------------------------
C
      SUBROUTINE FORD2(M, B)
      DIMENSION L(15), B(2)
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))
      N = 2**M
      L(1) = N
      DO 10 K=2,M
        L(K) = L(K-1)/2
  10  CONTINUE
      DO 20 K=M,14
        L(K+1) = 2
  20  CONTINUE
      IJ = 2
      DO 40 J1=2,L1,2
      DO 40 J2=J1,L2,L1
      DO 40 J3=J2,L3,L2
      DO 40 J4=J3,L4,L3
      DO 40 J5=J4,L5,L4
      DO 40 J6=J5,L6,L5
      DO 40 J7=J6,L7,L6
      DO 40 J8=J7,L8,L7
      DO 40 J9=J8,L9,L8
      DO 40 J10=J9,L10,L9
      DO 40 J11=J10,L11,L10
      DO 40 J12=J11,L12,L11
      DO 40 J13=J12,L13,L12
      DO 40 J14=J13,L14,L13
      DO 40 JI=J14,L15,L14
        IF (IJ-JI) 30, 40, 40
  30    T = B(IJ-1)
        B(IJ-1) = B(JI-1)
        B(JI-1) = T
        T = B(IJ)
        B(IJ) = B(JI)
        B(JI) = T
  40    IJ = IJ + 2
      RETURN
      END
```

```
C
      SUBROUTINE PART(INFILE,IPART)
C
C     USER SETS IPART FOR USE BY VERNAL
C
C             INPUT:  (VIA ARGLST)    INFILE
C             OUTPUT: (VIA ARGLST)    IPART
C
      CHARACTER*1 INFILE
C
 10   CALL TTYOUT ('TO PART (0-7): $')
      IPART = IANS (-1,7)
      IF (IPART .EQ.-1) RETURN
      IF (IPART .NE. 0) GOTO 20
      CALL TTYOUT (' ')
      CALL TTYOUT ('PART 1: FILE READ')
      CALL TTYOUT ('PART 2: HEADER LIST')
      CALL TTYOUT ('PART 3: SIGNAL STATISTICS')
      CALL TTYOUT ('PART 4: SIGNAL SPECTRA')
      CALL TTYOUT ('PART 5: TRANSFER FUNCTIONS')
      CALL TTYOUT ('PART 6: HISTOGRAM')
      CALL TTYOUT ('PART 7: ANALYSIS OUTPUT FILE')
      GOTO 10
C
 20   IF ((IPART .EQ. 1) .OR. (INFILE .EQ. 'Y')) RETURN
      CALL TTYOUT ('NEED AN INPUT FILE')
      RETURN
      END
```

```
C
      SUBROUTINE REMPWR (NCOMP,HARM,XDATA,KLOW,KHIGH,NREM,SUMREM)
C
C     REMPWR COMPUTES THE SUMMED REMNANT POWER OVER THE
C     HARMONIC WINDOW DEFINED BY (KLOW,KHIGH)
C     SUMREM EXCLUDES POWER AT THE SOS HARMONICS DEFINED
C     BY HARM, AND RETURNS THE NUMBER OF REMNANT FREQS SUMMED
C
C             INPUTS: (VIA ARGLST)    NCOMP,HARM,XDATA
C                     (VIA ARGLST)    KLOW,KHIGH
C             OUTPUTS:(VIA ARGLST)    NREM, SUMREM
C
      INTEGER HARM(1)
C
      DIMENSION XDATA(1)
C
C     * ZERO COUNTER & SUMMER
      NREM = 0
      SUMREM = 0.
C
C     * SUM FROM KLOW TO KHIGH
      DO 20 K = KLOW,KHIGH
C
C     * EXCLUDE SOS HARMONICS
      DO 10 L = 1,NCOMP
 10   IF (K .EQ. HARM(L)) GOTO 20
C
C     * GET REMNANT AMP
      INDEX = 2*K + 1
      AMPREM = XDATA (INDEX)
C     * ACCUMULATE 2*PWR
      SUMREM = SUMREM + AMPREM*AMPREM
C     * INCREMENT COUNTER
      NREM = NREM + 1
C
 20   CONTINUE
C
C     * CALC SUMMED REM PWR
      SUMREM = SUMREM/2.
      RETURN
      END
```

```
C
      SUBROUTINE RWDATA (IRW,LUNIT,NFRAME,NCHAN,IDATA)
C
C     RWDATA READS/WRITES THE DATA ARRAY IDATA FROM/TO FILE
C
C             INPUTS: (VIA ARGLST)    IRW (1 = READ, 2 = WRITE)
C                                     LUNIT, NFRAME, NCHAN, IDATA
C             OUTPUT: (VIA ARGLST)    IDATA
C
      COMMON /LUNCOM/ LUNTTY
C
      DIMENSION IDATA (1)
C
      DATA NCMAX/4/
C
      IF(NCHAN .LE. NCMAX) GOTO 10
      CALL TTYOUT('******RWDATA: NCHAN .GT. NCMAX******')
      STOP
C
 10   GOTO (100,200) IRW
C
C     READ DATA FROM FILE & LOAD IDATA
 100  IF (LUNIT .NE. LUNTTY) GOTO 105
      CALL TTYOUT ('******RWDATA: TRYING TO READ FROM TTY******')
      STOP
C
 105  READ (LUNIT, 999)
 999  FORMAT(/,/)
C
      INDEX = 0
      DO 120 I = 1, NFRAME
      READ (LUNIT,1000) IDUMMY, (IDATA(INDEX+J), J=1,NCHAN)
 1000 FORMAT (1X,5I5)
 120  INDEX = INDEX + NCHAN
      GOTO 300
C
C     WRITE ALL CHANNELS OF DATA FROM IDATA TO FILE (OR TTY)
 200  WRITE (LUNIT, 2000) NCHAN
 2000 FORMAT (/, 1X, '***RECORDED DATA OF ', I3, ' CHANNELS***')
      WRITE (LUNIT, 2001)
 2001 FORMAT (2X, 'IFRM','  C1 ','  C2 ','  C3 ','  C4 ')
      INDEX = 0
      DO 220 I=1,NFRAME
      WRITE (LUNIT,1000) I, (IDATA(INDEX+J),J=1,NCHAN)
 220  INDEX = INDEX + NCHAN
      IF (LUNIT .EQ. LUNTTY) RETURN
C
 300  CLOSE (UNIT=LUNIT,STATUS='KEEP')
      RETURN
      END
```

```
C
      SUBROUTINE RWHEAD(IRW,LUNIT,ICLOSE,NCHAN,IRUN,ISAMP,NPER,
     1                  CHNAME,NRUN,NCOMP,HARM,AMP,PMUL,ISEED)
C
C     READS/WRITES HEADER FROM/TO A DATA FILE
C     ALSO WRITES HEADER TO TTY
C
C     INPUTS: (VIA ARGLST)    IRW    (1=READ HEADER,2=WRITE HEADER)
C             (     "    )    LUNIT
C             (VIA ARGLST)    ICLOSE (1=CLOSE FILE, 2=LEAVE FILE OPEN)
C     I/O:    (VIA ARGLST)    IRUN, ISAMP
C             (     "    )    NPER, NRUN, NCOMP
C             (     "    )    HARM, AMP, PMUL, ISEED
C             (VIA TIMCOM)    PZERO, FZERO, TSAMP, TRUN
C
      COMMON /TIMCOM/ PZERO, FZERO, TSAMP, TRUN
      COMMON /TTLCM1/ FNAME, IDATE, ITIME, TITLE
      COMMON /TTLCM2/ NLINE
      COMMON /LUNCOM/ LUNTTY
C
      CHARACTER*1 IOPEN,MDUMY
      CHARACTER*8 ITIME
      CHARACTER*9 IDATE
      CHARACTER*10 FNAME
      CHARACTER*255 TITLE
      CHARACTER*4 CHNAME(4)
      INTEGER HARM(1), PMUL(1), HOURS, SECONS
      DIMENSION AMP(1)
C
      DATA NVERS /3/
      DATA IOPEN /'N'/
      DATA MDUMY /'P'/
C
      IF (LUNIT .EQ. LUNTTY) GOTO 201
C
C     * GET FILE NAME AND CLOSE IT IF OPEN
   5  CALL FILNAM (IRW, FNAME, NCHAR)
      IF (IOPEN .EQ. 'N') GOTO 10
      CLOSE (UNIT=LUNIT,STATUS='KEEP')
C     * AND INDICATE IT'S CLOSED
      IOPEN = 'N'
 10   GOTO (100, 200) IRW
C
C             READ FROM FILE
C
 100  CONTINUE
      OPEN (UNIT=LUNIT,FILE=FNAME,STATUS='OLD')
      IOPEN = 'Y'
      READ (LUNIT, 105) NVERS
 105  FORMAT (17X, I1,/,/)
      CALL TITLER (IRW, LUNIT, MDUMY, IRUN)
      READ (LUNIT, 107) (CHNAME(I),I=1,NCHAN)
 107  FORMAT (/,/,2X,4(2X,A4))
      READ (LUNIT, 110) ISAMP
 110  FORMAT (/, /, 25X, I4)
      READ (LUNIT, 115) FZERO, PZERO
```

```
 115   FORMAT (17X, 1PE12.3, 21X, 1PE12.3)
       READ (LUNIT, 120) TEMP, NPER
 120   FORMAT (17X, 1PE12.3, 28X, I5)
       READ (LUNIT, 125) TRUN, NRUN
 125   FORMAT (17X, 1PE12.3, 28X, I5)
       READ (LUNIT, 130) NCOMP, ISEED
 130   FORMAT (/,/, 22X, I4, 27X, I5, /)
       READ (LUNIT, 135)  (HARM(I), AMP(I), PMUL(I), I=1,NCOMP)
 135   FORMAT (10X, I5, 16X, F6.3, 5X, I6)
       GOTO 300
C
C              WRITE TO FILE (OR TTY)
C
 200   CONTINUE
       OPEN (UNIT=LUNIT,FILE=FNAME,STATUS='NEW')
       IOPEN = 'Y'
 201   WRITE (LUNIT, 205) NVERS
 205   FORMAT (1X, 'VERSION NUMBER: ', I1)
       WRITE (LUNIT, 206)
 206   FORMAT (/, 1X, '***RUN IDENTIFICATION***')
       CALL TITLER (IRW, LUNIT, MDUMY, IRUN)
       WRITE (LUNIT, 207)
 207   FORMAT (/, 1X, '***CHANNEL NAMES***')
       WRITE (LUNIT, 208) (CHNAME(I),I=1,NCHAN)
 208   FORMAT (2X,4(2X,A4))
       WRITE (LUNIT, 209)
 209   FORMAT (/, 1X, '***TIME BASE PARAMETERS***')
       WRITE (LUNIT, 210) ISAMP
 210   FORMAT (1X, 'SAMPLE PERIOD:  ', 8X, I4,' MSEC')
       WRITE (LUNIT, 215) FZERO, PZERO
 215   FORMAT (1X, 'BASE FREQUENCY: ',1PE12.3, '   HZ',
     1         4X, 'BASE PHASE: ',    1PE12.3, ' DEG')
       WRITE (LUNIT, 220) NPER*(ISAMP/1000.), NPER
 220   FORMAT (1X, 'SOS PERIOD:     ',1PE12.3, '  SEC',
     1         4X ,'WITH: ', 13X, I5, ' PTS')
       WRITE (LUNIT, 225) TRUN, NRUN
 225   FORMAT (1X, 'RUN LENGTH:     ',1PE12.3, '  SEC',
     1         4X, 'WITH: ', 13X, I5, ' PTS')
       WRITE (LUNIT, 229)
 229   FORMAT (/, 1X, '***SOS SIGNAL PARAMETERS***')
       WRITE (LUNIT, 230) NCOMP, ISEED
 230   FORMAT (1X, '# OF SOS COMPONENTS: ', I4,
     1         8X, 'RANDOM PHASE SEED: ', I5)
       WRITE (LUNIT, 234)
 234   FORMAT (2X, 'COMP', 5X, 'HARM', 7X, 'FREQ', 7X, 'AMP',
     1         8X, 'PMUL', 7X, 'PHS')
       WRITE (LUNIT, 235)  (J,HARM(J),FZERO*HARM(J),AMP(J),
     1                      PMUL(J),PZERO*PMUL(J), J=1,NCOMP)
 235   FORMAT (I5,5X,I5,5X,F6.2,5X,F6.3,5X,I6,5X,F6.1)
C
C      * RETURN IF JUST DONE TTY WRITE
       IF (LUNIT .EQ. LUNTTY) RETURN
C
 300   IF (ICLOSE .NE. 1)  RETURN
C      * CLOSE FILE
       CLOSE (UNIT=LUNIT,STATUS='KEEP')
```

```
C       * AND INDICATE CLOSED
        IOPEN = 'N'
        RETURN
        END
```

```
C
      SUBROUTINE SIGNAL (JCHAN,NSTART,NPER,NCHAN,LSIGNL,XDATA,IDATA)
C
C     SIGNAL LOADS XDATA WITH DATA CHANNEL JCHAN, TAKEN
C     FROM IDATA, STARTING AT POINT NSTART IN THE IDATA ARRAY
C
C             INPUTS: (VIA ARGLST)    JCHAN,NSTART,NPER,NCHAN,IDATA
C                     (VIA ARGLST)    LSIGNL (0=INIT PASS, 1=OTHERS)
C             OUTPUTS:(VIA ARGLST)    LSIGNL,XDATA
C
      COMMON /SIGCOM/ SCALE
C
      DIMENSION XDATA(1)
      DIMENSION IDATA(1)
C
      IF (LSIGNL .EQ. 1) GOTO 20
C
C     * INIT SCALES  FIRST TIME THRU
      DO 10 J = 1,NCHAN
 10   CONTINUE
C     * & INDICATE DONE
      LSIGNL = 1
C
 20   CONTINUE
      I = (NSTART-1)*NCHAN + JCHAN
C
      DO 30 K = 1,NPER
      CALL CONVRT (1,IDATA(I),XDATA(K))
 30   I = I + NCHAN
C
      RETURN
      END
```

```
C
      SUBROUTINE SPECT (JCHAN,NCOMP,HARM,NPER,LSPECT,XDATA)
C
C     FOR THE SIGNAL IN XDATA, SPECT CALCULATES, AT EACH SOS FREQ:
C             1)THE CORRELATED AMP AND PHS
C             2)THE CORRELATED & REMNANT POWER (PER MSMT BIN)
C             3)THE COR-TO-REM POWER RATIO     (PER MSMT BIN)
C     SPECT ALSO CALCULATES THE TOTAL CORRELATED AND REMNANT POWER
C
C             INPUTS: (VIA ARGLST)    JCHAN
C                     (    "     )    NCOMP,HARM,NPER
C                     (    "     )    LSPECT (0=INIT PASS, 1=OTHERS)
C                     (    "     )    XDATA
C             OUTPUTS:(VIA ARGLST)    LSPECT,XDATA
C                     (VIA SCRCOM)    AMPCOR,PHSCOR,CDIVR
C                     (VIA SCRCOM)    PWRCOR,PWRREM,TOTCOR,TOTREM,NREM
C
      COMMON /SPCCOM/ AMPCOR,PHSCOR,CDIVR,PWRCOR,PWRREM,
     1                TOTCOR,TOTREM,NREM,NHALF,DBTWO,RATIO
C
      INTEGER HARM(1)
      DIMENSION XDATA(1)
      DIMENSION AMPCOR(15,4),PHSCOR(15,4),CDIVR(15,4),
     1          PWRCOR(15,4),PWRREM(15,4),TOTCOR(4),TOTREM(4),NREM(15)
C
      DATA HALF /0.50/
C     * 1/4 OCTAVE REM WINDOW
      DATA WINDOW /0.25/
C     * ZERO & INF IN DB UNITS
      DATA DBZERO, DBINF /-99.99,+99.99/
C
      IF (LSPECT .EQ. 1) GOTO 10
C     * DO FIRST PASS CALCS
      NHALF = NPER/2
      DBTWO = 10.*ALOG10(2.)
      RATIO = 2.**(HALF*WINDOW)
C     * & INDICATE DONE
      LSPECT = 1
C
 10   CALL TTYOUT ('DOING FFT...')
C
      CALL FFT (NPER,XDATA)
C
C     * ZERO THE COR PWR SUM
      SUMCOR = 0.
C
      DO 30 J =1,NCOMP
C     * GET JTH HARMONIC
      KHARM = HARM(J)
C     * & ITS XDATA INDEX
      INDEX = 2*KHARM + 1
C
C     DO AMP, PHS, PWR CALCULATIONS FOR SOS FREQS (CORRELATED)
C
C     * GET AMP & ITS SQUARE
      AMPTMP = XDATA (INDEX)
```

```
      AMPSQR = AMPTMP*AMPTMP
C     * GET AMP & PWR IN DB
      AMPTMP = 20.*ALOG10(AMPTMP)
      PWRTMP = AMPTMP - DBTWO
C     * GET PHASE IN RAD
      PHSTMP = XDATA (INDEX+1)
C
C     * LOAD COR AMP,PHS,PWR
      AMPCOR (J,JCHAN) = AMPTMP
      PHSCOR (J,JCHAN) = PHSTMP
      PWRCOR (J,JCHAN) = PWRTMP
C
C     * ACCUMULATE 2*PWR
      SUMCOR = SUMCOR + AMPSQR
C
C     DO PWR CALCULATIONS FOR NON-SOS FREQS (REMNANT)
C
C     * GET LOW & HIGH HARMS
      KLOW = KHARM/RATIO + HALF
C     * WHICH DEFINE REM WINDOW
      KHIGH= KHARM*RATIO + HALF
C     * & LIMIT THEM
      IF ( KLOW .LT.     1)  KLOW = 1
      IF (KHIGH .GT. NHALF) KHIGH = NHALF
C
      CALL REMPWR (NCOMP,HARM,XDATA,KLOW,KHIGH,NCOUNT,SUMREM)
C
C     * CALC AVG REM PWR IN WINDOW
      PWRTMP = DBZERO
      IF ( (NCOUNT .GT. 0) .AND. (SUMREM .GT. 0.) )
     1 PWRTMP = 10.*ALOG10(SUMREM/NCOUNT)
      PWRREM (J,JCHAN) = PWRTMP
C     * LOAD # OF REM FREQS IN AVG
      NREM (J) = NCOUNT
C
C     DO CALCULATIONS FOR COR-TO-REM POWER RATIO
C
C     * GET COR & REM PWR
      TMPCOR = PWRCOR (J,JCHAN)
      TMPREM = PWRREM (J,JCHAN)
C     * SET C/R TO ZERO WHEN
      IF (TMPCOR .GT. DBZERO) GOTO 18
C     * COR PWR IS ZERO
      TMPCDR = DBZERO
      GOTO 20
C     * SET C/R TO INF WHEN
 18   IF (TMPREM .GT. DBZERO) GOTO 19
C     * REM PWR IS ZERO
      TMPCDR = DBINF
      GOTO 20
C     * SET C/R TO DIF IN DB
 19   TMPCDR = TMPCOR - TMPREM
C
C     * AND LIMIT
      CALL LIMIT (DBZERO,DBINF,TMPCDR)
C
```

```
      CLOSE(UNIT=LUNIN)
C
C     Get name of next data file
C     --------------------------
 40   CONTINUE
C
      CLOSE(UNIT=18)
C
C     Calculate the Standard Deviation of the Phase and the Gain and
C               Average the valid data for each frequency
C     ------------------------------------------------------------
C
      P1=0.0
      DO 45 I=1,NPOINT
        NVRPS=IARR(I,2)
        IF(NVRPS.EQ.0 .OR. NVRPS.EQ.1)GO TO 42
          SDG=SQRT((DFARR(I,4)-((DFARR(I,2)**2)/NVRPS))/(NVRPS-1))
          SDP=SQRT((DFARR(I,5)-((DFARR(I,3)**2)/NVRPS))/(NVRPS-1))
          GO TO 43
 42     CONTINUE
          IF(NVRPS.EQ.0)NVRPS=1
          SDG=0
          SDP=0
 43     CONTINUE
          DFARR(I,6)=DFARR(I,2)/NVRPS
          DFARR(I,7)=DFARR(I,3)/NVRPS
          DFARR(I,8)=SDG
          DFARR(I,9)=SDP
          DFARR(I,10)=DFARR(I,7)
          IF(IARR(I,3).NE.1)GOTO 45
 46       IF(DFARR(I,10).LE.P1)GOTO 47
            DFARR(I,10)=DFARR(I,10)-360.0
            GOTO 46
 47       P1=DFARR(I,10)
 45   CONTINUE
C
C     Save the results in a file
C     --------------------------
      WRITE(LUNTTY,1011)
      WRITE(LUNTTY,1000)
 1000 FORMAT(2X,'PROCESSING COMPLETE.')
      WRITE(LUNTTY,1011)
      CALL TTYOUT('ENSEMBLE AVERAGE ANALYSIS FILE (NAME.ENS)....$')
      CALL FILNAM(2,FNAME,NDUMMY)
      OPEN(UNIT=22,FILE=fname,STATUS='NEW')
C
C     Put on three lines of header
C     ----------------------------
      WRITE(22,1010)
 1010 FORMAT(10X,'This is an ENS average data file')
      WRITE(22,1011)
      WRITE(22,1012)
      WRITE(22,1013)NPOINT
 1011 FORMAT(10X)
 1012 FORMAT(25X,'AVG      S.D.                  ',
     1        'AVG      S.D.      VALID')
```

```
 1013  FORMAT(I4,2X,'COMP     FREQ       ',
     +'GAIN     GAIN     PHASE      PHASE     PHASE     REPS')
C
       DO 50 I=1,NPOINT
       WRITE(22,1020)IARR(I,1),DFARR(I,1),DFARR(I,6),DFARR(I,8),
     +                 DFARR(I,10),DFARR(I,7),DFARR(I,9),IARR(I,2)
  50   CONTINUE
 1020  FORMAT(6X,I4,F9.2,F10.1,F8.1,F11.1,F11.1,F10.1,I8)
C
       CLOSE(UNIT=22)
C
 3000  CONTINUE
       END
C
C
```

```
      IF(ICNT.EQ.1)GO TO 61
      IF(ICNT.EQ.2)GO TO 62
  61    OPEN(UNIT=18,FILE='FILB.TMP',STATUS='OLD')
        GO TO 63
  62  CONTINUE
        ICNT=0
        OPEN(UNIT=18,FILE='FILA.TMP',STATUS='OLD')
  63  CONTINUE
C
C     Check corrected file names for accuracy
C     ---------------------------------------
      go to 15
C
  70  CONTINUE
      WRITE(LUNTTY,103)
      WRITE(LUNTTY,120)
  120 FORMAT(2X,'CHECKING FOR FILES ON DISK.')
C
C     Check for files on disk
C     -----------------------
      CALL FILCHK(KFILS,ICNT)
C
 3000 CONTINUE
C
      RETURN
      END
C
C
```

```
C
      SUBROUTINE FILCHK(KFILS,ICNT)
      COMMON /LUNCOM/ LUNTTY
      CHARACTER*15 FNAME
      CHARACTER*15 FNAME2
      CHARACTER*1 ANS
C
C     Check for files on disk
C     -----------------------
   5  IF(ICNT.EQ.1)OPEN(UNIT=18,FILE='FILB.TMP',STATUS='OLD')
      IF(ICNT.EQ.0)OPEN(UNIT=18,FILE='FILA.TMP',STATUS='OLD')
      OPEN(UNIT=16,FILE='TMP.FIL',STATUS='NEW')
C
      N=0
      DO 25 I=1,KFILS
        READ(18,101)FNAME
        LENGTH=LEN1(FNAME)
        OPEN(UNIT=21,FILE=FNAME(1:LENGTH),STATUS='OLD',ERR=2000)
        CLOSE(UNIT=21)
        WRITE(16,100)FNAME
      GO TO 25
2000    N=N+1
        WRITE(LUNTTY,201)I,FNAME
201     FORMAT(2X,'File number',I2,' ,named ',A15,', is not',
     +             ' on this disk.')
10      CALL TTYOUT('Is the file typed correctly? $')
        READ(LUNTTY,115)ANS
        IF(ANS(1:1) .EQ. 'Y')GOTO 15
        IF(ANS(1:1) .EQ. 'N')GOTO 20
          WRITE(LUNTTY,103)
          WRITE(LUNTTY,202)
202       FORMAT(2X,'Invalid Response. Try again.')
          GO TO 10
15      CONTINUE
          WRITE(LUNTTY,203)
203       FORMAT(2X,'FILE NOT ON THIS DISK.',
     +              '  PROCESSING CANNOT BE DONE.')
          CLOSE(UNIT=18)
          ICNT=10
          GO TO 450
20      CONTINUE
          WRITE(LUNTTY,103)
          CALL TTYOUT('Type the proper file name. $')
          DO 26 L=1,15
26        FNAME2(L:L)=' '
          CALL FILNAM(1,FNAME2,NDUMMY)
          WRITE(16,100)FNAME2
25    CONTINUE
C
      CLOSE(UNIT=16)
      CLOSE(UNIT=18)
C
      OPEN(UNIT=16,FILE='TMP.FIL',STATUS='OLD')
      IF(ICNT.EQ.1)OPEN(UNIT=18,FILE='FILB.TMP',STATUS='OLD')
      IF(ICNT.EQ.0)OPEN(UNIT=18,FILE='FILA.TMP',STATUS='OLD')
C
```

```
      PROGRAM MODLER
C
      COMMON /PASCOM/ IPASS
C
      CHARACTER*1 LASK
      DIMENSION PVEC(20)
C
      DATA LUNOUT/0/
C
      IPASS = 1
      CALL JCOMP(PVEC,COSTJ)
C
    1 IPASS = 2
      CALL JCOMP(PVEC,COSTJ)
C
      IPASS = 3
      CALL JCOMP(PVEC,COSTJ)
C
      WRITE(LUNOUT,1000)
 1000 FORMAT(///)
      WRITE(LUNOUT,1010) COSTJ
 1010 FORMAT('MODEL MATCHING COST FCN:   ',E15.5,//)
C
      IPASS = 4
      CALL MODFIT(PVEC)
C
      IPASS = 5
      CALL MODFIT(PVEC)
C
      WRITE(0,1000)
      IF(LASK('ANOTHER TRY') .EQ. 'Y') GOTO 1
      END
C
C
      BLOCKDATA MODBLK
      COMMON /MAIN1/ NDIM,NDIM1,W(625)
      COMMON /DATCOM/ RTD,DTR,TWOPI,HALFPI,BIGVAL
C
      DATA RTD,DTR/57.295779513,0.017453292/
      DATA TWOPI,HALFPI,BIGVAL/6.283185307,1.570796,1.E6/
      DATA NDIM,NDIM1 /25,26/
      END
```

```
C
      SUBROUTINE JCOMP(PVEC,COSTJ)
C
      COMMON /PASCOM/ IPASS
      COMMON /DATCOM/ RTD,DTR,TWOPI,HALFPI,BIGVAL
      COMMON /BLOCKP/ NP,NP1,IT,MAXIT,PFI(2500)
      COMMON /NUMPTS/ NPTS
      COMMON /FRQCOM/ FREQ(25)
C
      CHARACTER*12 BLANK,FILNAM
      DIMENSION PVEC(1),GAIND(25),PHASED(25),SIGGND(25),SIGPHD(25)
      SAVE NDTMAX,GAIND,PHASED,SIGGND,SIGPHD
      DATA BLANK,NDTMAX/'            ',25/
C
      GO TO(100,200,300,400,500) IPASS
C
C     PASS1:  OPEN FILE, READ DATA, CLOSE FILE
C
  100 FILNAM = BLANK
      LUNIT = 5
      CALL OPEN4('FILE NAME FOR INPUT DATA',LUNIT,1,1,FILNAM)
C
      READ(LUNIT,110) NPTS
  110 FORMAT(2X,I3)
      IF(NPTS .GT. NDTMAX) STOP'****JCOMP: NPTS TOO BIG****'
C
      DO 120 I=1,NPTS
  120 READ(LUNIT,115) FREQ(I),GAIND(I),SIGGND(I),PHASED(I),SIGPHD(I)
  115 FORMAT(5F10.3)
      CLOSE(LUNIT)
      RETURN
C
C     PASS2: GET TRFCTN INITIALIZED
C
  200 CALL TRFCTN(PVEC,OMEGA,GAINM,PHASEM)
      RETURN
C
C     PASS3: COMP MODEL MATCHING COST, BETWEEN MODEL & DATA
C
  300 COSTJ=0.0
      DO 310 I=1,NPTS
      OMEGA=TWOPI*FREQ(I)
      CALL TRFCTN(PVEC,OMEGA,GAINM,PHASEM)
      TMP1=(GAIND(I)-GAINM)/SIGGND(I)
      PFI(I) = TMP1
      TMP2=(PHASED(I)-PHASEM)/SIGPHD(I)
      PFI(I + NPTS) = TMP2
  310 COSTJ=COSTJ + TMP1**2 + TMP2**2
      RETURN
C
  400 RETURN
  500 RETURN
      END
```

```
C
      SUBROUTINE MODFIT(PVEC)
C
      COMMON /PASCOM/ IPASS
      COMMON /DATCOM/ RTD,DTR,TWOPI,HALFPI,BIGVAL
      COMMON /NUMPTS/ NPTS
      COMMON /FRQCOM/ FREQ(25)
      COMMON /TRFCOM/ NTYPE,DCGAIN,TDEL,NA,A(25),NB,B(25)
C
      DIMENSION PVEC(1),ZREAL(10),ZIMAG(10),ZETANZ(10),WNZ(10),
     1                  PREAL(10),PIMAG(10),ZETANP(10),WNP(10)
C
      GOTO (100,200,300,400,500) IPASS
C
  100 RETURN
  200 RETURN
  300 RETURN
C
C     FIND 1ST & 2ND ORDER ZERO/POLE PARAMETERS FROM A/B COEFFS
C
  400 IF(NA .NE. 0) THEN
            CALL POLRT(A,NA,ZREAL,ZIMAG)
            CALL SCNVRT(NA,ZREAL,ZIMAG,ZETANZ,WNZ)
            A(1)=PVEC(3)
      ELSE
            A(1)=1.
      ENDIF
      IF(NB .NE. 0) THEN
            CALL POLRT(B,NB,PREAL,PIMAG)
            CALL SCNVRT(NB,PREAL,PIMAG,ZETANP,WNP)
            B(1)=PVEC(3+NA)
      ELSE
            B(1)=1.
      ENDIF
      DCGAIN = PVEC(1)*A(1)/B(1)
      TDELAY= PVEC(2)
C
      WRITE(0,4001)
 4001 FORMAT(/,18X,'DCGAIN',9X,'TDELAY')
      WRITE(0,4002) DCGAIN,TDELAY
 4002 FORMAT(10X,2E15.5,/)
      IF(NA .NE. 0) WRITE(0,4003)
 4003 FORMAT(/,18X,'ZEROES')
      DO 410 I=1,NA
      IF(ZIMAG(I) .EQ. 0.) THEN
            WRITE(0,4005) I,ZREAL(I),ZIMAG(I)
      ELSE
            WRITE(0,4005) I,ZREAL(I),ZIMAG(I),ZETANZ(I),WNZ(I)
      ENDIF
  410 CONTINUE
 4005 FORMAT(I10,4E15.5)
      IF(NB .NE. 0) WRITE(0,4004)
 4004 FORMAT(/,18X,'POLES')
      DO 420 I=1,NB
      IF(PIMAG(I) .EQ. 0.) THEN
            WRITE(0,4005) I,PREAL(I),PIMAG(I)
```

```
      ELSE
             WRITE(0,4005) I,PREAL(I),PIMAG(I),ZETANP(I),WNP(I)
      ENDIF
  420 CONTINUE
      RETURN
C
C     PASS5: WRITE OUT THE FITTED MODEL (FREQ,GAIN,PHASE) VALUES
C
  500 CONTINUE
      WRITE(0,5001)
 5001 FORMAT(//,20X,'MODEL FREQUENCY RESPONSE',//)
      WRITE(0,5002)
 5002 FORMAT(9X,'I',7X,'FREQ(HZ)',7X,'GAIN(DB)',5X,'PHASE(DEG)',/)
      DO 510 I=1,NPTS
      OMEGA = TWOPI*FREQ(I)
      CALL TRFCTN(PVEC,OMEGA,GAINM,PHASEM)
      WRITE(0,5005) I,FREQ(I),GAINM,PHASEM
 5005 FORMAT(I10,3F15.2)
  510 CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE TRFCTN(PVEC,OMEGA,GAIN,PHASE)
C
      COMMON /PASCOM/ IPASS
      COMMON /DATCOM/ RTD,DTR,TWOPI,HALFPI,BIGVAL
      COMMON /BLOCKP/ NP,NP1,IT,MAXIT,PFI(2500)
      COMMON /TRFCOM/ NTYPE,DCGAIN,TDEL,NA,A(25),NB,B(25)
C
      CHARACTER*1 LASK
      DIMENSION PVEC(1),C(3)
      COMPLEX UPVAL,DENVAL,FS
      SAVE NAMAX,NBMAX
C
      DATA NAMAX,NBMAX/25,25/
C
      GO TO (100,200,300,400,300) IPASS
C
  100 RETURN
C
  200 NTYPE = ISPEC('SYSTEM TYPE(0,1,2,..) : ',0,5)
      TFGAIN = SPECFY('D.C. GAIN :',-BIGVAL,BIGVAL)
      TD=SPECFY('TIME DELAY(SEC) :',0.,10.)
      NTEMP=0
C
  201 CALL ALTYP('SPECIFY NUMERATOR')
      NA=0
      NDUM=0
      CALL VECSET(0.,1,NAMAX,A)
      A(1)=1.
      IF(LASK('     FIRST-ORDER ZEROES').EQ.'Y')THEN
  202         ZA=SPECFY('          ZERO VALUE(RAD/S): ',-BIGVAL,BIGVAL)
              IF(ZA.EQ.0.) GOTO 203
              NTEMP = NA+1
              IF(NTEMP .GT. NAMAX) THEN
                      CALL ALTYP('EXCEEDED MAX ORDER OF NUM')
                      GOTO 205
              ENDIF
              NA=NTEMP
              NDUM=NDUM+1
              C(1)=ZA
              C(2)=1.
              TFGAIN=TFGAIN/C(1)
              CALL POLMUL(1,NDUM,C,A)
              GO TO 202
      ENDIF
C
  203 IF(LASK('     SECOND-ORDER ZEROES').EQ.'Y') THEN
  204         ZETA=SPECFY('          DAMPING RATIO: ',-1.,1.)
              WN=SPECFY('          NATURAL FREQUENCY(RAD/S): ',0.,BIGVAL)
              IF((ZETA.EQ.0.).AND.(WN.EQ.0.)) GOTO 205
              NTEMP=NA+2
              IF(NTEMP .GT. NAMAX) THEN
                      CALL ALTYP('EXCEEDED MAX ORDER OF NUM')
                      GOTO 205
              ENDIF
              NA=NTEMP
```

```
              NDUM=NDUM+2
              C(1)=WN**2
              C(2)=2.*ZETA*WN
              C(3)=1.
              TFGAIN=TFGAIN/C(1)
              CALL POLMUL(2,NDUM,C,A)
              GO TO 204
      ENDIF
C
  205 CALL ALTYP('SPECIFY DENOMINATOR')
      C(1)=0.
      C(2)=0.
      C(3)=0.
      NB=0
      NDUM=0
      CALL VECSET(0.,1,NBMAX,B)
      B(1)=1.
      IF(LASK('     FIRST-ORDER POLES').EQ.'Y') THEN
  206         PA=SPECFY('          POLE VALUE(RAD/S): ',-BIGVAL,BIGVAL)
              IF(PA.EQ.0.) GOTO 207
              NTEMP=NB+1
              IF(NTEMP .GT. NBMAX) THEN
                      CALL ALTYP('EXCEEDED MAX ORDER OF DEN')
                      GOTO 209
              ENDIF
              NB=NTEMP
              NDUM=NDUM+1
              C(1)=PA
              C(2)=1.
              TFGAIN=TFGAIN*C(1)
              CALL POLMUL(1,NDUM,C,B)
              GO TO 206
      ENDIF
C
  207 IF(LASK('     SECOND-ORDER POLES').EQ.'Y') THEN
  208         ZETA=SPECFY('          DAMPING RATIO: ',-1.,1.)
              WN=SPECFY('          NATURAL FREQUENCY(RAD/S): ',0.,BIGVAL)
              IF((ZETA.EQ.0.).AND.(WN.EQ.0.)) GOTO 209
              NTEMP=NB+2
              IF(NTEMP .GT. NBMAX) THEN
                      CALL ALTYP('EXCEEDED MAX ORDER OF DEN')
                      GOTO 209
              ENDIF
              NB=NTEMP
              NDUM=NDUM+2
              C(1)=WN**2
              C(2)=2.*ZETA*WN
              C(3)=1.
              TFGAIN=TFGAIN*C(1)
              CALL POLMUL(2,NDUM,C,B)
              GO TO 208
      ENDIF
C
C     STACKING 'P' VECTOR
  209 PVEC(1)=TFGAIN
      PVEC(2)=TD
```

```
      DO 210 II=1,NA
  210 PVEC(II+2)=A(II)
      DO 220 II=1,NB
  220 PVEC(II+NA+2)=B(II)
      NP = NA + NB + 2
      NP1 = NP + 1
      RETURN
C
C     UNSTACK 'P' VECTOR
  300 TFGAIN=PVEC(1)
      TD=PVEC(2)
      DO 310 II=1,NA
  310 A(II)=PVEC(II+2)
      DO 320 II=1,NB
  320 B(II)=PVEC(II+NA+2)
C
      CALL POLCMP(NA,A,OMEGA,UPVAL)
      CALL POLCMP(NB,B,OMEGA,DENVAL)
C
      FS=(TFGAIN/(OMEGA**NTYPE)) * (UPVAL/DENVAL)
C
      GAIN = 20.*ALOG10(CABS(FS))
      D1=REAL(FS)
      D2=AIMAG(FS)
      PHASE=RTD*(ATAN2(D2,D1) - OMEGA*TD - HALFPI*NTYPE)
C     write(0,9000) omega/twopi,gain,phase
C 9000 format(3e15.5)
      RETURN
C
  400 RETURN
  500 RETURN
C
      END
```

```
C
      SUBROUTINE POLCMP(N,COEFF,OMEGA,POLVAL)
C     This Routine Assumes Input Polynomials Of The Form:
C      S**n + C(n)S**n-1 + .... + C(2)S + C(1)
C     and generates one complex value (polval), assuming s=jw
C
      DIMENSION COEFF(1)
      COMPLEX POLVAL,SDUM
C
      IF(N .EQ. 0) THEN
              POLVAL = CMPLX(1.,0.)
              RETURN
      ENDIF
C
      POLVAL = CMPLX(COEFF(1),0.)
      SDUM = CMPLX(0.,OMEGA)
      DO 10 I=2,N
      XI=I-1
      POLVAL = POLVAL + COEFF(I)*(SDUM**XI)
 10   CONTINUE
      XN=N
      POLVAL = POLVAL + SDUM**XN
      RETURN
      END
C
      SUBROUTINE SCNVRT(N,SREAL,SIMAG,ZETAN,WN)
C
      DIMENSION SREAL(N),SIMAG(N),ZETAN(N),WN(N)
C
      DO 10 I=1,N
      WN(I) = SQRT((SREAL(I))**2 + (SIMAG(I))**2)
   10 ZETAN(I) = -SREAL(I)/WN(I)
      RETURN
      END
      SUBROUTINE VECSET(C,I1,I2,X)
C     SET VECTOR X = CONSTANT C FOR I=I1 TO I2
      DIMENSION X(1)
1     DO 105 I=I1,I2
      X(I)=C
105   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE POLMUL(NA,NB,A,B)
C
C     COMPUTES THE POLYNOMIAL POL(B)=POL(A)*POL(B)
C     WHERE POL(B) = B(NB+1)*X**NB + B(NB)*X**(NB-1) + ... + B(1)
C
      DIMENSION A(1), B(1)
C
      NC=NA+NB
      NNA=NA+1
      NNB=NB+1
      NNC=NC+1
      DO 190 L=1,NNC
      K=NNC-L+1
      TEMP=0.0
      DO 110 I=1,K
      J=K-I+1
      IF ((I.GT.NNA) .OR. (J.GT.NNB)) GO TO 110
      TEMP=TEMP+A(I)*B(J)
110   CONTINUE
      B(K)=TEMP
190   CONTINUE
      NB=NC
      RETURN
      END
```

```
      SUBROUTINE OPEN4(MESAGE,LUNIT,KTRIG,KFORM,NAME)
C
C     OPEN FILE ON LOGICAL UNIT LUNIT WITH MESSAGE 'MESAGE'
C     KTRIG =  1 FOR READ, 2 FOR WRITE, 3 FOR NAMING ONLY
C           <  0 NO INTERACTION WITH TTY
C     KFORM =  1 FOR FORMATTED, 2 FOR UNFORMATTED
C
      CHARACTER*1   LASK
      CHARACTER*3   CSTAT
      CHARACTER*12  CFORM
      CHARACTER*80  CHRDUM
      CHARACTER*(*) MESAGE, NAME
C
      NCHAR = INDEX(NAME,' ')-1
      IF(NCHAR .EQ. -1) NCHAR = LEN(NAME)
C     CHECK IF WANT TTY INTERACTION
      IF(KTRIG.LT.0) GO TO 110
      IF(NCHAR.GT.0) GO TO 100
      CHRDUM=MESAGE//': '
      CALL ALTYP0(CHRDUM)
      GO TO 107
100   CHRDUM=MESAGE//' '//NAME(1:NCHAR)
      IF(LASK(CHRDUM) .EQ. 'Y') GOTO 110
105   CALL ALTYP0('ENTER FILE NAME: ')
107   READ (0,1025) NAME
1025  FORMAT(A)
      NCHAR = INDEX(NAME,' ')-1
      IF(NCHAR .EQ. -1) NCHAR = LEN(NAME)
110   IF(KTRIG.EQ.3) RETURN
      IF(IABS(KTRIG) .EQ. 1) CSTAT = 'OLD'
      IF(IABS(KTRIG) .EQ. 2) CSTAT = 'NEW'
      IF(KFORM .EQ. 1) CFORM =   'FORMATTED'
      IF(KFORM .EQ. 2) CFORM = 'UNFORMATTED'
      OPEN(UNIT=LUNIT,STATUS=CSTAT,FORM=CFORM,FILE=NAME(1:NCHAR),ERR=115)
      RETURN
115   CHRDUM='CAN''T FIND FILE:'//NAME(1:NCHAR)
      CALL ALTYP(CHRDUM)
      GO TO 105
      END
```

```
      SUBROUTINE ALTYP(LETRS)
      CHARACTER *(*) LETRS
C
      LENGTH = LEN(LETRS)
      IF(LENGTH.EQ.0) RETURN
      CALL IDTB(LETRS,LENGTH)
      IF(LENGTH.GT.72) LENGTH = 72
      WRITE(0,1000) LETRS(1:LENGTH)
1000  FORMAT(1H ,A)
      RETURN
      END
      SUBROUTINE ALTYP0(LETRS)
      CHARACTER *(*) LETRS
C
      LENGTH = LEN(LETRS)
      IF(LENGTH.EQ.0) RETURN
      CALL IDTB(LETRS,LENGTH)
      IF(LENGTH.GT.72) LENGTH = 72
      WRITE(0,1000) LETRS(1:LENGTH)
1000  FORMAT(1H ,A $)
      RETURN
      END
      CHARACTER FUNCTION LASK(LET)
C     ASK A YES OR NO QUESTION (UP TO 72 LETTERS)
C     LASK APPENDS A '?: '
C     RETURNS EITHER Y OR N
      CHARACTER YES,NO,LYESNO
      CHARACTER *(*) LET
      DATA YES,NO/'Y','N'/
C
      LENGTH=LEN(LET)
      CALL IDTB(LET,LENGTH)
      CALL ALTYP0(LET(1:LENGTH))
      CALL ALTYP0('?: ')
      LASK=LYESNO()
      RETURN
      END
      CHARACTER FUNCTION LYESNO()
C
C     ACCEPTS UPPER AND LOWER CASE ANSWERS &  RETURNS EITHER Y OR N
C
      CHARACTER*1 LDUM
C
105   READ(0,1010) LDUM
1010  FORMAT(A1)
      IF((LDUM.EQ.'Y').OR.(LDUM.EQ.'y')) GO TO 110
      IF((LDUM.EQ.'N').OR.(LDUM.EQ.'n')) GO TO 115
      CALL ALTYP0('ANSWER Y or N: ')
      GO TO 105
110   LYESNO='Y'
      RETURN
115   LYESNO='N'
      RETURN
      END
```

```
      IF (E.GE.EMIN) GO TO 44
C      THIS FORCES EMIN TO HOLD STEADY FOR 5 ITERATIONS
      EMIN = E
      TOL = EMIN*0.7
      GO TO 45
C     THIS WILL ALLOW AN ERROR X*EMIN ONLY AFTER N ITERATIONS
C       WHERE X = (1.1)**N
   44 IF (E.LT.TOL) GO TO 70
   45 CBAR = V(2) - U(2)
      IF (NR.GT.3) V4 = V(4)
      D = V(3)**2 - CBAR*V4
      IF (D) 47,46,47
   46 P = P - 2.0
      Q = Q*(Q+1.0)
      GO TO 50
   47 P = P + (U(2)*V(3) - U(1)*V4)/D
      Q = Q + (-U(2)*CBAR + U(1)*V(3))/D
   50 U(NR) = A(NR) + R
      V(NR) = U(NR) + R
      I = NR - 1
   55 U(I) = A(I) + R*U(I+1)
      V(I) = U(I) + R*V(I+1)
      I = I-1
      IF (I.GT.0) GO TO 55
      E = ABS(U(1)/A(1))
      IF (E.LE.1.E-12) GO TO 60
      IF (E.GE.EMIN) GO TO 56
      EMIN = E
      TOL = EMIN*0.7
      GO TO 57
   56 IF (E.LT.TOL) GO TO 60
   57 IF (V(2).NE.0) GO TO 58
      R = R+1.
      GO TO 59
   58 R = R- U(1)/V(2)
   59 TOL = TOL*1.1
      GO TO 30
C       STORE A SINGLE REAL ROOT
   60 CALL VSCALE(A,U(2),NR-1,1.0)
      GO TO 62
   61 R = -A(1)
   62 U(NR) = R
      V(NR) = 0.0
      NR = NR-1
      GO TO 80
C      STORE A PAIR OF ROOTS
   70 CALL VSCALE(A,U(3),NR-2,1.0)
      GO TO 72
   71 P = A(2)
      Q = A(1)
   72 P = (-0.5)*P
      D = P*P - Q
      IF (D) 75,78,78
   75 U(NR) = P
      U(NR-1) = P
      V(NR) = -SQRT(-D)
```

```
      V(NR-1) = -V(NR)
      GO TO 79
   78 V(NR) = 0.0
      V(NR-1) = 0.0
      D = ABS(P) + SQRT(D)
      IF (P.LT.0.0) D = -D
      U(NR) = D
      U(NR-1) = Q/D
   79 NR = NR-2
   80 IF (NR.GT.0) GO TO 10
      RETURN
      END
      SUBROUTINE VSCALE(X,Y,N,C1)
      DIMENSION X(1),Y(1)
      L=0
      IF(C1.EQ.1.0) GO TO 5
      IF(C1.EQ.0.0) GO TO 8
      IF(C1.EQ.-1.) GO TO 13
1     L=L+1
      X(L)=C1*Y(L)
      IF(L.LT.N) GO TO 1
      RETURN
5     L=L+1
      X(L)=Y(L)
      IF(L.LT.N) GO TO 5
      RETURN
8     L=L+1
      X(L)=0.0
      IF(L.LT.N) GO TO 8
      RETURN
13    L=L+1
      X(L)=-Y(L)
      IF(L.LT.N) GO TO 13
      RETURN
      END
```

APPENDIX E: LISTING FOR LIBRARY IOLIB

```
      FUNCTION LANS(ANS1,ANS2)
C
C     CHECKS TO SEE THAT THE INPUT 'LANS' EQUALS EITHER
C     'ANS1' OR 'ANS2'
C
      CHARACTER*1 LANS,ANS1,ANS2
C
      LUNIT=0
    5 READ(LUNIT,100)LANS
  100 FORMAT(A1)
      IF((LANS.EQ.ANS1).OR.(LANS.EQ.ANS2))RETURN
      WRITE (LUNIT,200) ANS1,ANS2
  200 FORMAT(1X,'PLEASE ANSWER ',A1,' OR ',A1,': '$)
      GO TO 5
      END
C
C
      FUNCTION IANS(MIN,MAX)
C
C     CHECKS TO SEE THAT THE INPUTTED VALUE 'IANS' IS WITHIN
C     THE RANGE SPECIFIED BY 'MIN' AND 'MAX'
C
      LUNIT=0
    5 READ(LUNIT,100,ERR=300) IANS
  100 FORMAT(I6)
      IF((IANS.GE.MIN).AND.(IANS.LE.MAX))RETURN
      WRITE(LUNIT,200)MIN,MAX
  200 FORMAT(1X,'MIN=',I6,' AND MAX=',I6,'   TRY AGAIN: '$)
      GO TO 5
  300 CALL TTYOUT ('ERROR - INPUT NOT INTEGER. TRY AGAIN: $')
      GOTO 5
      END
C
C
      FUNCTION RANS(RMIN,RMAX)
C        CRA / IOLIB
C
C     FUNCTION TAKES A REAL NUMBER AS INPUT IN THE RANGE
C     SPECIFIED BY 'RMIN' AND 'RMAX'
C
      LUNIT=0
  100 READ (0,*,ERR=400) RANS
      IF ((RANS .LT. RMIN) .OR. (RANS .GT. RMAX)) GOTO 200
      RETURN
  200 WRITE (LUNIT, 10) RMIN, RMAX
   10 FORMAT ('MIN= ',1PE15.5,' AND MAX= ',1PE15.5,'  TRY AGAIN: ')
      GOTO 100
  400 CALL TTYOUT('ERROR - INPUT MUST BE A REAL NUMBER. TRY AGAIN: $')
      GOTO 100
      END

      SUBROUTINE FILIN (IDUM1,IDUM2,MSG,LUNIT)
C
C     READS LINES OF INPUT FROM THE TERMINAL OR FROM A FILE AND
C     COLLECTS LINES INTO THE STRING 'MSG', IN WHICH EACH LINE OF
C     INPUT IS SEPARATED BY THE SYMBOL '.  THE FINAL LENGTH
```

```
C       OF 'MSG' MUST BE LESS THAN OR EQUAL TO 255.
C
C       IDUM1 AND IDUM2 ARE DUMMY VARIABLES AND ARE NOT ACTUALLY
C       USED IN THIS SUBROUTINE.
C
        CHARACTER*80 TEMP
        CHARACTER*(*) MSG
C
        LDUM=LEN(MSG)
        DO 5 L=1,LDUM
           MSG(L:L)=' '
 5      CONTINUE
C
        INOW=0
C
 30     READ (LUNIT, 10) TEMP
 10     FORMAT (A)
C
        LENGTH=LEN1(TEMP)
        IF (LENGTH .EQ. 0) RETURN
C
        DO 20 I=1,LENGTH
           MSG (INOW+I:INOW+I) = TEMP (I:I)
 20     CONTINUE
        INOW = LENGTH+INOW+1
        MSG (INOW:INOW) = '
        GOTO 30
        RETURN
        END
        SUBROUTINE FILNAM(IOCHAN,NAME,NCHAR)
C
C       CHECKS TO SEE THAT THE INPUTTED FILNAM IS VALID.
C       A FILENAME CAN ONLY CONTAIN CHARACTERS WHICH ARE EITHER
C       SINGLE DIGIT INTEGERS OR LETTERS OF THE ALPHABET.
C       THE EXTENSION ON A FILENAME MUST BE 3 CHARACTERS IN LENGTH.
C       THE PART OF THE FILENAME BEFORE THE PERIOD CAN BE FROM 1 TO 6
C       CHARACTERS IN LENGTH.  THE FIRST CHARACTER MUST BE A LETTER.
C
C       Input is IOCHAN.  1 for Input filename, 2 for Output filename.
C       NAME is the array containing the name of the file.
C       NCHAR is the number of characters in the filename.
C
C       ** Next two lines used for multiple run mode. **
        COMMON /FILCOM/ NUMFIL, IFILE
        COMMON /LUNCOM/ LUNTTY
C
        CHARACTER*10 NAME
C
C       ** Next two lines used for multiple run mode. **
        IFILE = IFILE + 1
        IF (IFILE .GT. 1) GOTO 12
C
        CALL TTYOUT('ENTER FILENAME FOR $')
C       * Check for legitimate IOCHAN
        GOTO (5,10) IOCHAN
        STOP'****FILNAM:ILLEGAL IOCHAN VALUE****'
```

```
C
C               Print appropriate prompt and read filename.
C
   5    CALL TTYOUT('INPUT: $')
        GOTO 15
  10    CALL TTYOUT('OUTPUT: $')
C
C       ** Next two lines used for multiple run mode. **
        GOTO 15
  12    CALL TTYOUT ('NEXT FILENAME: $')
C
  15    READ (LUNTTY, 20) NAME
  20    FORMAT(A)
C
C       ** Next line used for multiple run mode. **
        IF (IFILE .EQ. NUMFIL) IFILE=0
C
C               Is the first character a letter? (not <a or >b)
C
        I = 1
        IF ((NAME(I:I) .LT. 'A') .OR. (NAME(I:I) .GT. 'Z') .AND.
     1      (NAME(I:I) .LT. 'a') .OR. (NAME(I:I) .GT. 'z')) GOTO 100
C
C       Now check the rest of the name to see if it is all alphanumeric
C       characters, and set NCHAR = to 3 places after the '.'
C
        DO 200 I = 2,10
        IF (NAME(I:I) .EQ. '.') GOTO 50
        IFLAG = -1
        IF ((NAME(I:I) .LT. 'A') .OR. (NAME(I:I) .GT. 'Z') .AND.
     1              (NAME(I:I) .LT. 'a') .OR. (NAME(I:I) .GT. 'z'))
IFLAG=IFLAG+1
        IF (NAME(I:I) .LT. '0' .OR. NAME(I:I) .GT. '9') IFLAG=IFLAG+1
        IF (IFLAG) 200, 200, 100
  50    IF ((I .EQ. 1) .OR. (I .GE. 8)) GOTO 100
        NCHAR = I + 3
        DO 110 J = I+1, NCHAR
        IFLAG = -1
        IF ((NAME(J:J) .LT. 'A') .OR. (NAME(J:J) .GT. 'Z') .AND.
     1      (NAME(J:J) .LT. 'a') .OR. (NAME(J:J) .GT. 'z')) IFLAG=IFLAG+1
        IF (NAME(J:J) .LT. '0' .OR. NAME(J:J) .GT. '9') IFLAG=IFLAG+1
 110    IF (IFLAG .EQ. 1) GOTO 100
C       * Legal File Name.  Return.
        NAME=NAME(1:NCHAR)
        RETURN
 200    CONTINUE
C
C               Bad filename: deal with it...
C
 100    CALL TTYOUT('INVALID FILENAME.  TRY AGAIN: $')
        GOTO 15
        END
        SUBROUTINE FILOUT (MSG, LUNIT)
C
C
C       This subroutine outputs the string MSG onto the user's terminal.
```

```
C
 5000  IF (LUNIT .NE. 0) THEN
          WRITE (LUNIT,4000)
 4000     FORMAT ()
       ENDIF
C
       RETURN
       END
       FUNCTION LASK (MSG)
C
C      THIS PRINTS MSG AS A PROMPT OF UP TO 255 CHARACTERS, THEN
C      ACCEPTS EITHER Y OR N AS A RESPONSE.
C
       CHARACTER*1 LANS, LASK
       CHARACTER*(*) MSG
       CHARACTER*255 MSG1
C
       LUNIT=0
C
       LENGTH=LEN1(MSG)
       MSG1=MSG(1:LENGTH)//' $'
       LENGTH1=LEN1(MSG1)
       CALL TTYOUT (MSG1(1:LENGTH1))
       LASK = LANS ('Y','N')
       RETURN
       END
       INTEGER FUNCTION LEN1 (TEMP)
C
C      COMPUTES THE NUMBER OF NON-BLANK CHARACTERS
C      OF A GIVEN STRING
C
       CHARACTER*(*) TEMP
C
       LENGTH=LEN(TEMP)
 100   IF (LENGTH .EQ. 0) THEN
          LEN1=0
          RETURN
       ENDIF
       IF (TEMP(LENGTH:LENGTH) .EQ. ' ') THEN
          LENGTH=LENGTH-1
          GOTO 100
       ENDIF
       LEN1=LENGTH
       RETURN
       END
       SUBROUTINE TTYIN (IDUM1,IDUM2,MSG)
C
C      DIRECTS USER TO 'FILIN' WITH LUNIT=0
C      IDUM1 AND IDUM2 ARE DUMMY VARIABLES AND ARE NOT ACTUALLY
C      USED IN THIS SUBROUTINE
C
       CHARACTER*(*) MSG
C
       LUNIT=0
       CALL FILIN (IDUM1,IDUM2,MSG,LUNIT)
       RETURN
```

```
      END
      SUBROUTINE TTYOUT (MSG)
C
C     DIRECTS USER TO 'FILOUT' WITH LUNIT=0
C
      CHARACTER*(*) MSG
C
      LUNIT=0
      CALL FILOUT (MSG, LUNIT)
      RETURN
      END
      SUBROUTINE VECTIN(MODE,VECNAM,VECDIM,VECTOR,VECMIN,VECMAX)
C
C     LOADS A VECTOR VARIABLE (VECTOR) COMPONENT BY
C     COMPONENT FROM THE TTY, IN A PROMPTING MODE,
C     CHECKING THAT THE TTY INPUT VALUE IS BETWEEN VECMIN
C     AND VECMAX.
C     VECNAM IS A ONE-CHARACTER LITERAL ASSOCIATED WITH THE
C     VECTOR, AND VECDIM IS THE VECTOR'S DIMENSION; BOTH
C     ARE ASSUMED SUPPLIED BY THE CALLING ROUTINE.
C     WHEN MODE=1 SEQUENTIAL ENTRY & CORRECTION ARE DONE
C                 =2 CORRECTION ONLY IS DONE
C
      CHARACTER*1 LASK,VECNAM
      INTEGER VECDIM
      DIMENSION VECTOR(1)
C
      GO TO(5,15)MODE
      STOP'*****VECTIN:ILLEGAL VALUE FOR MODE*****'
C
C      * READ-IN SECTION
    5 CALL TTYOUT('INPUT REAL VECTOR VALUES:')
      DO 10 J=1,VECDIM
      I = J
   10 VECTOR (I) = VECVAL (I, VECNAM, VECMIN, VECMAX)
C
C1030 IF (LASK ('LIST VECTOR VALUES? ') .EQ. 'N') GOTO 1000
C     DO 1010 I=1,VECDIM
C        WRITE (0,1020) VECNAM,I,VECTOR(I)
C1020    FORMAT (1X,A,'(',I2,')=',F15.7)
C1010 CONTINUE
C
 1000 CALL TTYOUT (' ')
      IF (LASK ('ANY CHANGES? ') .EQ. 'N') RETURN
C
C     * CORRECTION SECTION
   15 CALL TTYOUT('ENTER COMPONENT INDEX')
   20 CALL TTYOUT('I=$')
      I=IANS(1,VECDIM)
      WRITE (0,1050) VECNAM,I,VECTOR(I)
 1050 FORMAT (1X,A,'(',I2,')=',F15.7)
      VECTOR (I) = VECVAL (I, VECNAM, VECMIN, VECMAX)
      CALL TTYOUT (' ')
      IF (LASK ('MORE? ') .EQ. 'Y') GOTO 20
      RETURN
      END
```

```
C
C
C
      FUNCTION VECVAL (I, VECNAM, VECMIN, VECMAX)
C
      CHARACTER*1 VECNAM
      LUNIT=0
C
    5 WRITE (LUNIT,100) VECNAM, I
  100 FORMAT(1X,A1,'(',I2,')='$)
      VECVAL = RANS(VECMIN, VECMAX)
      END
```

APPENDIX F: LISTING FOR LIBRARY UTLLIB

```
        .TITLE ATOD
;       SUBROUTINE ATOD(ICHAN,IDATA)
;
;       IN FILE ATOD.MNC
;
;       ICHAN SPECIFIES CHANNEL NO. FROM 0 TO 15
;       IDATA IS DATA WORD, BETWEEN 0 AND 4095,
;       INCLUSIVE
;
;
        .GLOBL  ATOD

;       HPL/SAT DEFINITION    (!!!COMMENT OUT FOR MNC!!!)
;       LPSADS = 170400          ;A/D CONVERTER STATUS

;       MNC     DEFINITION    (!!!COMMENT OUT FOR HPL/SAT!!!)
        LPSADS = 171000          ;A/D CONVERTER STATUS

;       COMMON DEFINITION
        LPSADB = LPSADS+2        ;A/D CONVERTER BUFFER

ATOD:   TST     (R5)+            ; SKIP PAST PARAMETER COUNT
        MOV     (R5)+,R0         ; GET ADDRESS OF BUFFER
        MOV     (R5),R1          ; GET DATA BUFFER ADDRESS
        CLR     LPSADS          ; INITIALIZE CONVERTER
        MOV     (R0), R2         ; GET CHANNEL NUMBER
        ASH     #10,R2           ; SHIFT TO LEFT BYTE
        MOV     R2,LPSADS
        INC     LPSADS          ; START CONVERSION

1$:     TSTB    LPSADS          ; WAIT FOR CONVERSION
        BPL     1$               ; TO FINISH

        TSTB    LPSADS+1        ; CHECK FOR CONVERSION
        BMI     2$               ; ERROR

        MOV     LPSADB,(R1)     ; SAVE DATA IN BUFFER
        RTS     PC

2$:     MOV     #-1,(R1)         ; FORCE ERRONEOUS DATA TO -1
        RTS     PC

        .END
```

```
        .TITLE  CLOCK
;SIMPLE MSEC CLOCK ROUTINES FOR LPS-11 ON HPL, SAT, MNC

;HPL/SAT DEFINITIONS     (!!!COMMENT OUT FOR MNC!!!)
;STATUS=170404
;MODE1= 400
;RATSHF=1        ;NUMBER OF BITS RATE MUST BE LEFT-SHIFTED

;MNC DEFINITIONS         (!!!COMMENT OUT FOR HPL/SAT!!!)
STATUS=171020
MODE1=  2
RATSHF=3        ;NUMBER OF BITS RATE MUST BE LEFT-SHIFTED

;COMMON DEFINITIONS
PRESET= STATUS+2  ;NO INTERRUPT VECTORS USED
RUN=    1
DONEFL= 200

;CLSTRT(IRATE,NTICKS): SET CLOCK FOR NTICKS AT IRATE, MULTIPLE INTERVAL MODE
;IRATE: 1=1MHZ, 2=100KHZ, 3=10KHZ, 4=1KHZ, 5=100HZ, 6=SCHMITT-TRIGGERED,
7=LINE
CLSTRT::CLR     STATUS                  ;CLEAR ANY EXISTING STATE
        TST     (R5)+             ;SKIP ARG COUNT
        MOV     R5)+,R1         ;GET RATE
        ASH     #RATSHF,R1        ;SHIFT TO REQUIRED POSITION

        BIS     #MODE1+RUN,R1     ;SET MODE, RUN BITS
        MOV     R5)+,R0         ;GET NO OF CLOCK TICKS IN PERIOD
        BEQ     CLSX              ;DO NOWT IF NO TICKS..
        NEG     R0
        MOV     R0,PRESET       ;SET COUNTER
STMOD1: MOV     R1,STATUS
CLSX:   RTS     PC

;LOGICAL FUNCTION CLWAIT() RETURNS R0 .FALSE. IF TIMED-OUT ON ARRIVAL,
;  ELSE, WAITS TILL CLOCK TIMES OUT, RETURNS R0 .TRUE. FOR GOOD INTERVAL
CLWAIT::CLR     R0                ;SET FLAG FOR BAD INTERVAL
        BIT     #DONEFL,STATUS          ;ARE WE DONE?
        BNE     WAITX             ;YES
        BIT     #RUN,STATUS     ;IS AN INTERVAL SET UP?
        BEQ     WAITX                     ;NO: ABORT
WTLOOP: BIT     #DONEFL,STATUS          ;YES: WAIT FOR DONE FLAG
        BEQ     WTLOOP
        COM     R0                ;FLAG GOOD INTERVAL
WAITX:  BIC     #DONEFL,STATUS
        RTS     PC

;CLSTOP() STOPS CLOCK DEAD
CLSTOP::CLR     STATUS
        RTS     PC
        .END
```

```
        .TITLE DTOA
;       SUBROUTINE DTOA(ICHAN,IDATA)
;
;
;       ICHAN SPECIFIES CHANNEL NO. FROM 0 TO 5
;       IDATA IS DATA WORD, ASSUMED BETWEEN ZERO AND
;       4095 INCLUSIVE
;

        .GLOBL DTOA

;       HPL/SAT DEFINITION      (!!!COMMENT OUT FOR MNC!!!)
;       EXTDA=170420

;       MNC DEFINITION          (!!!COMMENT OUT FOR HPL/SAT!!!)
        EXTDA=171060

DTOA:   TST     (R5)+           ;SKIP ARGUMENT COUNT
        MOV     R5)+,R0        ;GET CHANNEL NUMBER
        ASL     R0              ;AND MPY BY 2
        MOV     R5)+,EXTDA(R0);LOAD DA
        RTS     PC

        .END
        SUBROUTINE LIMIT (XLOW, XHIGH, X)
C
        IF (XLOW .LE. XHIGH) GOTO 10
        CALL TTYOUT ('*****LIMIT: LOW/HIGH LIMITS REVERSED*****')
        STOP
C
 10     IF (X .LT.  XLOW) X = XLOW
        IF (X .GT. XHIGH) X = XHIGH
        RETURN
        END
```

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-4140		

4. Title and Subtitle	5. Report Date
Identification of Visual Evoked Response Parameters Sensitive to Pilot Mental State	April 1988
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
G. L. Zacharias	R8701
	10. Work Unit No. 505-67-11-01

9. Performing Organization Name and Address	11. Contract or Grant No.
Charles River Analytics Inc. 55 Wheeler Street Cambridge, MA 02138	NAS1-17816
	13. Type of Report and Period Covered: Contractor Report 3/85 - 8/87

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665	

15. Supplementary Notes

LaRC Technical Monitor: Alan T. Pope
Final Report

16. Abstract

The primary objective of this study is the development and demonstration of systems analysis techniques for modeling the electroencephalographic (EEG) steady-state visual evoked response (ssVER), for use in EEG data compression and as an indicator of mental workload. The study focused on steady-state frequency-domain stimulation and response analysis, implemented with a sum-of-sines (SOS) stimulus generator and an off-line describing-function response analyzer.

Three major tasks were conducted: 1) VER-related systems identification material was reviewed; 2) software for experiment control and data analysis was developed and implemented; and 3) ssVER identification and modeling was demonstrated, via a mental loading experiment.

It was found that a systems approach to ssVER functional modeling can serve as the basis for eventual development of a mental workload indicator. The review showed how transient visual evoked response (tVER) and ssVER research are related at the functional level, the software development showed how systems techniques can be used for ssVER characterization, and the pilot experiment showed how a simple model can be used to capture the basic dynamic response of the ssVER, under varying loads. Further work is required, however, to determine if the observed loading sensitivity holds up over a larger subject base and a wider range of tasks.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Electroencephalograph, Steady-State Evoked Response, Transient Evoked Response, Mental Workload	Unclassified-Unlimited Subject Category 54

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	176	A09

NASA FORM 1626 OCT 86

NASA-Langley, 1988

For sale by the National Technical Information Service, Springfield, Virginia 22161-2171

☆U.S. GOVERNMENT PRINTING OFFICE: 1988-526-064/61819