THEORY AND APPLICATIONS OF SENSITIVITY ANALYSIS TO ENZYME KINETICS

By

Thomas Henry Pierce III

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Chemistry

1981

ABSTRACT

THEORY AND APPLICATIONS OF SENSITIVITY ANALYSIS
TO ENZYME KINETICS

By

Thomas Henry Pierce III

The theory of Sensitivity Analysis and its applications to Enzyme Kinetics are examined. The Walsh Sensitivity Analysis Procedure, WASP, is developed and shown to be a powerful probe of the theory of Sensitivity Analysis as well as the preferred method for discrete models. The Fourier Analysis Sensitivity Test, FAST, method is reviewed and shown to be the preferred method of Sensitivity Analysis for continuous models. The linear Taylor series approach to Sensitivity Analysis is given as an aid in interpreting Sensitivity Analysis results.

The theory of Walsh function Sensitivity Analysis is derived and its advantages are investigated. The Walsh technique is shown to be an exact technique for discrete model output functions. For continuous model output functions the Walsh method yields an averaged finite difference Taylor series with respect to the parameters. Walsh Analysis and 2-point discrete Fourier Analysis are shown to be identical. Since Walsh Analysis is easily

related to both the Fourier method and to the linear Taylor series method, it is a valuable tool for further development of Sensitivity Analysis.

The applications of the mass action laws of chemical kinetics are used to develop models which are analyzed with respect to their parameters. Enzyme Kinetics models for hysteresis and allosterism are investigated by the techniques of Sensitivity Analysis. The mechanism of hysteresis in the Frieden Model and the Ainslie model is clearly shown to be an effect of the rates of isomerization of the inactive enzyme-substrate complex to the active enzyme-substrate complex. The Ainslie model is dynamically equivalent to the simpler Frieden model for a large set of rate constants. The Frieden model also displays apparent allosterism if the "correct" set of rate constants and initial condititons are used. Therefore the Frieden model is the simplest one-site enzyme kinetic model which displays both burst and lag hysteresis as well as both positive and negative cooperativity (allosterism).

Fourier Sensitivity Analysis was applied to a pH Tryptophanase model where the parameters and their variations were obtained from experimental data. This type of analysis gave insight to the design of future experiments. Over the experimentally accessible range of pH, the lower pH region is shown to contain the most

information on the parameters which can be examined in future experiments.

A computer program is given which is used for routine application of Sensitivity Analysis, both Fourier and Walsh, to other models. This program has been extensively revised to clarify its logic and to simplify its use. Any mathematical model which can be simulated on a computer may be directly inserted into this program.

Suggestions for future work are discussed. Research in the connections between Statistics and Sensitivity Analysis should lead to insight into both areas. The investigation of "approximate" Walsh Sensitivity Analysis may lead to faster algorithms for Sensitivity Analysis.

ACKNOWLEDGEMENTS

A great many people assisted me in this endeavor and I am especially grateful to those mentioned here. I would like to thank Rex Kenner, Bill Waller, Barb Kennedy and Aristides Mavridis for their help in getting me started; Jim Mehaffey and Mark Ondrias for keeping me sane. My parents and Grandmother Pierce I thank for their encouragement. I am grateful to Cathy Stewart for her conviction and care during these difficult times. I appreciate the everpresent support and guidance of Professors Robert Cukier and James Dye and acknowledge Dr. Robert Ball (Dr. Bob) for his penetrating discussions, computer programs (GETNUM, PLOT) and especially his friendship without which this work would have been impossible.

I am also indebted to Michigan State University, Chemistry Department, for an assistantship as well as support from NSF Grant No. PCM 78-15750.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. SENSITIVITY ANALYSIS - AN OVERVIEW

INTRODUCTION

Mathematical models have exerted tremendous influence in science. Many people have commented on the "seemingly exact" way that mathematical equations model nature (Benacerraf, et al. 1964). It is this ability to 'describe' physical processes that makes mathematics so useful in science.

Mathematical models are composed of four parts; independent variables through which the model evolves, dependent variables which change as a function of the independent variables, parameters which are constant during a simulation but may change from one simulation to another, and constants which never change, such as the velocity of light in a vacuum.

Once a mathematical model is proposed, if it is 'correct', we can use it to predict the future behavior of a physical system. It may be used to explain previous behavior of the physical system. To do this many models require the 'adjustment' of parameters. It is this ability of these parameters to describe different physical systems by changing their values which gives great generality to

mathematical models and causes confusion as to whether or not the model is 'right'. Often two or more models give the correct results using only different parameters. Mathematical models depend on their parameters.

Sensitivity analysis describes precisely how the mathematical model depends on its parameters. An intuitive sensitivity analysis method would be to vary a parameter over two values and observe how the model changes. If a particular value of the model output increases as the parameter increases, we say that the output is positively affected. This can be generalized by asking for the quantitative sensitivity of a particular output function to a parameter. The most popular method is to take the derivative of the output function with respect to the parameter 'p', evaluated at a nominal value p_o.

$$\frac{\Delta f}{\Delta p} \sim \left(\frac{\partial f}{\partial p}\right)_{p=p_o} \tag{1.1}$$

Many models have more than one parameter. A collection of the derivatives with respect to the parameters permits observations to be made about the model. For example, we can order the parameters according to their significance. The most significant parameter is the one which has the largest effect on the value of the model output function. This leads to an ordering of the parameters with respect to their effect on the model, from most significant to

least significant.

Often models have parameters with at least one independent variable. An example is the temperature dependence of a rate constant:

$$k = Ae^{-E_a/RT} \qquad (1.2)$$

Here A and E_a are parameters, R is a constant, and temperature, T, is the independent variable. The model output function is the rate constant k, the dependent variable. In such cases the sensitivity of the output function depends not only on the parameters but also on the independent variable. The model output function is the rate constant k, the dependent variable. In such cases the sensitivity of the output function depends not only on the parameters but also on the independent variable. When measurements are repeated at different values of the independent variable, sequences of parameter sensitivity values can be collected over a temperature range of interest.

A sequence of parameter sensitivity values may also give other useful information. From it we may be able to identify regions of sensitivity. Using the previous example, there may be temperature ranges where the sensitivity ordering of parameters changes. In one region the parameter 'A' may be the most important, while in a different region the parameter 'E_a' may be the most important.

Therefore, to measure 'A', we should measure it in the first region where the model is most sensitive to 'A'.

For a more complex model it may happen that in the region of interest the model has no significant sensitivity to one or more of the parameters. In this case the model may be reduced to a simpler model by formally fixing the value of insensitive parameter to zero (or one): For example, it may be possible to remove the step corresponding to the parameter in question from a mechanism.

Application of sensitivity analysis can also help to validate a model. Knowing the rank-order of the parameters' sensitivities and the region of maximum sensitivity permits the design of experiments to probe the test model over these regions. A fit of the model to the new data, gives further evidence that the model is correct.

As described above, sensitivity analysis can lead to better understanding of the model. Since many models are complex an intuitive understanding of model behavior may be difficult to obtain. It is useful to have quantitative mathematical tools which help to crack a complex model into separate parts, which can then be independently analyzed and understood.

Classical linear sensitivity analysis is a useful first approach which is straightforward. One examines the change in the model output function caused by a unit change in a parameter, $\frac{\partial f}{\partial p}$ (Beck 1977). However, this

method is only rigorously correct for linear models; those models whose output functions are linear in their parameters, since the first derivative is the only variable effect attributable to a parameter.

With nonlinear models, higher order effects may be important. The presence of higher order terms can be verified by generalizing classical sensitivity analysis to a Taylor series with respect to the parameters.

$$f(p_1,p_2,\ldots p_n) = f(\underline{p})\Big|_{\underline{p}=\underline{p}_o} + \sum_{i=1}^{n} \left(\frac{\partial f}{\partial p_i}\right)\Big|_{p=p_o} \Delta p_i$$

$$+ 1/2 \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f(\underline{p})}{\partial p_i \partial p_j}\Big|_{p=p_o} \Delta p_i \Delta p_j$$

$$+ \ldots \qquad (1.3)$$

In linear models higher derivatives, f", f''', etc, are zero so that changes caused by parameters are weighted only by f' evaluated at the nominal value, $\underline{p}=\underline{p}_o$. For nonlinear models, the first derivative approximation is good if all higher derivatives are small or if the region of variation is so small that $(\Delta p)^2 \approx 0$.

Both of these conditions are very restrictive. We

would like to vary a parameter over its entire valid range, which is often many orders of magnitude. In some cases higher derivatives are as large or larger than first derivatives. With these problems in mind the idea of alternate representations of the model output function in terms of the parameters is a natural step.

In 1973 Cukier et al. derived a technique which represents the model function in terms of a Fourier series. They related the sensitivity of each parameter to a separate Fourier coefficient. Since any expansion of a well-behaved function is identical to another expansion which has been rearranged, it can be seen (Appendix 1) that these Fourier coefficients are functions of all the higher derivatives of the model function with respect to the parameters. This is the ideal relationship required for nonlinear functions. It allows sensitivity measures where the parameters are varied over orders of magnitude with no restrictions on the model output function.

The implementation of the Fourier method on a computer requires approximations as explained in Chapter Two. The approximations limit the method through an accumulation of approximation error. However the sources of the error have been described by Cukier et al. (1975). These errors are controllable and they can be bounded by a maximum error estimate.

Other expansions are possible and in Chapter Three we

will investigate Walsh series expansions (Walsh 1923, Fine 1949). It will be shown that for a discrete model we incur no errors in analyzing the sensitivity of a model. However for continuous models it will be shown that a Walsh sensitivity expansion is identical to a finite-difference Taylor series.

In Chapter Four the three approaches are compared, and we can see that they give similar results when the parameter variation approaches zero. In the case of a global analysis of a strongly nonlinear model only the Fourier method gives the correct results for the sensitivities.

In Chapter Five we apply the Fourier technique to some steady-state enzyme kinetics models. Here we show that two apparently different models are dynamically identical over a rather extensive range of parameter variation. Also the sensitivity analysis of progress curves shows that some parameters may 'accumulate' sensitivity in time. At short times they are relatively insensitive, but as the reaction proceeds they become the most important parameters in the model.

In Chapter Six we apply the Fourier technique to a recently-studied transient-state enzyme model (June et al. 1979, 1980).

The future work and development of sensitivity analysis techniques is discussed in Chapter Seven. It is suggested

that the study of approximate Walsh sensitivity analysis and the frequency sets used in approximate analysis will extend both methods. Also the relationship of sensitivity analysis and statistics is discussed. It is our belief that these questions will direct research into fruitful areas which will advance the usefulness of sensitivity analysis techniques with extremely complex models.

Appendix 8 contains the various programs and their operation instructions for the application of both Fourier and Walsh Sensitivity Analysis. The programs are model independent so that any type of numerical model may be used. It is my hope that sufficient theory and examples are given here to encourage others to use these powerful techniques.

II. FOURIER SENSITIVITY ANALYSIS

In this chapter we will discuss the Fourier method of sensitivity analysis. Only an overview of the theory will be given as this technique was extensively reviewed by Cukier et al. (1978). Here we will examine the details of the implementation and review the approximations and limitations of this method. The particular model chosen in this implementation is derived from the laws of mass action kinetics.

Chemical rate equations as derived from postulated mechanisms can be described by sets of first order in time, coupled ordinary differential equations of the form,

$$\frac{dC_i}{dt} = F_i(\underline{C},t,\underline{k}) \qquad (2.1)$$

$$i = 1,2,3,\ldots,m$$

with prescribed initial conditions,

$$C_i(t=0) = C_i^o \qquad (2.2)$$

In Equation (2.1) $C_i(t)$ is the concentration of the ith species at time t, $\underline{C} = (C(1),C(2),\ldots,C(m))$, is a

vector of all the species concentrations and $\underline{k}$ = (k(1), k(2),...,k(n)) is a vector of all the rate constants (parameters). The function F_i symbolizes the rate law for the species i. We shall assume that these rate equations can be solved for $\underline{C}(t)$, given the initial conditions, the values of the rate constants, and the specific functional form of the rate laws. If this cannot be done analytically, it can almost always be done numerically.

We require the sensitivities of the concentration $C_i(t)$ to uncertainties in the values of k_ℓ, the rate coefficients. The uncertainties in the rate coefficients are, in this method, represented statistically. That is, we assign a probability density, $p_\ell\ (k_\ell)dk_\ell$ as the probability that the ℓ'th rate coefficient lies between k_ℓ and $k_\ell + dk_\ell$. These probability densities reflect our knowledge of the possible values of the rate coefficients in a given elementary chemical reaction. If one has accurate data, then the probability density can be chosen to be narrow to reflect this information. However, if data are sparse or not reliable, the uncertainty can be chosen as large as desired.

The joint distribution function may then be written

$$P(\underline{k}) = \prod_{\ell=1}^{n} p_\ell\ (k_\ell) \qquad (2.3)$$

as we require the rate constants' probability distributions to be linearly independent, but not necessarily identically distributed. Once the joint distribution is known we may construct moments of the multidimensional function by multidimensional integration. The first moment, the average value, is written

$$\langle C(\underline{k})\rangle = \int d\underline{k}\ C(\underline{k})P(\underline{k}) \qquad (2.4)$$

where $d\underline{k} = dk_1 dk_2 \ldots dk_n$ is the multidimensional volume element in the rate coefficient space. In the present example $\langle C(\underline{k})\rangle$ represents the average concentration of a given species as calculated from the rate laws and the rate constants, where the rate constants are varied over their entire set of possible values.

Similarly we may construct multidimensional variances of the function, by calculating

$$(\sigma^2)_i = \langle C_i(\underline{k})^2\rangle - \langle C_i(\underline{k})\rangle^2 \qquad (2.5)$$

This would represent the expected spread of concentrations over values accessible to species $\underline{i}$ because of uncertainties in the rate constants. Similarly partial variances, the variance along only one parameter dimension, say the first

parameter can be computed as

$$(\sigma_1^2)_i = \langle (C_i^*(k_1))^2 \rangle - \langle C_i^*(k_1) \rangle^2 \qquad (2.6)$$

where $C_i^*(k_1)$ is the function averaged over $\underline{k} = (k_2 \ldots k_n)$ and the integration is over k_1. This gives the spread of concentrations caused by uncertainties in k_1. This idea may be extended to coupled partial variances, variances over more than one parameter at a time.

These variances would be very informative. We would be able to characterize the extent that the model depended on the parameters. This also would tell us which parameters were most important (those whose variances were largest). If a parameter's variance were small then the effect of a parameter changing over its entire range is negligible to the behavior of the model. This means that the model may be simplified by excluding parameters whose variances are small.

The coupled partial variances would tell us how the parameters interact. If these coupled variances are large then the model also depends on the relationship of coupled parameters. The effect of one parameter acts in concert with another parameter. These coupled variances may be extended to arbitrary number of parameters coupled together (but less than n).

The only requirement here is the construction of the

joint probability distribution and its multidimensional integration. Presumably this could be done by numerical quadrature. We would sample each parameter over its domain, then solve the model for each different combination of the parameters. The solutions would be numerically integrated against the joint probability distribution to give the desired moments.

The required amount of calculation to accomplish this is enormous. If we chose 10 different values for each parameter, and there were only five parameters in the model, we would have to solve the model 5^{10} times, approximately 10 million times. Even so, if the ranges of the parameters were large or the model highly structured, we would need still more sampled points to accurately carry out these variance calculations by such a brute force method. To calculate the variances in finite time we need a different approach. We need a way to compute the multidimensional integrals without exhaustive sampling of the output function.

In 1938 Hermann Weyl (Weyl, 1938) derived an integral identity which, under certain conditions, reduces a multidimensional integral to a single path integral. To apply his theorem we must return to our definition of parameters and transform them into periodic functions.

The rate constants, considered as random variables, may be related to a generating function,

$$k_\ell = g_\ell (u_\ell) \tag{2.7}$$

where g_ℓ is the generating function and u_ℓ is the independent variable. As u_ℓ is varied from $-\infty$ to ∞, k_ℓ is varied over all its possible values. Consequently, u_ℓ also has a probability distribution. Since the k_ℓ's are independent functions of the u_ℓ's, the u_ℓ's are also independent. We may then write the total joint probability density function in u-space as

$$P(\underline{u}) = \prod_{\ell=1}^{n} P_\ell(u_\ell) \tag{2.8}$$

It is convenient to let u_ℓ be related to θ_ℓ through the transformation function

$$u_\ell = G_\ell(\theta_\ell) \tag{2.9}$$

such that as θ_ℓ traverses $-\infty$ to ∞, u_ℓ also goes from $-\infty$ to ∞, so no information has been lost. Now we further write $\theta_\ell = (\omega_\ell s)$ with $-\infty < s < \infty$. The new parameter, ω_ℓ, is called the ℓ'th frequency. This procedure relates each parameter to a frequency, ω_ℓ, so that by varying s over its range all the parameters vary <u>simultaneously</u>, at different frequencies, over their ranges.

The probability distributions in u-space are easily related to a probability distribution in θ-space. It is desirable for all unit lengths in θ-space to be equiprobable, i.e., we want to insure that $P(u_\ell)du_\ell = P(\theta_\ell)d\theta_\ell$. Note that we are using the same symbol for the probabilities even though we have transformed the independent variable u_ℓ to θ_ℓ. This is done to simplify notation.

The chain rule may be used to derive the equation relating these two probabilities.

$$P(\theta_\ell)d\theta_\ell = P(u_\ell)du_\ell$$

$$du = \frac{du}{dx}\frac{dx}{d\theta}\,d\theta; \quad x = \sin\theta, \quad \theta_\ell = \omega_\ell s$$

$$P(\theta_\ell) = P(u_\ell)\frac{du}{dx}\frac{dx}{d\theta} \tag{2.10}$$

$$\frac{dx}{d\theta} = \cos\theta = (1-x^2)^{1/2}$$

$P(\theta) = \frac{1}{\pi}$ for the half interval

Writing $u_\ell = G_\ell$ we obtain

$$\frac{1}{\pi} = P(G_\ell)(1 - x^2)^{1/2} \frac{dG_\ell}{dx} \tag{2.11}$$

This is a first order differential equation whose solution is the transformation function G_ℓ. To uniquely solve this equation we need an initial condition.

If we define $k_\ell = k_\ell^o \exp(u_\ell)$, and note that when $s = 0$, $x = \sin(\omega_\ell s) = 0$, we see that $u_\ell = G(\sin(\omega_\ell s)) = 0$ which implies that $G(0) = 0$. This is the required initial condition. This means that u_ℓ is restricted to a polynomial in $\sin(\omega_\ell s)$ with no additive constants. Some possible transformation functions are given in Appendix 2 along with the distributions that they generate.

Now that the rate constants are related to the search variable, s, we can apply Weyl's theorem.

$$\left(\frac{1}{2\pi}\right)^n \int d\underline{\theta} F(\underline{\theta}) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} F(s)ds \tag{2.12}$$

Weyl showed that this integral identity would be exact if

$$\theta_\ell = \omega_\ell s \text{ and } \sum_{\ell=1}^{n} \alpha_\ell \omega_\ell \neq 0 \tag{2.13}$$

for all possible integer values of α_ℓ. In this case the frequency set, $\{\omega_\ell\}$, is called incommensurate.

An incommensurate frequency set is easily constructed if we use irrational numbers for the frequencies. However, since we will be using a computer for solving the model, irrational numbers are not feasible. What is done is to define an order of accuracy 'M' such that

$$\sum_\ell \alpha_\ell \omega_\ell \neq 0 \quad \text{for} \quad \sum_i |\alpha_i| \leq M + 1 \tag{2.14}$$

or more concretely

$$\sum_\ell \alpha_\ell \omega_\ell = 0 \quad \text{for} \quad \min_{\underline{\alpha}} \left[\sum_\ell |\alpha_\ell| = M + 2\right]$$

Once we have defined an order of accuracy it can be seen that irrational numbers for the frequency set are no longer required. Now a frequency set will be associated with its value of M. In fact, we may use integer frequencies as it simplifies further calculations.

The finding of arbitrarily accurate frequency sets is apparently quite difficult. For the special case of 4th order accurate sets, <u>i.e.</u>, M = 4, we may exploit the

idea of sums and differences of two frequencies to find the sets. Schailbly et al. have tabulated 4th order accurate frequency sets for parameter sets of up to fifty parameters. These sets are stored in the program in Appendix 8.

Since the parameters are proportional to sines, it is prudent to use only odd frequencies in order to exploit the periodicity of the sine of an odd frequency (see Cukier 1978). This helps in the search for frequencies by eliminating half of the integers.

Given the frequency set, we can approximate the multidimensional θ-space integral in s-space. But the s-space integral has other valuable properties. We have related each parameter to a function of a sine of a frequency. Since sines of different frequencies are part of an orthonormal family, the Fourier series, the effect of each parameter may be easily projected out of the s-space integral.

Expanding the output function in a Fourier series (Zygmund 1959), given

$$C^i(\underline{k}(s)) = \frac{A_o}{2} + \sum_{j=1}^{\infty} [A_j \cos(js) + B_j \sin(js)] \qquad (2.15)$$

$$A_j^i = \frac{1}{\pi} \int_{-\pi}^{\pi} C^i(s)\cos(js)ds$$

$$B_j^i = \frac{1}{\pi} \int_{-\pi}^{\pi} C^i(s)\sin(js)ds$$

or equivalently in exponential format,

$$C^i(\underline{k}(s)) = \sum_{j=0}^{\infty} C_j \exp(i2\pi js)$$

where

$$C_j = \frac{1}{\pi} \int_{-\pi}^{\pi} C^i(\underline{k}(s)) \exp(i2\pi js)ds \qquad (2.16)$$

Note that $C_j = \frac{A_j + B_j}{2}$ separates the output function into its frequency components. ("i" denotes the ith concentration). Since the th parameter is associated with the ℓth frequency, the magnitude of the Fourier coefficients of this frequency and its harmonics measure the sensitivity of the ℓth parameter.

We can illustrate this with a simple example. Consider a reaction scheme with three rate coefficients associated with three frequencies, w_1, w_2, and w_3. Since we vary the rate coefficients as $\sin(w_\ell s)$ the ith concentration as a function of s will consist of sums and products of these $\sin(w_\ell s)$ factors. When strings of these sines are multiplied together the result is sines and cosines of sums of the $w_\ell s$ factors. If we assume that the frequencies in the sums of the $w_\ell s$ factors are incommensurate, no linear combination of the frequencies can be formed which sums to zero. That is, there are always three

independent components in these sums. If the Fourier decomposition of C(s) is performed the Fourier coefficients of indices w_1, $2w_1$, $3w_1$,... can only be due to the rate coefficient k_1 since only k_1 varies with w_1 and no combination of w_2 and w_3 can add up to a multiple of w_1 to cause an interference. Thus, the Fourier analysis enables us to isolate the effect on the ith concentration of uncertainty in the th rate coefficient.

The above definitions of A_j and B_j are exact only if we are able to analytically integrate the equations. Since we will numerically integrate the model equations for $C_i(s)$ only concentration-time points will be available. We must then use a discrete Fourier transform instead of a continuous one. The concentration points may be numerically integrated into discrete Fourier coefficients.

There are two kinds of error involved when this approach is used. We obtain the largest error by exchanging integers for irrational numbers in Weyl's integral identity (Cukier 1975). By choosing a value of M and its frequency set we postpone addition errors, the sum of the harmonics equalling zero, beyond the combination of M frequencies or harmonics. Since Fourier coefficients decrease by at least (1/n), error from the combination of harmonics can be maintained at a low level.

This form of error also depends on the model. If there is no combination of M parameters multiplied together

in the model then the possible error from different frequencies adding to the same frequency, breaking a type of linear independence, is eliminated. Also if the output function of the model does not have parameters raised to high powers, such as k^5 or k^{10}, then the frequencies will not add in the high harmonics to give error. In any case one may always increase the value of M to decrease this type of error, provided the frequency set is known.

The second source of error is from the use of a finite discrete transform instead of an infinite continuous one. This error comes about by sampling the function at equally spaced points, $\frac{2\pi j}{N}$. This equal spacing gives an aliasing error, frequencies which oscillate faster than the sampling rate fold their effects into lower frequencies (Cukier 1975)

$$a_k^{calc} = A_k + A_{2N-k} + \sum_{m=0}^{\infty} (A_{2Nm-k} + A_{2Nm+k})$$

$$b_k^{calc} = B_k + B_{2N-k} + \sum_{m=0}^{\infty} (B_{2Nm-k} + B_{2Nm+k}) \qquad (2.17)$$

Aliasing sets a limit on the maximum frequency that one may compute from a sampled function. This maximum

frequency is determined from the Shannon Sampling Theorem (A. J. Jerri 1977) and is called the Nyquist Frequency. One can reduce this type of error by sampling more points. In the literature (Cukier 1975, Jerri 1977) the usual number of samples taken to insure accuracy is $4w_{max}$, where w_{max} is the largest frequency desired.

$$2w_{max} < N \tag{2.18}$$

This sampling rate tells us the minimum number of samples that are required to compute a particular frequency.

Having examined the error terms we find that the number of points chosen is important. However adding a single point implies that we will do an additional simulation with a new parameter vector. The cost in computation time in each simulation can be high. If the model is composed of ordinary differential equations then the computation of the required simulations accounts for about 90% of the total required computation time involved in sensitivity analysis. We would, therefore, like to minimize the number of simulations necessary to calculate the Fourier coefficients. If odd frequencies are chosen, the number of simulations necessary is reduced by one-half. It was shown (Cukier et al. 1978) that the symmetry relations for sines of odd frequencies

$$f(\pi - s) = f(s)$$

$$f(s - \pi) = f(-s)$$

$$f(s + \tfrac{\pi}{2}) = f(s - \tfrac{\pi}{2})$$

$$f(s - \tfrac{\pi}{2}) = f(-s - \tfrac{\pi}{2}) \qquad (2.19)$$

allow us to sample the output function in the range $[-\pi/2, \pi/2]$ and then reflect these values into $[0,2\pi]$. In this way we get twice as many Fourier coefficients as sampled points.

Given a finite number of simulations we can construct a finite Fourier series approximation to the function in s-space. The coefficients in this approximation may be evaluated in two possible ways. The direct application of the transformation, a brute force approach, would require N^2 multiplications on the computer. This can be seen by examining the equations for the Fourier coefficients given below.

$$C_p = \sum_{j=1}^{N} C(s_j)\exp(i2\pi jP/N) \qquad (2.20)$$

or

$$a_j = \frac{2}{N} \sum_{j=1}^{N} C(s_j)\cos(\frac{2\pi jP}{N})$$

$$b_j = \frac{2}{N} \sum_{j=1}^{N} C(s_j)\sin(\frac{2\pi jP}{N})$$

N an odd integer

Alternatively we could use the Fast Fourier Transform algorithm, FFT, of Cooley and Tukey (Cooley, Tukey 1965).

This method uses approximately Nlog(N) multiplications. Usually this algorithm is applied when the number of samples is a power of two. In this case the algorithm is at its most efficient. Since we want to minimize the number of required samples, sampling the function a power of two times is too strict a requirement. This usual restriction in the number of samples is <u>not</u> a requirement for use of the algorithm. In fact, as long as the number of samples taken <u>is not a prime number</u> the FFT algorithm is a much faster and more accurate technique than the direct method. If the number of samples, N, is factored into its prime factors, $n_1 n_2 n_3 \ldots n_\ell = N$, then the number of operations that this generalized FFT algorithm takes is <u>at most</u> $(n_1 N + n_2 N + \ldots n_\ell N)$ (Dahlquist 1974).

By using the FFT method on R points in s-space, we extract 2R coefficients. Some of these coefficients are

ambiguously related to the parameters. This ambiguity depends on the order of accuracy M. The ambiguity implies that more than one linear combination of the w-set frequencies adds to the same frequency. We don't, then, know to which linear combination to assign this frequency. Hence it is considered as an error term and only used in the calculation of the total variance.

Obviously for a Mth-accurate frequency set, only those combinations of w-frequencies whose α-set sum to less than $\frac{M+2}{2}$ may be unambiguously defined. That is, we know, to Mth-accuracy, what combination of parameters add to these frequencies. These Fourier coefficients may be easily combined into the desired variances as was earlier proposed.

Parseval's formula for Fourier Series (Zygmund 1959) gives us the total variance of the model output function

$$\sigma^2_{total} = \sum_{i=1}^{N-1} (a_i^2 + b_i^2) + (\frac{a_o}{2})^2 \qquad (2.21)$$

where N = 2R, and R is the number of simulations.

The contribution of the ℓth parameter to this total variance is contained in the coefficients of the ℓth frequency and their harmonics (Cukier et al. 1978)

$$\sigma^2_\ell = \sum_{p=1}^{K} a^2_{pw_\ell} + b^2_{pw_\ell} \qquad (2.22)$$

where K = M/2, for a Mth accurate frequency set.

From this we can construct the reduced partial variances

$$S_\ell = \frac{\sigma_\ell^2}{\sigma_{total}^2} \qquad (2.23)$$

The contribution of the coupling between parameters is contained in the coefficients of the combination frequencies, $jw_k + pw_\ell$ (Cukier et al 1978).

$$\sigma_{\ell,k}^2 = \sum_{i=-\beta_1}^{\beta_2} \sum_{\substack{j=-\beta_2 \\ i,j \neq 0}}^{\beta_2} a_{jw_\ell+iw_k}^2 + b_{jw_\ell+iw_k}^2 \qquad (2.24)$$

Where $\max[\beta_1 + \beta_2] = (M+1) - M/2$, thereby preventing double counting of frequencies which may be included in single partial variances. This does allow for the possibility of some double counting in the coupled partial variances. However, these high harmonics of the fundamental frequencies are usually attenuated so the error is very slight.

From this we can construct the reduced coupled partial variances,

$$S_{\ell,k} = \frac{\sigma^2_{\ell,k}}{\sigma^2_{total}} \qquad (2.25)$$

Cukier (1975) has related the B_{w_ℓ} coefficient to the linear sensitivity coefficient $(\frac{\partial c}{\partial p_\ell})$ in the limit of small parameter variation.

$$B_{w_\ell} = \langle (\frac{\partial C(u)}{\partial u})(\frac{\partial u}{\partial p}) \rangle + O(p^2) \qquad (2.26)$$

This equation shows a <u>direct</u> relationship between the <u>averaged</u> linear sensitivity coefficient and the Fourier coefficient, B_{w_ℓ}, which is used to construct the w_ℓ partial variance.

The Fourier method of sensitivity analysis is implemented in a program given in Appendix 8. This program is not restricted to models involving ordinary differential equations. Rather any type of mathematical model may be inserted in the program through use of the subroutine MODEL. Hence the global parameter sensitivities of any type of model are easily obtained.

III. HADAMARD-ORDERED WALSH FUNCTIONS

Cukier et al. (1978) proposed that alternate orthogonal expansions also may be possible, leading to other types of sensitivity analysis. This approach was investigated and an alternate expansion was found. It was discovered that a Walsh function expansion (Walsh 1923) could be used for sensitivity analysis. If we consider a model whose parameters take on a finite set of values, then we may use Walsh Sensitivity Analysis. Here the model output function has a finite set of discrete responses dependent on the parameters. For these discrete model functions the use of Walsh functions eliminates the approximations inherent in the Fourier method. With continuous model output functions (those functions whose parameters vary continuously over a domain) the Walsh method is closely related to Taylor Series Sensitivity Analysis. In this chapter we develop the theory of the Walsh method.

A Walsh function (Ahmed and Rao 1975) may be defined as a function of two arguments, a time variable and a sequency variable, similar to frequency in Fourier analysis. Walsh functions form a complete orthonormal set of step functions. Here the Hadamard definition (Ahmed and Rao 1975) is used to represent Walsh functions.

$$WALH(w,t) = (-1)^{\sum_{i=1}^{p} w_i t_i} \qquad (3.1)$$

Where $t=(t_1,t_2,\ldots,t_p)$ is the binary representation of t and $(w_1,w_2,\ldots w_p)$ is the binary representation of w.

Walsh functions are defined over the time range [0,1], i.e., the time variable is a real number less than 1. However, the sequency variable is an integer less than 2^p. This means that the binary point for the time variable is placed to the left of t_1 and the binary point for the sequency variable is placed to the right of w_p. Note that the indexing chosen here labels the most significant digit of the variable first.

With this definition of Walsh functions, the time variable is defined with respect to the sequency variable in order to cancel dimensions. It should be noted that the time referred to here is not "model time". Model time is defined as the independent variable in a model such as the model of a chemical reaction which evolves in time.

To clarify the evaluation of a Walsh function, let us consider the Walsh function, WALH(2, .75). Only two binary digits are necessary to represent these arguments, therefore let p=2. Writing out the binary expansions of the arguments we obtain

$$w = 2_{10} = (10.)_2 \tag{3.2}$$

$$t = .75_{10} = (.11)_2$$

Substituting these binary representations into the defining Equation 3.1 we obtain

$$WALH(2, .75) = (-1)^{w,t_1+w_2t_2} = (-1)^{1.1+0.1} = (-1)^1 = -1 \tag{3.3}$$

Walsh functions are constrained by the number of binary digits required to represent w and t. For this reason we get different groups of functions for each choice of p, the number of binary digits used in the representation. Each group is closed with respect to ordinary multiplication. If we multiply one Walsh function by another Walsh function from the same group, we obtain a third member of the group. This means that multiplication of a p-digit Walsh function with a k-digit Walsh function is not defined.

This group property can be illustrated by examining a Walsh function multiplication table for p-2. Here we have a group of Walsh functions with only four members, WALH(0,t), WALH(1,t), WALH(2,t), and WALH(3,t). By using

Equation 3.1 we can generate Table 1.

Figure 1 gives the plots of these four Walsh functions. They are piecewise continuous functions. These functions may be integrated but they may not be differentiated without the introduction of distributions which include delta functions.

Walsh functions have some useful properties. They are invariant to an exchange of arguments, i.e.,

$$\mathrm{WALH}(w,t) = \mathrm{WALH}(t,w) \tag{3.4}$$

Proof:

$$\mathrm{WALH}(w,t) = (-1)^{\sum_i w_i t_i} = (-1)^{\sum_i t_i w_i} = \mathrm{WALH}(t,w) \tag{3.5}$$

As has been claimed, the Walsh functions are orthogonal (Walsh, 1923),

$$\int_0^1 \mathrm{WALH}(n,t)\ \mathrm{WALH}(w,t)dt = 2^p \delta_{n,w} \tag{3.6}$$

Proof:

Table 3.1. Multiplication Table for the p=2 Group of Walsh Functions.

*	WALH(0,t)	WALH(1,t)	WALH(2,t)	WALH(3,t)
WALH(0,t)	WALH(0,t)	WALH(1,t)	WALH(2,t)	WALH(3,t)
WALH(1,t)	WALH(1,t)	WALH(0,t)	WALH(3,t)	WALH(2,t)
WALH(2,t)	WALH(2,t)	WALH(3,t)	WALH(0,t)	WALH(1,t)
WALH(3,t)	WALH(3,t)	WALH(2,t)	WALH(1,t)	WALH(0,t)

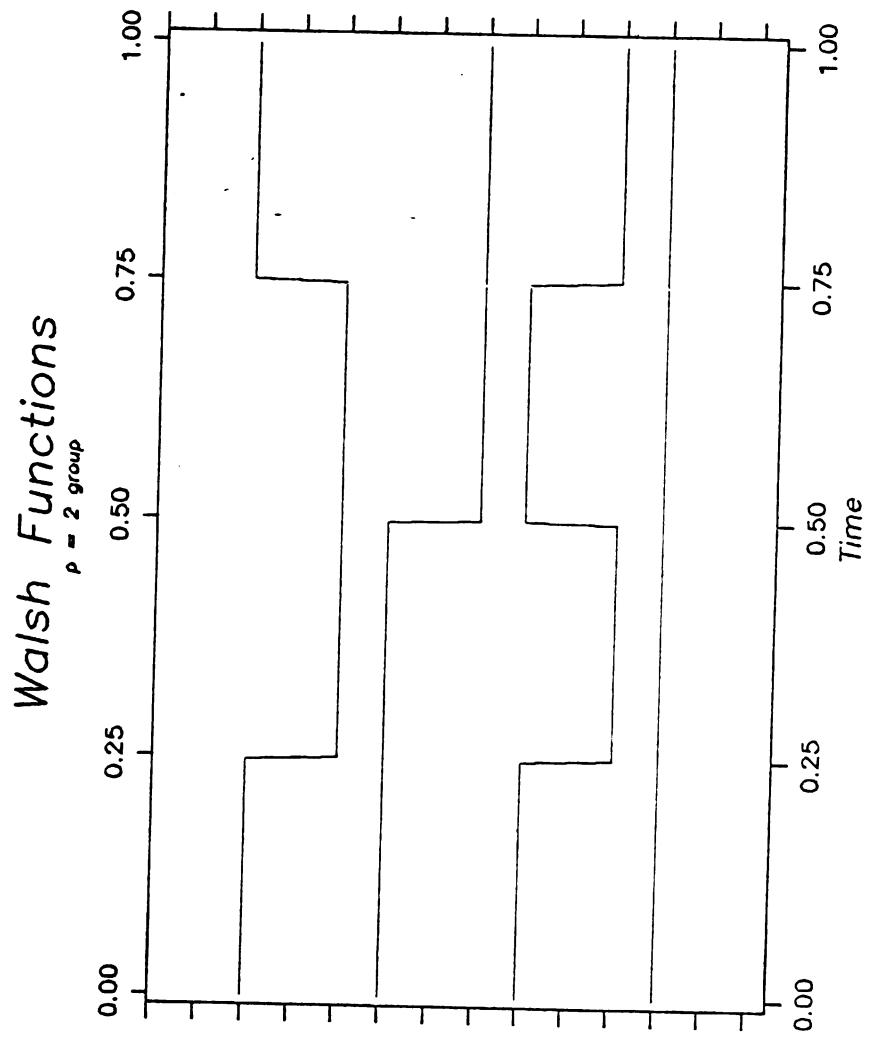


Figure 3.1 The four Walsh functions defined by two binary digits.

$$\int_0^1 \mathrm{WALH}(m,t)\mathrm{WALH}(w,t)\,dt$$

$$= \sum_{t_1=0}^{1} \sum_{t_2=0}^{1} \cdots \sum_{t_p=0}^{1} (-1)^{\sum_{i=1}^{p} n_i t_i} (-1)^{\sum_{i=1}^{p} w_i t_i}$$

$$= \sum_{t_1=0}^{1} \cdots \sum_{t_p=0}^{1} (-1)^{\sum_{i=1}^{p} (n_i+w_i) t_i}$$

$$= \sum_{t_2=0}^{1} \cdots \sum_{t_p=0}^{1} \left(1 + (-1)^{n_1 w_1}\right) (-1)^{\sum_{i=2}^{p} (n_i+w_i) t_i}$$

$$= 2^p \prod_{i=1}^{p} \delta_{n_i,w_i} = 2^p \delta_{n_1 w}$$

If we divide each Walsh function by two we obtain an orthonomal set of functions. Completeness of Walsh functions was shown by Walsh (1923).

Utilizing the orthonormality and completeness properties of Walsh functions we know that any continuous function, f(t), can be expanded into an infinite Walsh series with $0 \leq t < 1$. This exact infinite expansion of an

arbitrary function may be approximated by a finite expansion:

$$f(t) = \sum_{n=0}^{N-1} C_n WALH(n,t) \qquad (3.8)$$

Here we restrict to to a discrete set of N points (t_i) where,

$$Nt = t_i \qquad (3.9)$$

Note that $0 \leq t_i < N-1$ and t_i is an integer. The error incurred by this approximation is

$$error = \sum_{n=N}^{\infty} C_n WALH(n,t) \qquad (3.10)$$

Since the coefficients of a Walsh expansion decrease in magnitude by (1/n) as shown by Fine (1955), we can approximate the major structure of any arbitrary function by using a finite expansion of Walsh functions.

Walsh coefficients are a linear transformation of the function sampled at each t_i. We can compute the coefficients by exploiting the orthogonality of Walsh functions.

By writing the function in its finite Walsh expansion, Equation 3.8, then multiplying both sides by WALH(m,t) and integrating over "t", we will project out the C_m coefficient of the Walsh expansion.

$$\frac{1}{2^p} \int_o^1 f(t)\mathrm{WALH}(m,t)dt$$

$$= \sum_{n=0}^{N-1} C_m \frac{1}{2^p} \int_o^1 \mathrm{WALH}(n,t)\mathrm{WALH}(m,t)dt$$

$$= \sum_{n=0}^{N-1} C_n \delta_{n,m} = C_m \qquad (3.11)$$

This procedure for calculation of the N coefficients of the Walsh expansion from the N sampled values of the function requires N^2 multiplications. Using matrix factorization techniques an algorithm may be developed where this transformation only requires Nlog(N) multiplications. This is known as the Fash Walsh Transform (Ahmed and Rao 1975, Andrews and Caspari 1970) (See Appendix 3).

To use Walsh functions in sensitivity analysis we must be able to calculate multidimensional moments, which are averages of the output function over all its parameters. To calculate these moments we must express the

function in terms of its parameters. If we relate each parameter, u_i, to a generating function in t_i,

$$u_i = g_i(t_i) \tag{3.12}$$

we may expand each parameter dimension in a Walsh series. Thereby we obtain a multi-dimensional Walsh series expansion.

For example, let $f(u_1,u_2)$ be an output function with two parameters, u_1 and u_2. If we relate the parameter u_1 to a generating function, $g_1(t_1)$, we may write $f(\underline{u})$ as

$$f(u_1,u_2) = f(g_1(t_1),u_2) = f(t_1,u_2) \tag{3.13}$$

Note again that the same symbol for the function, f, is retained. This will be done throughout this chapter when the meaning is obvious.

This function may be formally expanded in a Walsh series in t_1 with u_2 treated as a parametric constant.

$$f(t_1,u_2) = \sum_j C_j(u_2)\mathrm{WALH}(j,t_1) \tag{3.14}$$

Since the coefficients are now functions of u_2 we may relate u_2 to a generating function in t_2 and expand the coefficients in a Walsh series in t_2.

$$f(t_1,u_2) = f(t_1,g_2(t_2)) = \sum_j \sum_k C_{jk} WALH(j,t_1)WALH(k,t_2) \tag{3.15}$$

This yields a two-dimensional WAlsh series expansion for $f(u_1,u_2)$.

The key to the utility of Walsh functions in sensitivity analysis is the multiplication identify. For an expansion to be efficient, products of the orthogonal basis set must reduce easily. In the case of Walsh functions, the product of two Walsh functions in the same group is a third Walsh function, given by

$$WALH(n,t)WALH(w,t) = WALH(n + w,t) \tag{3.16}$$

where + is binary addition without carry. That is, $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, and $1 + 1 = 0$.

Proof:

$$WALH(n,t)WALH(w,t) = (-1)^{\sum_i n_i t_i} (-1)^{\sum_i w_i t_i}$$

$$= (-1)^{\sum_i (n_i+w_i)t_i} = (-1)^{\sum_i (n_i + w_i)t_i}$$

$$= WALH(n + w,t) \tag{3.17}$$

We can apply this multiplication property to reduce the multidimensional integral for the multidimensional Walsh series coefficients to a one dimensional integral. This will allow us to connect the multidimensional parameter space to a single dimensional line.

In summary, if $f(\underline{u})$ is a multidimensional function in $\underline{u}$ with $\underline{u} = (u_1, u_2, \ldots u_p)$ we may expand the function in a finite multidimensional Walsh series. Each dimension of $f(\underline{u})$ will be related to a dimension t_i which will be expanded in a single dimensional Walsh series. We may write a generalization of Equation 3.15 as a Cartesian product.

$$f(\underline{u}) = f(\underline{t}) = \sum_{w_1=0}^{m_1-1} \sum_{w_2=0}^{m_2-1} \cdots \sum_{w_p=0}^{m_p-1} C_{\underline{w}} \prod_{i=1}^{p} WALH(w_i, t_i) \tag{3.18}$$

The coefficients may be expanded as finite sums

$$C_{\underline{w}} = C_{w_1 w_2 \ldots w_p} \frac{1}{N} \sum_{t_1=0}^{m_1-1} \cdots \sum_{t_p=0}^{m_p-1} f(\underline{t}) \prod_{i=1}^{p} WALH(w_i, t_i) \tag{3.19}$$

$$N = 2^{m_1+m_2-m_p}$$

Let us specialize to the case where each parameter

has only two distinct values. In this case each Walsh sequency expansion of a parameter will consist of only two sequencies. Since there are only two distinct points in each parameter dimension only two samples will be taken from each dimension, i.e., $m_i = 2$. In this case we will need only one binary digit to represent a particular parameter dimension (Kunz 1979).

In the multidimensional parameter space, the function is then defined only on the binary hypercube. Each separate Walsh series requires only a one digit representation. This one digit Walsh function is written

$$\mathrm{WALH}(k,t) = (-1)^{\sum_{i=1}^{p} k_i t_i} = (-1)^{\sum_{i=1}^{1} k_i t_i} = (-1)^{k_1 t_1} \tag{3.20}$$

When the coefficient equation is rewritten with 1-digit Walsh functions the result is

$$C_{\underline{w}} = \frac{1}{2^p} \sum_{t_1=0}^{1} \sum_{t_2=0}^{1} \cdots \sum_{t_p=0}^{1} f(t_1 t_2 - t_p) \prod_{i=1}^{p} (-1)^{w_i t_i} \tag{3.21}$$

By applying lexicographic ordering (Kunz 1979) to w_i and t_i we may define W and T as

$$W = w_p 2^{p-1} + w_{p-1} 2^{p-2} + \ldots + w_1 2^0$$

$$T = t_p 2^{p-1} + t_{p-1} 2^{p-2} + \ldots + t_1 2^0 \qquad (3.22)$$

Therefore, T and W are p-digit binary numbers which range from 0 to 2^{p-1} as t_i and w_i take on their allowed values of 0 and 1.

Upon substitution of W and T the coefficient equation becomes

$$C_W = \frac{1}{N} \sum_{T=0}^{N-1} f(T)(-1)^{\sum_{i=1}^{p} w_i t_i}; \quad N = 2^p \qquad (3.23)$$

Note that the sum over T encounters all of the two-term sums in Equation 3.21.

We may now associate the finite multidimensional expansion with a finite single dimensional expansion.

$$f(T) = \sum_{W=0}^{N-1} C_W \, WALH(W,T) \qquad (3.24)$$

From Equation 3.24 we may derive the required

relationship for varying the parameters. Each parameter is associated with a separate dimension. Each dimension may be expanded in a Walsh sequency expansion as in Equation 3.18. Since these sequencies belong to different parameter dimensions we have the requirement that the binary sum of the sequencies be unique for any combination of sequencies.

$$\pi_i \mathrm{WALH}(w_i,t) = \mathrm{WALH}(w_1+w_2+\ldots+w_p, t_1+t_2+t_3+\ldots+t_p)$$

$$= \mathrm{WALH}(W,T) \qquad (3.25)$$

This means that the binary sum of the w_2's must never add to the same W-value for different w_i's. Analogous to the Fourier method, this restriction is called "binary incommensurate". Note that this procedure involves the conversion of p one-digit Walsh functions to one p-digit Walsh function.

Since, for exact analysis, we must sample from each dimension so that we never repeat the search curve, we then must assign a unique sequency to each parameter where the sequencies are binary incommensurate. The simplest set of such sequencies is the set of powers of two,

$$2^0, 2^1, 2^2, \text{ etc.} \qquad (3.26)$$

In this way, for a model with 'p' parameters, parameter u_1 would be associated with the least significant binary digit in a p-digit number, u_2 with the next most significant digit, etc.

Each parameter can take on two values given by the generating function $g_i(t_i)$,

$$g_i(t_i) = u_i = u_i^0 + \Delta WALH(w^{i-1}, t_i 2^{i-1}) \qquad (3.27)$$

where u_i^0 is the average value of u_i and Δ determines the range. With this generating function, when t_i takes on its values of 0 and 1, u_i oscillates between $u_i^0 + \Delta$ and $u_i^0 - \Delta$ at the 2^{i-1} sequency.

By defining the parameters of a model to be functions of different sequency Walsh functions, as in the Fourier method, we can expand the model output function in an infinite Walsh series. By truncating the expansion to a finite Walsh series we incur no error. This can be illustrated by a two dimensional example. If we set u_1 = WALH(m,t) and u_2 = WALH(k,t), we may then write

$$f(u_1,u_2) = f(WALH(m,t), WALH(k,t)) \qquad (3.28)$$

By expanding Equation 3.28 in a two dimensional Walsh series, we will obtain a finite set of terms. Using the multiplication identity we can see that there are only four possible terms, a 0-sequency term, a k-sequency term, a m-sequency term, and a (k + m)-sequency term. No other terms are possible regardless of the nature of the function f.

In the calculation of the Walsh coefficients with the Fast Walsh Transform, we use 2^p equally spaced samples. This enables us to calculate 2^p different sequency. Coefficients, $C_0, C_1, \ldots, C_{n-1}$. Therefore, we will have, as a subset of these coefficients, all four of the required sequencies. Of the 2^p computed coefficients only these four will be non-zero. In summary, for any Walsh driven function, there will be no error incurred in approximating the function by using finite Walsh expansions.

From Equation 3.24 we can derive the total variance of the model output function (See Appendix 4). The variance is

$$\sigma_T^2 = \sum_{i=1}^{N-1} C_i^2 \qquad (3.29)$$

This is the same formula as in the Fourior method. In an analogous manner, the single parameter variance is

constructed by calculation of the variance with respect to only one parameter, say the first, using the model output function which has been averaged over all the other parameters (See Appendix 5).

$$\sigma_1^2 = C_{w_1}^2 \tag{3.30}$$

Nowever, in the Walsh case we get only one term! There is no infinite series to truncate as there is in the Fourier method. In the Walsh expansion the reduced partial variance is given by

$$S_1 = \frac{\sigma_1^2}{\sigma_T^2} = \frac{C_{w_1}^2}{\sum_{i=1}^{N-1} C_i^2} \tag{3.31}$$

which is <u>exact</u>.

In a similar vein we can construct coupled partial variances. The coupled partial variances are the Walsh coefficients whose sequency is that of the desired single sequencies added together by binary addition without carry (See Appendix 6). Then we divide by the total variance to get the reduced coupled partial variances.

$$S_{\ell,k} = \frac{C^2_{w_\ell + w_k}}{\sum_{i=1}^{N-1} C_i^2} \tag{3.32}$$

For the partial variances to be rigorously correct, i.e., to contain no error terms, we must sample the entire parameter space. This will only be true if we are using a discrete model whose parameters can only take on two values.

When Walsh Sensitivity Analysis is applied to a continuous model, an approximation is made. This a-proximation is that the influence on the output function of the range of parameter variation in a continuous model may be approximated by using only two values of a parameter chosen from a continuous range of possible values. A good choice, when comparing the Walsh method with a continuous method, would be the extremes of the interval over which the parameter is varied in the continuous analysis.

To illustrate this we will examine a one-parameter model. First we write down the one dimensional Walsh expansion in terms of the parameter, u_1, where u_1 = WALH(1,t).

$$ul \rightarrow u_1$$

$$f(u_1) = \sum_{w=0}^{1} C_w (-1)^{wt} \tag{3.33}$$

From this expansion we can calculate two terms. The C_0 term and the C_1 term. The C_0 term is the sum of all the sampled function values

$$C_0 = \frac{1}{2} (f(u_1=1) + f(u_1=-1)) \tag{3.34}$$

This may be interpreted as the average value of the function at $f(u_1=0)$, where we note that when t=0, f(WALH(1,0)) = f(1) and when t = 1, f(WALH(1,1)) = f(-1).

To calculate C_1 we subtract the two function values:

$$C_1 = \frac{1}{2} (f(1) - f(-1)) \tag{3.35}$$

If we were to do a Walsh sensitivity analysis on a one parameter model, the minimal sequency with which to vary that parameter is $w_1 = 1$. So the C_1 coefficient would be proportional to the "sensitivity" of the model to its parameter. In linear analysis, if we used a central difference formula for the first derivative of the function with respect to a parameter, we would get the same equation for the sensitivity of the parameter.

With a two dimensional model the possible Walsh

coefficients are C_0, C_1, C_2, C_3. We can write out the Walsh transform, from Equation 3.23, for the Walsh coefficients as

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \frac{1}{2^2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{pmatrix} \tag{3.36}$$

The sampled parameter sets can be plotted in the two-dimensional parameter space (Figure 2). This identifies the sampled function values in the parameter space.

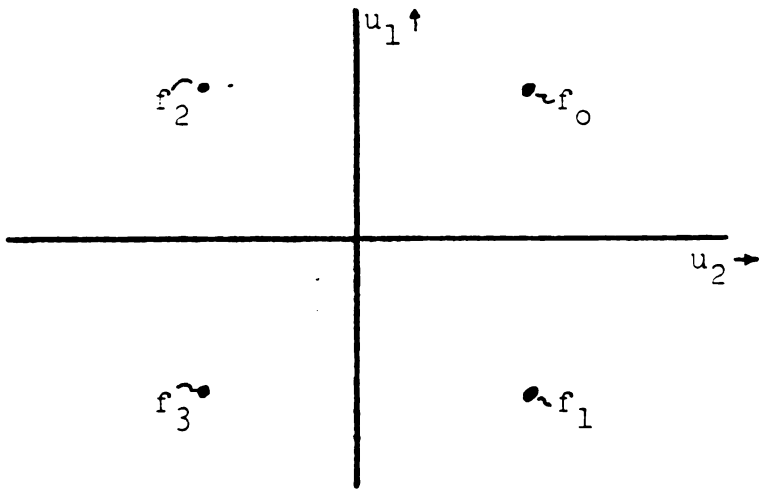


Figure 2. Plot of the sampling points in the multidimensional u-space.

The equation for C_1 may be rewritten as

$$C_1 = \frac{1}{2}\left(\frac{f_0-f_1}{2} + \frac{f_2-f_3}{2}\right) \tag{3.37}$$

which is the average of the two central difference approximations to the first derivative with respect to the first parameter (See Figure 2).

Similarly, for C_2

$$C_2 = \frac{1}{2}\left(\frac{f_0-f_2}{2} + \frac{f_1-f_3}{2}\right) \tag{3.38}$$

which is the average of the two central difference approximations of the first derivative with respect to the second parameter.

Finally, the C_3 coefficient is exactly as expected, a central difference approximation to $\Delta^2 f/\Delta u_1 \Delta u_2$.

$$C_3 = \frac{1}{4}\left(f_0-f_1-f_2+f_3\right) \tag{3.39}$$

If we examine the general expression for the Walsh coefficient of the kth parameter, w^{k-1}, we note that for

an P-dimensional system the coefficient is always the central difference approximation average to the derivative with respect to the kth dimension (without loss of generality we may set k = 1).

$$C_{2^{k-1}} = C_{w_1 \ldots w_p} = \frac{1}{2^p}\Big[\sum_{t_1=0}^{1} \ldots \sum_{t_p=0}^{1} f(t_1 t_2 \ldots t_p)(-1)^{\sum_{i=1}^{p} w_i t_i}\Big]$$

$$C_{10\ldots0} = \frac{1}{2^{p-1}} \sum_{t_2=0}^{1} \ldots \sum_{t_p=0}^{1} \left\{\frac{f(0t_2\ldots t_p)-f(1t_2\ldots t_p)}{2}\right\}$$

$$= \left\langle \frac{\Delta f}{\Delta u_i} \right\rangle \qquad (3.40)$$

Similarly for all coupled Walsh coefficients ($2^{k-1} + 2^{\ell-1}$-sequency) we get the approximation to the average of the mixed derivative (See Abramowitz and Stegun pp-884).

$$C_{110\ldots0} = \frac{1}{2^p} \sum_{t_1=0}^{1} \ldots \sum_{t_p=0}^{1} f(t_1 \ldots t_p)\{(-1)^{t_1+t_2}\}$$

$$= \frac{1}{2^{p-2}} \sum_{t_3=0}^{1} \ldots \sum_{t_p=0}^{1} \left\{\frac{f(00t_3..t_p)-f(01t_3..t_p)-f(10t_3..t_p)+f(11t_3..t_p)}{4}\right\}$$

$$= \left\langle \frac{\Delta^2 f}{\Delta u_1 \Delta u_2} \right\rangle \qquad (3.41)$$

This implies that a Walsh function expansion to a continuous function is the first 2^p terms of a Taylor series using average derivatives.

$$f(\underline{u}) = f(\underline{t})$$

$$= \langle f \rangle + \sum_{i=1}^{p} \left\langle \frac{\Delta f}{\Delta u_i} \right\rangle (-1)^{t_i} + \sum_{i=1}^{p} \sum_{j}^{p} \left\langle \frac{\Delta^2 f}{\Delta u_i \Delta u_j} \right\rangle (-1)^{t_i + t_j}$$

$$+ \sum_{i}^{p} \sum_{j<k}^{p} \sum_{k}^{p} \frac{\Delta^3 f}{\Delta u_i \Delta u_j \Delta u_k} (-1)^{t_i + t_j + t_k} + \ldots \qquad (3.42)$$

with the restriction that each term be at most of degree one in any parameter.

IV. EXAMPLES IN SENSITIVITY ANALYSIS

This chapter will consider the application of the different types of sensitivity analysis to some simple models. To understand these models does not require rigorous sensitivity analysis. In fact, many of the results are intuitively obvious. However, the application of the different sensitivity analysis techniques to these models will help to distinguish the domains of applicability of these techniques. The analyses will also assist in the interpretation of the results of sensitivity analyses of more complex models.

The simplest mathematical basis for sensitivity analysis (Beck 1977) is a Taylor series of the model output function in terms of the parameters of the function. Exapnding the output function around a nominal value, $\underline{k}^0$, we can write the Taylor series as

$$f(\underline{k}^0+\Delta) = f(\underline{k}^0) + f'(\underline{k}^0)\Delta + \frac{1}{2!} f''(\underline{k}^0)\Delta^2 + \ldots \tag{4.1}$$

'Classical' sensitivity analysis is concerned with the first derivative term in this expansion, usually

multiplied by a scaling factor to remove the dimension of the parameter. The sensitivity coefficient of k_1, X_{k_1}, is written,

$$X_{k_1} = k_1^0 \left(\frac{\partial f(k)}{\partial k_1}\right)\Bigg|_{\underline{k}=\underline{k}^0} \qquad (4.2)$$

With such first derivatives, classical sensitivity analysis attempts to explain the effects on the model function from changing the parameters' values. In order for this to work, the higher order terms must be small with respect to the first derivative term, which can be guaranteed if $(k_i - k_i^0) << k_i^0$. In this case all higher order terms are multiplied by a number close to zero.

The requirement that higher order terms be small restricts the domain of classical sensitivity analysis to regions localized around the nominal value, $\underline{k}^0$. However, <u>if</u> the function is linear in the parameters all higher derivatives in the expansion <u>are</u> zero. This special case has been developed into a widely-used practical method (Beck 1972). From this viewpoint of ranges of parameter variations, the different sensitivity analysis techniques may be segregated. This can be seen by examining models where different ranges of parameter variation are used.

The first model is a straight line with two parameters,

the slope, m, and the intercept, b.

$$y = mt + b \tag{4.3}$$

This model has a Taylor series expansion

$$y = f(m,t,b) = f(m_0,t,b_0) + t(m-m_0) + b - b_0$$

$$= mt + b \tag{4.4}$$

As expected, the expansion is exact for the linear problem. The scaled sensitivity coefficients are

$$X_m = tm_0; \qquad X_b = b_0 \tag{4.5}$$

A sensitivity coefficient is large when a change in the parameter changes the value of the output function to a large degree. In this case the value of the output function depends critically on the value of the parameter. In the linear model X_m is large when t is large. Hence,

for accurate 'm' estimation, measurements should be taken at large values of t where the output function is most 'sensitive' to the value of 'm'. The b-sensitivity coefficient shows that measurements at t = 0 or small t values, where sensitivity to m is small, will allow an accurate estimation of b. These coefficients are plotted in Figure 4.1. This confirms previous knowledge (Acton 1966).

Applying Walsh sensitivity analysis to the linear model results in the same sensitivity coefficients as the Taylor series approach. This will happen since the average finite-difference expansion calculates exact derivatives in the linear case (Lanczos 1955).

A Walsh sensitivity analysis of the linear model requires a set of nominal values, and a range of variation for the parameters. The nominal values m = 0.0, b = 0.0 along with a range of variation of ±10 for each parameter were chosen. In order to vary two parameters over this range, four simulations were required ($N = 2^p = 2^2$).

Figure 4.2 is a plot of the average value of y, averaged over the four simulations. This plot shows 'typical' values of the output function over the selected parameter space. Note that for this simple case the average and nominal values are the same. Figure 4.3 shows the standard deviation of the four simulations (square root of the total variance) from the average value. If the

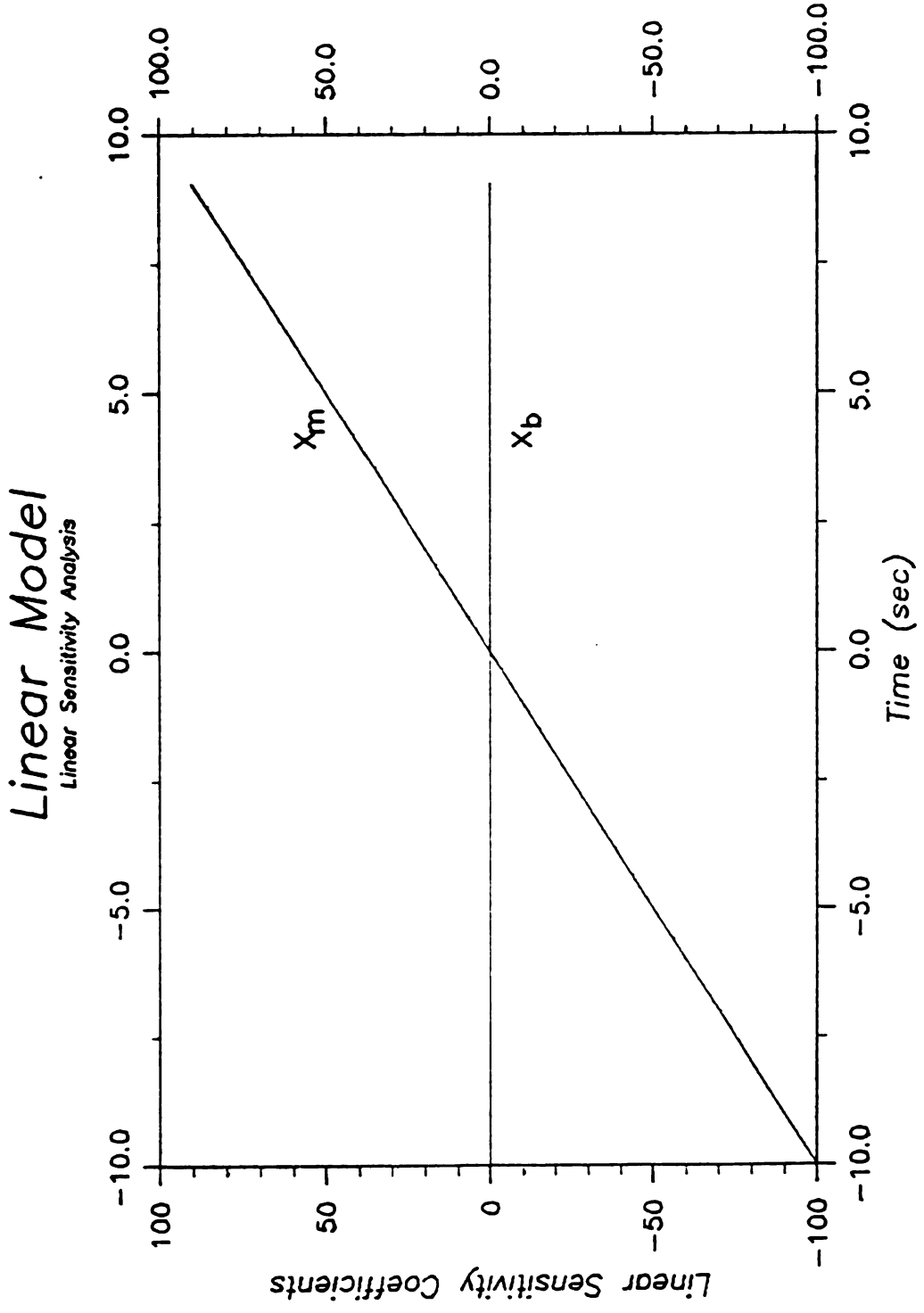


Figure 4.1 The linear sensitivity coefficients from the Linear Model.

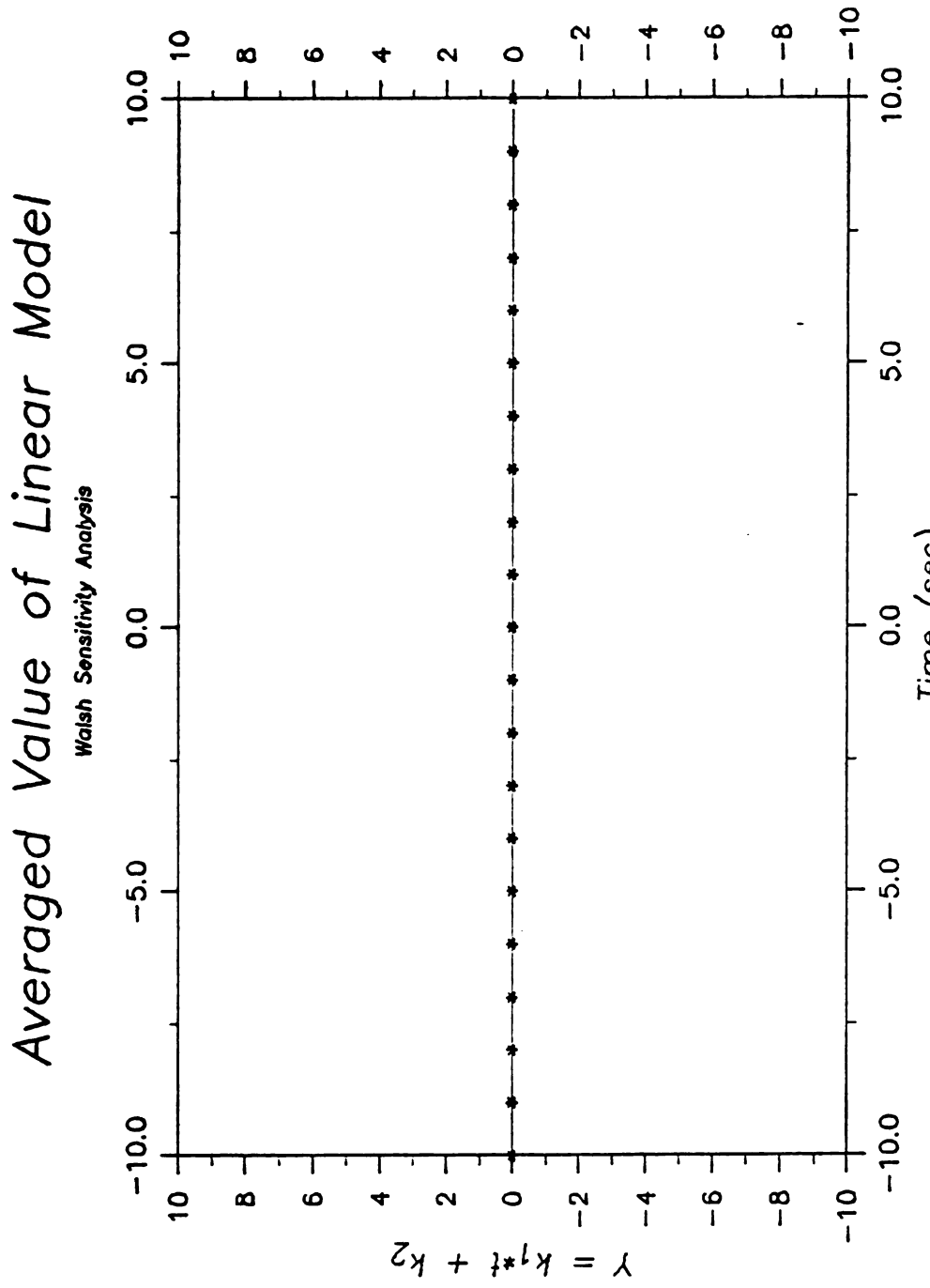


Figure 4.2 The averaged value of the Linear Model from a Walsh Sensitivity Analysis.

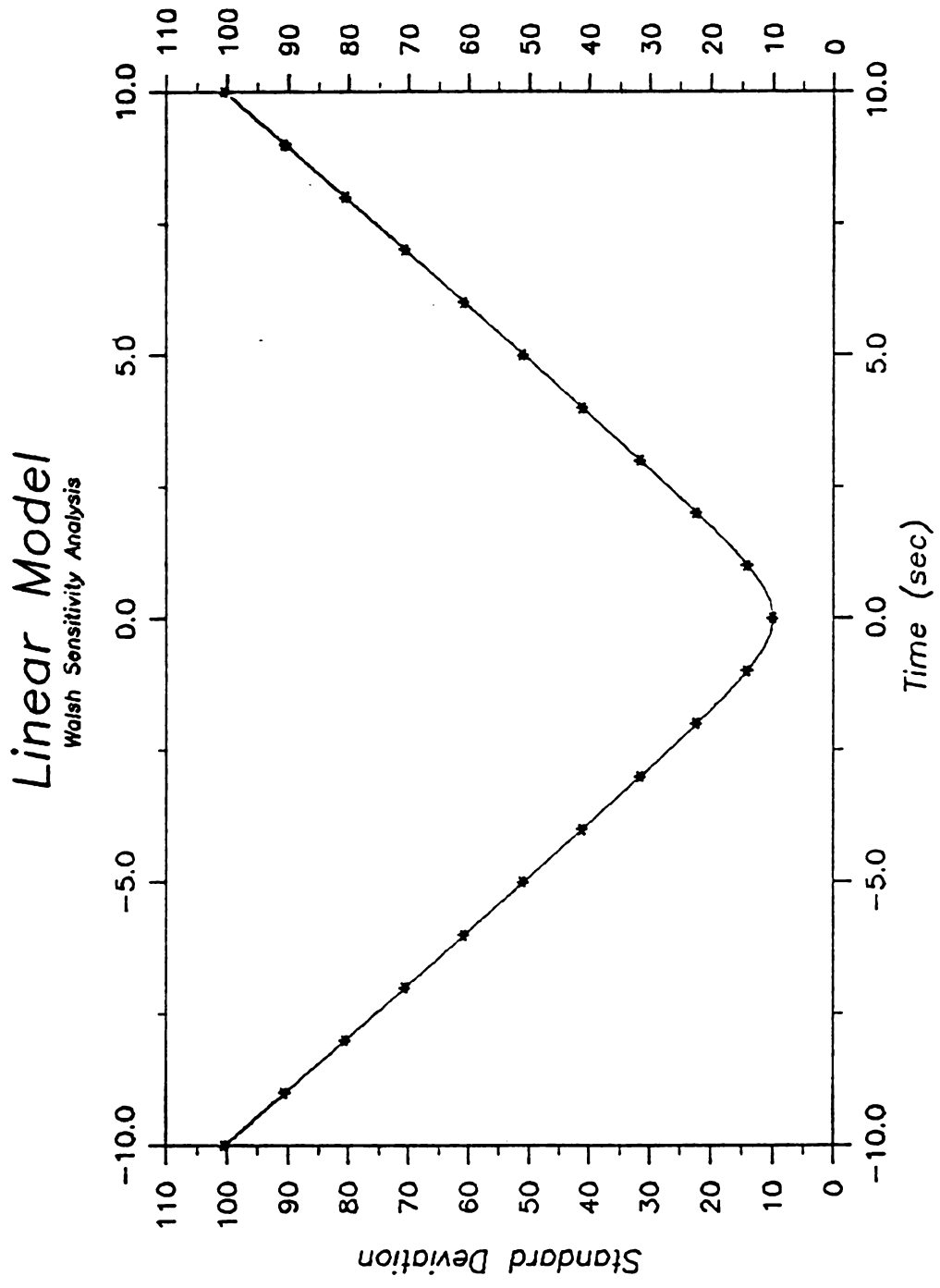


Figure 4.3 The standard deviation of the simulations sampled in the Walsh Analysis of the Linear Model.

standard deviation is small then the range of parameter space examined has little effect on the output function.

Figure 4.4 is a plot of the Walsh expansion coefficients. These coefficients are related to the linear sensitivity coefficients shown in Figure 4.1. A Walsh expansion coefficient may be thought of as an averaged derivative of the model output function as shown in Chapter Three. However, the equation depends on the particular transformation function used in the analysis. In Chapter Three the transformation function $u_\ell = WALH(2^{\ell-1},T_\ell)$ was used. In this case, by using the chain rule, it can be shown that $du = dt$. Similarly, the conversion of u_ℓ to t_ℓ must be accounted for when a different transformation function is used.

In the linear model the transformation function used was the arithematic transformation function $u_\ell = u_\ell^0 + \Delta WALH(2^{\ell-1},t_\ell)$. Applying the chain rule to the equation for a Walsh expansion coefficient, 3.40, results in

$$C_{2^{\ell-1}} = \left\langle \frac{\partial f(u_\ell)}{\partial u_\ell} \cdot \frac{\partial u_\ell}{\partial t_\ell} \right\rangle \qquad (4.6)$$

which in the case of the arithematic Walsh transformation function yields

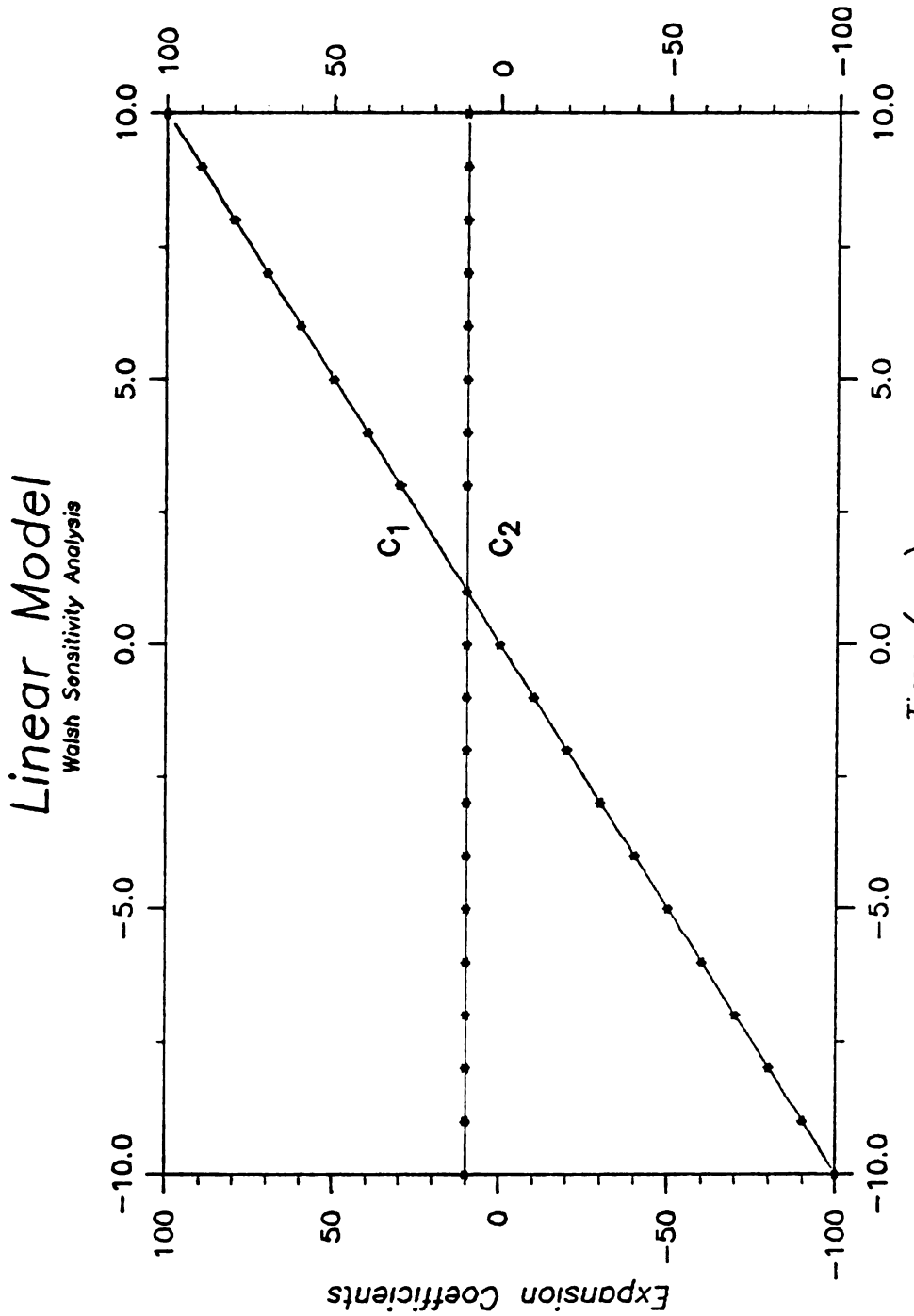


Figure 4.4 The Walsh expansion coefficients from the Linear Model.

$$\frac{\partial}{\partial t_\ell}\{u_\ell = u_\ell^0 + \Delta WALH(2^{\ell-1}, t_\ell)\}$$

$$\frac{\partial u_\ell}{\partial t_\ell} = \Delta \tag{4.7}$$

therefore

$$C_{2^{\ell-1}} = \Delta\left\langle \frac{\partial f(u_\ell)}{\partial u_\ell} \right\rangle \tag{4.8}$$

From this equation the relationship to the linear sensitivity coefficient is easily shown.

$$X_{u_\ell} = u_\ell^0 \left(\frac{\partial f(u_\ell)}{\partial u_\ell}\right) = \left(\frac{u_\ell^0}{\Delta}\right) C_{2^{\ell-1}} \tag{4.9}$$

Therefore the Walsh expansion coefficients, using an arithmetic transformation function, are equal to linear sensitivity coefficients scaled by a scale factor which is the nominal value, u_ℓ^0, divided by the range of parameter variation, Δ. In the linear model the scale factors for the parameters are (m^0/Δ_m) and $b^0/\Delta_b)$ which are 1 and 0,

respectively.

Alternatively the reduced partial variances may be plotted, (henceforth the reduced partial variances will be called partial variances with the assumption that they are divided by the total variance). These partial variances are shown in Figure 4.5. Using the notation developed in Chapter Two, the partial variance of the first parameter, m, is written S_1. Similarly the partial variance of the second parameter, b, is written S_2. The point at which the curves intersect, here at $t = \pm 1$, depends on the range of parameter variation and the nominal values chosen for the Walsh Sensitivity analysis. Here both parameters were varied over [-10.,10.].

In the linear model the interpretation is simple. To estimate the b-parameter, measurements should be taken near $t = 0$ where S_2 is largest, near $t = 0$. Measurements far from $t = 0$ are highly sensitive to the value of m as shown by S_1. These measurements taken from this region would be best for accurate estimation of m.

Treating the linear model with the Fourier method gives similar results. The nominal values and parameter variations were the same as in the Walsh analysis. However, more simulations were required to estimate the Fourier coefficients. The frequency set used was the 6th-order accurate set, [3,5], which requires, at a minimum, 11 simulations (21 simulations were used). The expansion

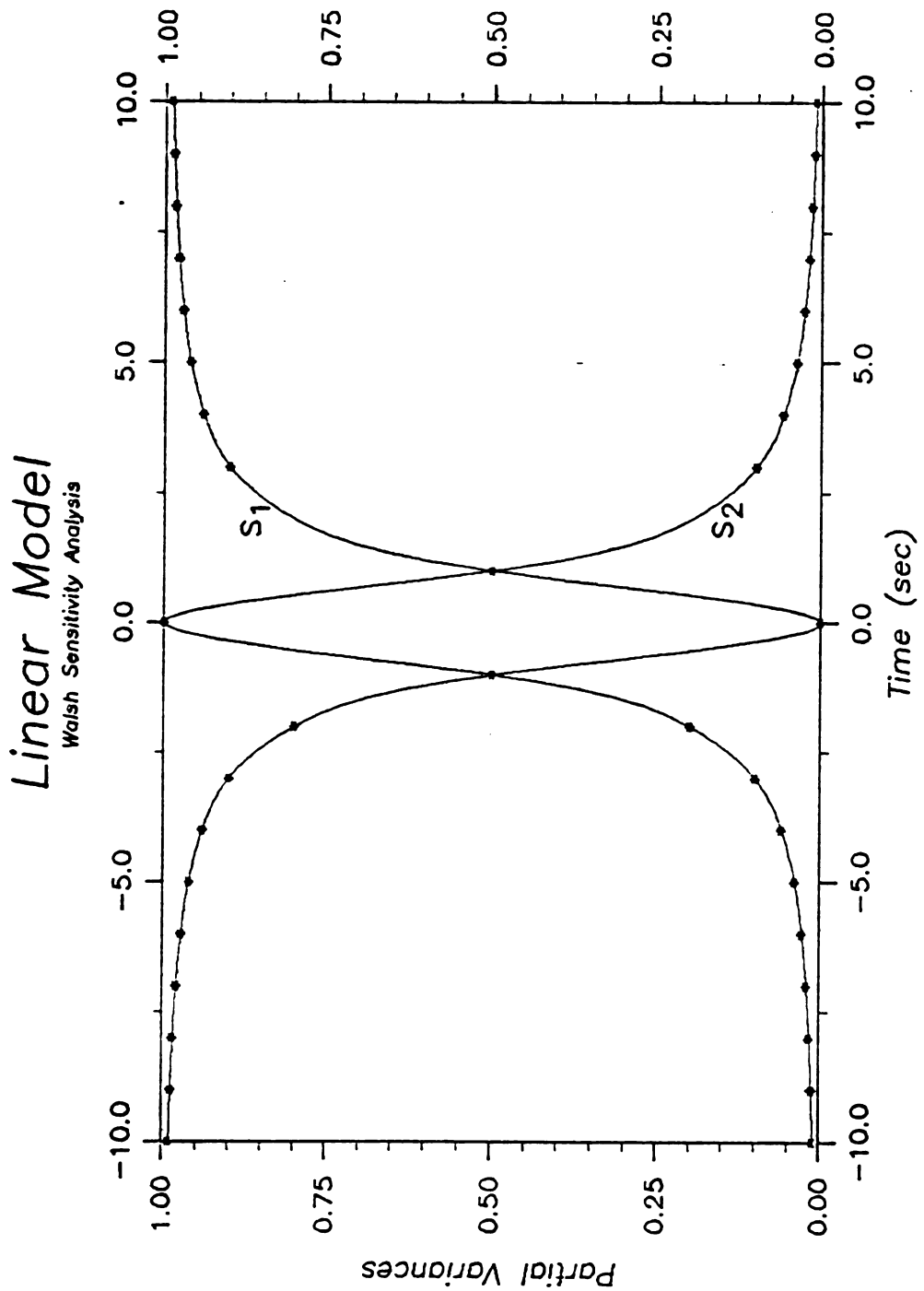


Figure 4.5 Walsh partial variances from the Linear Model.

coefficients versus t are plotted in Figure 4.6. These expansion coefficients are proportional to the first derivatives in the Taylor series.

The partial variances shown in Figure 4.7 are nearly identical to those derived from Walsh analysis, Figure 4.5. The minor differences are a result of the approximate nature of the Fourier method, since a Fourier expansion of an angled line requires an infinite number of terms. However, the partial variances shown capture more than 99% of the total variance.

The standard deviation curve given in Figure 4.8 also shows nearly the same range of variation in the output function as was examined by the Walsh method. However, the standard deviation curve weights simulations far from the average simulation more than those close to the average, causing the Fourier standard deviation curve to be smaller in magnitude than its Walsh counterpart. This happens as the parameter vectors are chosen throughout the parameter variation interval in the Fourier analysis while the parameter vectors are chosen at the extremes in the Walsh method.

Now let us examine a simple nonlinear model. Here nonlinear means that the Taylor series expansion of the output function with respect to the parameters is composed of terms containing second or higher derivatives of the output function. Perhaps the most commonly used nonlinear

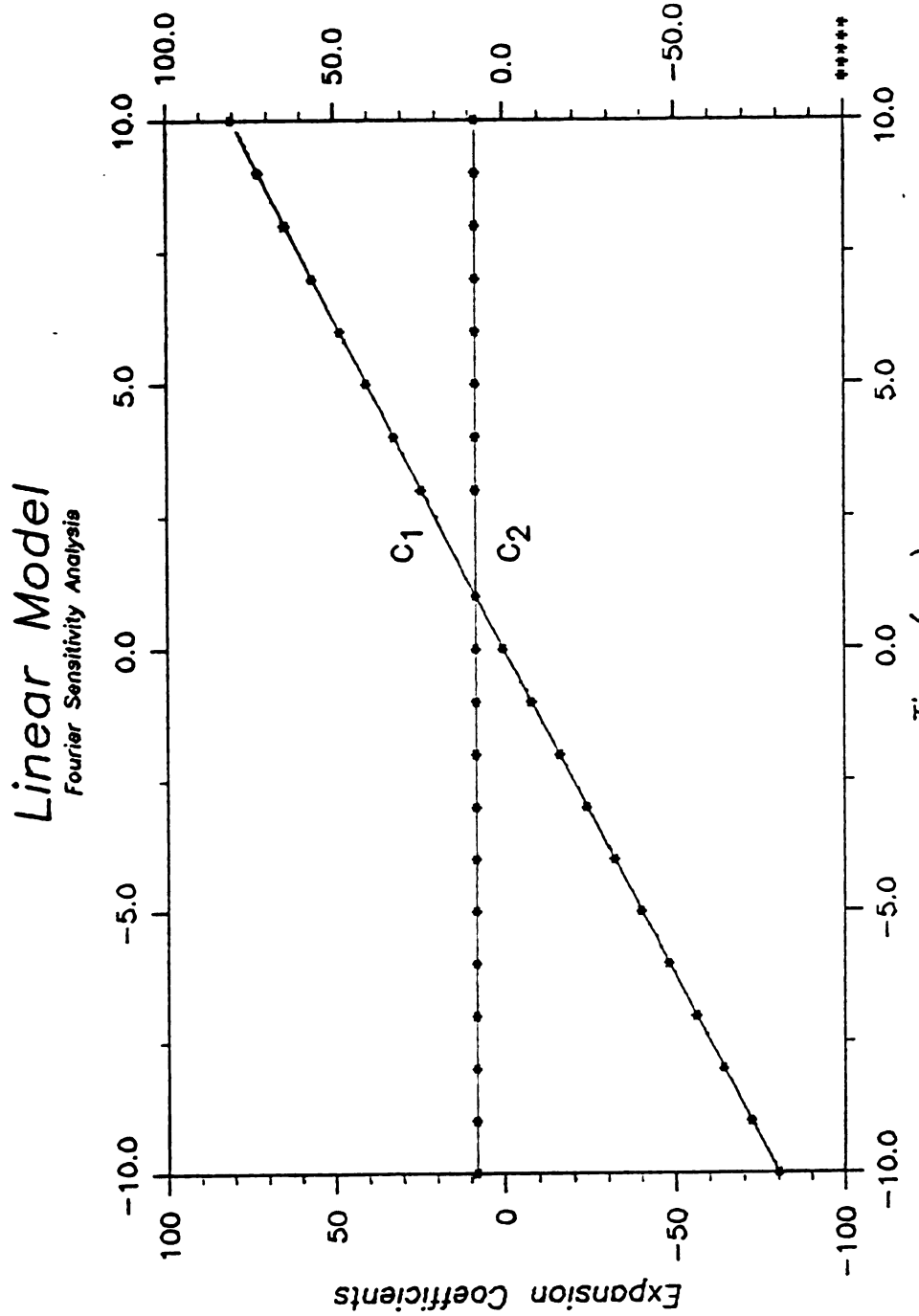


Figure 4.6 Fourier expansion coefficients from the Linear Model.

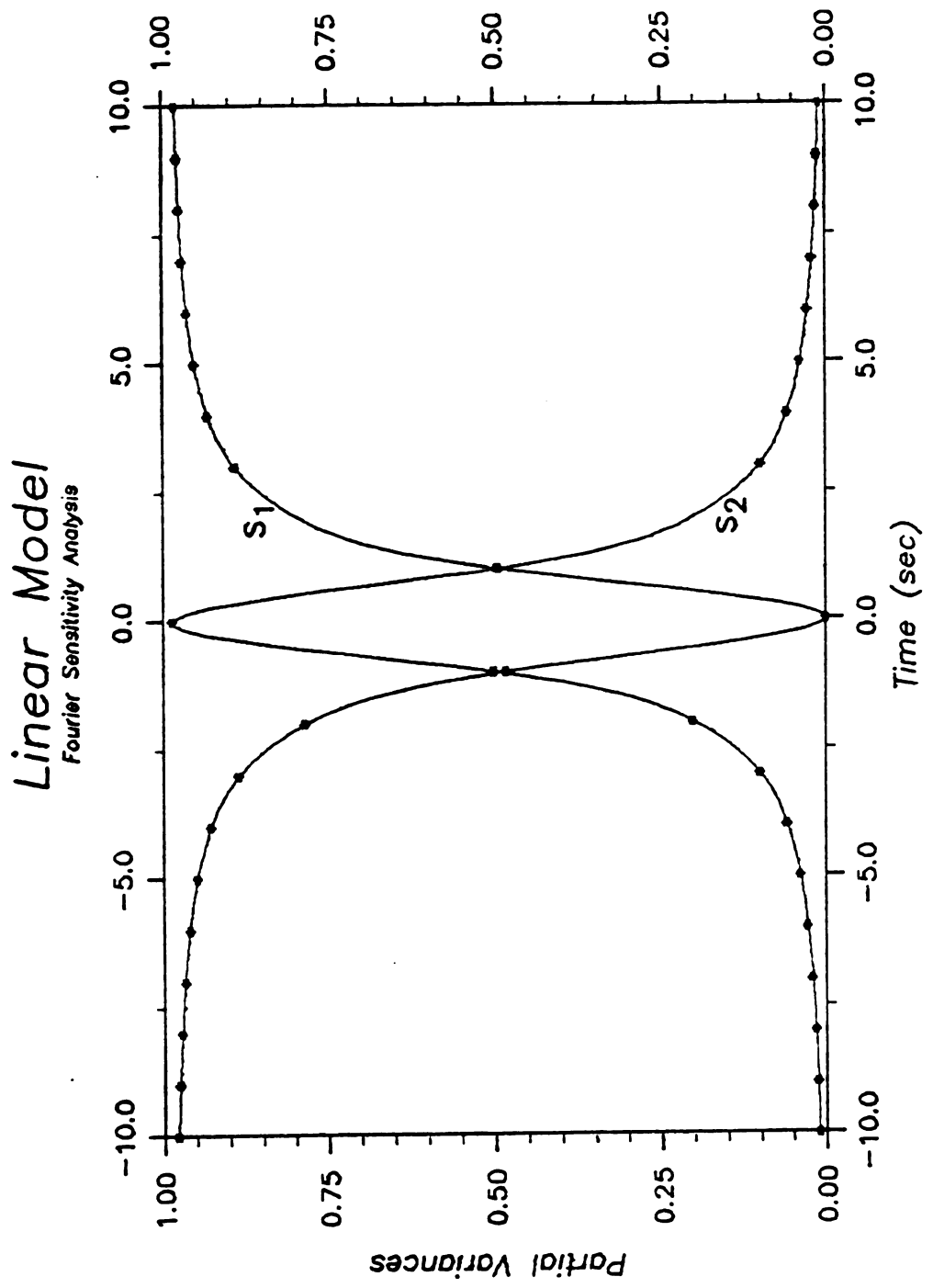


Figure 4.7 Fourier partial variances from the Linear Model.

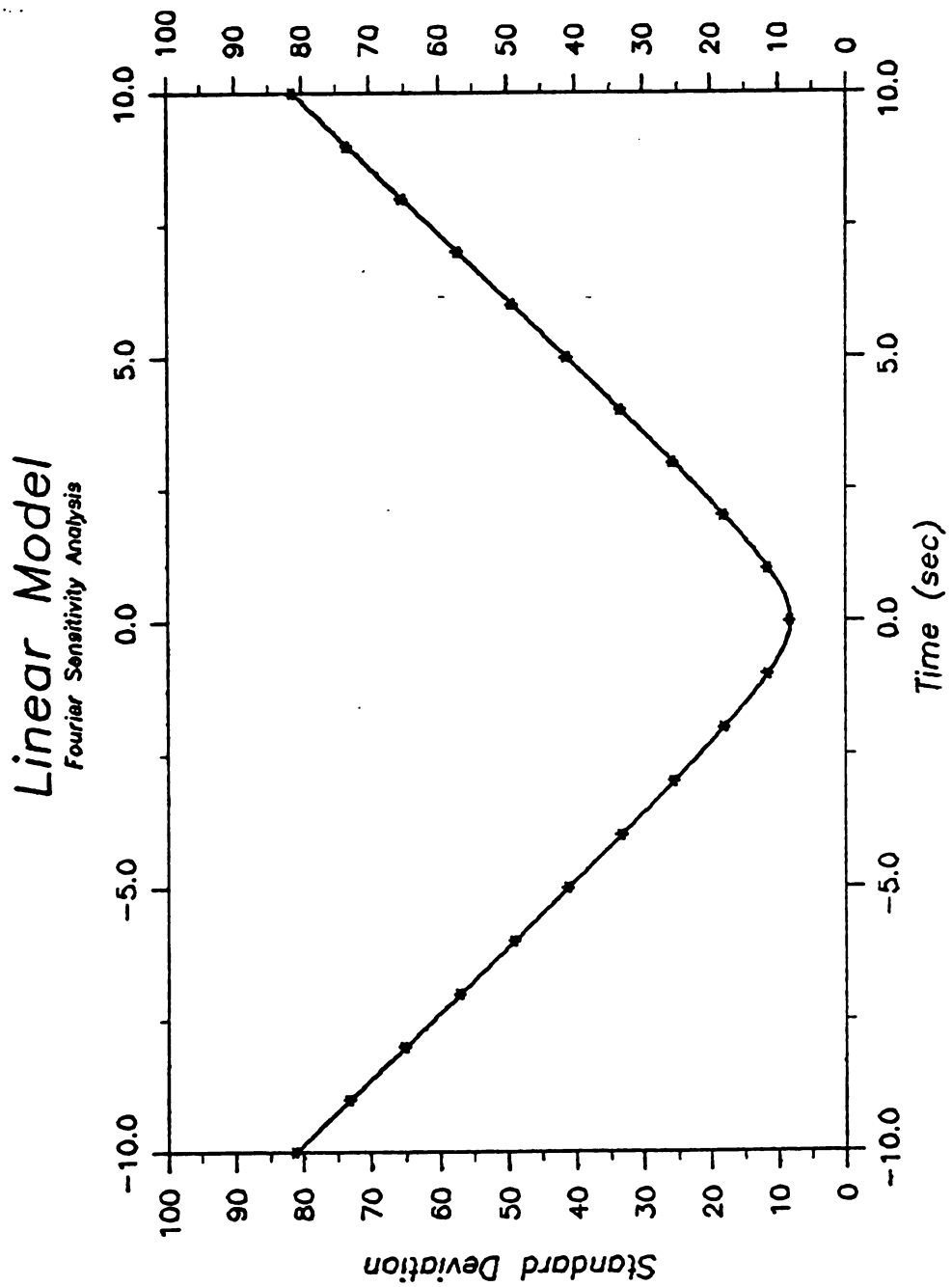


Figure 4.8 The standard deviation of the sampled simulations from the Linear Model using Fourier Analysis.

model in chemistry is the single exponential.

$$f = k_2 e^{k_1 t} \tag{4.10}$$

This function can be expanded in its Taylor series as

$$f(k_1,k_2,t) = f(k_1^0,k_2^0,t) + e^{k_1^0 t}(k_2-k_2^0) + k_2^0 t e^{k_1^0 t}(k_1-k_1^0)$$

$$+ t e^{k_1^0 t}(k_2-k_2^0)(k_1-k_1^0)$$

$$+ k_2^0 t^2 e^{k_1^0 t}(k_1-k_1^0)^2 + \ldots \tag{4.11}$$

Although the expansion continues for an infinite number of terms, a linear sensitivity analysis would only examine the first derivative terms.

$$X_{k_2} = k_2^0 e^{k_1^0 t}; \quad X_{k_1} = k_1^0 k_2^0 t e^{k_1^0 t} \tag{4.12}$$

These sensitivity coefficients are plotted in Figure 4.9 using k = -0.25 seconds and k = 1000.0 as the nominal values.

From this sensitivity analysis we can say that the best measurements for k_2 are at small t, and the best measurements for k_1 are at t = 4 seconds, the maximum of the curve. However, if higher order terms are considered note that they may be large and could affect the value of the output function.

To use Walsh sensitivity analysis on this model a range of parameter variation is needed. First, let us examine 'local behavior'; behavior of the model when the parameters are varied only slightly. In this case, for small variations in the parameters, the Walsh coefficients should be equivalent to the results of classical sensitivity analysis. But for large variations in the value of the parameters the Walsh method will give different results as the higher derivatives become significant.

Again a parameter set must be chosen. Figure 4.10 shows the plot of the averaged value for the four simulations of the exponential model with $k_2 = 1000 \pm 100$ and $k_1 = -0.25 \pm 0.025$ seconds, i.e., 10% variation. Since there are two parameters in this model, the curve in Figure 4.10 is the average of four different simulations where each simulation has a unique combination of parameters.

The expansion coefficients are plotted in Figure 4.11.

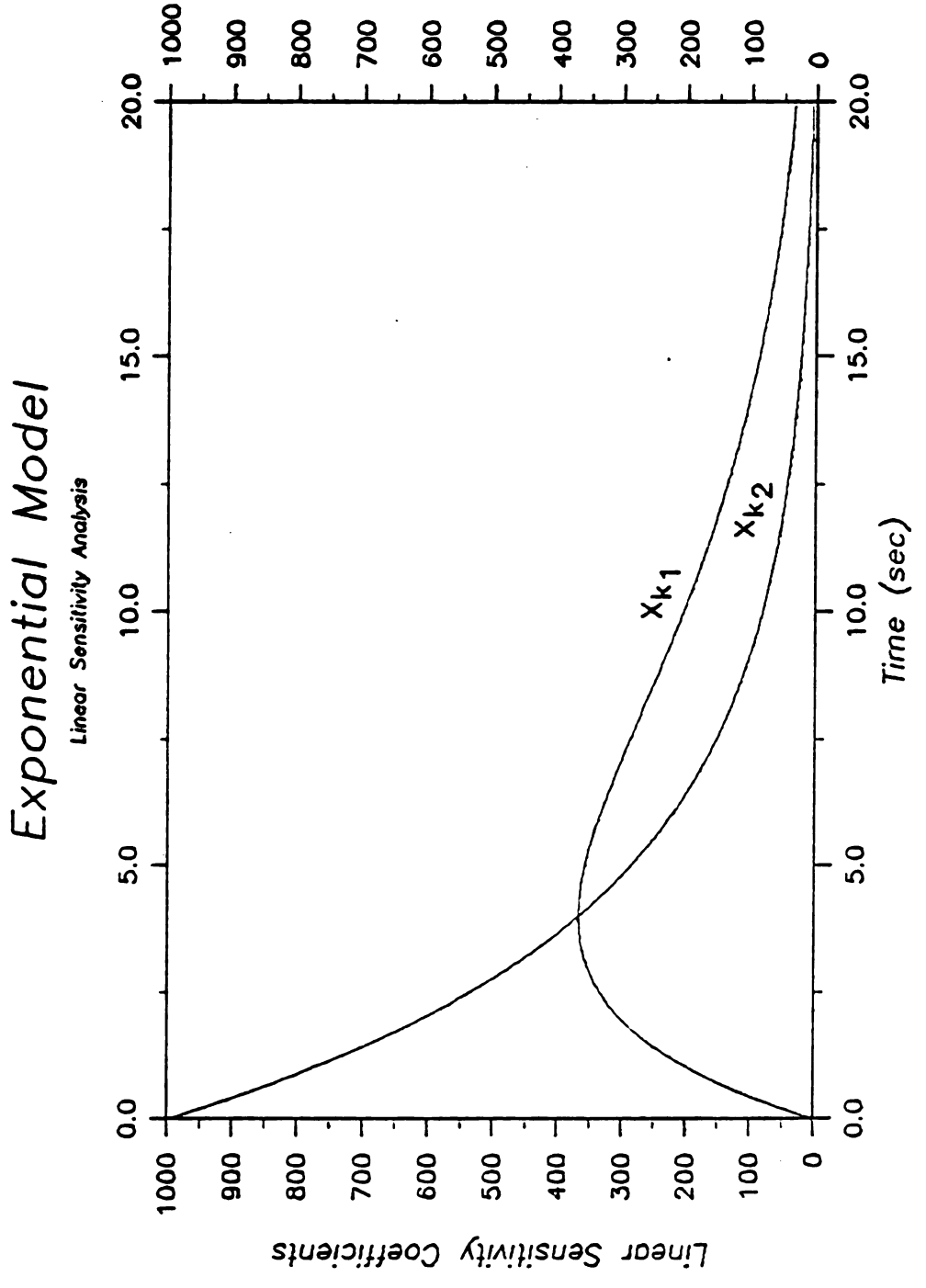


Figure 4.9 Linear sensitivity coefficients from the Exponential Model.

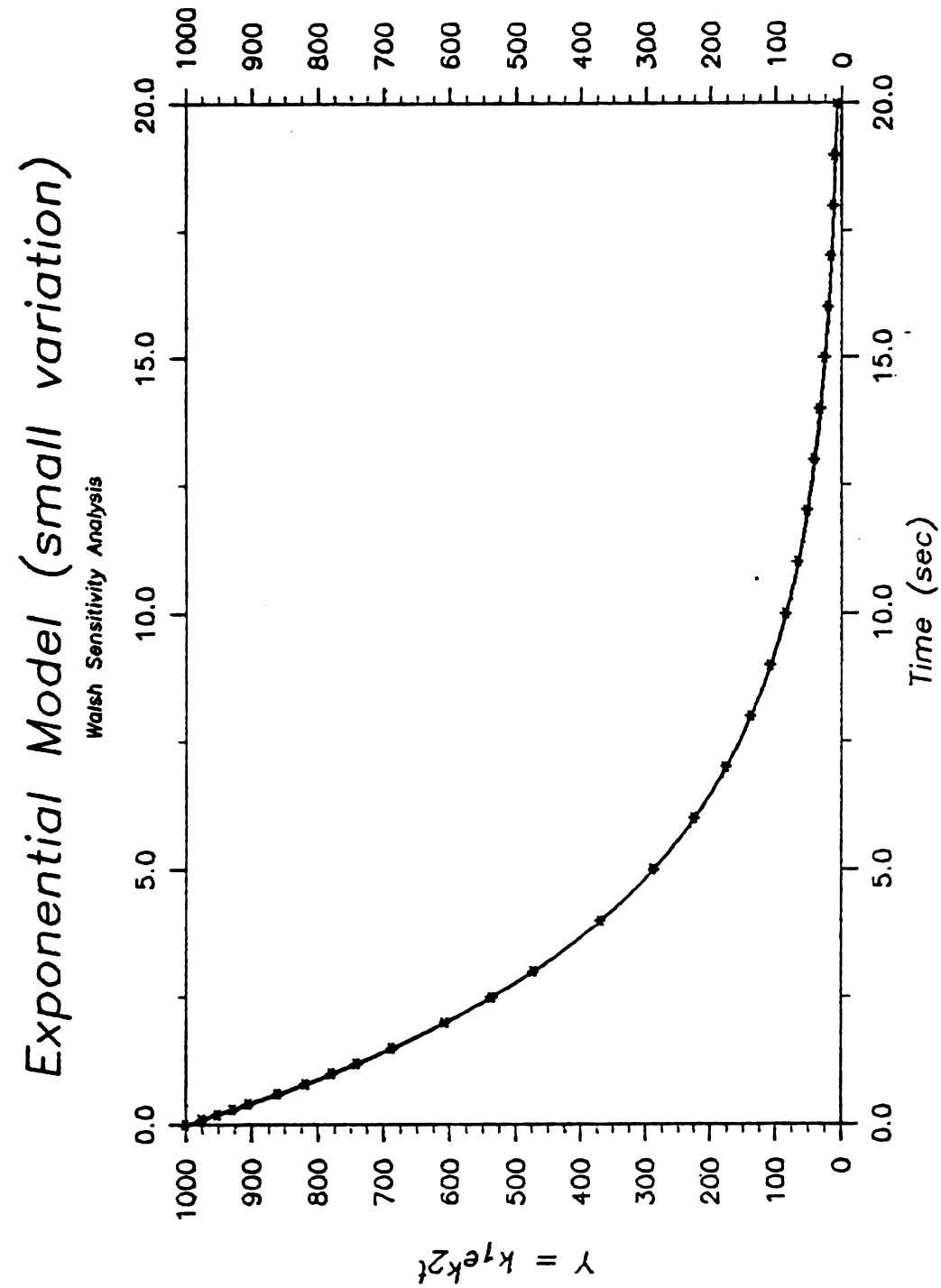


Figure 4.10 The averaged value of the Exponential Model using Walsh Sensitivity Analysis with 10% variation in the parameters.

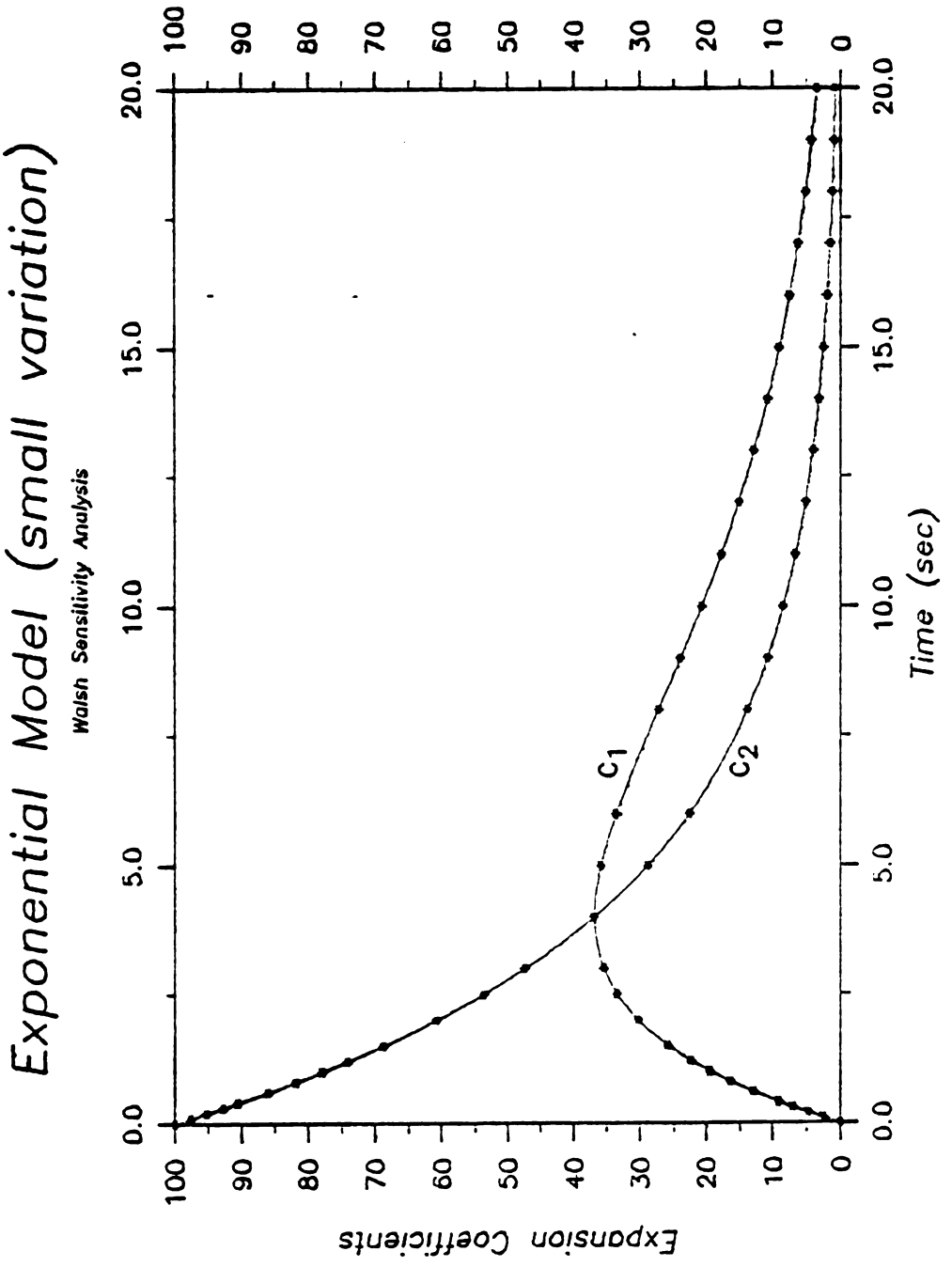


Figure 4.11 Walsh expansion coefficients from the Exponential Model (10% variation).

The C_1 coefficient is that Walsh expansion coefficient which, in the case of a continuous model, is an averaged finite-difference measure of the first derivative of the output function with respect to the first parameter, k_1. Similarly, the C_2 coefficient is a finite-difference approximation to the first derivative of the output function with respect to the second parameter, k_2. Comparing Figure 4.11 with Figure 4.9 we see that they are identical curves to within a constant scaling factor.

To display the sensitivities of the parameters it is more convenient to examine the partial variances shown in Figure 4.12. As before, these plots also show the most sensitivity to the second parameter at short times, and to the first parameter at long times. This figure also shows that there is very little coupling in the sensitivity between the two parameters. This can be observed by noting that the sum of the two partial variances (S_1 + S_2) is nearly 1.0. This means that almost all of the variance in the output function is assigned to S_1 or S_2.

The standard deviation curve for this analysis is given in Figure 4.13. This curve, which looks like an exponential decay, reflects the decay of the output function. It is tempting to claim that since the standard deviation is only 5% of its maximum at 20 seconds that statements about the sensitivity of parameters at these long times are meaningless. However, upon examining

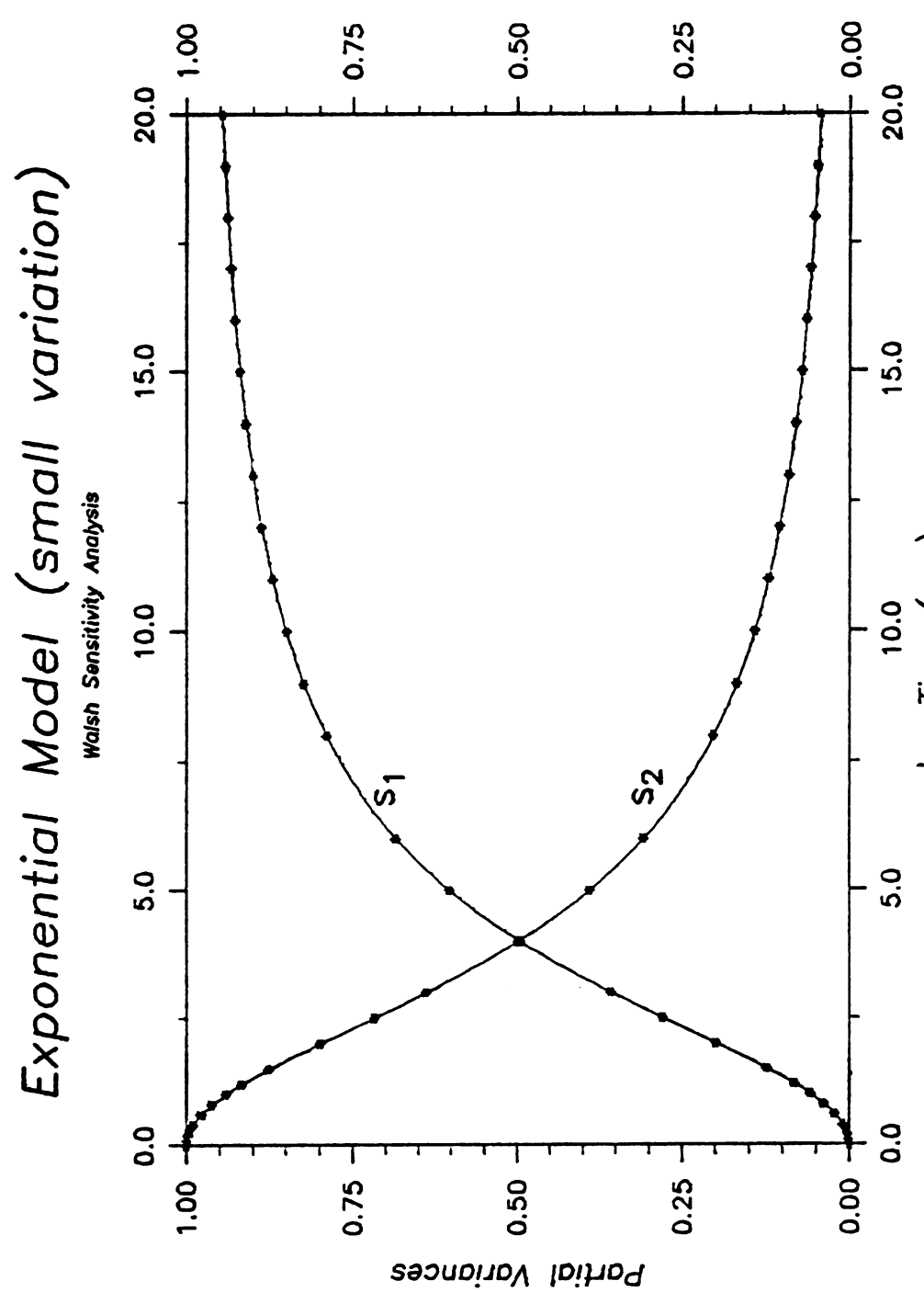


Figure 4.12 Walsh partial variances from the Exponential Model (10% variation).

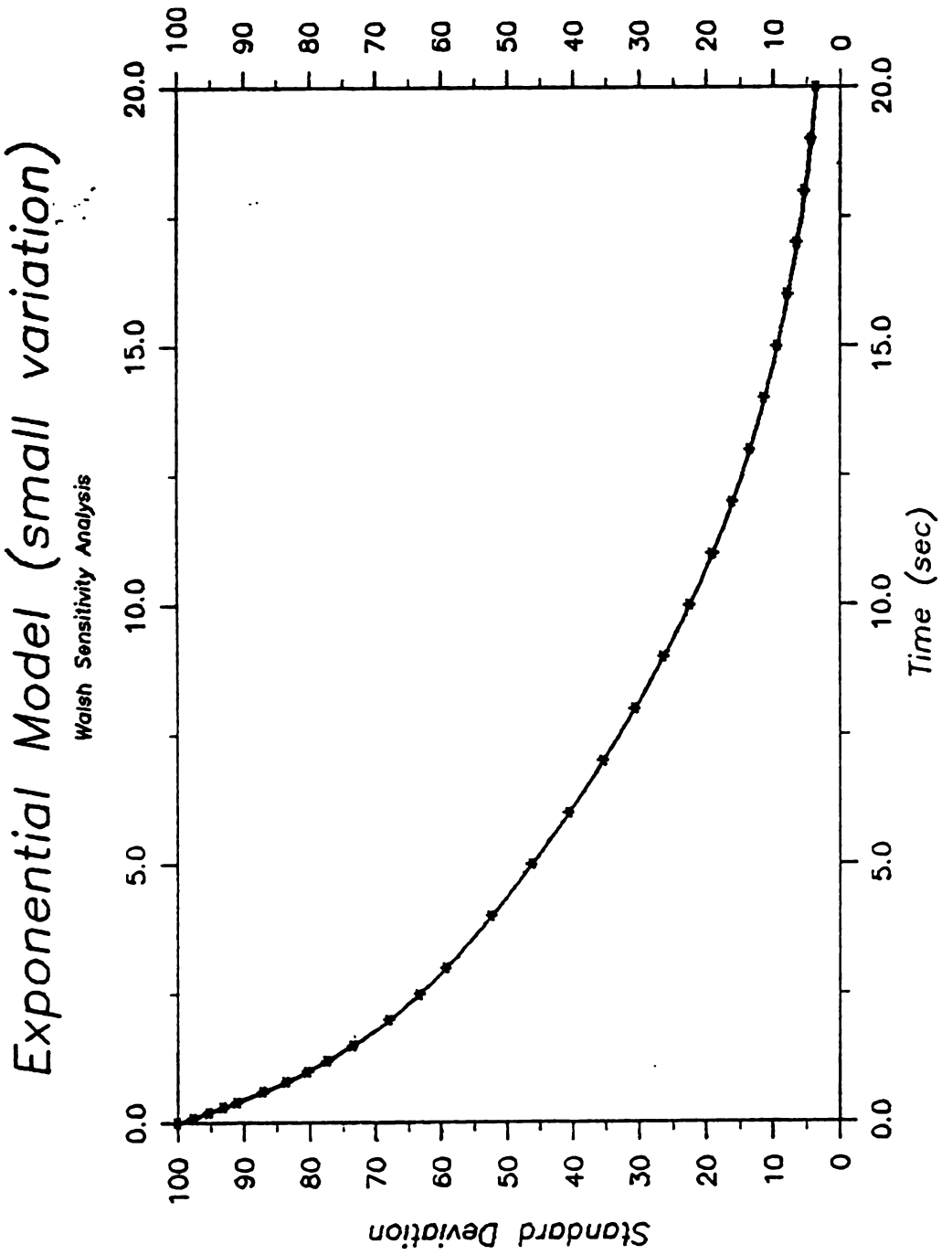


Figure 4.13 The standard deviation from the Walsh Analysis of the Exponential Model (10% variation).

the dimensionless plot in Figure 4.14 we see that the relative deviation, which is the standard deviation divided by the average value, actually increases in time. This means that as the magnitude of the output function decreases the relative variation grows. Therefore, the sensitivities of the parameters at long times can be significant if the relative deviation, rather than the absolute deviation, is nearly constant.

One advantage that the Walsh method has over linear analysis is that it explicitly uses a range of variation for the parameters. If this range of variation is increased (from 10% to 60%) and the mathematical model reanalyzed it can be seen from Figure 4.15 that slightly different behavior results. In Figure 4.15 the average value does not decay away as fast as the earlier analysis. Figure 4.16 shows that the coefficient curves have shifted the maximum sensitivity of the decay constant, k_1, to longer times, 5 seconds, reflecting the effect of the nonlinear behavior of the model. The sensitivity of the pre-exponential parameter, k_2, also decays away more slowly. The partial variances in Figure 4.17 also show the effect of a larger range of variation by shifting the crossover point from 4 seconds to 6 seconds. Note that the nonlinear effect of the model is to delay the sensitivity to k_1 into longer times. However, the standard deviation curve, Figure 4.18, and the relative deviation curve, Figure 4.19,

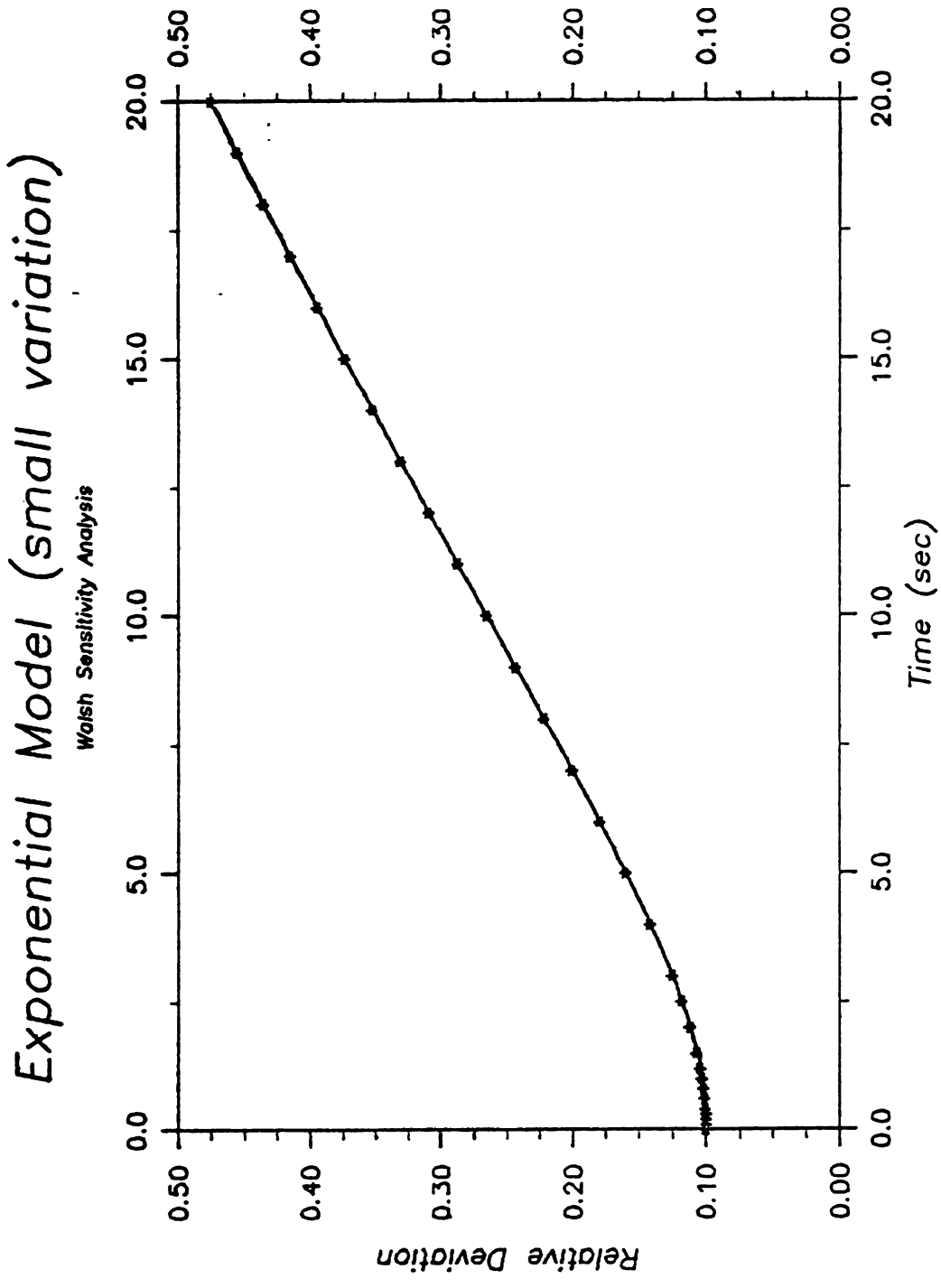


Figure 4.14 The relative deviation from the Walsh Analysis of the Exponential Model (10% variation).

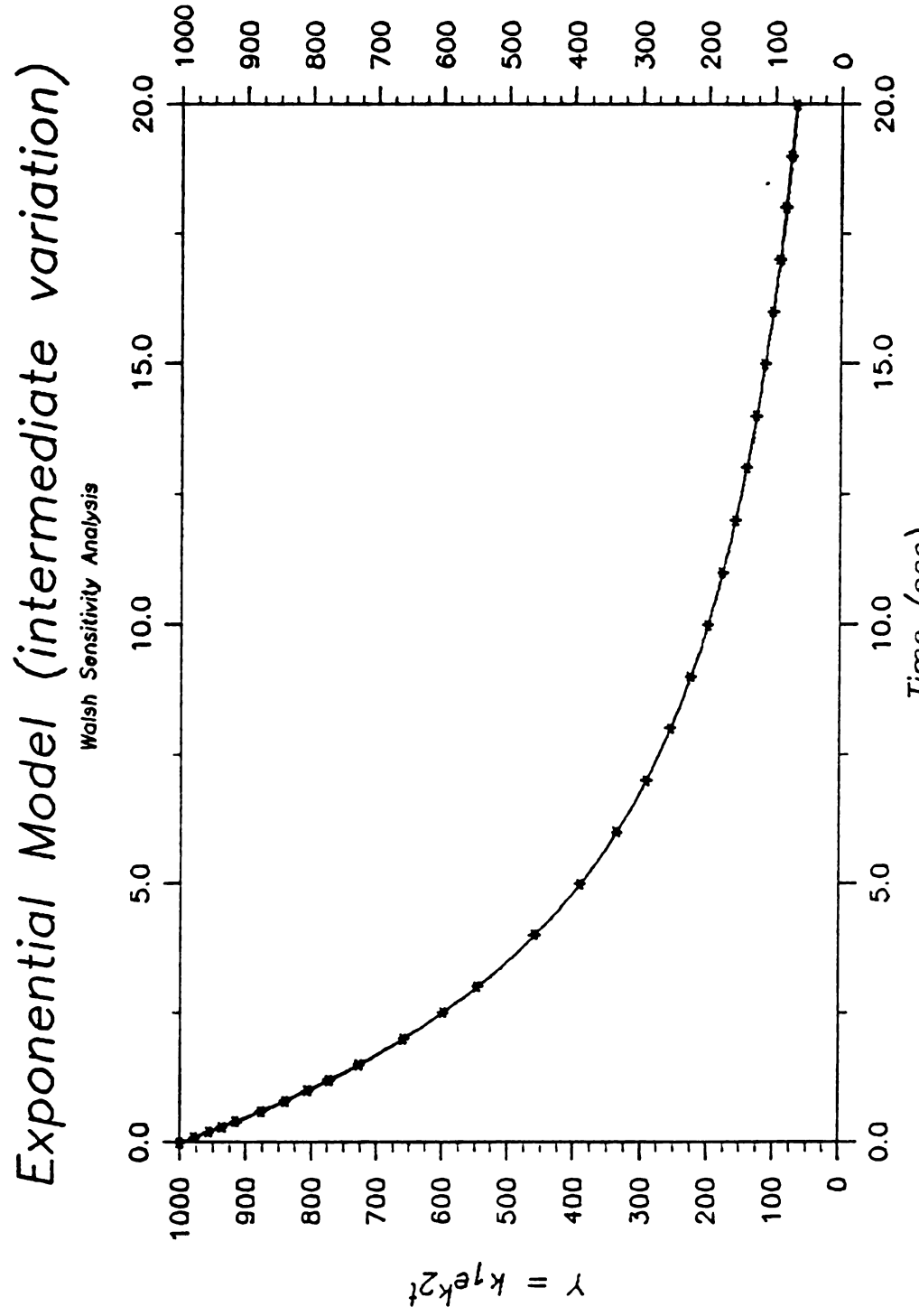


Figure 4.15 The averaged value from the Walsh analysis of the Exponential Model with the parameters varied by 60%.

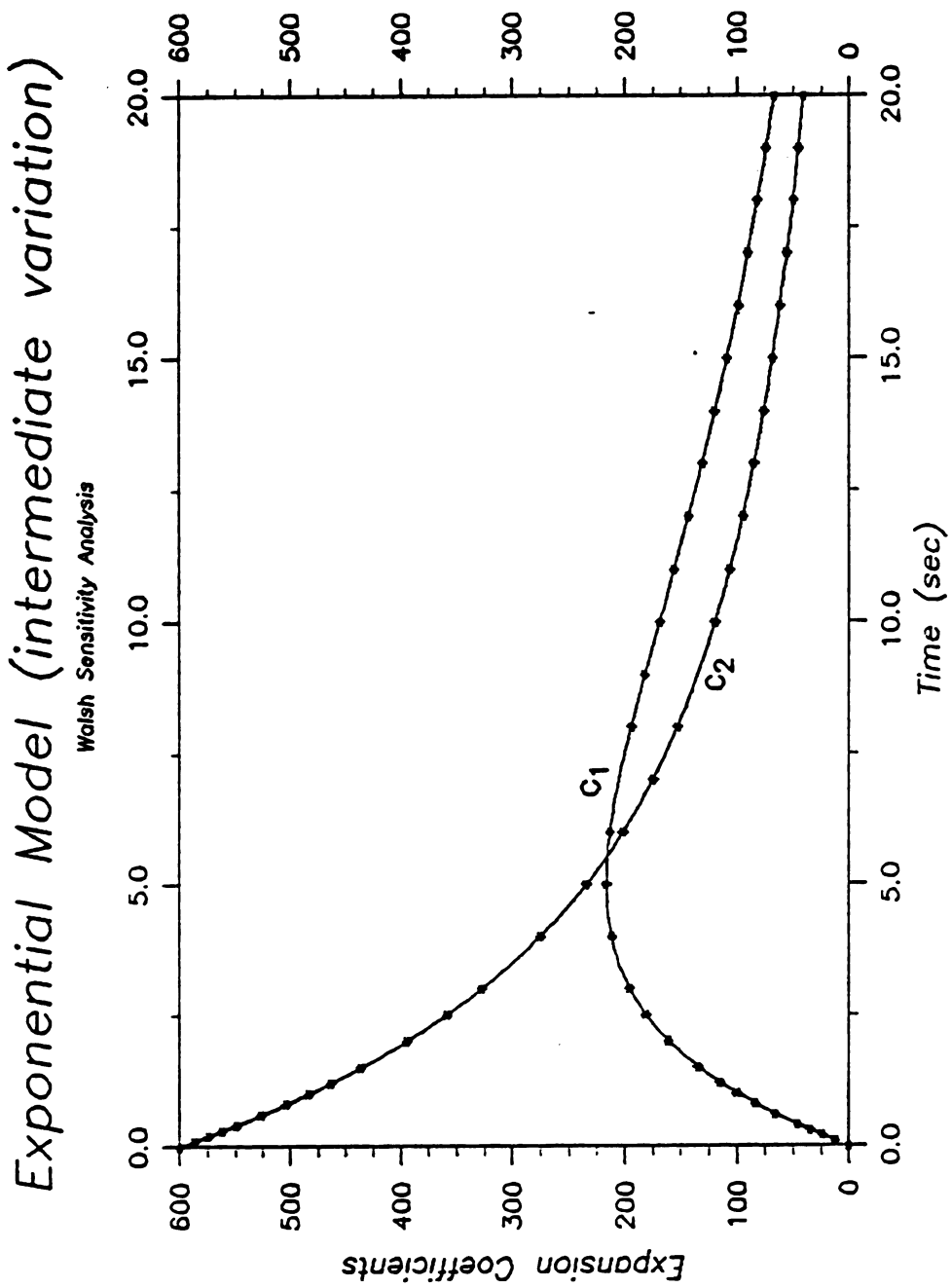


Figure 4.16 Walsh expansion coefficients from the Exponential Model (60% variation).

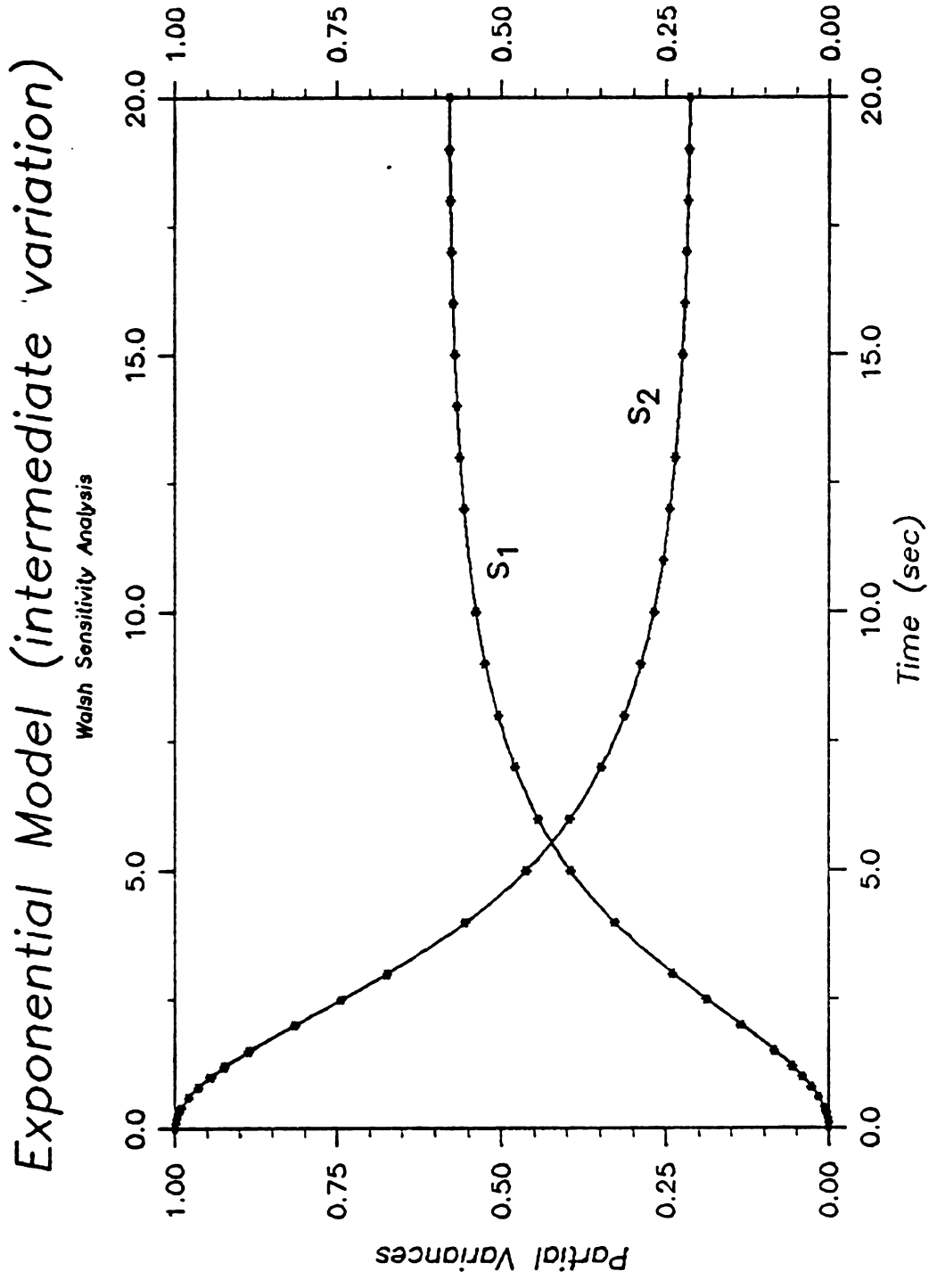


Figure 4.17 The Walsh partial variances from the Exponential Model (60% variation).

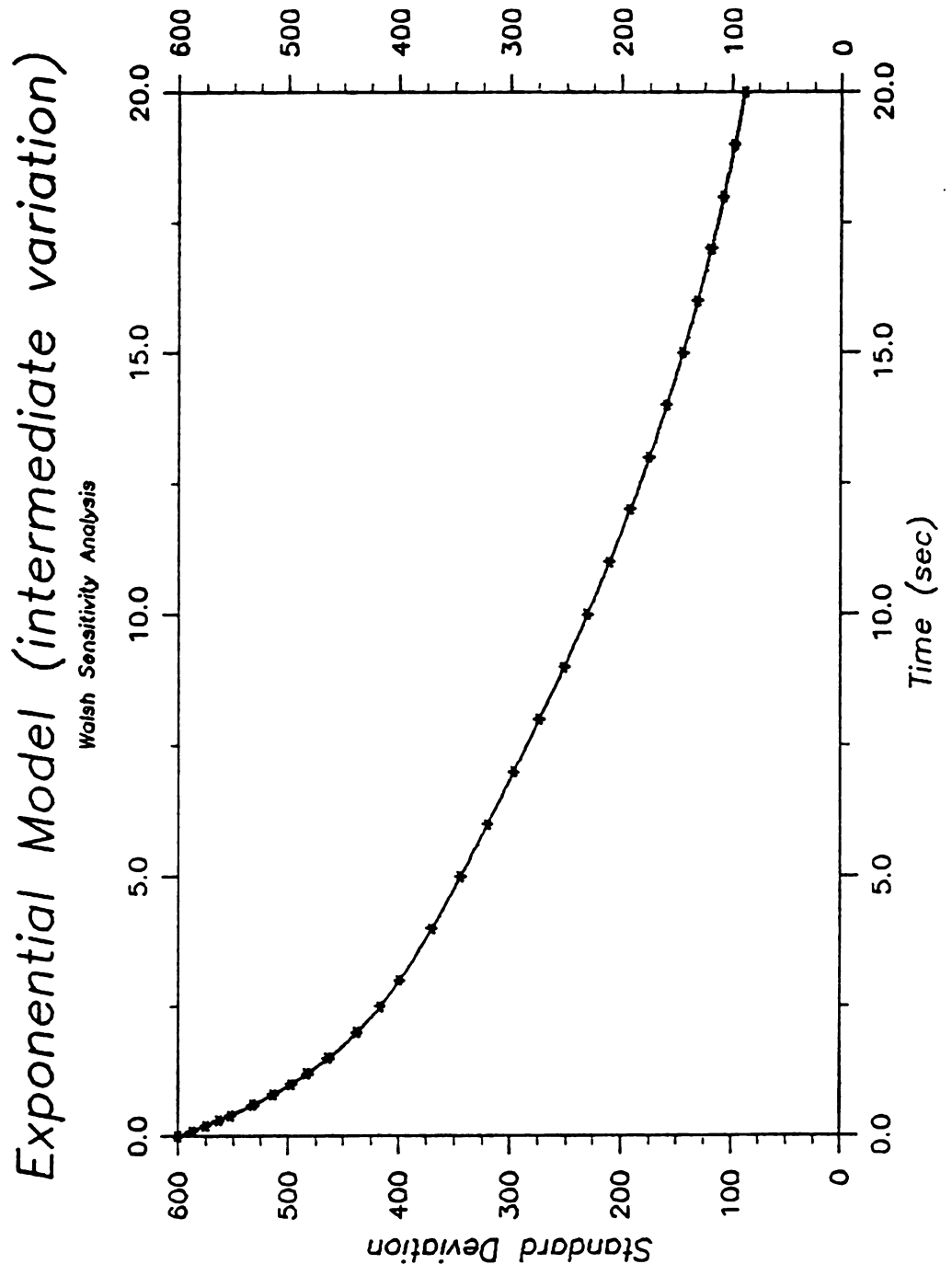


Figure 4.18 The Walsh standard deviation from the Exponential Model (60% variation).

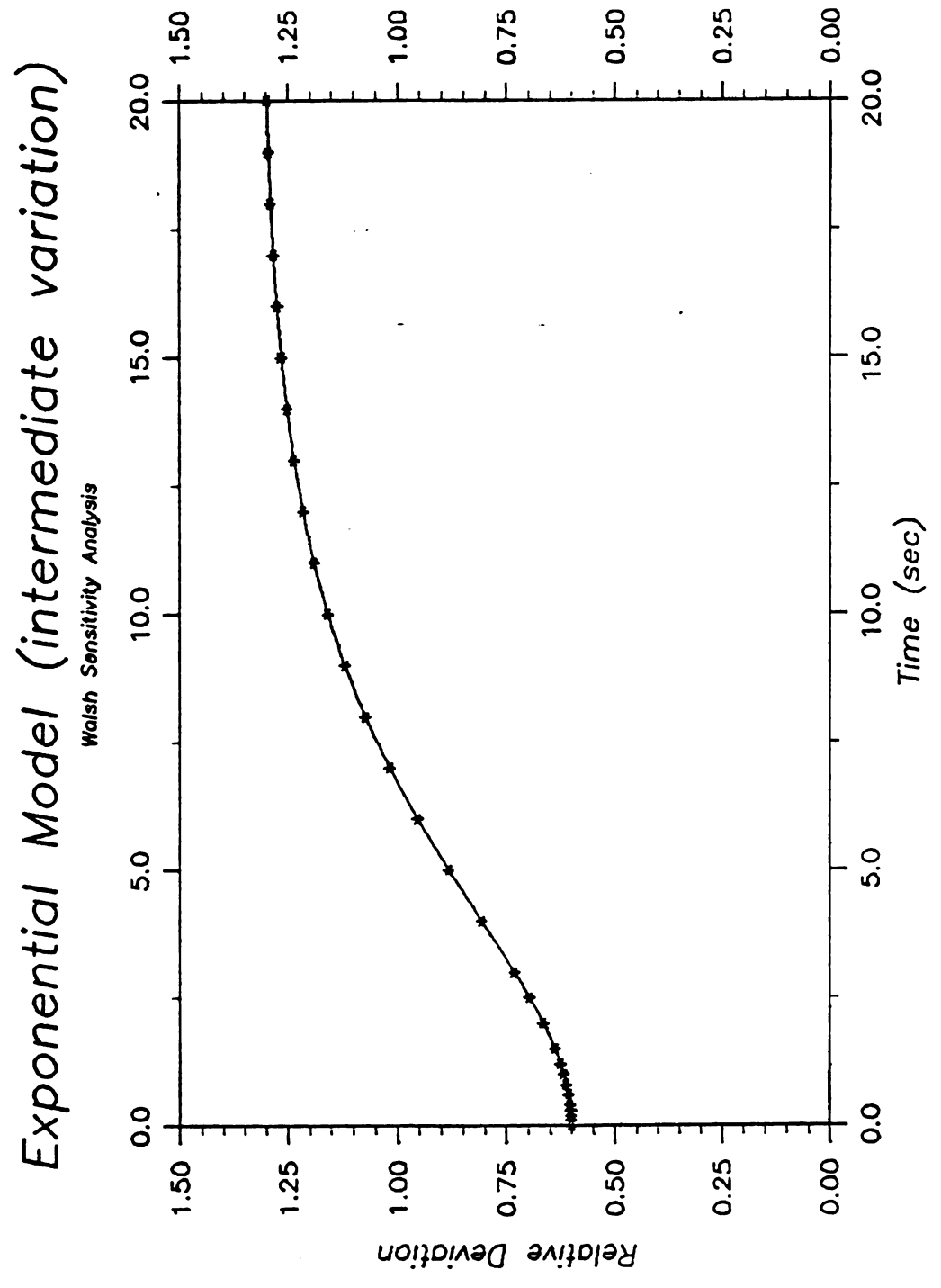


Figure 4.19 The Walsh relative deviation from the Exponential Model (60% variation).

have the same behavior as for the corresponding cases of a small range of the parameters although the magnitudes have changed.

Choosing an even larger range of variation for the parameters of the model ($k_2 = 1000 \pm 1000$, $k_1 = -0.25 \pm 0.25$ $(\text{seconds})^{-1}$) results in the curves in Figures 4.20-4.24. In this analysis the average value does not even decay away to zero! This is not typical behavior as shown in the previous two analyses. This behavior is caused by the particular sets of rate constants used in this analysis. At 2.5 seconds two simulations have reached their final values, and at 12 seconds the other two simulations have reached completion. This is the danger encountered when the analysis uses only the extremes of the parameter variation intervals. If the intervals are large enough the behavior of the model at the extremes of the parameter intervals may be completely different than its behavior closer to the nominal value.

With the Walsh method we can examine the onset of nonlinear behavior by expanding the range of analysis from the nominal value. This is important, especially for models which are numerically solved so that the degree of nonlinearity in the model solution is unknown.

When the analysis was repeated using the Fourier method with the same set of nominal parameters and over the same small range of variation, $k = -0.25 \pm 0.025$ $(\text{seconds})^{-1}$, $k_2 = 1000 \pm 100$, the same results as were

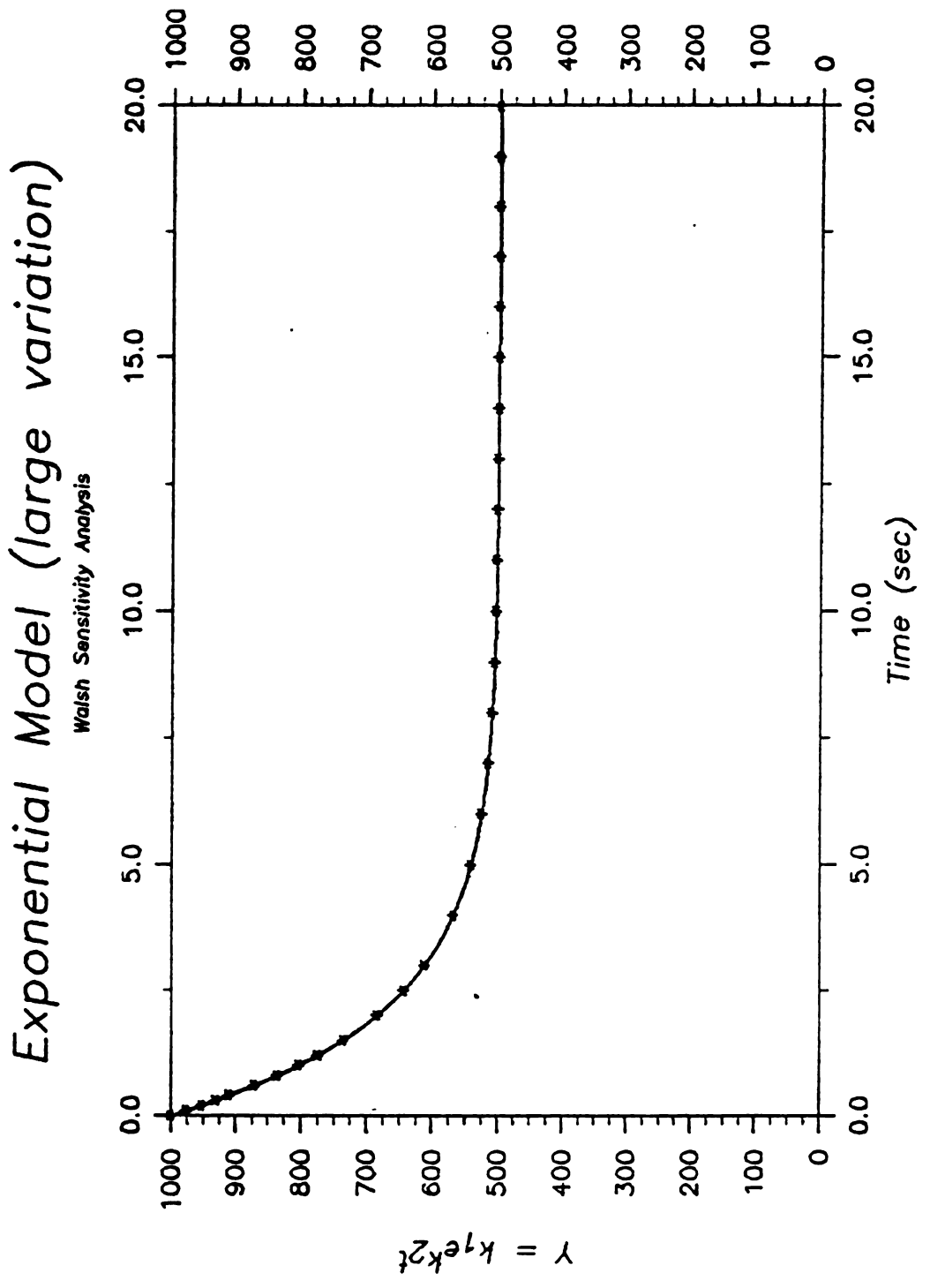


Figure 4.20 The Walsh averaged value from the Exponential Model with 100% parameter variation.

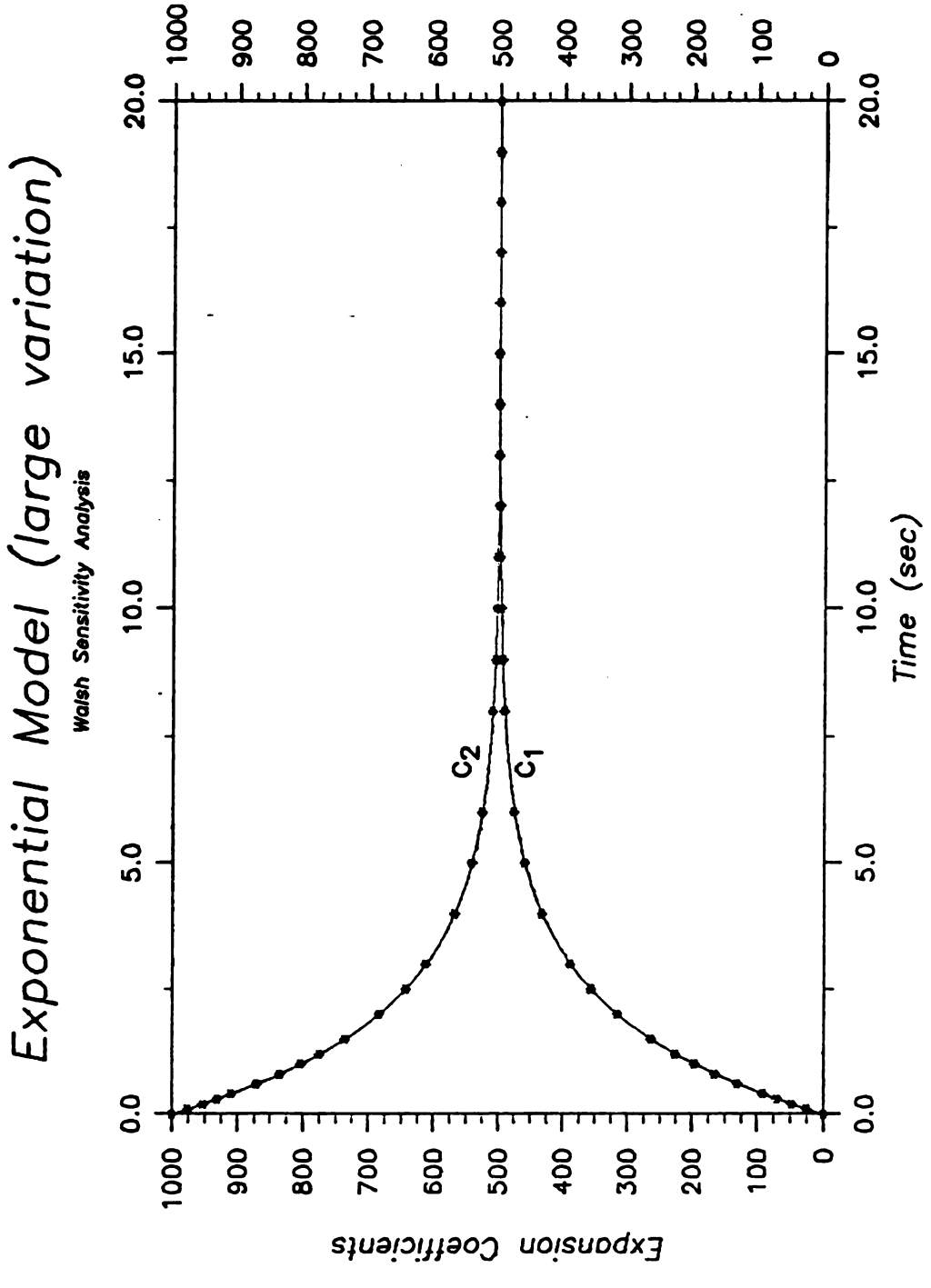


Figure 4.21 Walsh expansion coefficients from the Exponential Model (100% variation).

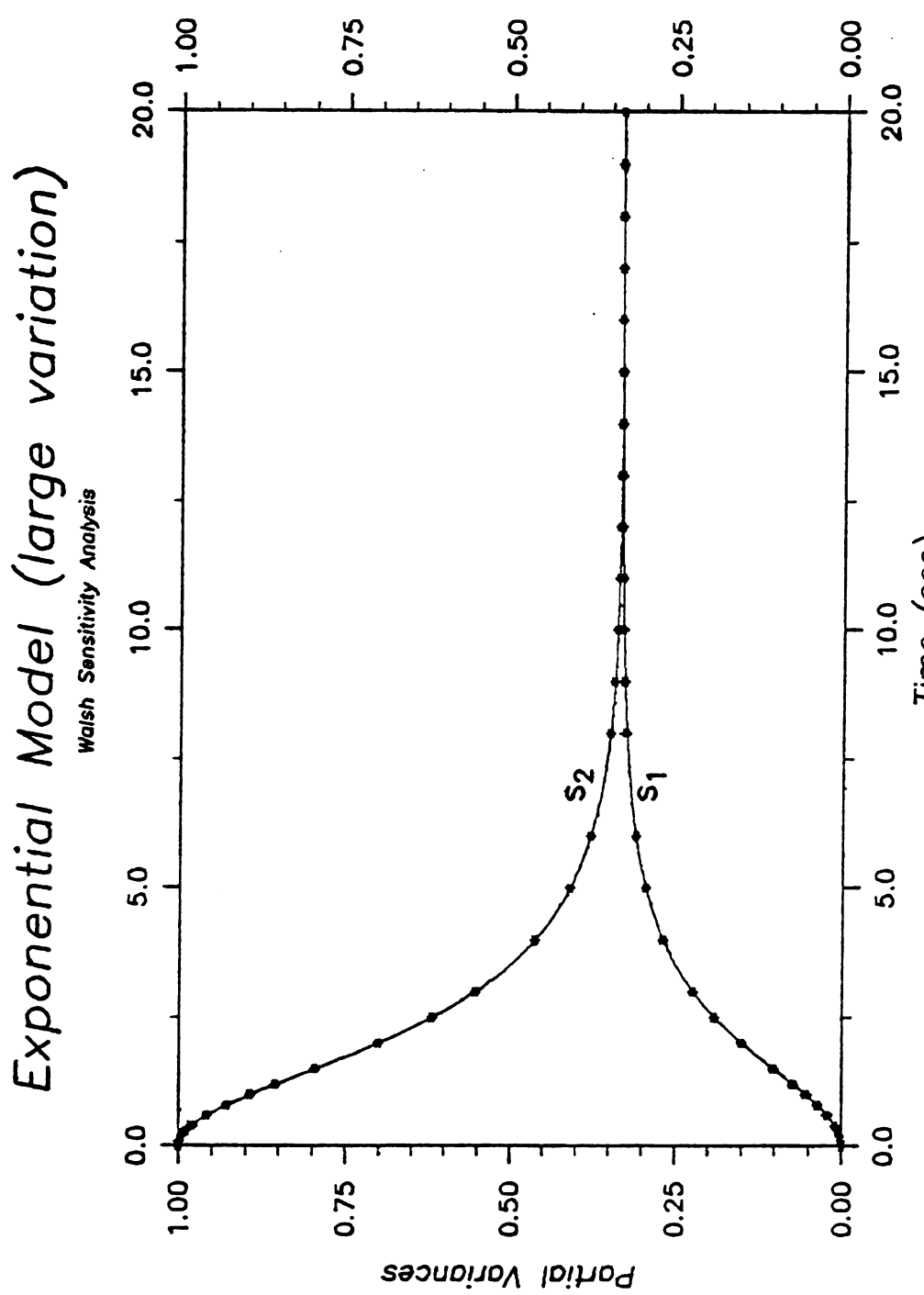


Figure 4.22 Walsh partial variances from the Exponential Model (100% variation).

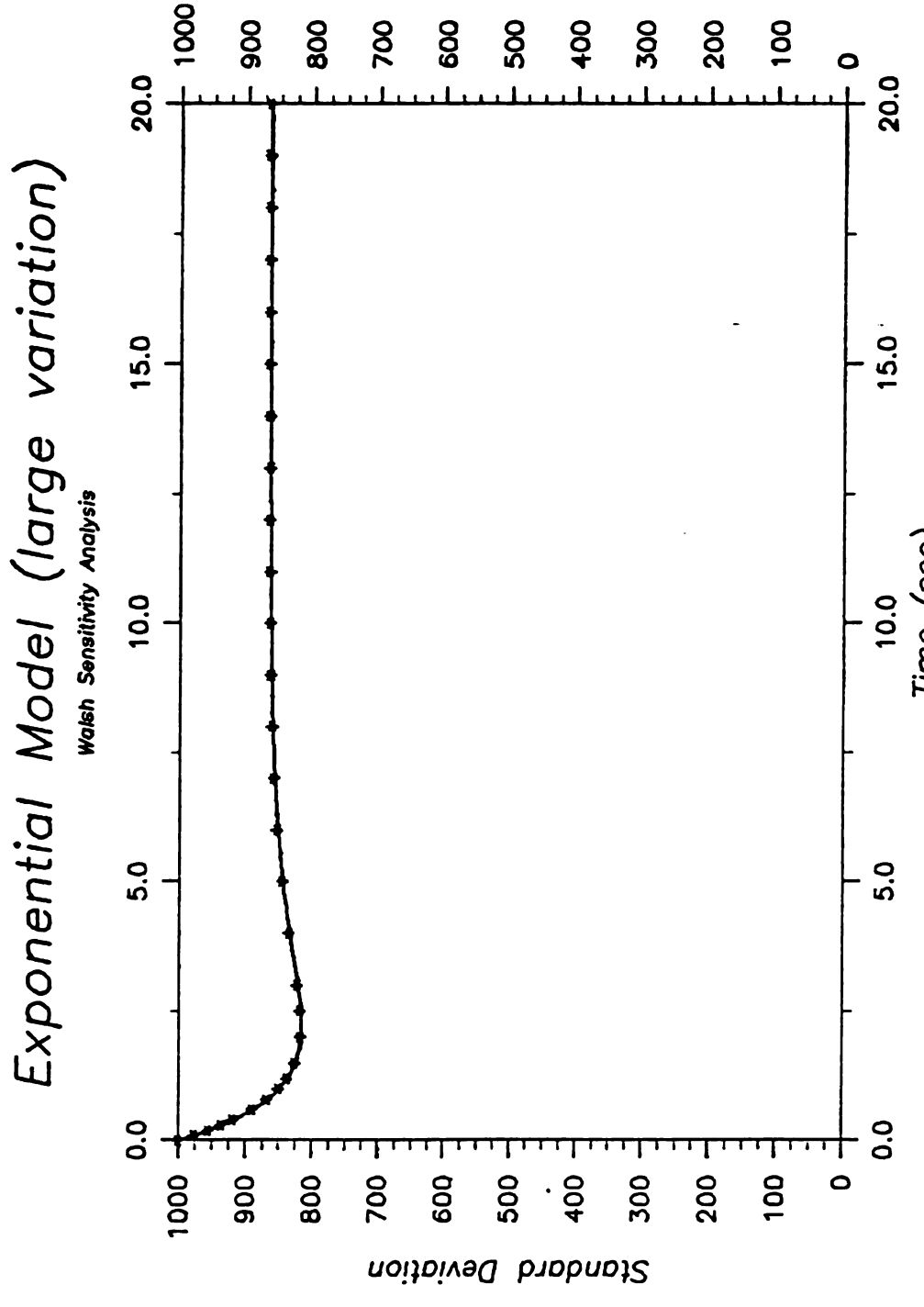


Figure 4.23 The Walsh standard deviation from the Exponential Model (100% variation).

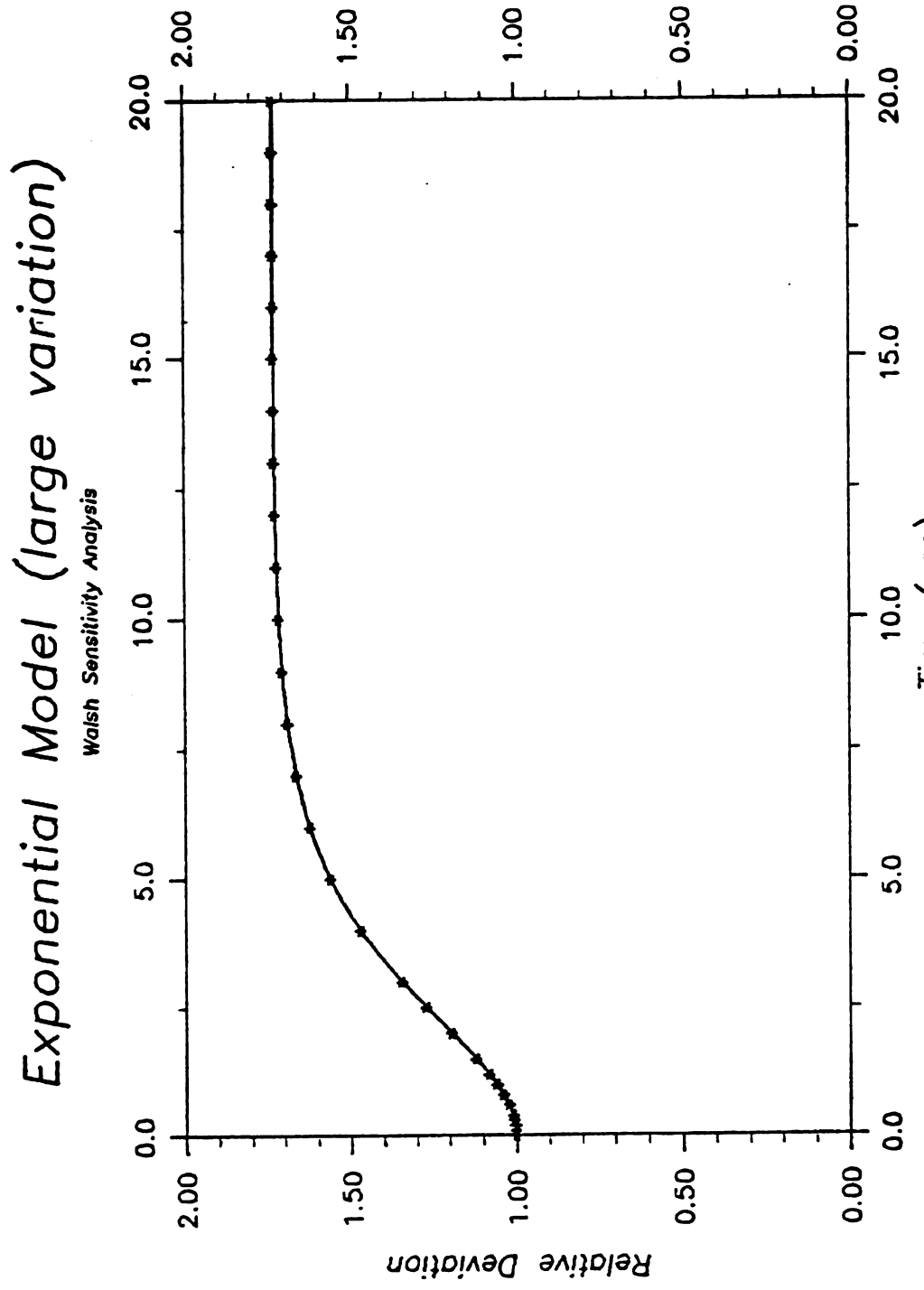


Figure 4.24 The Walsh relative deviation from the Exponential Model (100% variation).

obtained for the average value of the output function as with linear and Walsh techniques (Figure 4.25). The expansion coefficients, Figure 4.26, have different magnitudes but the behavior is the same. The maximum of the C coefficient occurs at the same time point for both the Walsh and Fourier methods.

The Fourier partial variances, Figure 4.27, are also identical to those of the small variation Walsh analysis. There is one slight difference in the two partial variances at very early times. This is caused by the slightly different parameter ranges used. The Fourier method used a log-uniform transformation function which varied the parameters over [-0.221, -0.227] and [1095, 905]. However, within one second, the Walsh partial variances and the Fourier partial variances, both normalized by their respective total variances which are different (compare Figure 4.13 with Figure 4.28), reach identical values. Consequently the Fourier partial variances have the same interpretation as did the previous Walsh partial variances.

For comparison purposes, the relative deviation curve for the small variation Fourier analysis is plotted in Figure 4.29. Note that it is always smaller than the corresponding Walsh curve, Figure 4.14. This reflects both the slightly restricted range of parameter variation and the increased number of simulations used in the Fourier method.

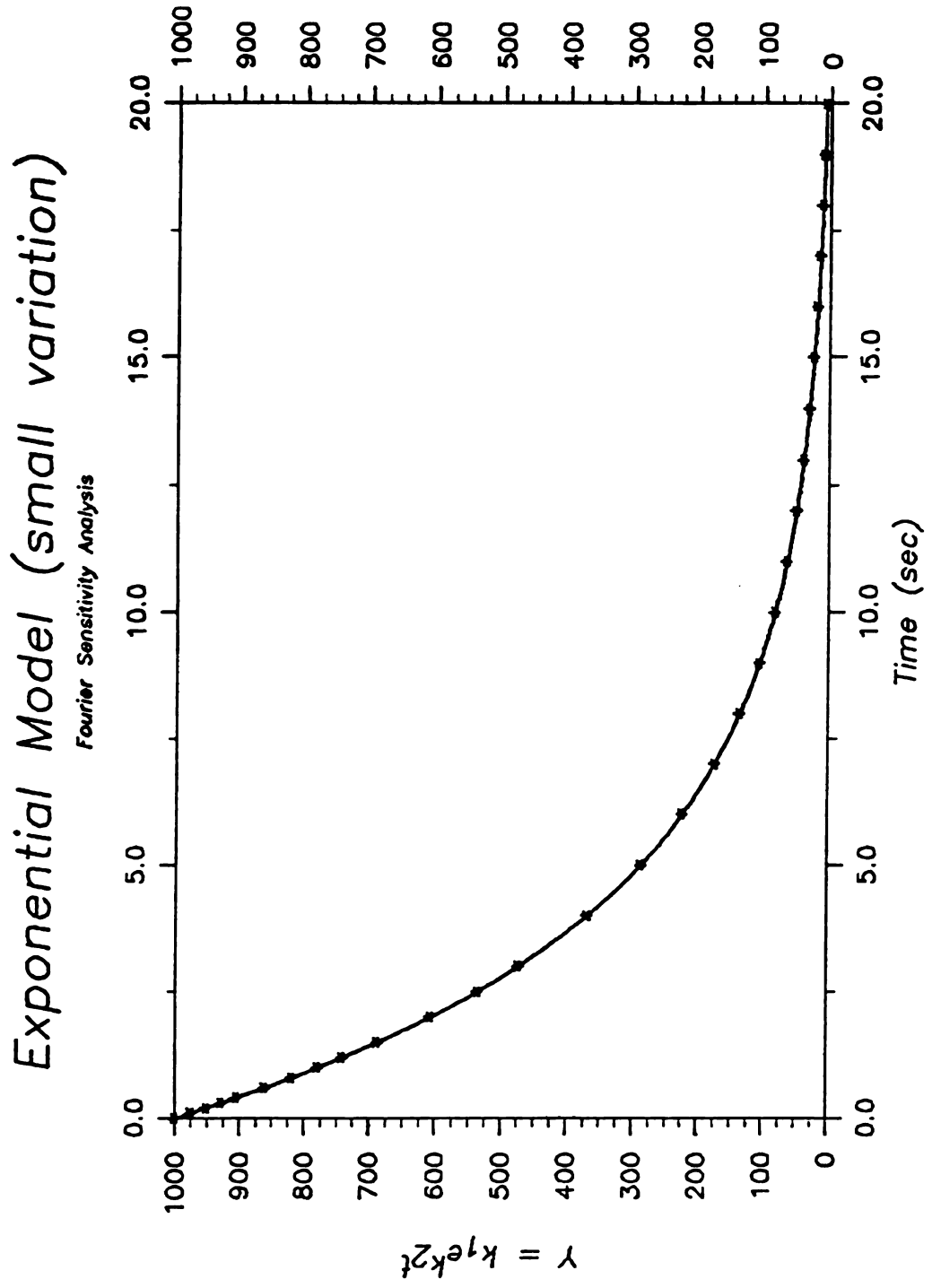


Figure 4.25 The Fourier averaged value from the Exponential Model (10% variation).

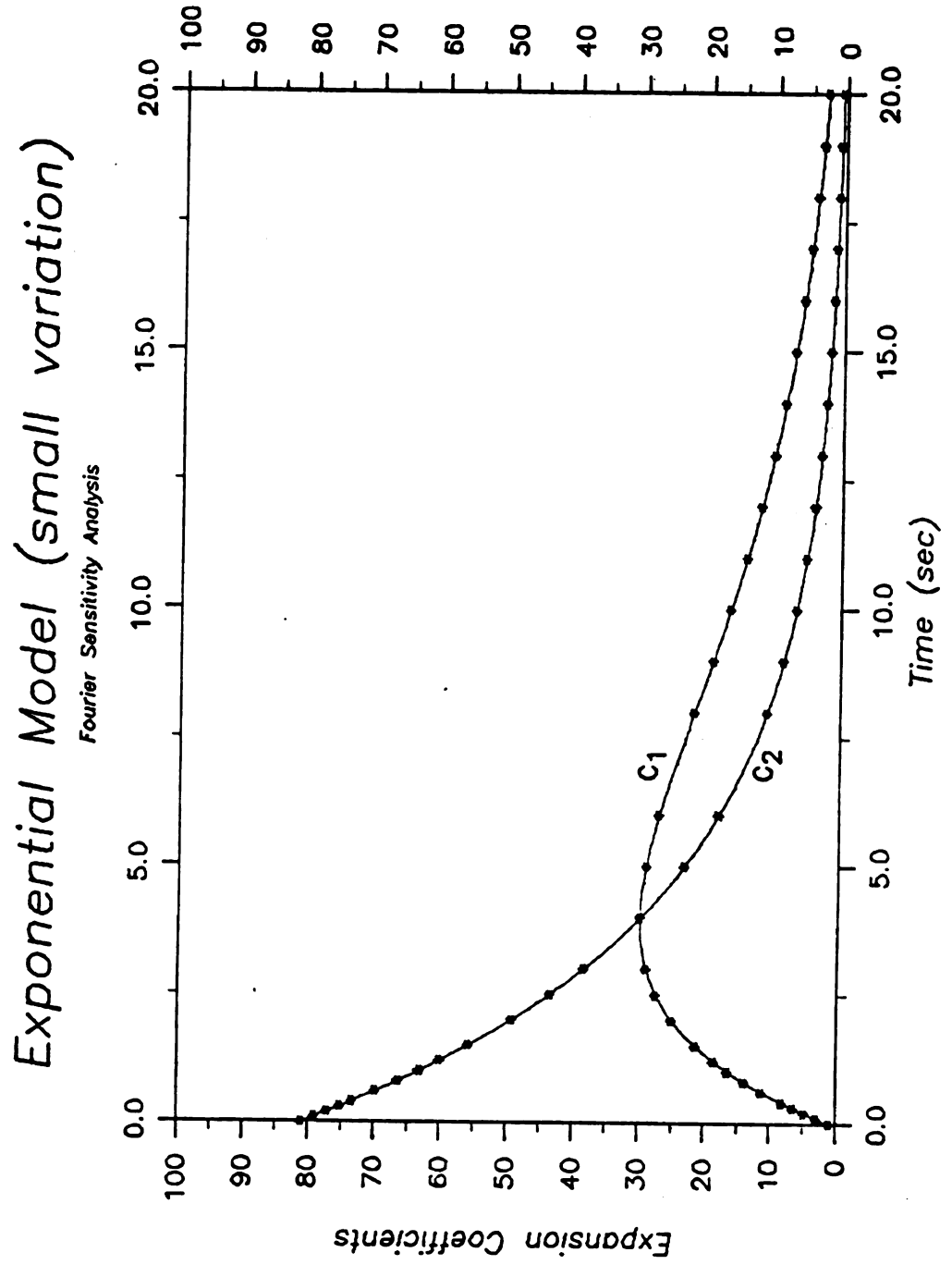


Figure 4.26. Fourier expansion coefficients from the Exponential Model (10% variation).

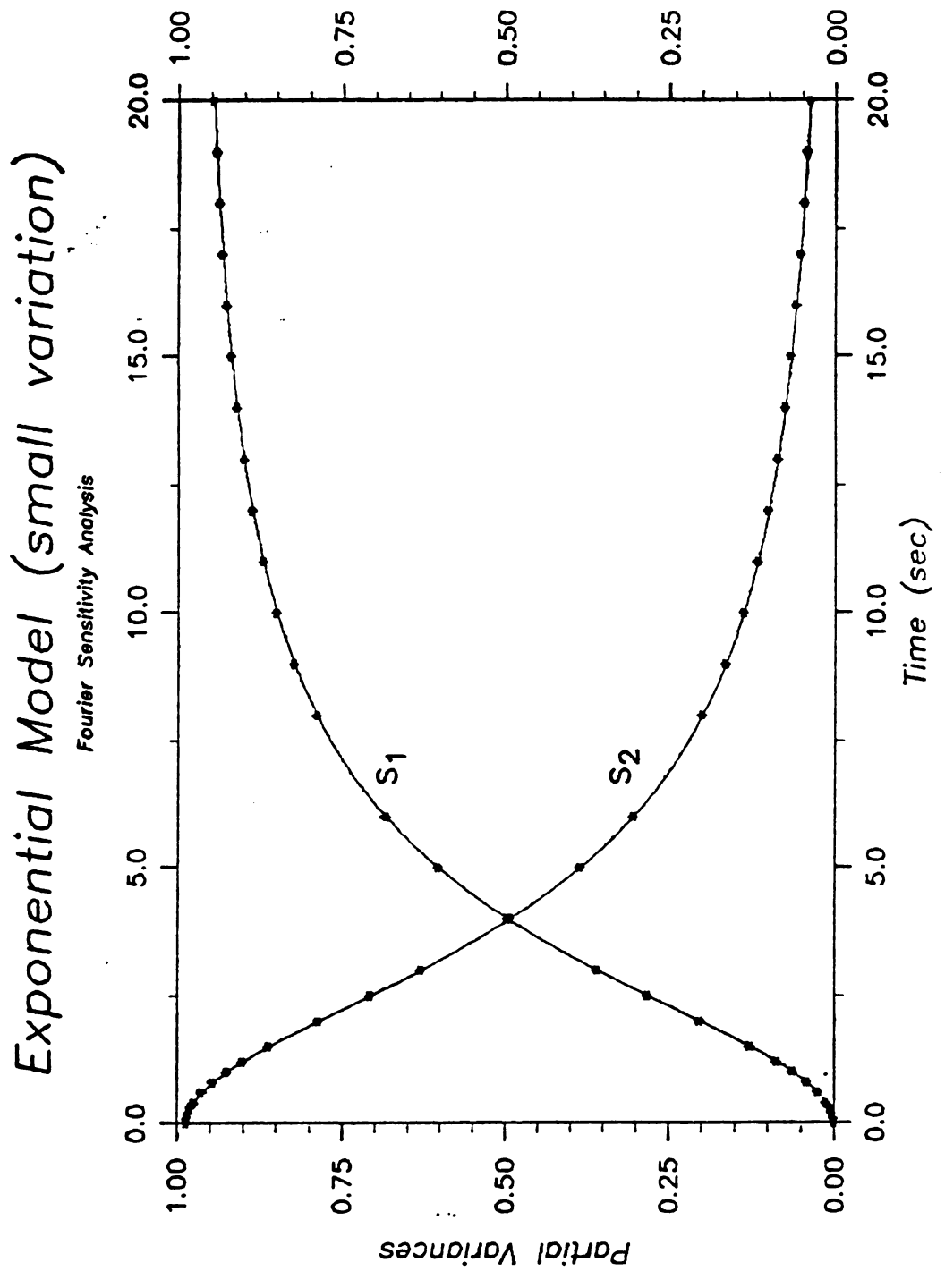


Figure 4.27 Fourier partial variances from the Exponential Model (10% variation).

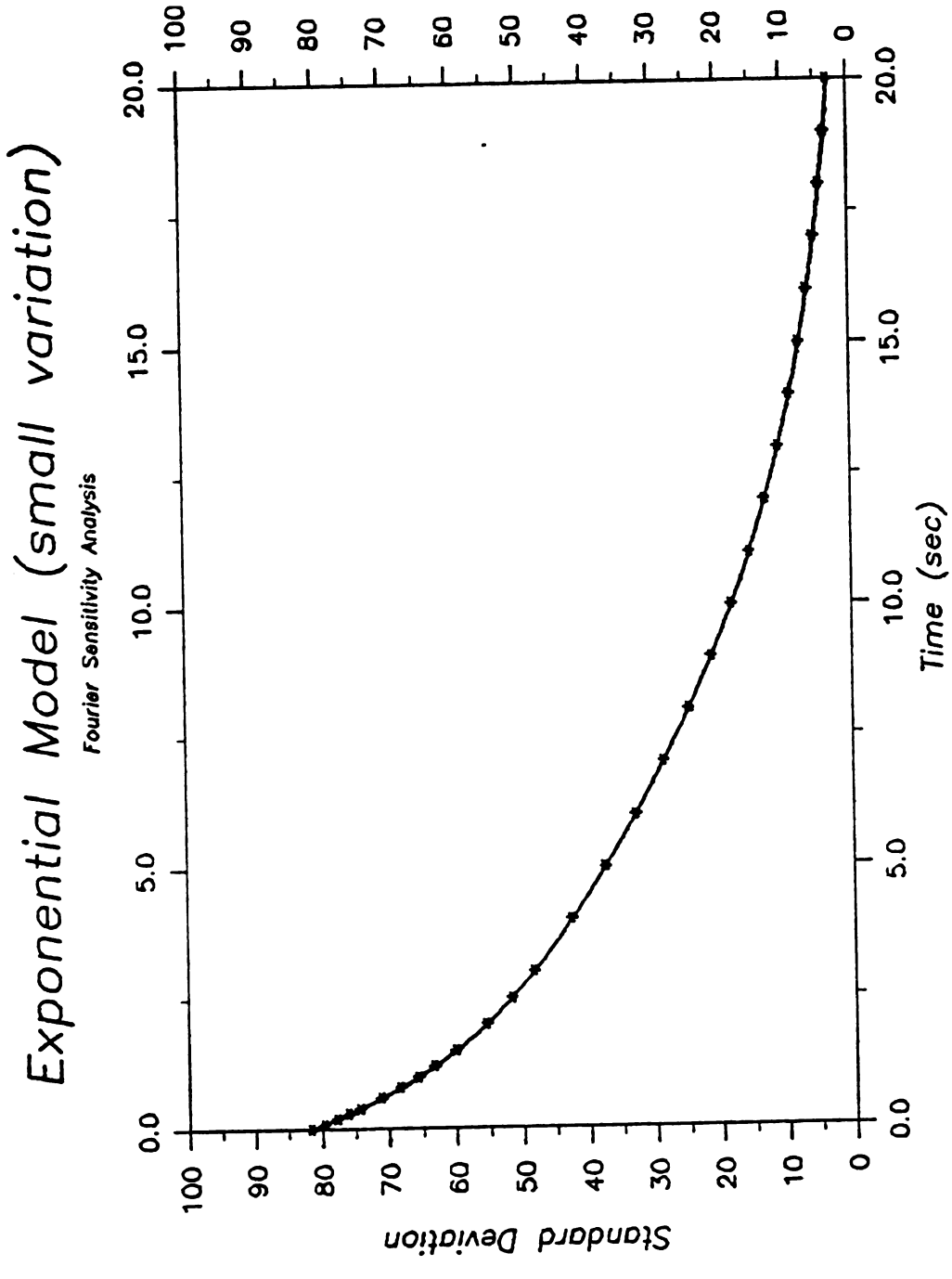


Figure 4.28 The Fourier standard deviation from the Exponential Model (10% variation).

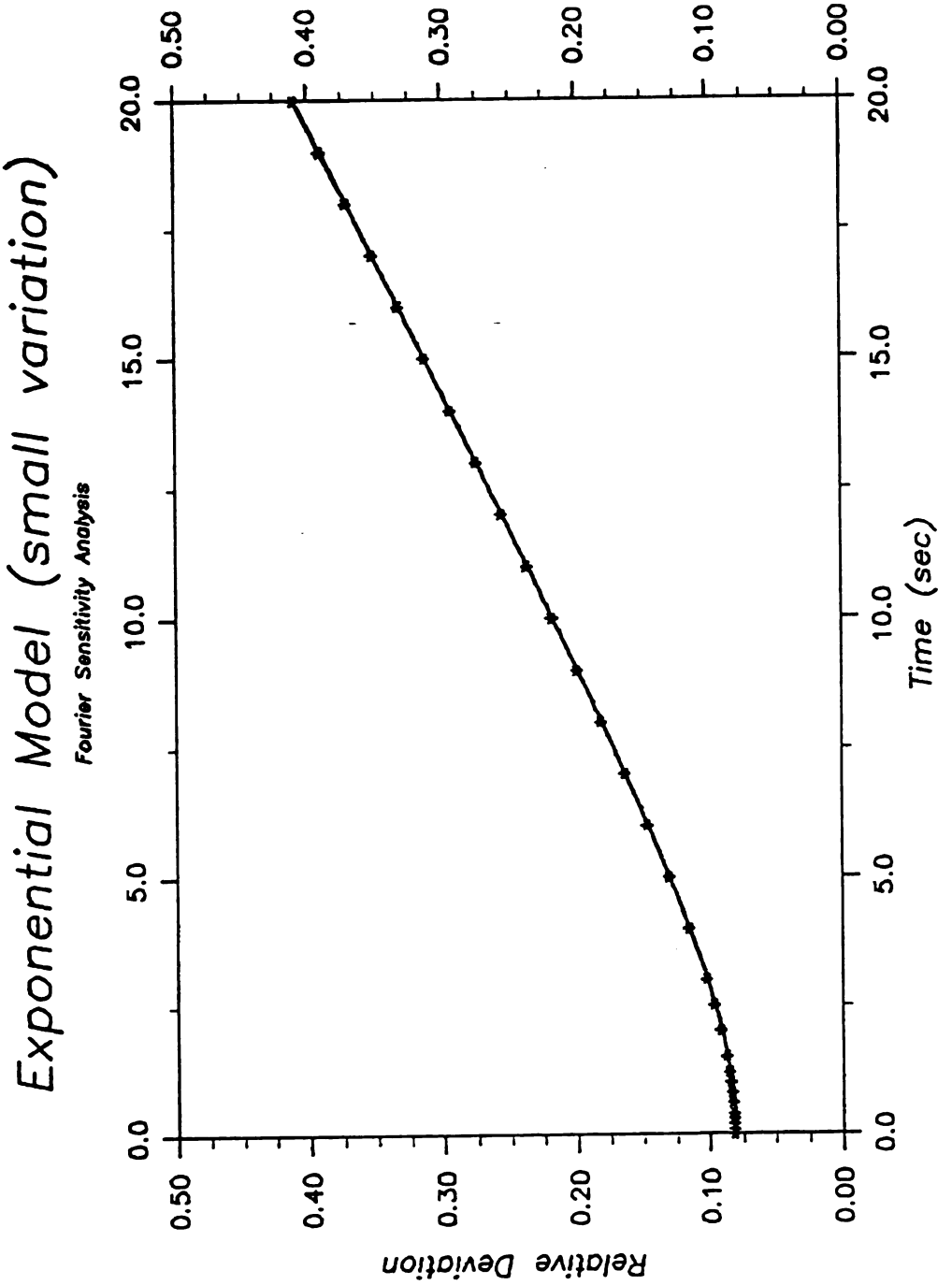


Figure 4.29 The Fourier relative deviation from the Exponential Model (10% variation).

Figures 4.30-4.34 give the results of Fourier analysis of the exponential model when a large range of variation of the parameters is used, ($k_1 = -0.25 \pm .25$ (seconds)$^{-1}$, $k_2 = 1000 \pm 1000$). This analysis reveals the nonlinear aspects of the model. Figure 4.30 shows a slower averaged decay of the output function than for the small range case. The expansion coefficients for the case of large variations, Figure 4.31, have nearly the same behavior as those obtained with the small variations (Figure 4.26). However, the maximum value of C_1 coefficient has shifted to longer times, similar to the behavior of the Walsh C_1 coefficient shown in Figure 4.16, although the shift is not as great. The large variation C_2 coefficient doesn't decay away as fast as the small variation C_2 coefficient does. Even more striking are the partial variances plotted in Figure 4.32. The partial variance of k_1, S_1, reaches a maximum at about 13 seconds and then slowly decays away. The maximum is important since it selects a time region which is optimal for the measurement of that parameter. Also over this larger range of parameter space there is significant coupling between the sensitivities of the two parameters. This is shown in Figure 4.32 by the coupled partial variance $S_{1,2}$.

Coupled partial variances indicate the degree of linear dependence between pairs of parameters. When a coupled partial variance is large it is difficult to separate the

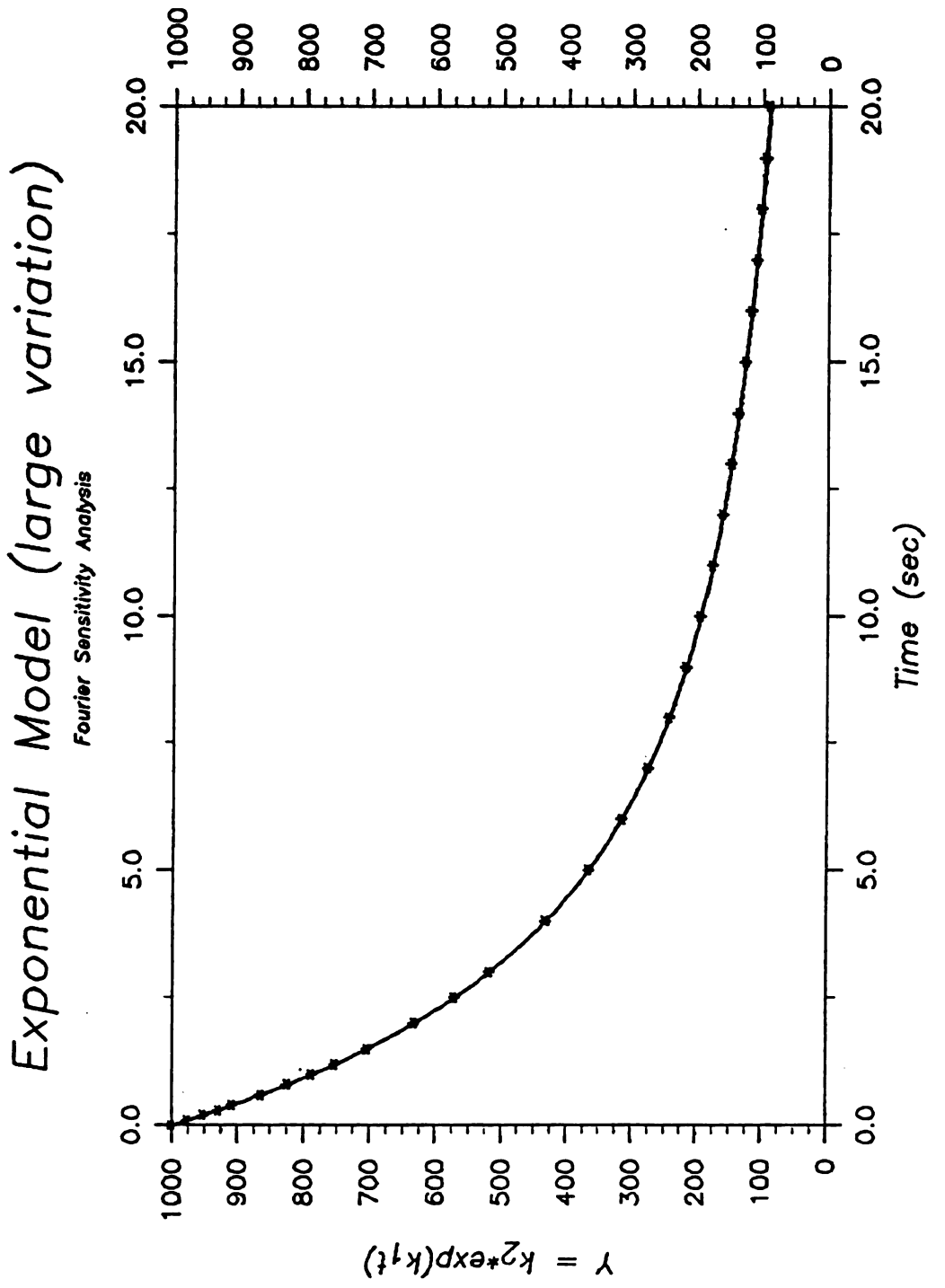


Figure 4.30 The Fourier averaged value from the Exponential Model (100% variation).

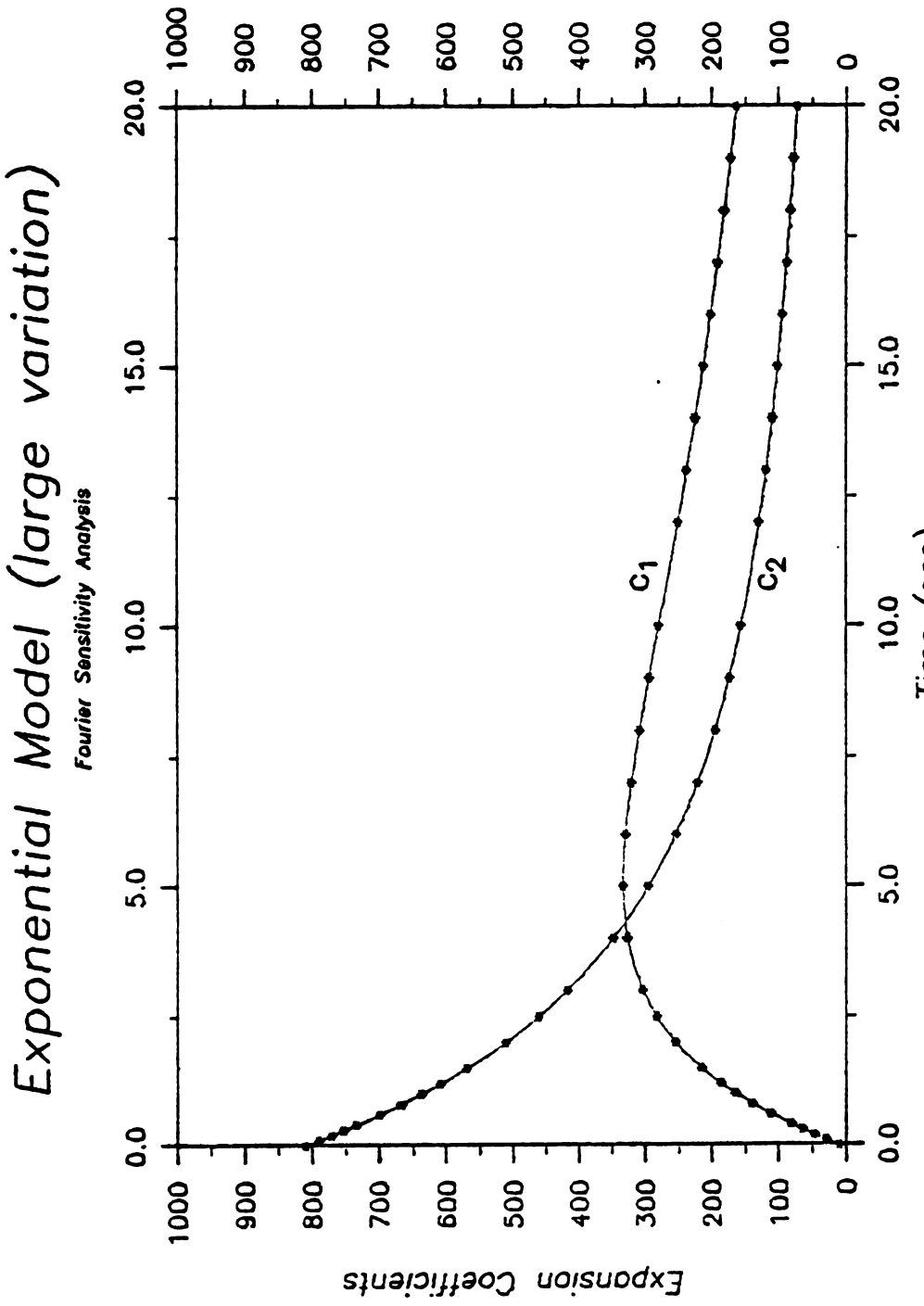


Figure 4.31 Fourier expansion coefficients from the Exponential Model (100% variation).

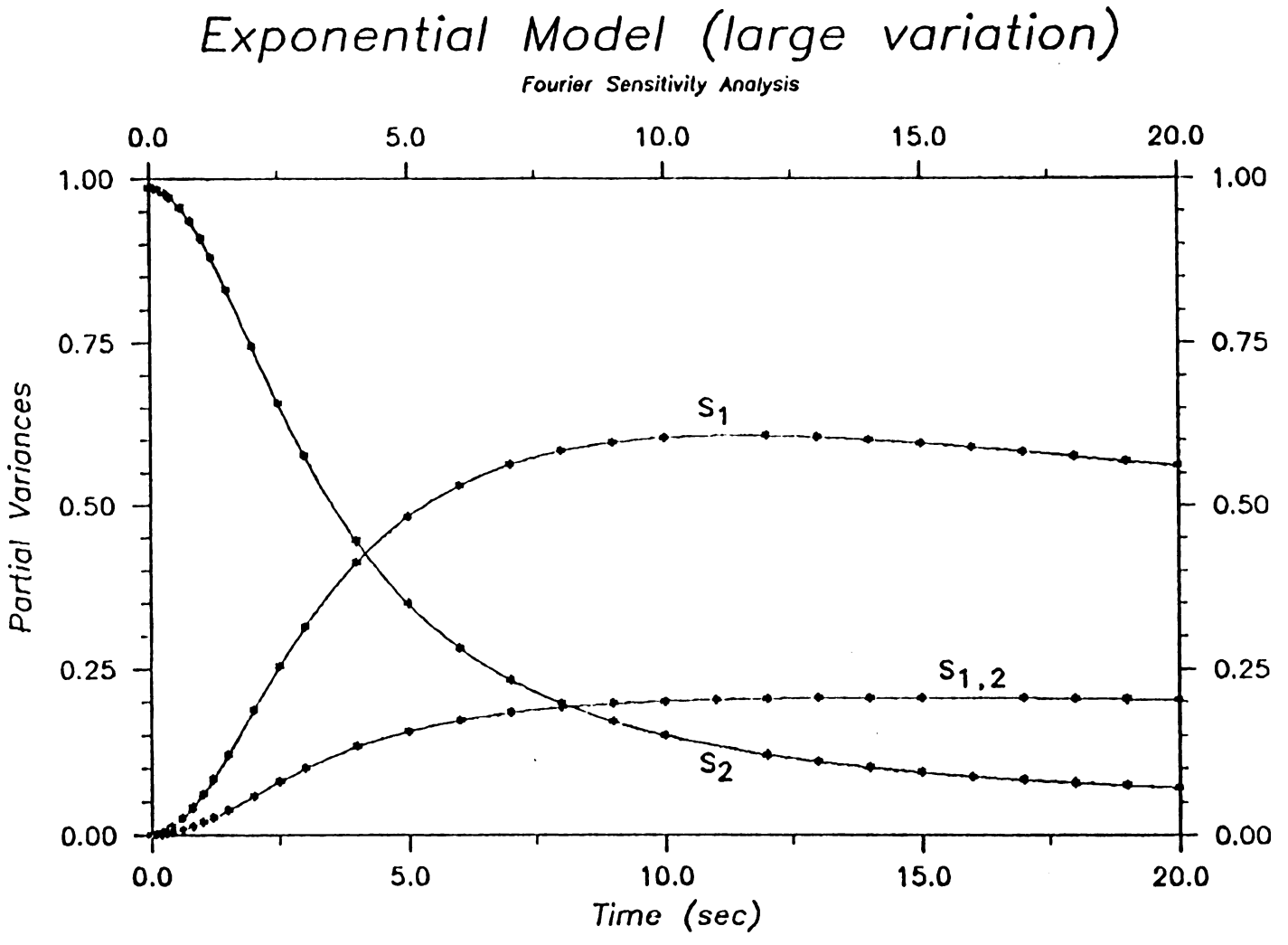


Figure 4.32 Fourier partial variances from the Exponential Model (100% variation).

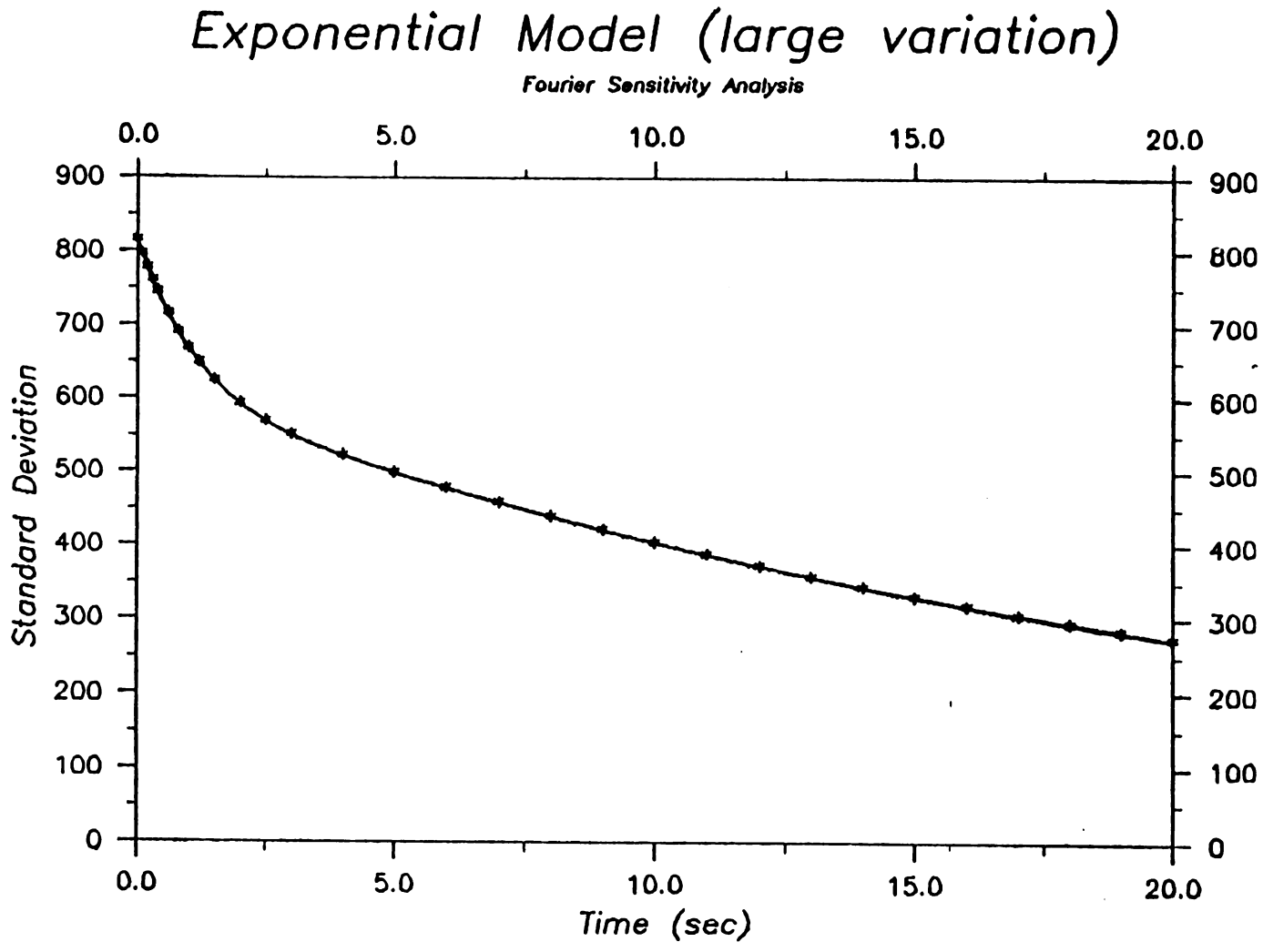


Figure 4.33 The Fourier standard deviation from the Exponential Model (100% variation).

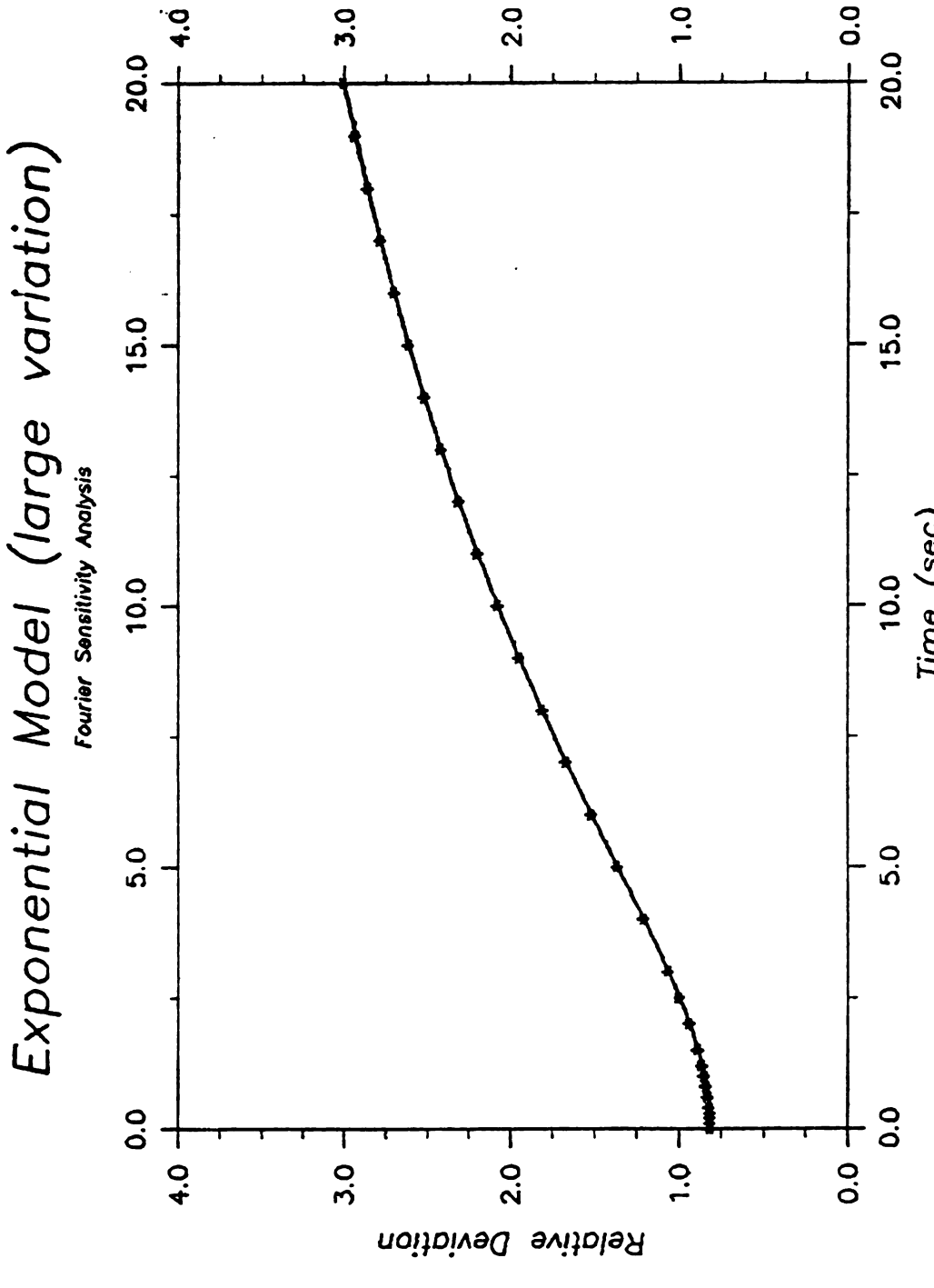


Figure 4.34 The Fourier relative deviation from the Exponential Model (100% variation).

effect of one parameter from that of the other. Note that because the partial variances and coupled partial variances are relative measures of sensitivity, they must be used in conjunction with the total variance to provide an understanding of the sensitivity. In particular, if the total variance is very small, there is little point in carefully examining its components since they are just a partitioning of this very small total variance into the individual contributions.

In examining the exponential model it can be seen that the Walsh method is equivalent to the linear analysis for small variations in the parameters. Its advantage over linear analysis is that as the range of parameter variation increases it also picks up the nonlinear effects in the model. The Fourier method is also similar to linear analysis, in the limit of small variations of the parameters. However, for large variations in the parameters, since it samples the whole of parameter space, it gives correct results while the Walsh and linear methods fail.

The Fourier method requires more simulations to achieve its results than does the Walsh method, for models with a small number of parameters; i.e., fewer than seven. The number of simulations required in Fourier analysis is heavily dependent on the order of accuracy of the frequency set. For a 6th-order accurate set for 10 parameters the Fourier method requires, at a minimum, 2843 simulations,

whereas a 4th-order set requires only 411 simulations. It should be noted that the Walsh method is exact for discrete models and for 10 parameters requires only 1024 simulations.

By applying the Fourier method to the kinetics of simple chemical reactions we can further improve our understanding of the interpretation of partial variances. One of the simplest reaction schemes in chemical kinetics is the unimolecular first-order decay of species A to species B.

$$A \overset{k_1}{\rightarrow} B \tag{4.13}$$

However, the mathematical model for this reaction is the exponential model which we have already examined. A slightly more complicated model reaction has two coupled first-order reactions, which may be written

$$A \overset{k_1}{\rightarrow} B \overset{k_2}{\rightarrow} C \tag{4.14}$$

Choosing $k_1 = 0.1 \pm 0.01$ (seconds)$^{-1}$ and $k_2 = 0.01 \pm 0.001$ (seconds)$^{-1}$ with A = 10000, B = C = 0 we are able to simulate a reaction with a 'bottleneck' step, ($B \overset{k_2}{\rightarrow} C$), since $k_1 >> k_2$.

Figure 4.35 displays the averaged concentrations for this reaction. Application of linear sensitivity analysis

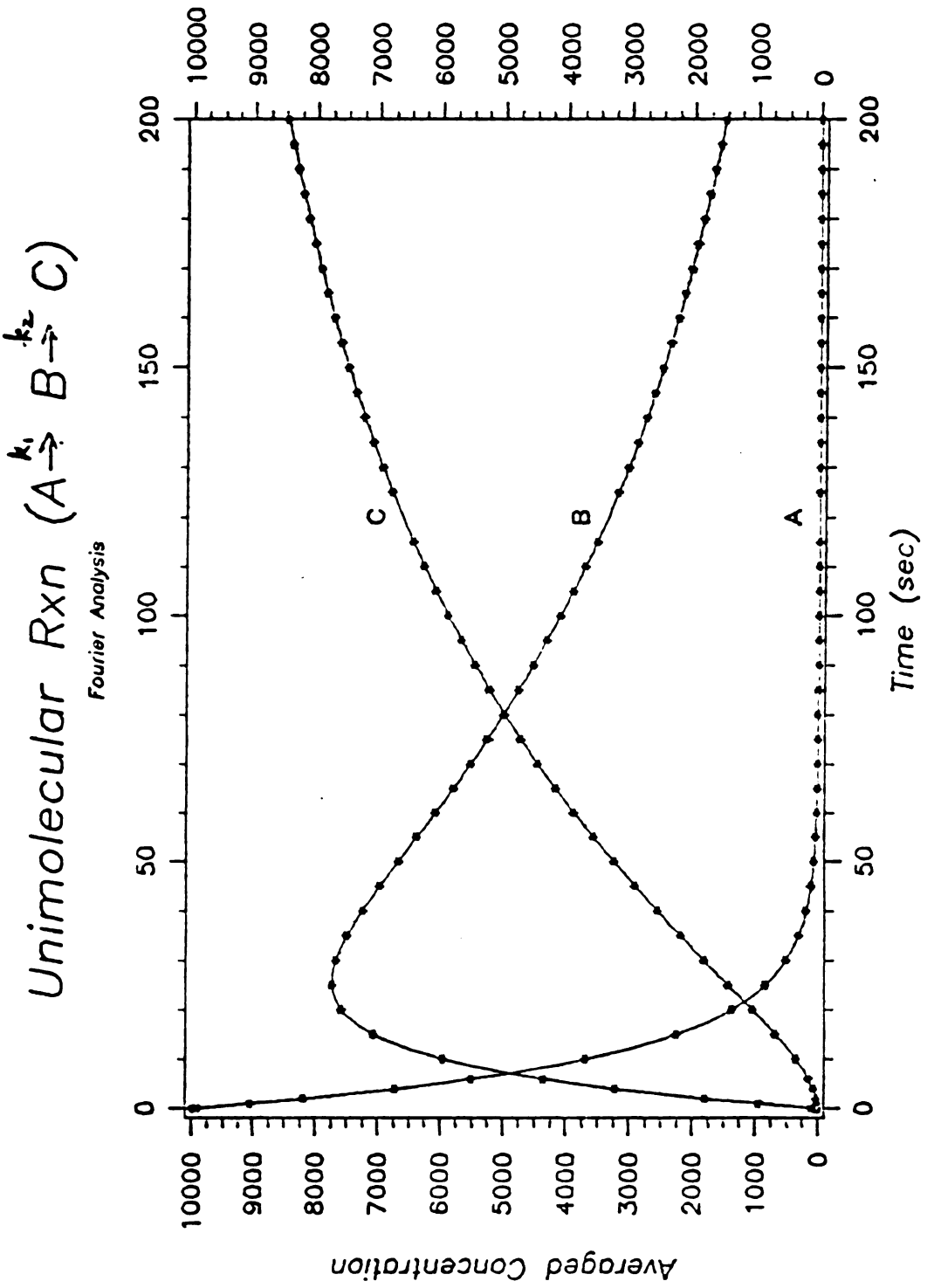


Figure 4.35 The averaged concentrations from the Unimolecular Model.

to this model to examine the sensitivities of the B-concentration would result in sensitivity coefficients similar to the expansion coefficients shown in Figure 4.36. Since the curves are not normalized they are difficult to interpret. One might be tempted to say that since C_1 is at a maximum at 20 seconds measurements in this time region are optimal for the determination of k_1. Similarly C_2 has its most effect on the concentration of B at 90 seconds. Therefore measurements of B near 90 seconds would pin down the k_2 rate constant.

If we examine the partial variances for the B-concentration we clearly get different results. From Figure 4.37 we see that measurements of the B-concentration before 10 seconds have elapsed and after 45 seconds will give accurate estimates of k_1 and k_2 respectively.

The sensitivity of the C-concentration in linear analysis gives curves shown in Figure 4.38. Here we see that, since the k_2 step is a bottleneck, we will have a difficult time measuring k_1 because the effect on C from k_2 is so large. Only at short times are the sensitivities of k_1 significant.

Figure 4.39 is even more revealing. This plot of the partial variances of the C-concentration clearly demonstrates that k_2 is the most important parameter in the model. It also shows that k_1 contributes to the C-concentration only at short times. Therefore to estimate k_1

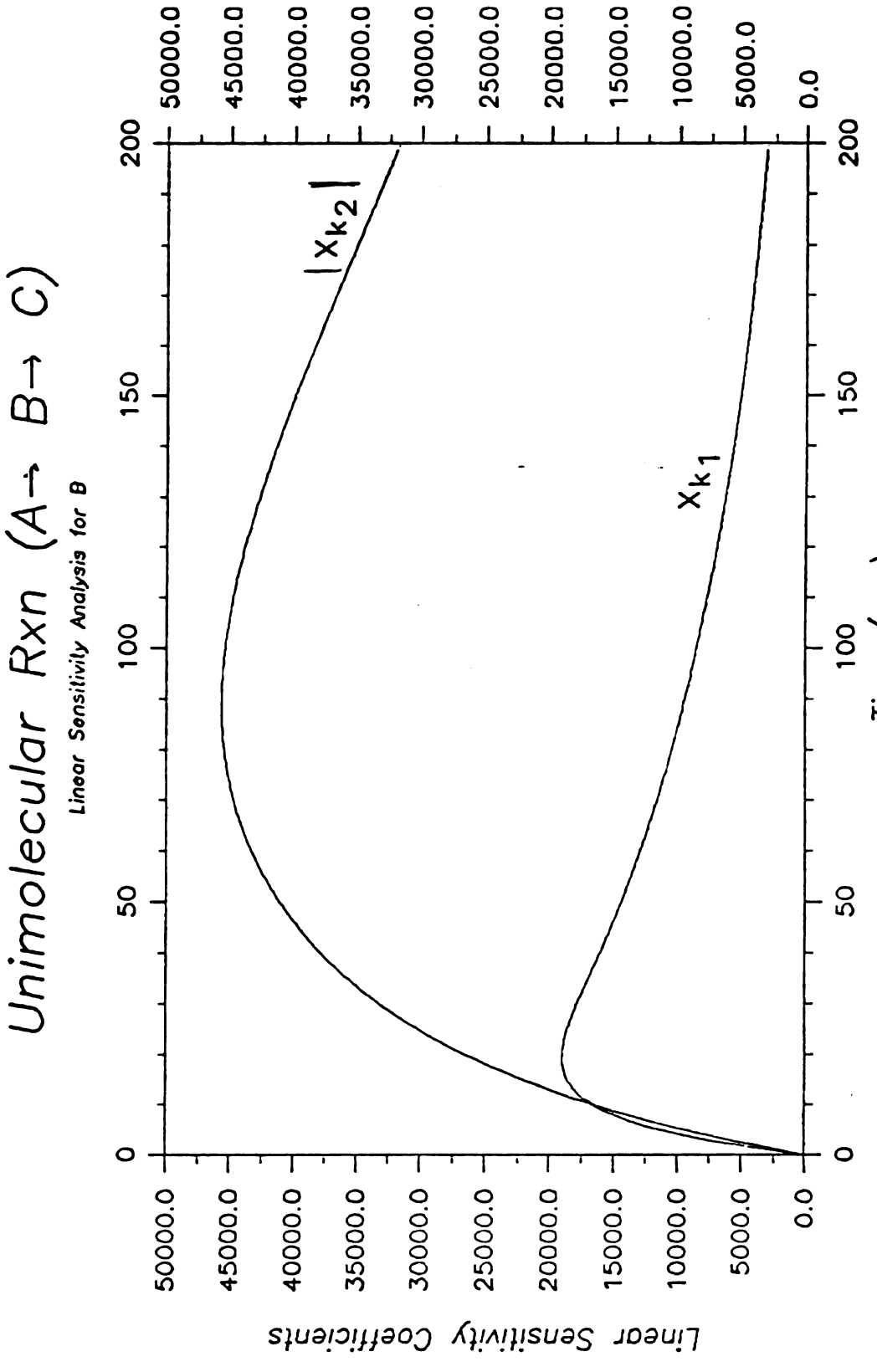


Figure 4.36 Linear sensitivity coefficients for [B] in the Unimolecular Model.

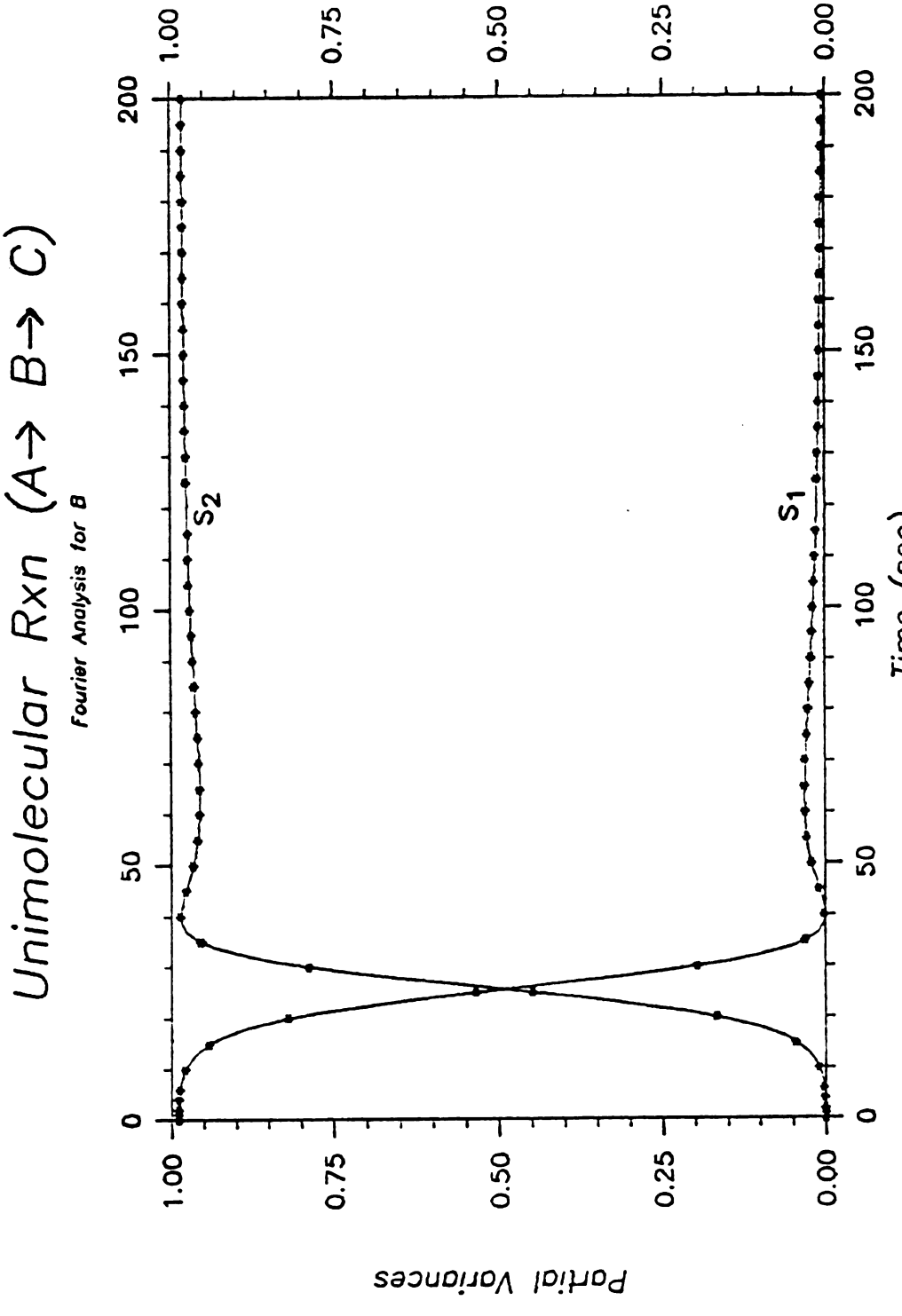


Figure 4.37 Fourier partial variances for [B] in the Unimolecular Model.

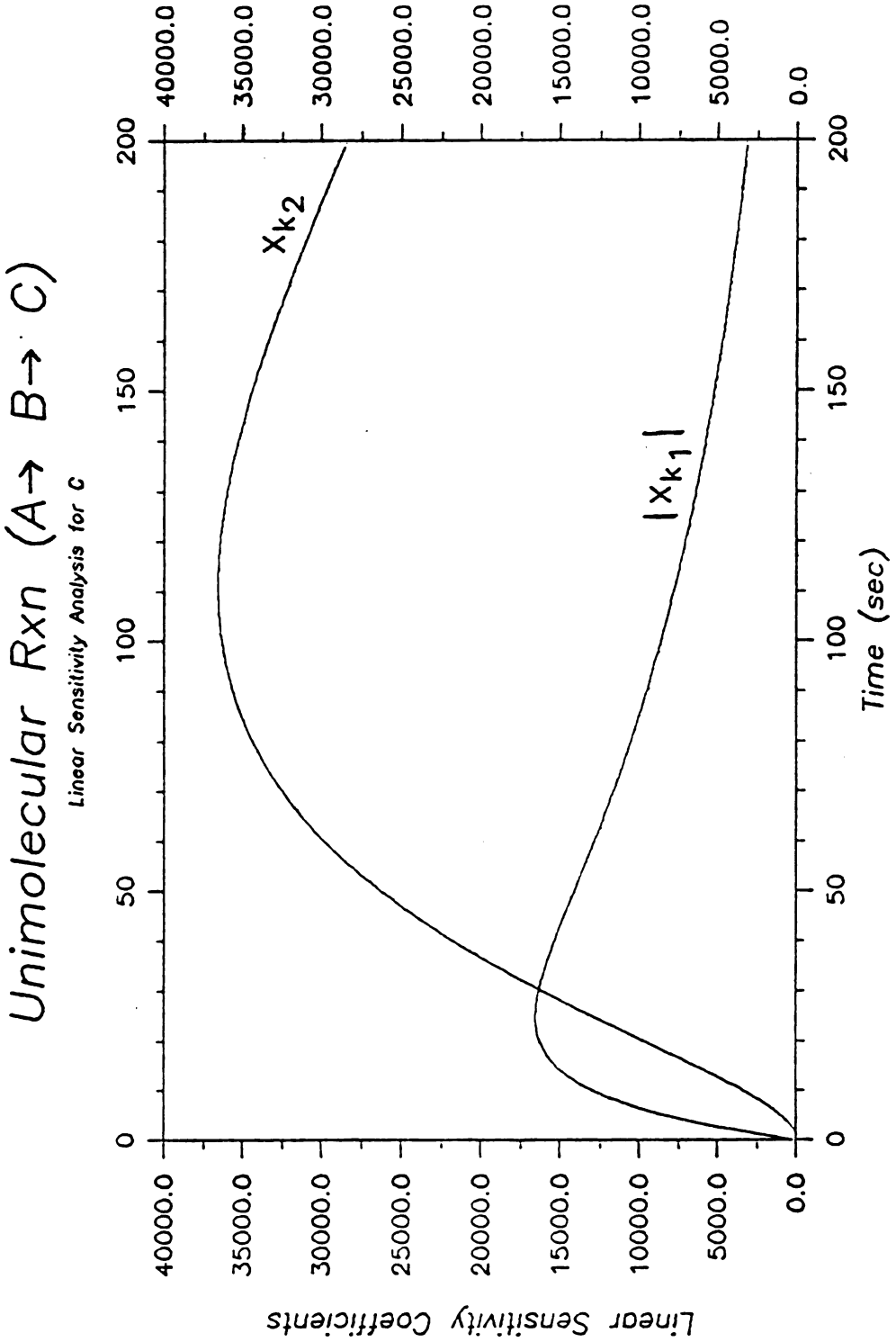


Figure 4.38 Linear sensitivity coefficients for [C] in the Unimolecular Model.

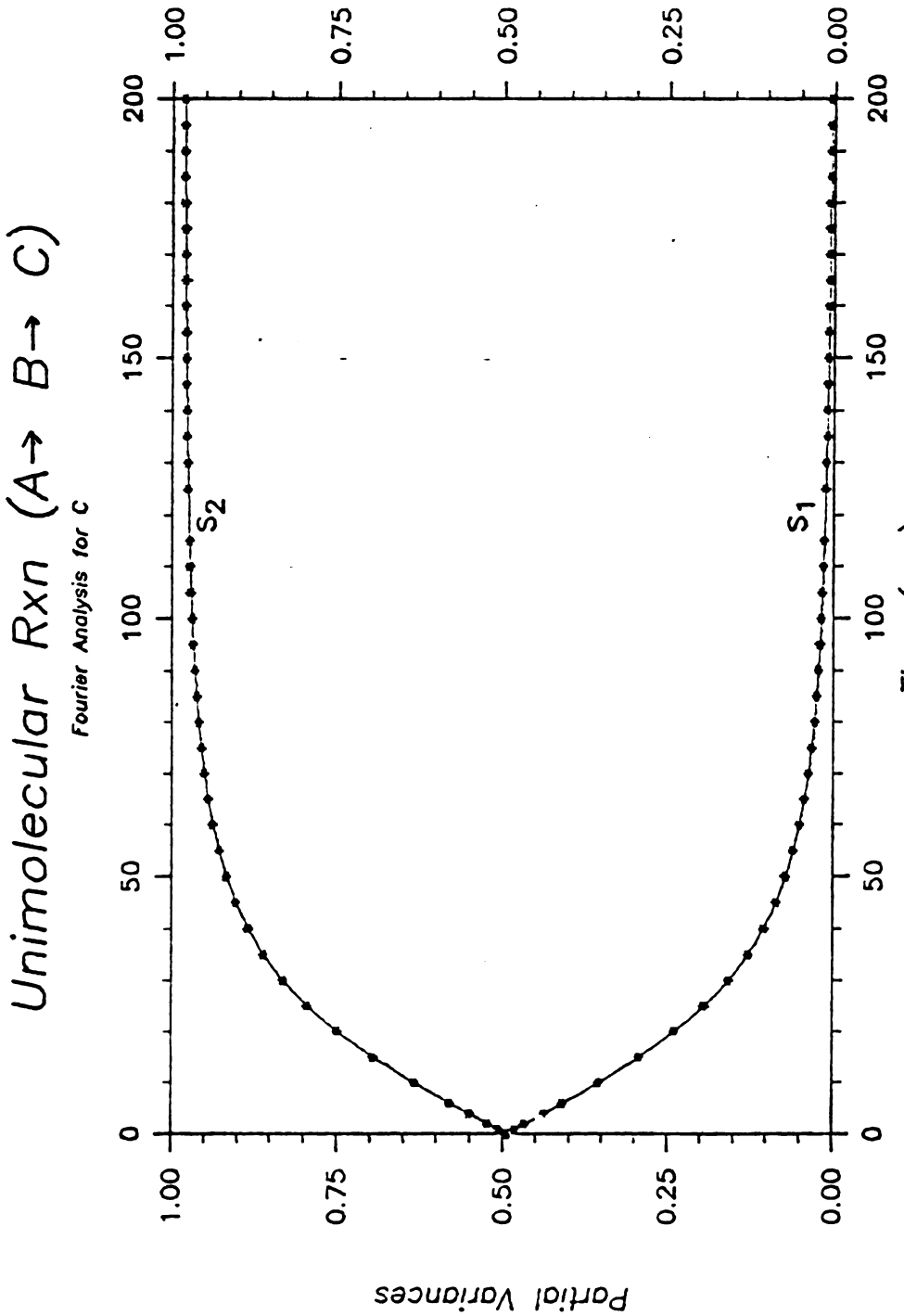


Figure 4.39 Fourier partial variances for [C] in the Unimolecular Model.

from measurements of C we should take the measurements at very short times. To estimate k_2, measurements after 50 seconds are adequate.

Figure 4.40 and 4.41 show the standard deviation and relative deviation curves, respectively, of the C-concentration. From the relative deviation curve we see that we are varying the C-concentration only slightly. Therefore, the Fourier expansion coefficients are equivalent to the linear sensitivity coefficients.

The basic difference in the ease of interpretation of the partial variances over the expansion coefficients is that the partial variances explicitly account for the range of parameter variation by being normalized by the total variance. The linear sensitivity coefficients or their equivalent, the expansion coefficients, do not account for the amount of variability introduced by varying the parameters. This is a great weakness in linear analysis.

A common feature in chemical kinetics models is the occurrence of a competing reaction. This is a reaction in which two steps compete for the same reactant. Which step predominates is dependent on the rate constants for the two steps. A simple scheme for this problem is written

$$\begin{array}{l} A \overset{k_1}{\rightarrow} B \overset{k_2}{\rightarrow} C \\ \\ B \overset{k_3}{\rightarrow} D \end{array} \qquad (4.15)$$

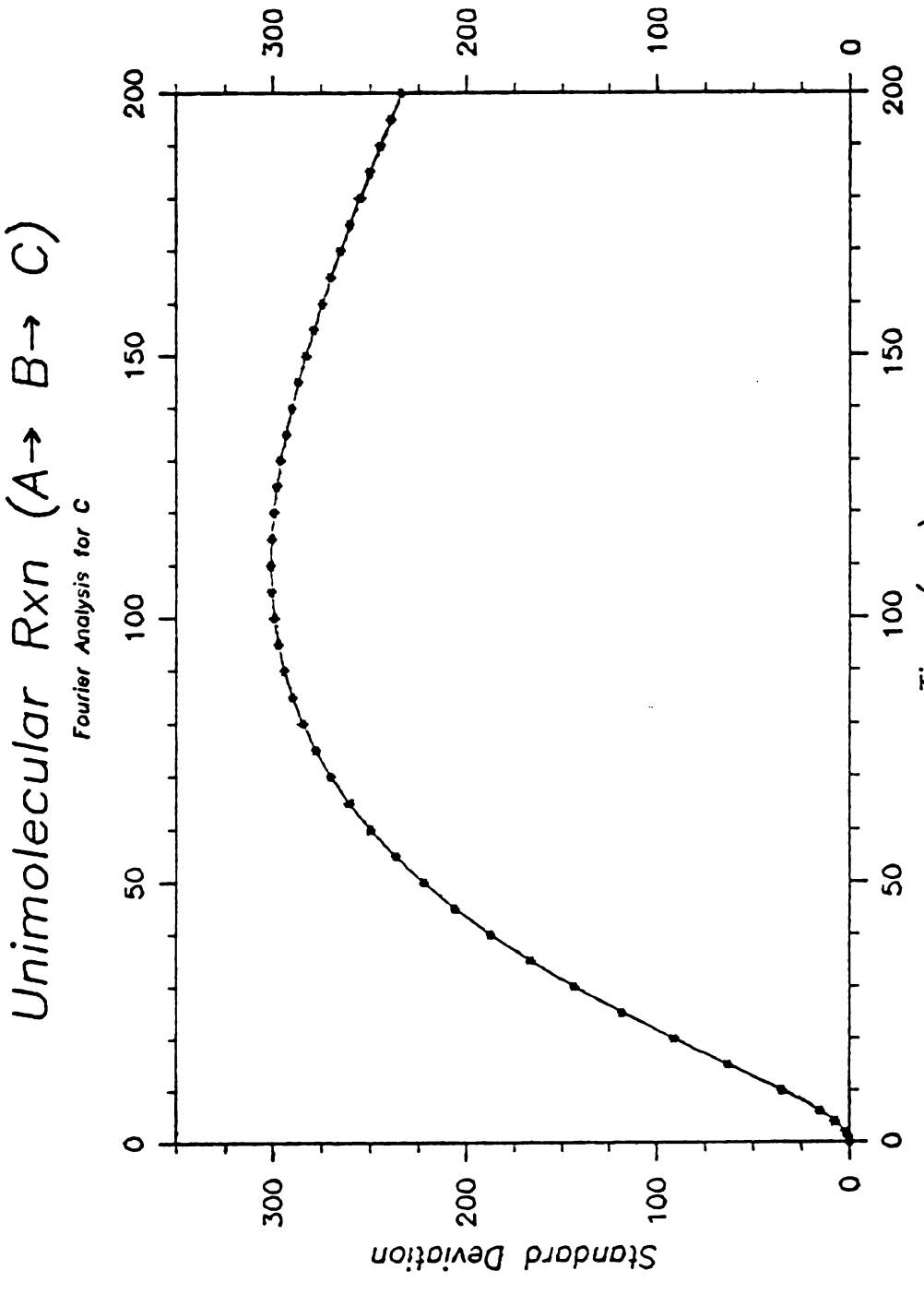


Figure 4.40 The Fourier standard deviation for [C] in the Unimolecular Model.

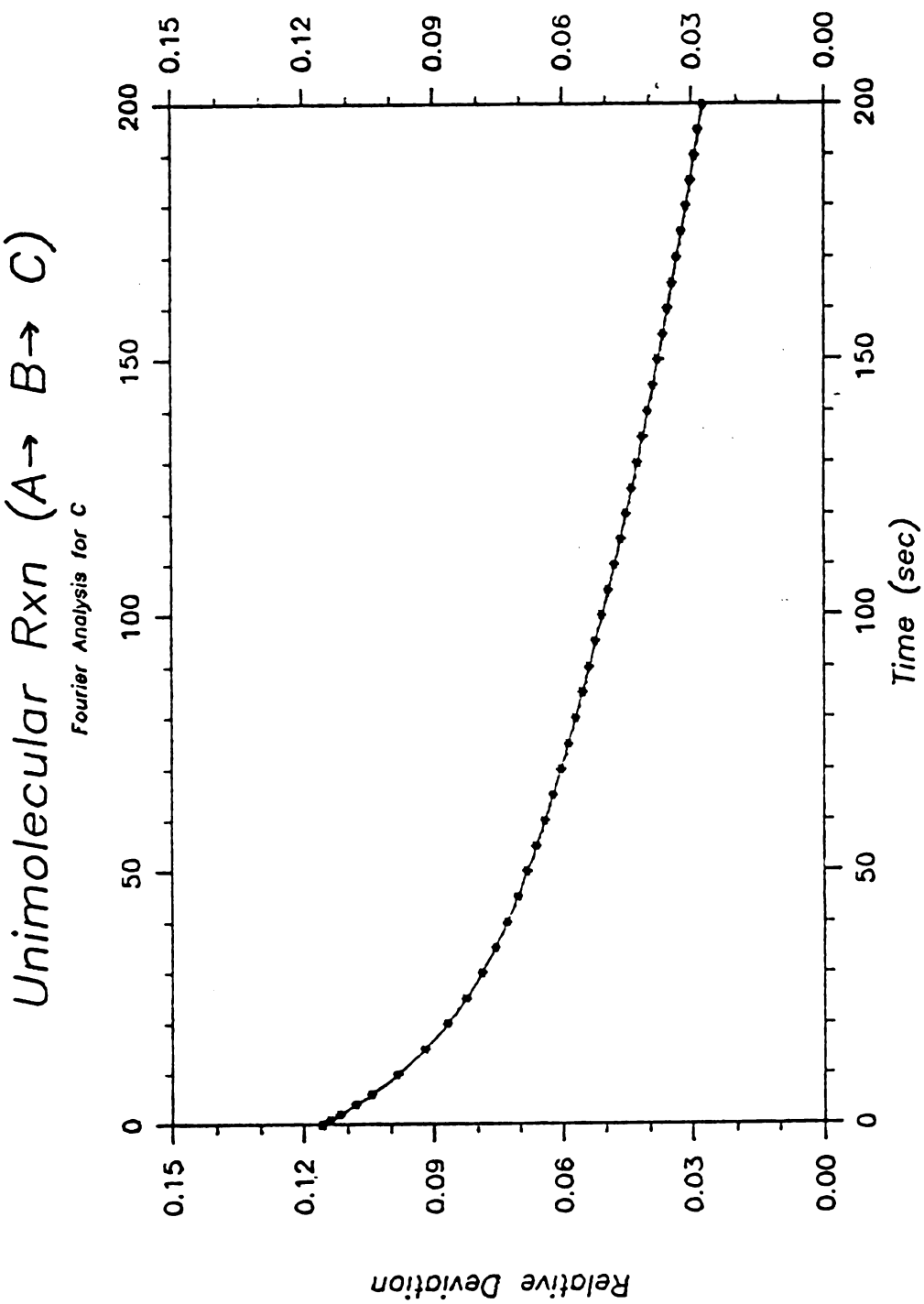


Figure 4.41 The Fourier relative deviation for [C] in the Unimolecular Model.

Here the final products are a mixture of C and D. The ratio of C to D depends on the two rate constants k_2 and k_3. Since this is a reaction problem often found in chemical kinetics models a sensitivity analysis on this scheme was done in order to determine the nature of the partial variances.

Fourier Sensitivity Analysis was applied to this scheme with $k_1 = 0.1 \pm 0.01$ seconds, $k_2 = 0.01 \pm 0.001$, seconds, and $k_3 = 0.003 \pm 0.00003$ seconds with a log-uniform transformation function to vary the rate constants uniformly in log-space. A 6th-order frequency set was used, [9, 15, 19], with 37 simulations.

The averaged concentrations of the four chemical species are shown in Figure 4.42. From this figure it can be seen that the time range chosen covers virtually all of the reaction. Since the concentration of species B both grows and decays it is the most active. Inspection of this model shows that k_2 and k_3 cannot be separated by measuring only the concentration of B. In fact, only the sum $k_2 + k_3$ could be determined. As expected, Figure 4.43 shows that the sensitivity of B to the rate constants is very similar to that of the coupled first-order model, Figure 4.37, since B does not "know" which path it will take and both paths are treated as a single sink. The only difference between this model and the coupled reaction model, with respect to B, is that 10% of the sensitivity

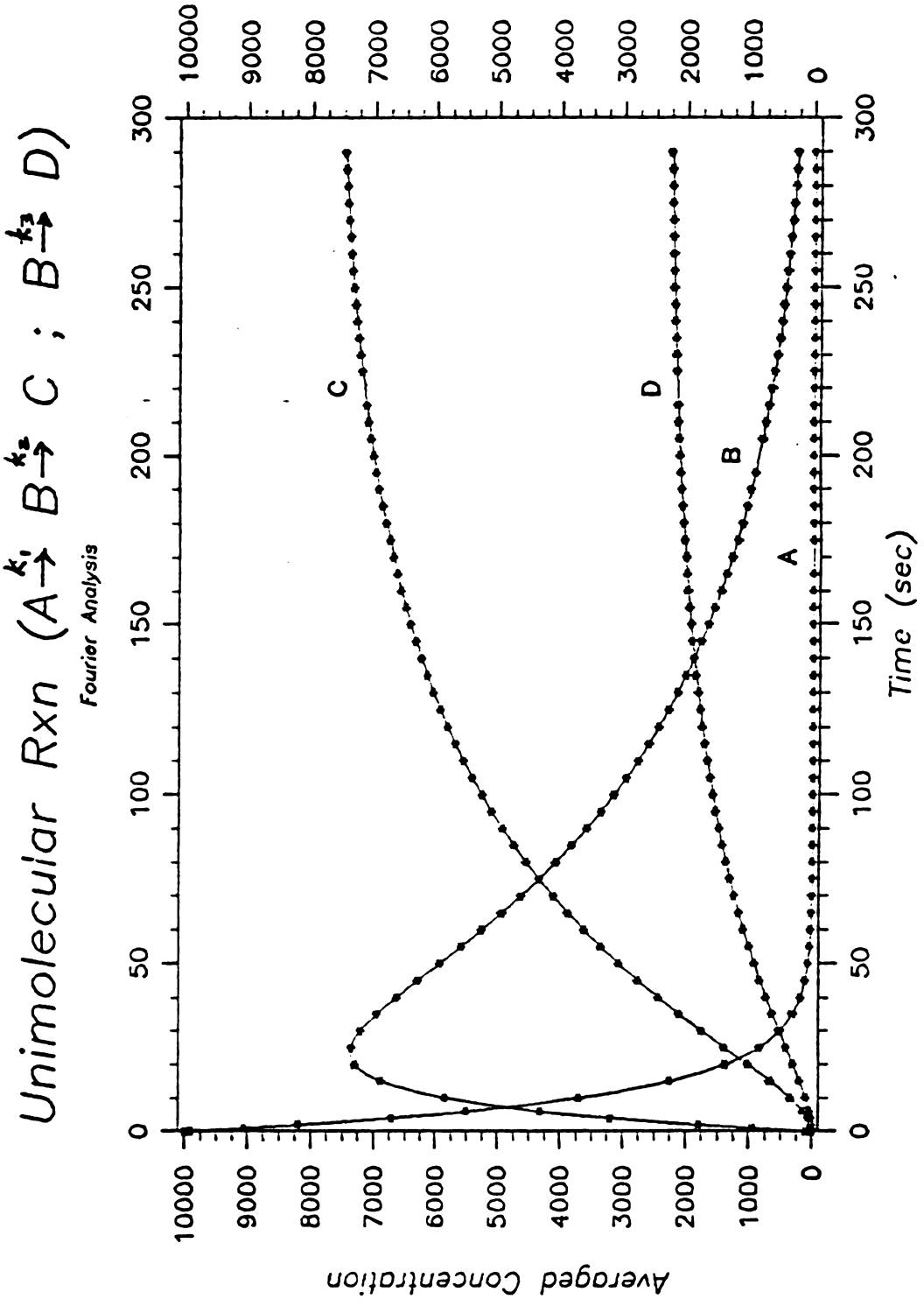


Figure 4.42 The averaged concentrations in the Branched Unimolecular Model.

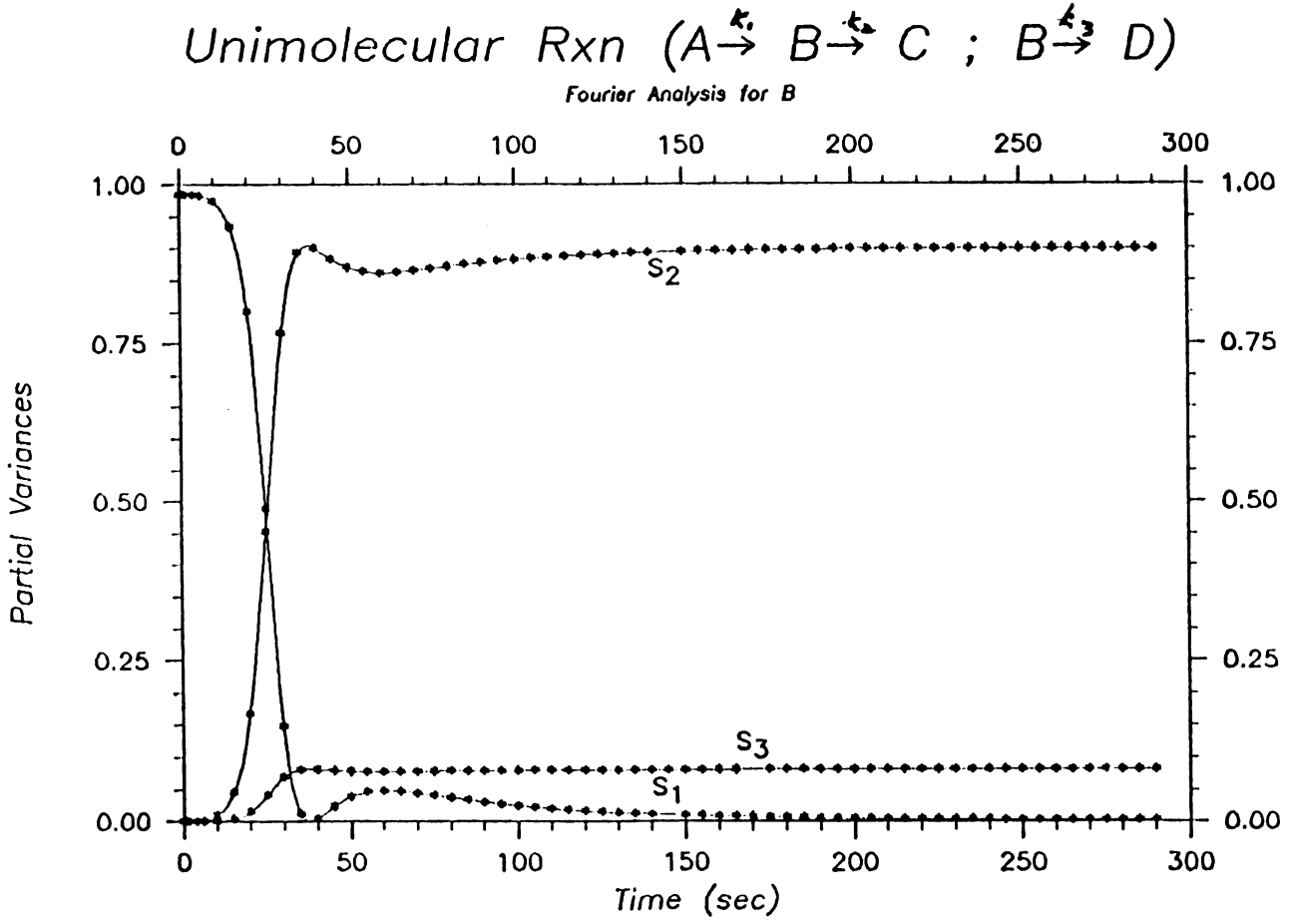


Figure 4.43 Fourier partial variances for [B] in the Branched Unimolecular Model.

to the 'sink' parameter k_2 is given to k_3. This small sensitivity to k_3 which is just the ratio of the nominal rate constants means that any measurements of B over the whole time range even in conjunction with measurements of C or D would be of little help in estimating k_3 since the sensitivity of B to k_2 is so large.

The partial variances of the C concentration, Figure 4.44, are as expected for a competing reaction. Here k_2 is the most important rate constant for C. This is, of course, expected as k_2 controls the only path for the production of the C concentration. Note however that k_3 'accumulates' sensitivity over the time course of the reaction. This is interpreted as showing how k_3 controls, to a lesser extent than k_2, the amount of C produced by the <u>end</u> of the reaction. Hence to estimate k_3 in this model measurements of C near the end of the reaction are required.

The partial variances of the D concentration, Figure 4.45, are subject to similar interpretations. During the first 50 seconds of the reaction, the formation and initial decay of the B concentration, the partial variances of D exhibit the same structures as those of the coupled scheme. Here k_3 controls the concentration of D, and the sensitivity to k_1 quickly decays away as B is created faster than destroyed. In this case the sensitivity to the rate constant from the competing reaction, k_2, accumulates

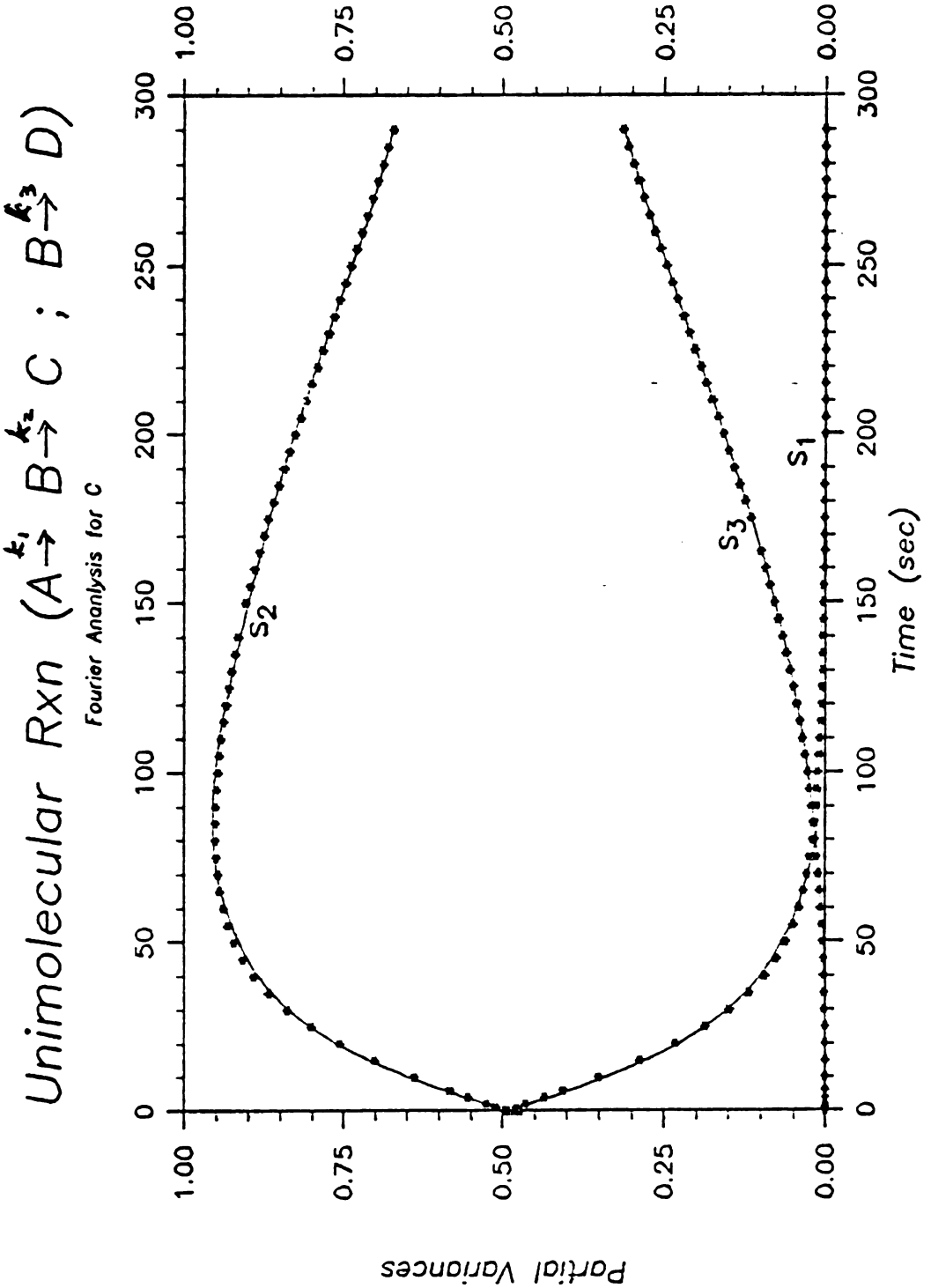


Figure 4.44 Fourier partial variances for [C] in the Branched Unimolecular Model.

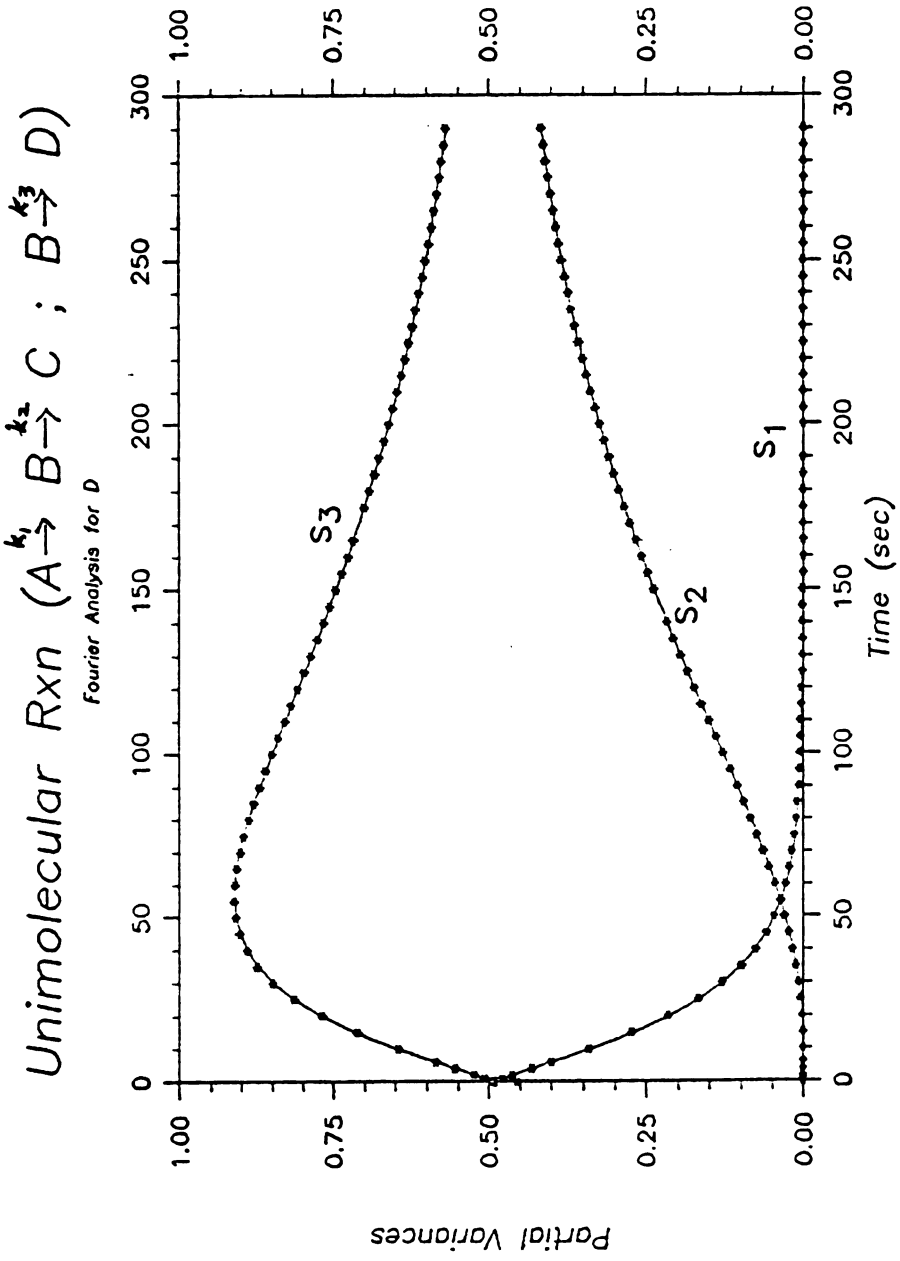


Figure 4.45 Fourier partial variances for [D] in the Branched Unimolecular Model.

faster and to a larger degree than k_3 did in the partial variance of C. This is because k_2 controls a more rapid step which removes B from the pool available to k_3, thus preventing the conversion of a larger amount of B into D. It is then the nominal value of k_2, which is greater than the nominal value of k_3, which causes the sensitivity of k_2 to accumulate faster over the same time range. Note that this model is simple enough to permit conclusions of this type to be made by inspection. However, the verification of these conclusions by sensitivity analysis lends confidence to the treatment of more complex models.

In enzyme kinetics the most commonly used model is the Michaelis-Menten model. This model may be written

$$E + S \underset{k_2}{\overset{k_1}{<=>}} ES \xrightarrow{k_3} E + P \qquad (4.16)$$

Often one starts with an excess of Substrate, S, to enzyme, E, in order to make a steady-state assumption on the concentration of ES. This results in a simplified rate equation for the change in substrate in time, often called the 'velocity' of the reaction (Fersht 1977).

$$\frac{dS}{dt} = \frac{-V_{max}\, S}{K_m + S} \quad S(t=0) = S_0 \qquad (4.17)$$

Where $V_{max} = k_3E_0$ and $K_m = (k_2 + k_3)/k_1$. This equation may be integrated to obtain progress curves of substrate versus time. If substrate is measured as a function of time either the integrated equation or the expression for the velocity could be used to determine K_m and V_{max}.

To determine which equation, the integrated form or the differential form, would be better suited for estimating K_m and V_{max} a Fourier sensitivity analysis was performed. Figure 4.46 shows the averaged values of substrate for this analysis with $K_m = 11000 \pm 110$ and $V_{max} = 50 \pm 5$ and $S_0 = 11000 \pm 110$. The relative deviation curve given in Figure 4.47 shows that the substrate was only varied over a small range by using these parameters.

Figure 4.48 shows the partial variances of the substrate. The first parameter, V_{max}, is the most important parameter over this entire time range with the second parameter, K_m, being much less important and the third parameter, S_0, even less important.

For the velocity equation we get different results as shown in Figure 4.49. Here at long times the K_m parameter is twice as important as it was in the integrated equation. Therefore to measure K_m we should take velocity data at long times and fit to the velocity equation. If estimation of V_{max} is our only concern then use of the integrated equation with measurements during the initial phase of the reaction is sufficient. Note

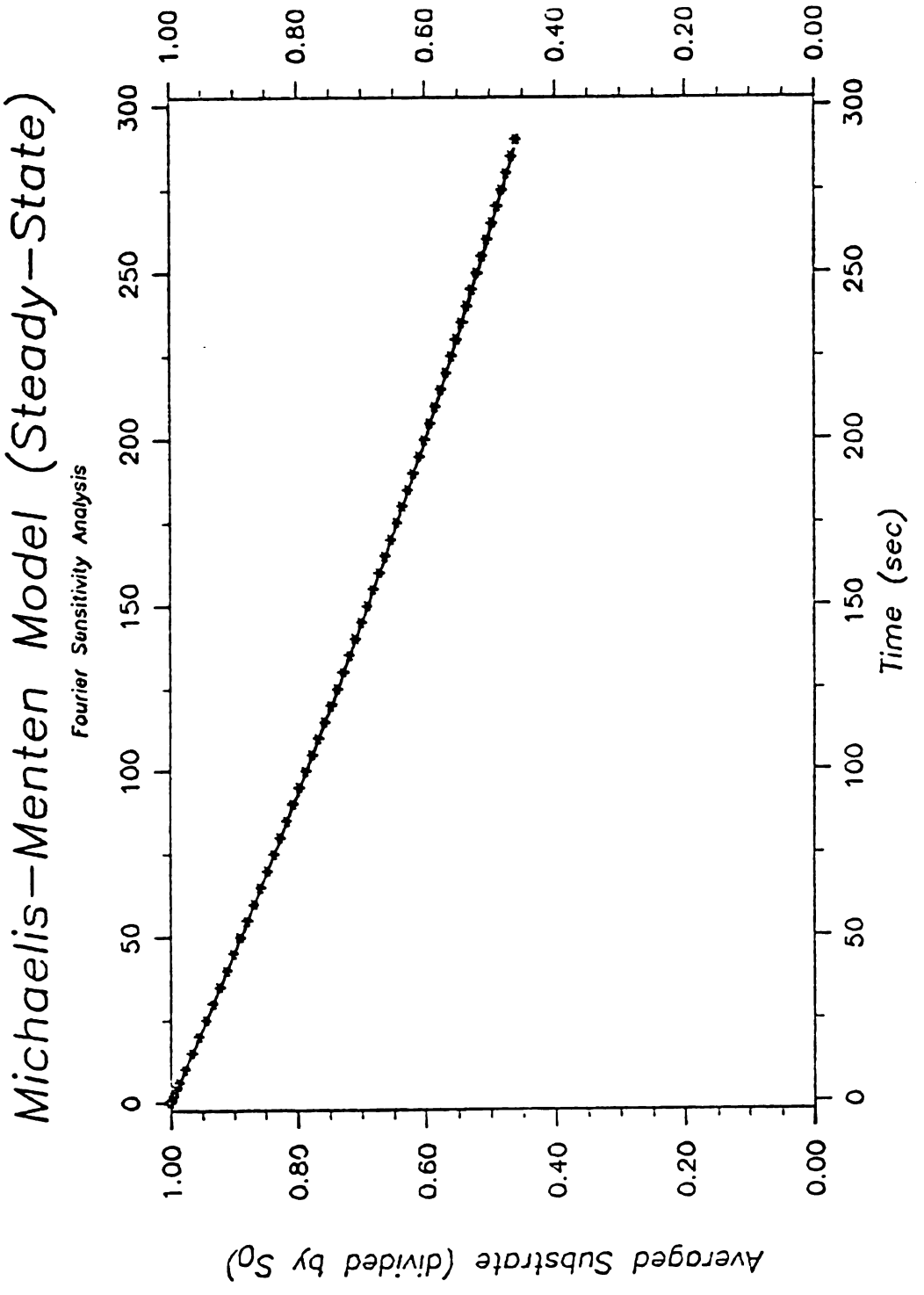


Figure 4.46 The averaged Substrate concentration in the Michaelis-Menten Model (1% variation).

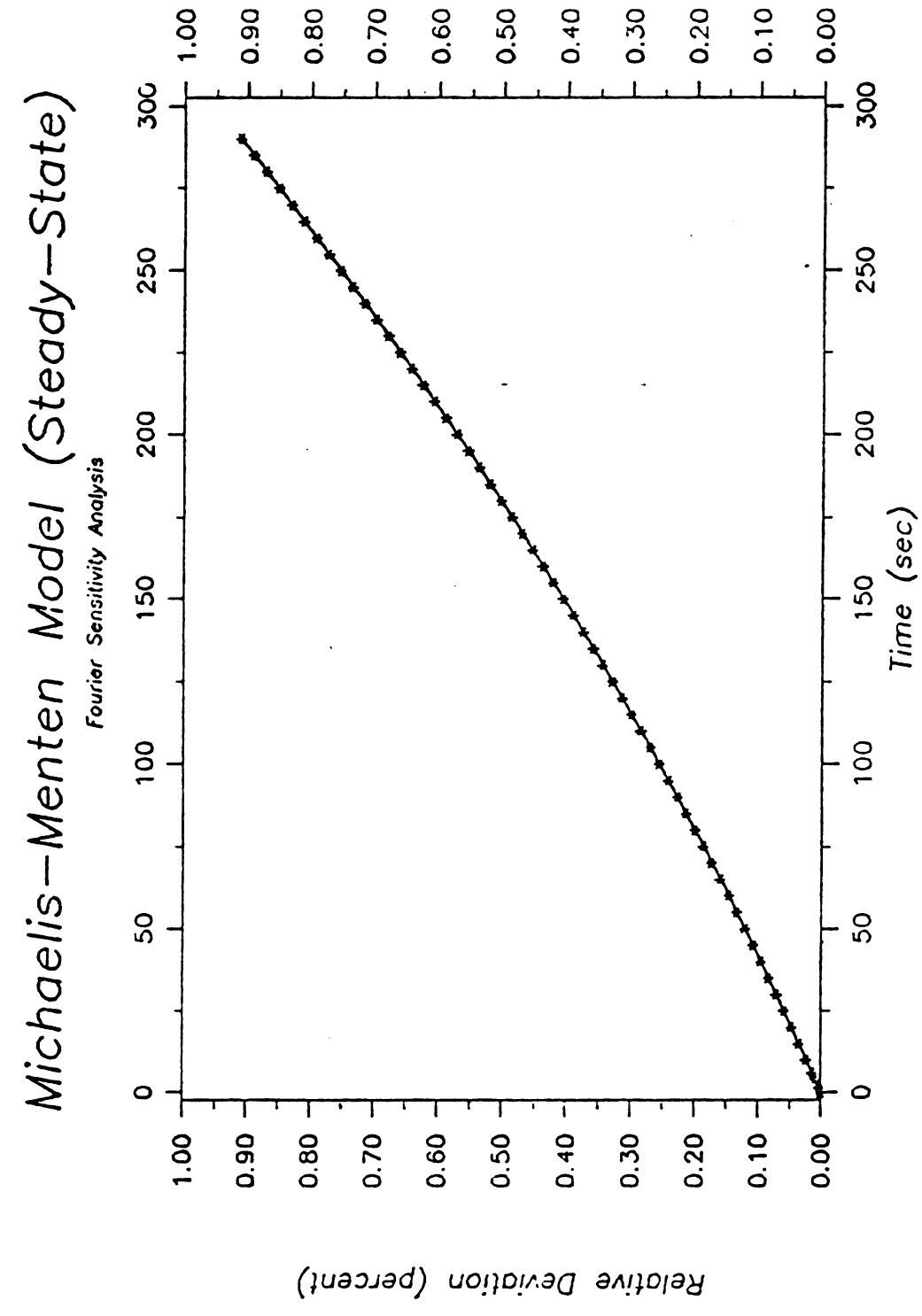


Figure 4.47 The relative deviation of Substrate in the Michaelis-Menten Model (1% variation).

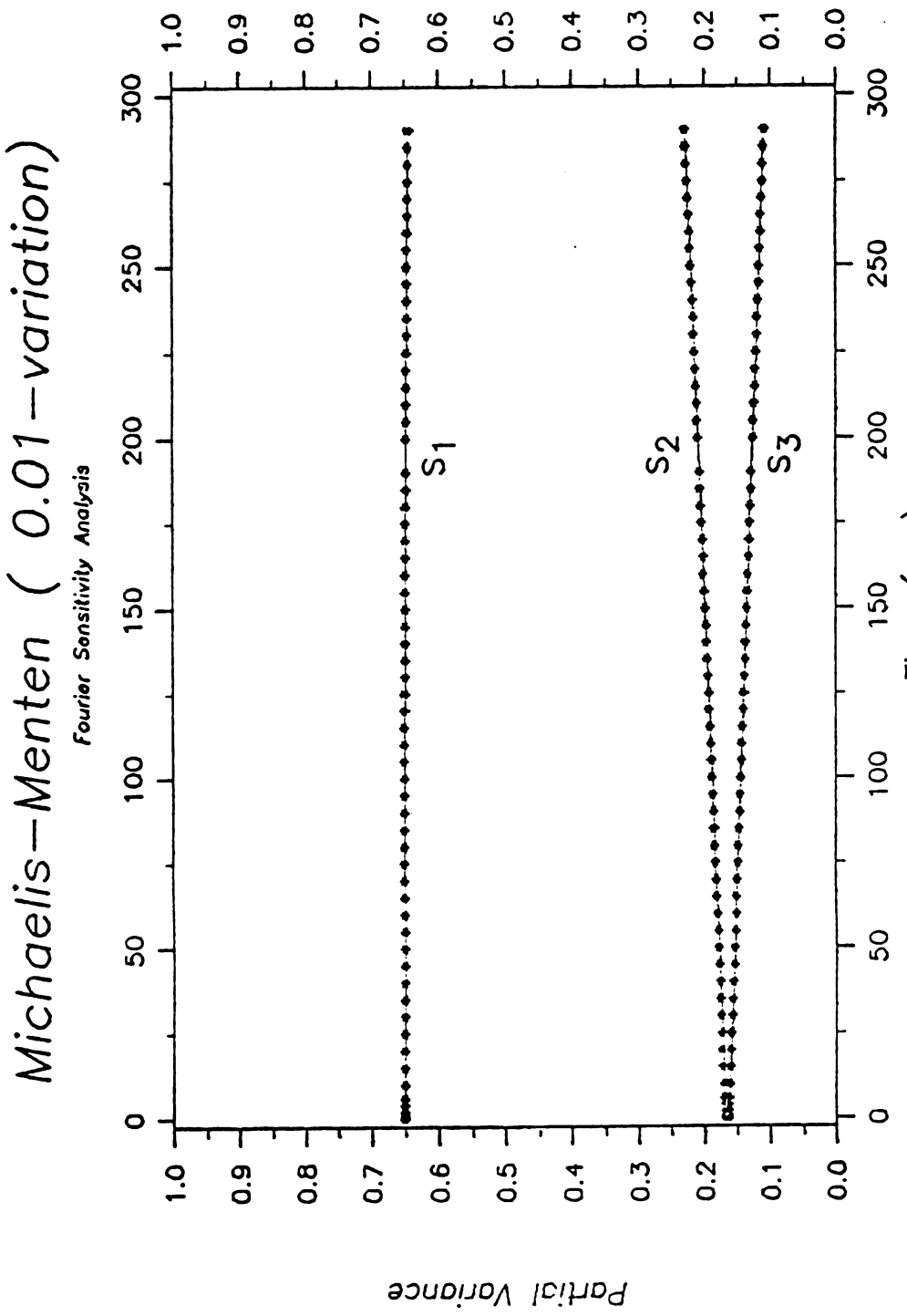


Figure 4.48 Partial variances of the rate constants for Substrate in the Michaelis-Menten Model (1% variation).

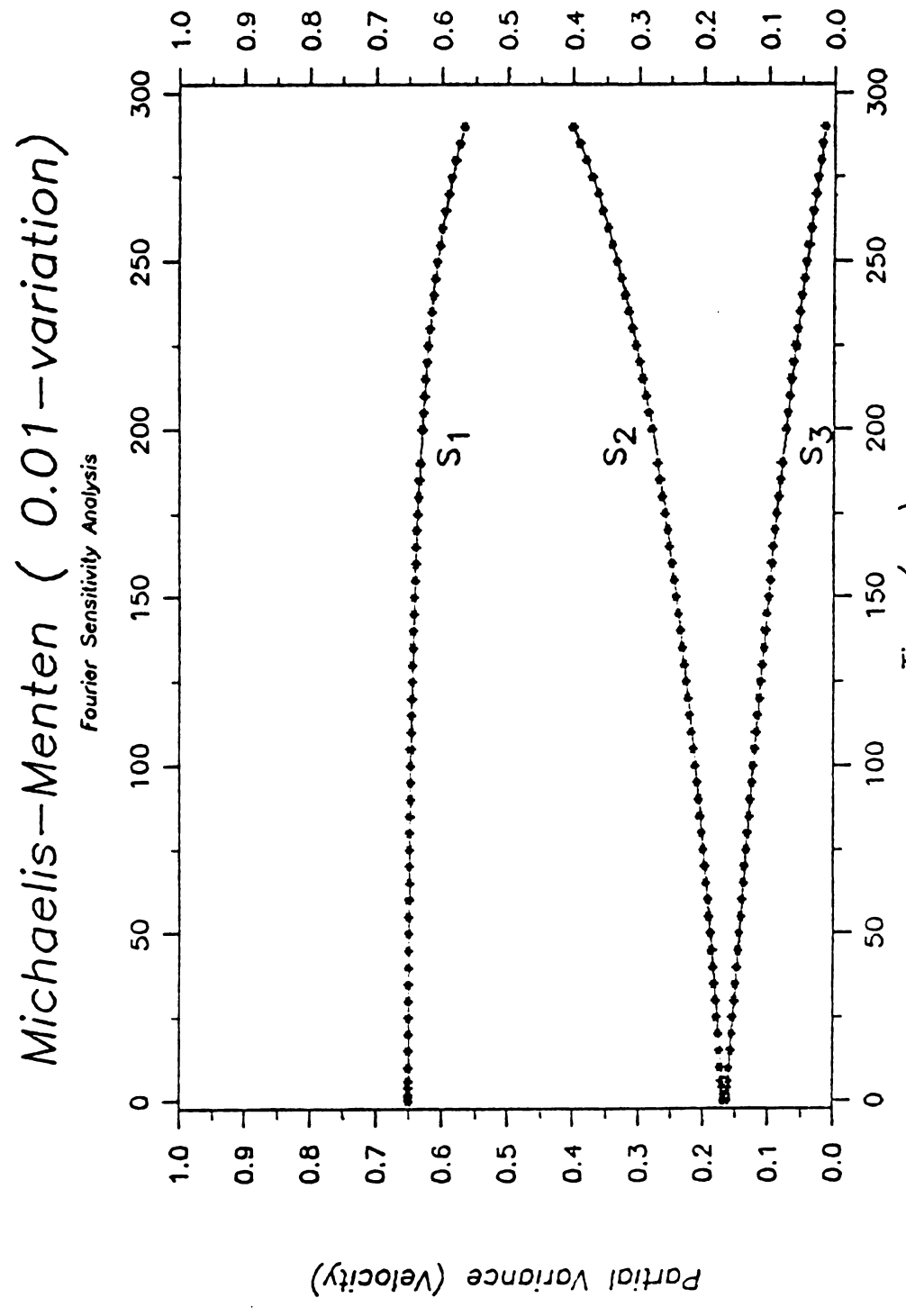


Figure 4.49 Partial variances of the rate constants for Velocity in the Michaelis-Menten Model (1% variation).

also that the use of the Michaelis-Menten model to estimate the initial concentration of substrate, S_0, requires that the other two parameters, K_m and V_{max}, be previously known since the model is more sensitive to these parameters.

Repeating the analysis over a larger parameter range (80% of the nominal value) resulted in essentially the same observations, Figures 4.50-4.52. S_0 is still the least sensitivity parameter as shown by the partial variances of the substrate, Figure 4.51. The velocity equation appears to be the more sensitive formulation of the Michaelis-Menten model. This is dramatically shown by the partial variances in Figure 4.52. V_{max} is the most important parameter in this section of parameter space. Even its couplings with K_m and V_{max} are more important than K_m or V_{max} in the region where the reaction has gone to 40% completion.

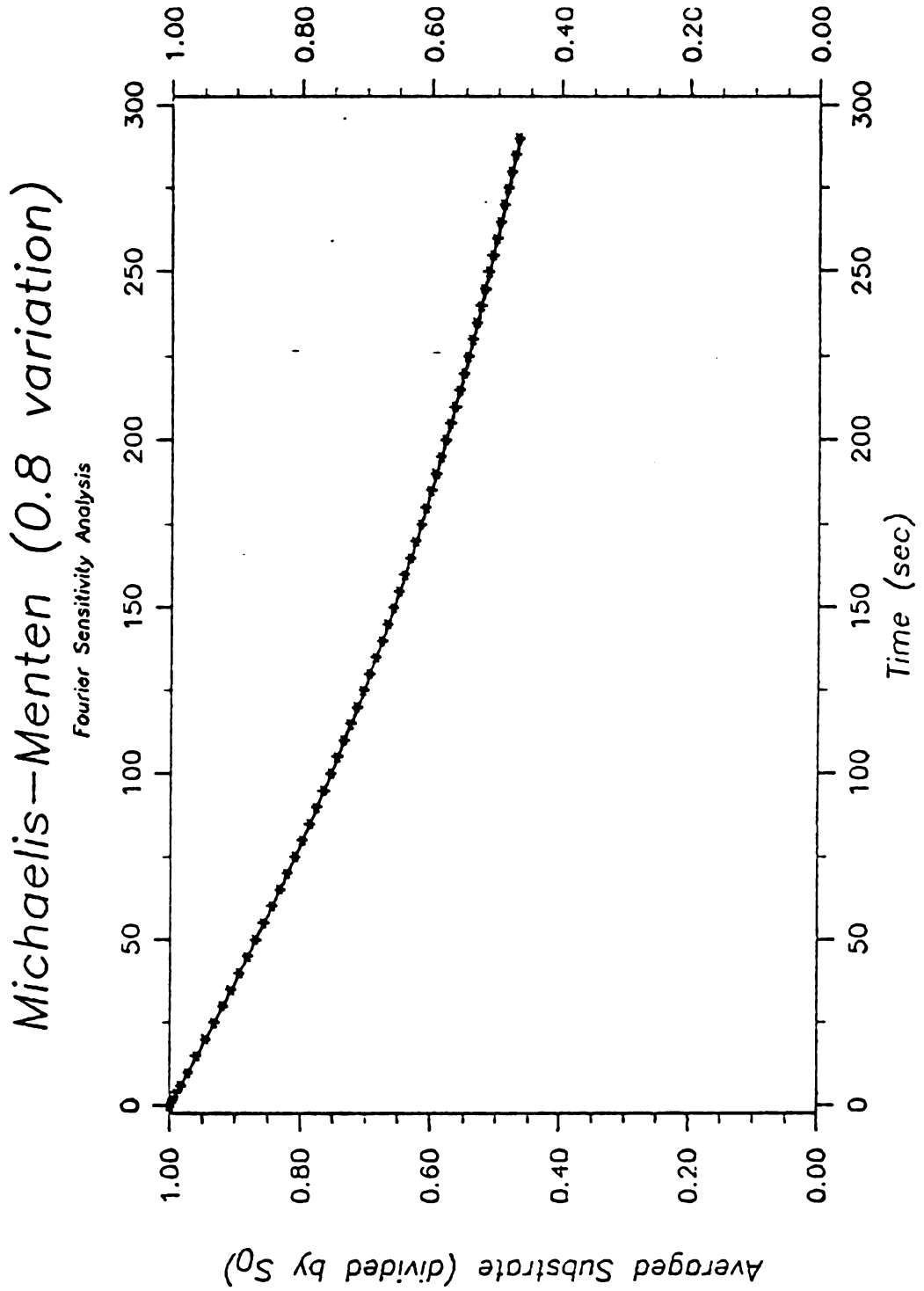


Figure 4.50 Averaged Substrate concentration in the Michaelis-Menten Model (80% variation).

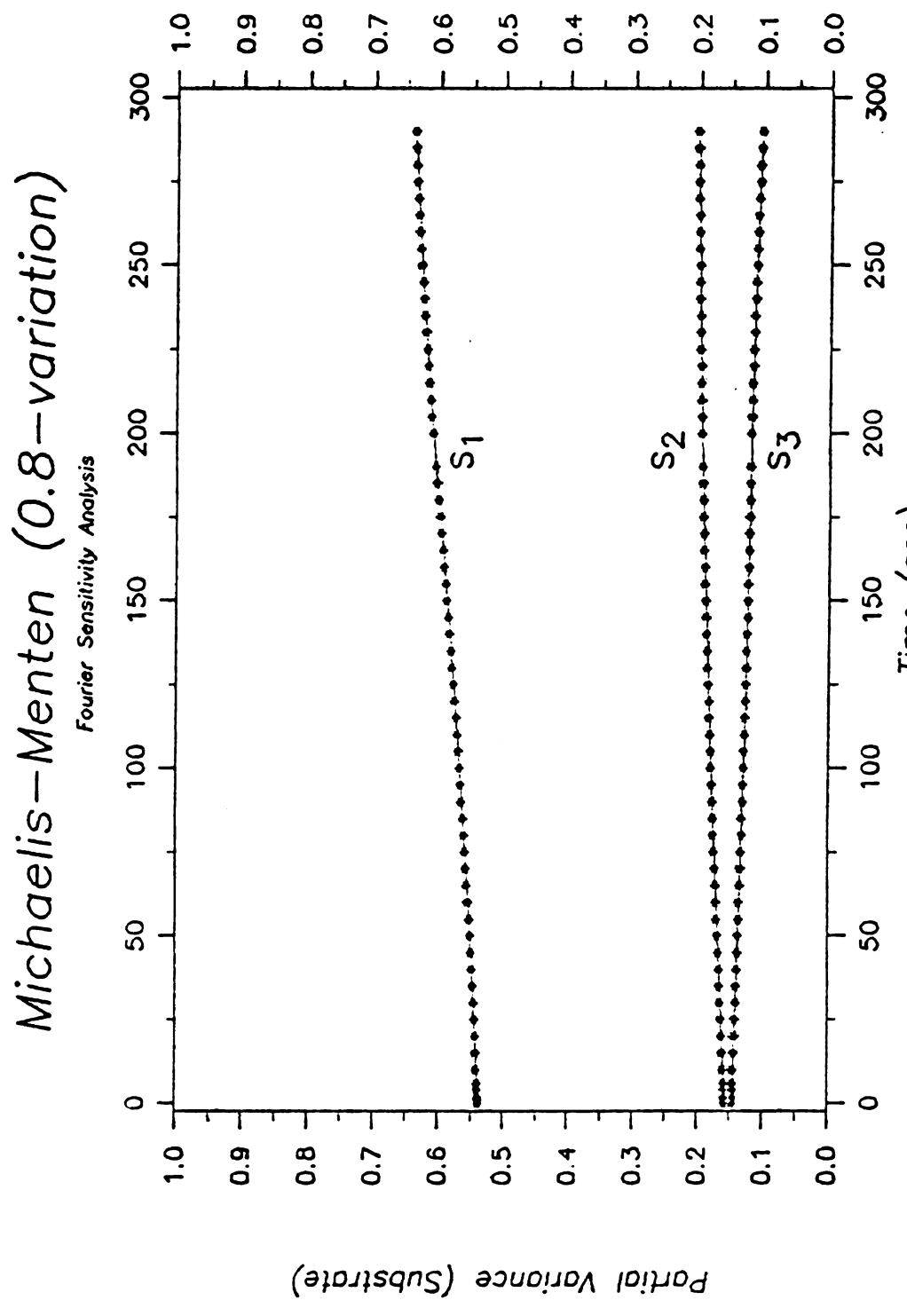


Figure 4.51 Partial variances for Substrate in the Michaelis-Menten Model (80% variation).

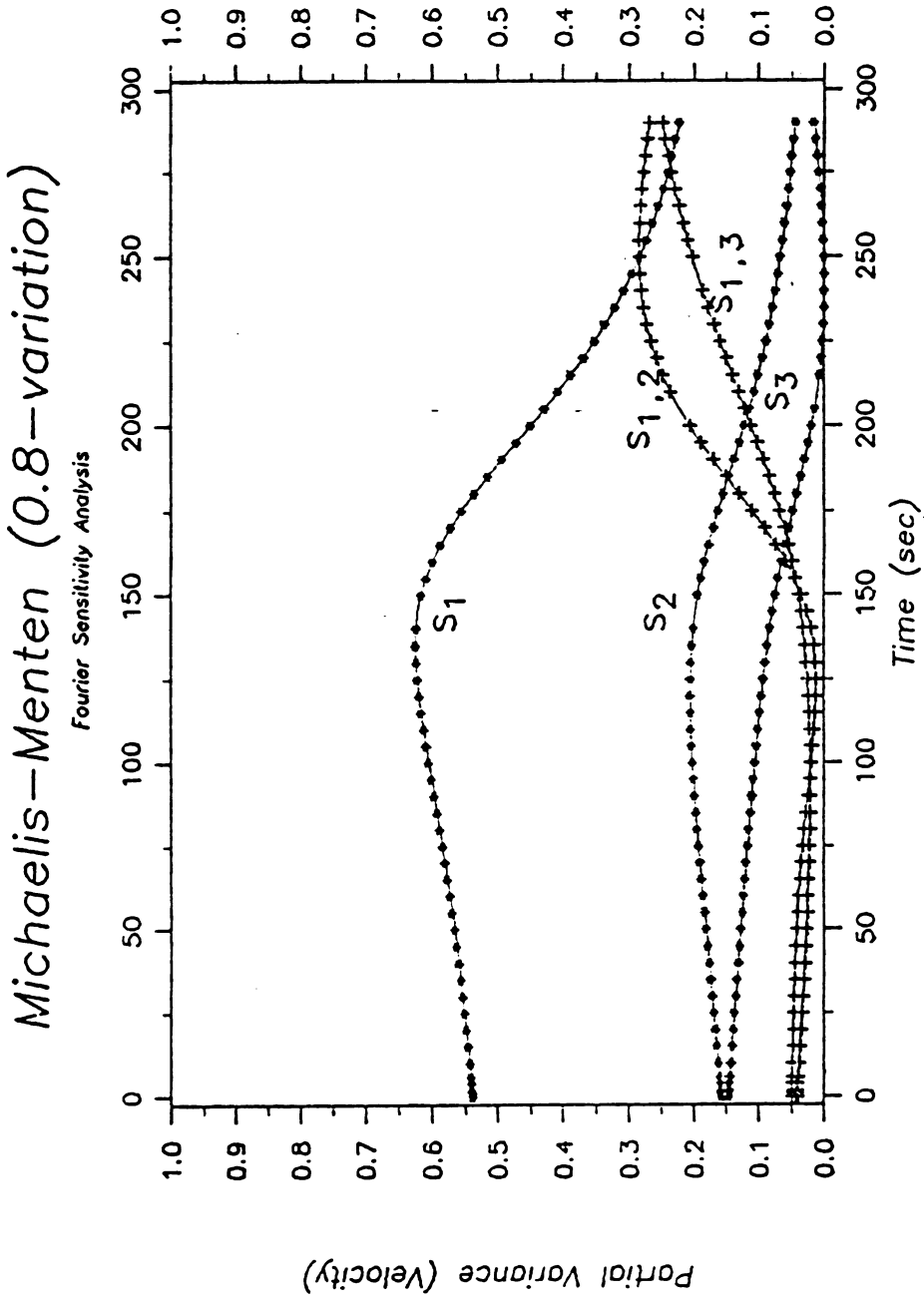


Figure 4.52 Partial variances for Velocity in the Michaelis-Menten Model (80% variation).

V. SENSITIVITY ANALYSIS OF SIMPLE ENZYME KINETICS MODELS

Many enzyme kinetics models are composed of interlocked Michaelis-Menten models (Segel 1975). These models are proposed to explain kinetic behavior patterns which the single Michaelis-Menten model alone is unable to do (Segel 1975). This behavior is even defined as "non-Michaelis-Menten" (Whitehead 1970). It was our desire to examine models which presumably exhibited non-Michaelis-Menten behavior patterns and were composed of linked Michaelis-Menten models. In this chapter we investigate four enzyme kinetics models, the irreversible Michaelis-Menten model, the reversible Michaelis-Menten model (Michaelis et al. 1913), the Ho-Frieden model (Ho 1976, Bates & Frieden 1973), and the Ainslie, Shill, and Neet model (Ainslie et al 1972). Of course, to fully understand the linked Michaelis-Menten models we must first examine the Michaelis-Menten model itself.

Michaelis-Menten Model

The simplest model used in enzyme kinetics is the irreversible Michaelis-Menten model:

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_3} E + P. \qquad (5.1)$$

In order to illustrate the utility of the partial variances to evaluate the sensitivities of concentrations to the rate constants, we applied the Fourier sensitivity analysis method, FSAM, to this simple model. Although the range of rate constants used and the substrate and enzyme concentrations selected insure that steady-state conditions are established very rapidly, the model was solved numerically without including any steady-state or equilibrium assumptions. Of course, in this situation, if one could observe only the substrate or product concentrations, it would be possible to determine only k_3 and $(k_2 + k_3)/k_1$ since the steady-state assumption yields [S] and [P] in terms of these two "constants". We use here the time-development of (E), (S), (ES), and (P) in terms of k_1, k_2, and k_3 in order to illustrate the method and permit comparison with more complex models.

Examination of the range of values of k_1, k_2, and k_3 tabulated (Fersht, 1977) for a variety of enzyme reactions which follow Michaelis-Menten kinetics shows that most lie within an interval of four orders of magnitude centered on the nominal values listed in Table 1. Because FSAM was designed to apply to situations with arbitrarily

Table 5.1. Parameter Values for the Irreversible Michaelis-Menten Model.

	$k_i^{(o)}$	Δk_i	
k_1	1.0 $(\mu M\ sec)^{-1}$	$10^{\pm 2}$	nominal $K_M^{(o)} = \frac{k_2^o + k_3^o}{k_1^o} = 11000\ \mu M$
k_2	$10^4\ sec^{-1}$	$10^{\pm 2}$	
k_3	$10^3\ sec^{-1}$	$10^{\pm 2}$	$1.1\ M \leq K_M \leq 1.1 * 10^3\ \mu M$
	$S_o = 11{,}000\ \mu M$ $E_o = 0.05\ \mu M$	(Assay Conditions)	

Table 5.2. Parameter Values for the Reversible Michaelis-Menten Model.

	k_i^o	Δk_i	
k_1	1.0 $(\mu M\ sec)^{-1}$	$10^{\pm 2}$	
k_2	$10^4\ sec^{-1}$	$10^{\pm 2}$	$10^{-9} \leq K_T^o = \frac{k_1^o\ k_3^o}{k_2^o\ k_4^o} = 0.1 \leq 10^7$
k_3	$10^3\ sec^{-1}$	$10^{\pm 2}$	
k_4	1.0 $(\mu M\ sec)^{-1}$	$10^{\pm 2}$	$S_o = 11{,}000\ \mu M$
			$E_o = 0.05\ \mu M$

large ranges in the rate constants, it was possible to explore the sensitivities of the concentrations to the three rate constants while each was allowed to vary independently by up to four orders of magnitude. The rate constant rang ranges and initial conditions given in Table 1 were used in these simulations. The initial concentrations correspond to "assay conditions" ($S_0 \gg E_0$). It is important to note that the equilibrium constant $K_1 = k_1/k_2$ was <u>not</u> held constant when the rate constants were varied. This permitted exploration of the overall sensitivity of the model to a range of maximum velocities which covered four orders of magnitude and a range of Michaelis constants which spanned eight orders of magnitude. It would also be possible to test a more restricted model by fixing the equilibrium constant as is done later for more complex models.

In Figures 5.1a and 5.1b we have plotted the average concentrations, which are the concentrations summed over all the different rate constant sets divided by the number of simulations, and the standard deviations [square root of the total variance defined in Equation (2.21) for the irreversible Michaelis-Menten Model]. All these curves are scaled to the percent of the total enzyme concentration E_0 for enzymatic species and to the percent of the initial substrate concentration, S_0, for the product and substrate. Two concentrations of the four are linearly

Figure 5.1. Average concentrations and standard deviations of the concentrations Michaelis-Menten Models. The symbols represent: ◇, S; ○, P; Δ, E; +, ES.

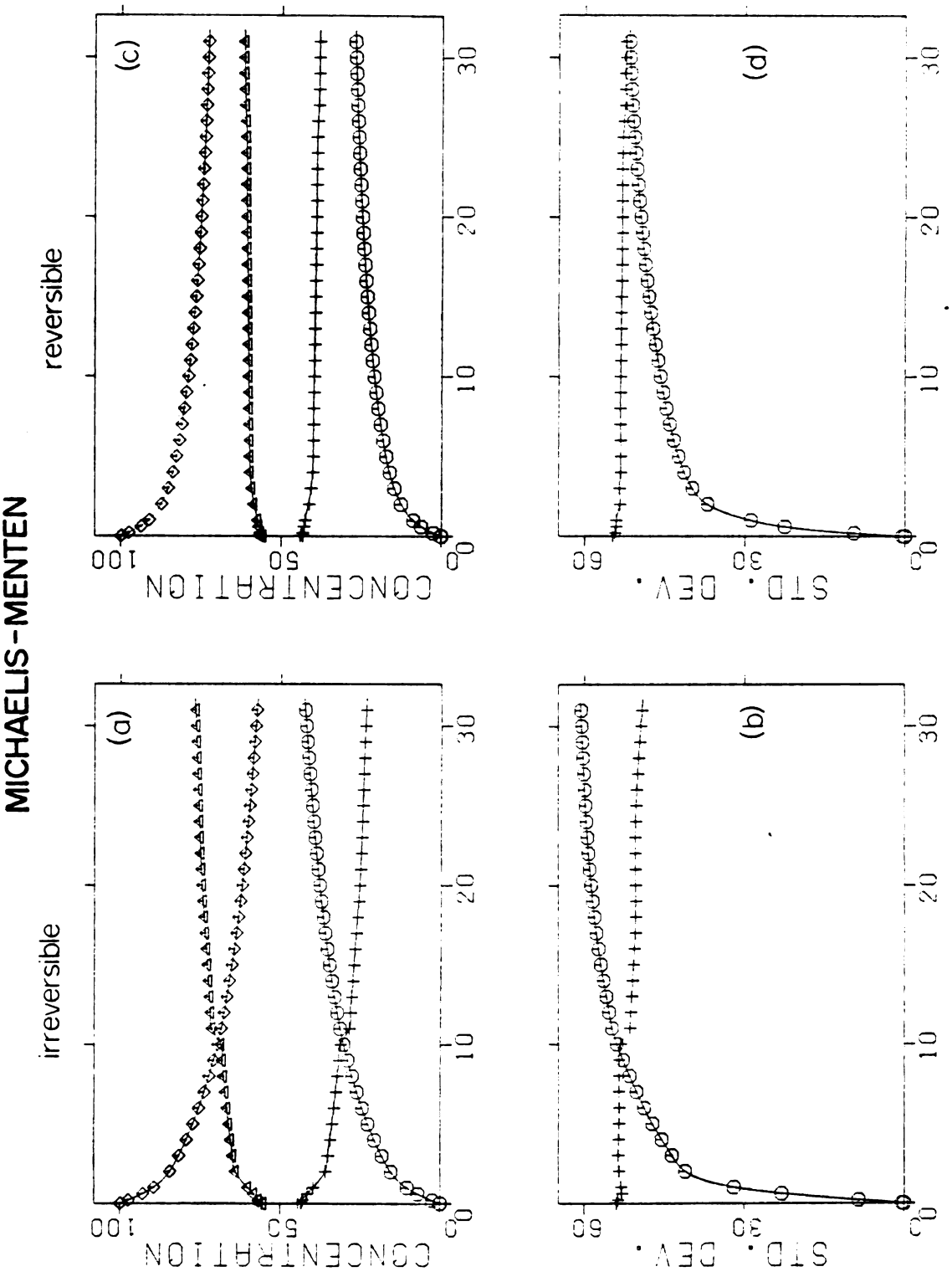

MICHAELIS-MENTEN
irreversible
reversible
(a)
(b)
(c)
(d)
CONCENTRATION
STD. DEV.
TIME (sec) x 10⁻¹

Figure 5.1

related by the mass balance equations. Hence only two standard deviations, those of P and ES, are shown in Figure 5.1b.

Figure 5.1a shows that on the average the concentration of substrate is decreased by only 57% in 300 seconds. However, the rapid growth of the standard deviation curve for substrate, shown in Figure 5.1b, indicates that a large spread of calculated concentrations would be rapidly attained for this range of rate constants. In fact, the wide range of concentrations of substrate becomes so pronounced that a substantial number of simulations go to completion after 20 seconds and another group after 110 seconds. This returns the enzyme concentration to its initial value for these simulations, and results in apparent breaks in the curves of the total standard deviation for E and ES. The origin of this effect will be more fully discussed in connection with the partial variances.

In Figure 5.2a we have plotted the sensitivity of the product concentration to the uncertainties in the three rate constants as a function of time. The values of the partial variances S_1, S_2, S_3 indicate that for this range of rate constants, the product concentration depends most strongly on the value of k_3, the rate constant for the formation of product from the ES-complex. Next in importance is the reverse step with rate constant k_2. Thus the value of k_3 is the most important in determining

Figure 5.2. Partial variance plots for the Michaelis-Menten Models. A number represents the partial variance for that rate constant. Coupled partial variances are represented as follows: in (a) by *, $S_{1,3}$; in (b) by *, $S_{1,3}$; + $S_{1,2}$; X, $S_{2,3}$; in (c) by *, $S_{1,3}$; +, $S_{2,3}$.

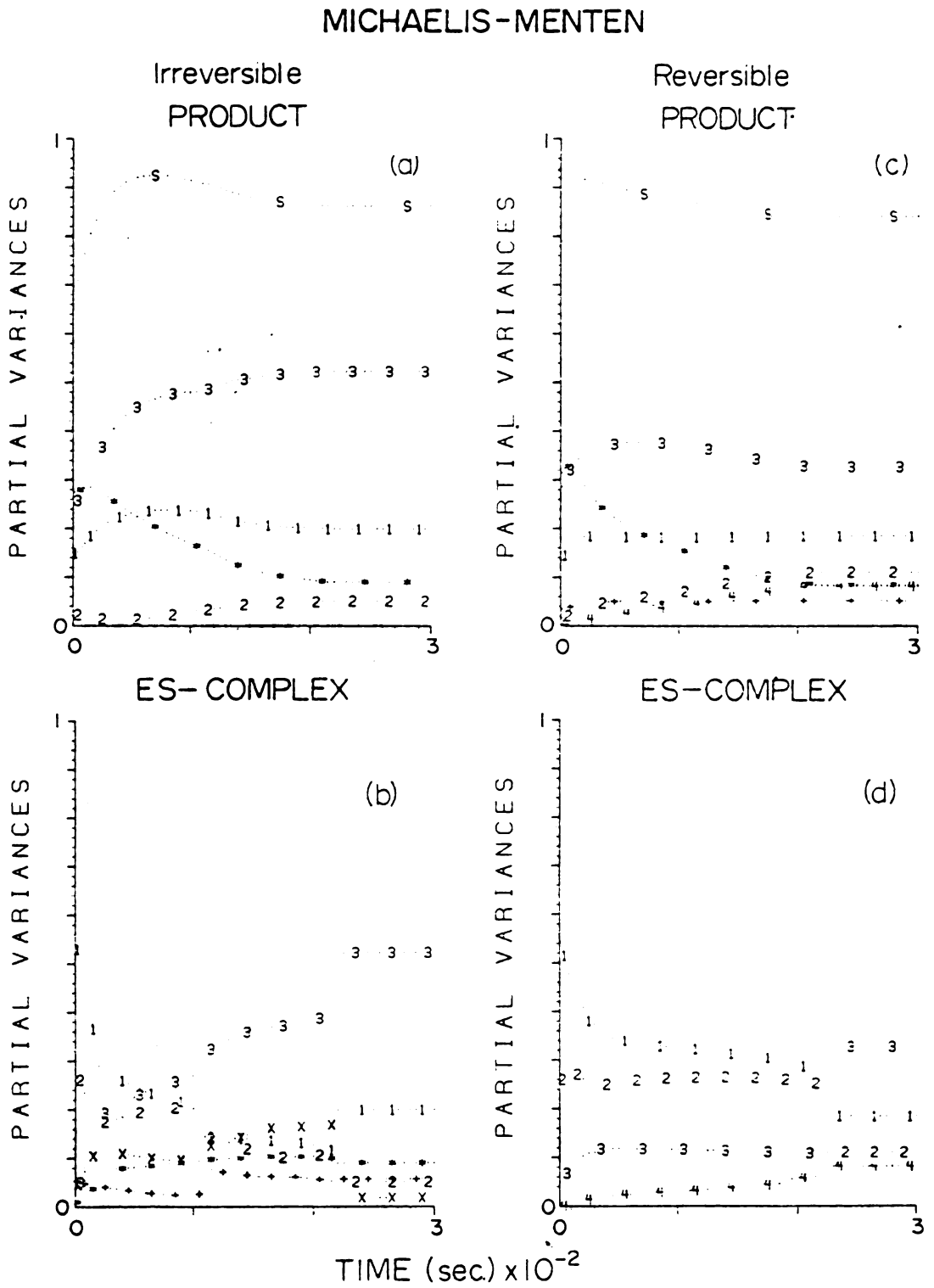

MICHAELIS-MENTEN
Irreversible
PRODUCT
Reversible
PRODUCT
(a)
(c)
(b)
(d)
ES- COMPLEX
ES-COMPLEX
PARTIAL VARIANCES
TIME (sec.) x10^-2
0
3

Figure 5.2

the product concentration. This is not surprising since k_3 controls how fast substrate is converted into product, while the binding step yields a steady state ES concentration essentially instantaneously on the time scale of Figure 5.2a. As time increases, the specific k_3 value chosen has an even greater effect on the accumulated product concentration so the relative importance of k_3 increases with time.

Figure 5.2a also indicates that the product concentration is sensitive to the coupling between k_1 and k_3, especially at early times. This indicates that the accurate determination of k_3, for example, from the early portion of a single progress curve would be hampered by coupling to k_1, resulting in significantly larger marginal deviations of the rate constant than would be obtained by using the entire progress curve. It must be emphasized that the entire analysis described here applies to full time-course behavior rather than only initial rate behavior. If one wished to study the sensitivity of initial rates to substrate concentration for example, a different procedure would be used.

The analysis shows that if the concentration of ES were measurable, it would be most sensitive to k_1 at short times as indicated by Figure 5.2b. This reflects the fact that the formation of ES is the dominant initial step. However, the sensitivity to k_3 grows rapidly as

substrate is depleted. There is also strong coupling between k_1 and k_3, implying that the errors in determining these rate constants from the time-dependence of the concentration of ES would be strongly correlated. As was found for the product sensitivities, ES is not very sensitive to the value of k_2 over the range examined.

Figure 5.2b shows apparent discontinuities in the sensitivity of the ES concentration to the rate constants. One might be tempted to attribute this to numerical errors or instabilities in the calculation, but this is not the case. When the ranges over which the rate constants can vary are drastically reduced the discontinuities disappear. The discontinuous curves (obtained with these time steps) originate because of the large ranges available to the three rate constants. Within a narrow time span an appreciable number of simulation runs reach completion. When this occurs the ES complex disappears and the concentration of E builds up for these simulations. This completely eliminates the sensitivity of ES to the rate constants for this subset of simulations. The result is a series of apparent breaks in the partial variances. It is possible to eliminate these breaks in either of two ways. As indicated above, the ranges of the rate constants can be restricted so that the number of simulations which reach completion is insignificant. Alternatively, the initial substrate to enzyme ratio can be made so

large that the simulations do not reach completion even for the most favorable combination of rate constants. Neither of these alternatives is entirely satisfactory since they both impose restrictions on the model.

In order to be certain that the origin of the greatest sensitivity of product concentration to k_3 and least sensitivity to k_2 is not just the large ranges permitted for the rate constants, sensitivity analyses were performed over several reduced ranges about the same nominal values. The general results were the same except that, as noted above, the apparent breaks in the sensitivity of ES to the rate constants disappeared.

Under the assay conditions modeled here, the concentration of the enzyme-substrate complex, ES, assumes a steady-state value at very short times, as shown in Figure 5.1a. However, even under steady-state conditions the concentration of the complex changes with time as substrate is used up. This is reflected in sensitivities which also change with time. The growing sensitivity to k_3 means that in a full time-course analysis, this rate constant would be relatively more accurately determined by following the progress curve for an extended period of time than could the other rate constants or combinations of these rate constants.

Reversible Michaelis-Menten Model

Only slightly more complex than the irreversible Michaelis-Menten model is the reversible model in which equilibrium is ultimately reached. In testing this model, the same nominal rate constants and ranges were used as for the irreversible case but a reverse step was added:

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \underset{k_4}{\overset{k_3}{\rightleftharpoons}} E + P \ . \qquad (5.2)$$

Table 5.2 gives the nominal rate constants, their ranges, the initial conditions, and the frequency set used. Figures 5.1c and 5.1d show the average concentrations and standard deviations for the reversible case.

As shown in Figure 5.2c, the first 40 seconds gives approximately the same product sensitivities as the irreversible model. The reverse step k_4 only begins to become important at later times as the concentration of product becomes large enough to bind to the enzyme.

Initially the sum of the product partial variances and the higher partial variance which couples k_1 and k_3 accounts for 93% of the total variance. During approximately the first 100 seconds, this sum decays to 85% and remains constant. The other 15% of the sensitivity is spread over couplings among the parameters but no

individual coupling is large enough to appear on the graph.

The major difference between the irreversible model and the reversible model is seen in the sensitivity of ES to the rate constants. In the irreversible case the sensitivity to k_3 grows, while in the reversible model k_1 remains most important. Apparently the reverse step (k_4) can serve to stabilize the concentration of ES as the reaction approaches equilibrium. Since substrate is present in excess, the concentration of the complex continues to be dominated by sensitivity to k_1. As with the irreversible case, apparent breaks in the sensitivity of ES to the rate constants are observed (see Figure 5.2d). These are again caused by a subset of the simulations which in this case reach their equilibrium values.

Models with Slow Conformational Changes

Except for a (usually undetectable) lag in product formation caused by storage of substrate as the ES complex, the Michaelis-Menten model is not capable of describing bursts or lags. Nor can it lead to allosteric behavior since the phenomenon of allosterism as defined (Segel, 1975; Fersht, 1977) in terms of deviations of the reaction velocity from the predictions of the Michaelis-Menten model. In order to examine these

phenomena it is necessary to devise more complex models.

The most common interpretations (Monod et al., 1965; Koshland et al., 1966) of allosteric behavior involve multi-subunit enzymes in which interactions among the subunits make the addition of another substrate molecule easier or more difficult than those which were previously bound. These models are intrinsically thermodynamic in nature since they refer to interactions which affect binding constants. It was suggested some time ago (Whitehead, 1970) that allosterism could arise without subunit interactions as a natural consequence of kinetic models which involved slow steps such as conformational changes. Such models have also been proposed to describe bursts and lags in product production.

In this section we apply sensitivity analysis to a model first examined by Ainslie, Shill and Neet (1972) using steady-state methods. Our sensitivity analysis of the model showed that similar behavior can be obtained with less complex models. Therefore, these simpler models are also examined in some detail.

Model of Ainslie, Shill and Neet

In 1972, Ainslie et al. proposed an enzyme model which they studied by using steady-state techniques coupled to slow conformational changes. They showed that appropriate choices of the 16 rate constants could

be made so that the model displayed either bursts or lags in product production. They also showed that the variation of the final steady-state velocity with substrate concentration could be made to exhibit allosterism, leading to behavior similar to either positive or negative cooperativity depending upon the choice of rate constants. Because of the wide variations in behavior exhibited by this model, brought about merely by changing the values of the rate constants, we felt that this model would provide an excellent test of the methods of sensitivity analysis.

This model, which we refer to as the Ainslie model, is described by the following scheme:

$$\begin{array}{ccccccc} & & \overset{1}{\underset{2}{\rightleftharpoons}} & & \overset{16}{\underset{15}{\rightleftharpoons}} & & \overset{14}{\underset{13}{\rightleftharpoons}} \\ E + S & & ES & & EP & & E + P \\ 10\downarrow\uparrow 9 & & 3\downarrow\uparrow 4 & & & & \downarrow\uparrow \\ & & \underset{12}{\overset{11}{\rightleftharpoons}} & & \underset{6}{\overset{5}{\rightleftharpoons}} & & \underset{8}{\overset{7}{\rightleftharpoons}} \\ E^* + S & & E^*S & & E^*P & & E^* + P \end{array} \qquad (5.3)$$

The numbering sequence for the 16 rate constants is also given in scheme (14) above. Equilibrium constants, K_1, K_3, K_5... are defined as k_1/k_2, k_3/k_4, k_5,k_6, etc.

With its 16 rate constants, this model is complex

enough to defy intuitive understanding of its detailed dynamic behavior. By applying SAM to this model, with the nominal rate constants and ranges already tested by Ainslie, et al., we can determine the important pathways which lead to bursts and lags. The analysis also showed that the model need not be this complex to yield the same general behavior.

Ainslie, et al. separated the rate constants into two sets: those which gave lags in product growth and those which gave bursts. Each of these sets was also divided into two groups which showed allosteric behavior similar to positive cooperativity and negative cooperativity, respectively. Hill plots (Hill, 1925; Segel, 1975; Fersht, 1977) were used to classify the cooperativity. Negative cooperativity gives Hill coefficients less than one while positive cooperativity leads to Hill coefficients greater than one.

In this sensitivity analysis it was only necessary to use two groups of rate constant ranges corresponding respectively to bursts and lags in order to cover the entire range studied by Ainslie, et al. In order to decrease the complexity of the problem, simplify the interpretation, and include the thermodynamic constraints demanded by the presence of mechanistic loops, we maintained all of the equilibrium constants at fixed values in each of the two sets studied. This corresponds

approximately to the choices made by Ainslie, et al. who used constant values for most of the equilibrium constants while varying the rate constants. This simplification reduces the number of independent parameters to eight but does not alter the general behavior of the model.

The eight differential equations which describe the time-dependence of the concentrations of the eight species in this scheme can be reduced to six coupled non-linear differential equations by using the two algebraic equations of mass balance. The nominal values of the rate constants, the values of the equilibrium constants used, and the initial conditions are given for the lag and burst sets in Table 5.3, while the frequency sets and computer data are given in Table 5.4. The ranges allowed for each rate constant were $10^{\pm 1}$ times the nominal value.

In Figure 5.3 the average concentrations and the standard deviations of the two sensitivity analysis runs are displayed. In the lag set the product growth is initially slow but it rapidly increases reaching 27% of its equilibrium value after 120 seconds. In contrast, the product growth of the burst set starts out fast and then slows down, reaching only 11% of its equilibrium value after 120 seconds. This leads to a large range of concentrations in the lag set, but to a restricted set in the burst case. In both cases the less active free enzyme, E, is a minor species.

Table 5.3. Parameter Values[a] for the Ainslie Model.

	(Lag Set)		(Burst Set)	
i	k_i^o	$K_{eq}^{(i)}$	k_i^o	$K_{eq}^{(i)}$
1	10 $(\mu Ms)^{-1}$	$10^{-2}(\mu M)^{-1}$	10. $(\mu Ms)^{-1}$	$0.1(\mu M)^{-1}$
3	10^{-2} s^{-1}	3.0	10^{-3} s^{-1}	10^{-2}
5	10^{4} s^{-1}	3.0	10^{5} s^{-1}	1.0
7	10^{3} s^{-1}	30.0 $(\mu M)^{-1}$	10^{3} s^{-1}	100.0 $(\mu M)^{-1}$
9	10^{-1} s^{-1}	10	10^{-2} a^{-1}	10.0
11	10.0 $(\mu Ms)^{-1}$	0.3 $(\mu M)^{-1}$	10 $(\mu Ms)^{-1}$	10^{-2} $(\mu M)^{-1}$
13	1.0 $(\mu Ms)^{-1}$	10^{-3} μM	10 $(\mu Ms)^{-1}$	10^{-3} μM
15	10 s^{-1}	112.7	10^{3} s^{-1}	100
	E_o = 0.5 μM[b] S_o = 4000.0 μM		E_o = 0.05 μM S_o = 4000.0μM	
	$K_T = 27 = \frac{[P]_{eq}}{[S]_{eq}}$		$K_T = 1.0 = \frac{[P]_{eq}}{[S]_{eq}}$	

[a]The range of the rate constants was $10^{\pm 1}$ times the nominal value, k_i^o.

[b]Initial distribution: 90% E, 10% E*.

Table 5.4. Summary of SAM and Computer Data.

	Frequency Set	Order of Set	Number of Parameter Vectors	Time Range (s)	Number of Time Points	CDC 6500 CPU Time for Integrations[a] (s)	Number of Fourier Coeff.	CPU Time to Compute Fourier Coeff. (s)
Michaelis-Menten Irreversible	9,15,19	6^{th}	37	[0,315]	70	127.7	10,360	6.5
Michaelis-Menten Reversible	13,31,37,41	6^{th}	121	[0,315]	70	403.0	33,880	16.5
Ainslie et al	151,313,463 529,555,573, 579,583	6^{th}	1743	[0,115]	30	6300	313,740	120.
Frieden Model	11,25,41,43 49,53	4^{th}	107	[0,115]	30	312.5	25,680	30.3

[a]Recent modifications have reduced this time by a factor of approximately three.

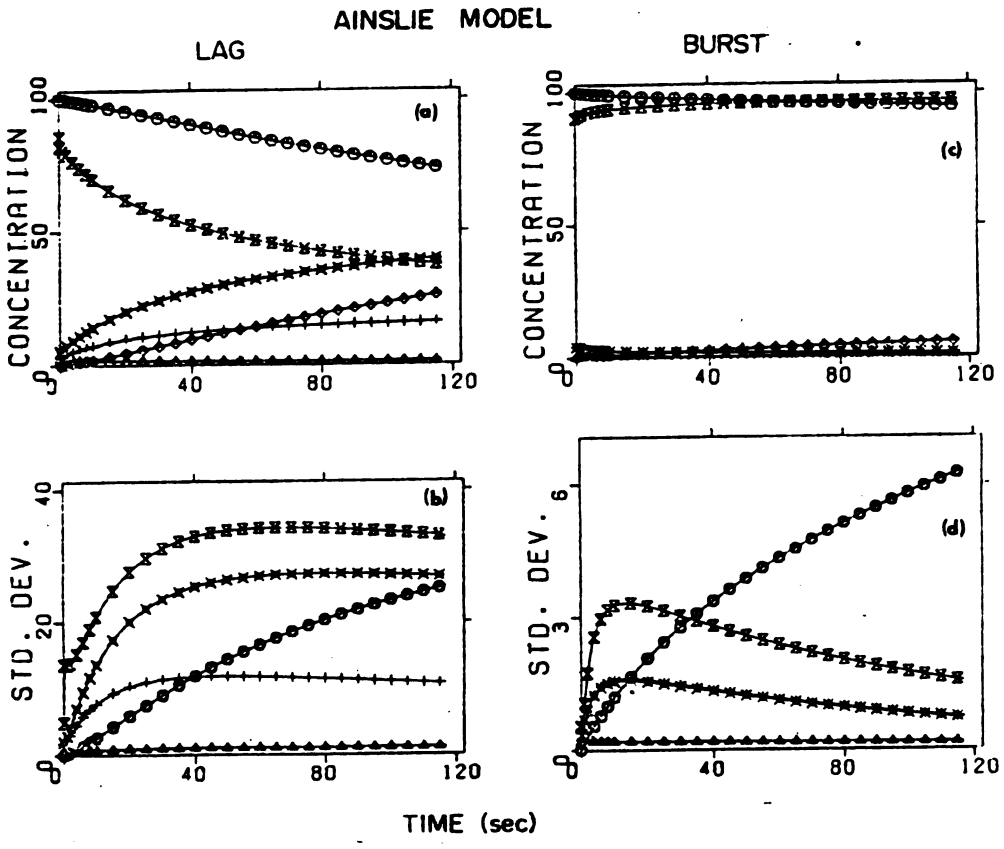


Figure 5.3. Average concentrations and the standard deviations of the concentrations for the Ainslie models. The symbols represent: ⓪, S; △, E: +, E*S: ◇, P; ⧖, ES: ⧓, E*P.

Careful examination of the partial variances shown in Figure 5.4 shows that the lag mechanism operates by shuttling the ES complex to the E*S complex which then rapidly forms product. The top (E+S → E+P) cycle has slow turnover relative to the bottom (E* + S → E* + P) cycle and the important bridge between them is the isomerization step of the enzyme-substrate complex. The small amount of E* present initially starts turning over substrate so that the substrate concentration is most sensitive to k_7, the product formation rate constant in the bottom cycle. As the reaction proceeds the total concentration of enzyme in the bottom cycle is increased by the conversion of ES to E*S; this increases the sensitivity to k_3 and to the binding step E* + S → E*S (k_{11}). This shift to the bottom cycle is verified by the rapid decay of substrate sensitivity to k_{15}.

The coupled partial variances of Figure 5.4b reinforce the above conclusions. The rapid decay in time of the coupled partial variance $S_{5,7}$ is consistent with the growing importance of the depletion of the substrate concentration via the bottom cycle. Furthermore, the isomerization step which increases the total active enzyme concentration and thus increases the importance of the k_{11} step results in the rapid growth in time of the coupled partial variance $S_{7,11}$.

Turning to the sensitivity of the enzyme concentration,

Figure 5.4. Partial variance plots for the Ainslie lag model. A number represents the partial variance for the rate constant. Other partial variances are represented by: B, S_{11}, D, S_{13}; F, S_{15}. Coupled partial variances are represented as follows: in (b) by +, $S_{5,7}$; *, $S_{7,11}$; in (c) by +, $S_{13,15}$; *, $S_{1,15}$; in (d) by *, $S_{5,7}$; in (e) by *, $S_{13,15}$.

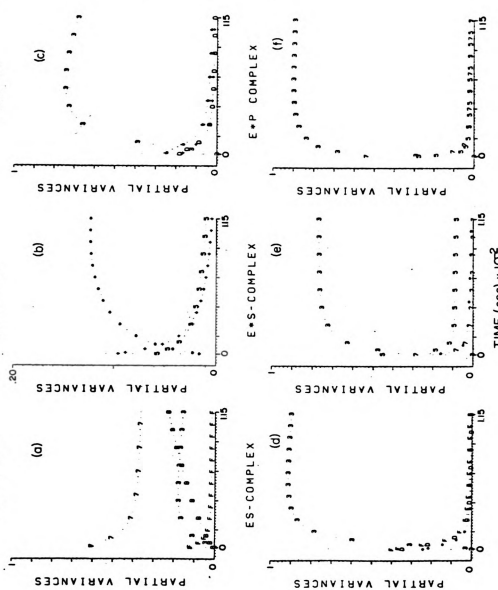

AINSLIE LAG
SUBSTRATE
(a)
(b)
ENZYME
(c)
ES-COMPLEX
(d)
E+S-COMPLEX
(e)
E+P COMPLEX
(f)
PARTIAL VARIANCES
TIME (sec) x 10⁻²

Figure 5.4

displayed in Figure 5.4c, we note that there is negligible sensitivity to k_1, since the binding step reaches equilibrium so rapidly on the time scale of this display that variations in k_1 cannot change the concentration of E. (Recall that the equilibrium constant K_1 is fixed.) On the other hand, the relative amounts of enzyme present as E, ES and EP strongly depend on the values of the other rate constants k_{13} and k_{15} in the top cycle at early times when the product concentration is low. The large initial values of the partial variances S_{13} and S_{15} and the coupled partial variance $S_{13,15}$ support this assertion.

The sensitivity to the top cycle rate constants is then lost to k_3 as the inactive enzyme isomerizes to E*S. Since 90% of the enzyme is initially in the inactive form, this transfer to E*S changes the E concentration significantly. All the other complexes also display this feature of a rapid rise to a large sensitivity to the isomerization rate k_3.

The sensitivity plots for E*S and E*P in Figures 5.4e and 5.4f respectively, are consistent with the interpretation of the other sensitivity plots. Once again the sensitivity to the binding step (k_{11}) for the bottom cycle is negligible due to the rapidity of this step. The steps with rate constants k_5 and k_7 are important initially but the k_3 isomerization step grows to major

importance. The similarity of the E*S and E*P sensitivities suggests that the inclusion of both intermediate complexes may not be necessary in the formulation of a mechanism that leads to lag behavior.

The burst mechanism operates by forming product initially via the fast bottom cycle. The rate constants are such that, as time progresses, enzyme is shunted from the lower cycle to the upper cycle primarily through the enzyme-substrate complex isomerization step. Since the top cycle is relatively slow, the turnover of substrate slows after the initial period, hence the burst behavior. The bottom E* cycle remains the major route for substrate turnover as shown by the large sensitivity to k_7 in Figure 5.5a. Though the partial variance S_7 does drop from 88% to 60% of the total variance at 115 seconds while S_{15} grows somewhat, it is k_7 that dominates the substrate sensitivity even more than in the lag case.

In notable contrast to the lag analysis, the enzyme sensitivity displayed in Figure 5.5b does involve k_1. However, examination of the total variance, that is the sum of all the partial and coupled partial variances, reveals that it is very small. Since the partial variances are all defined relative to this total variance we obtain non-negligible values for the partial variances even though, as just noted, the total variance is

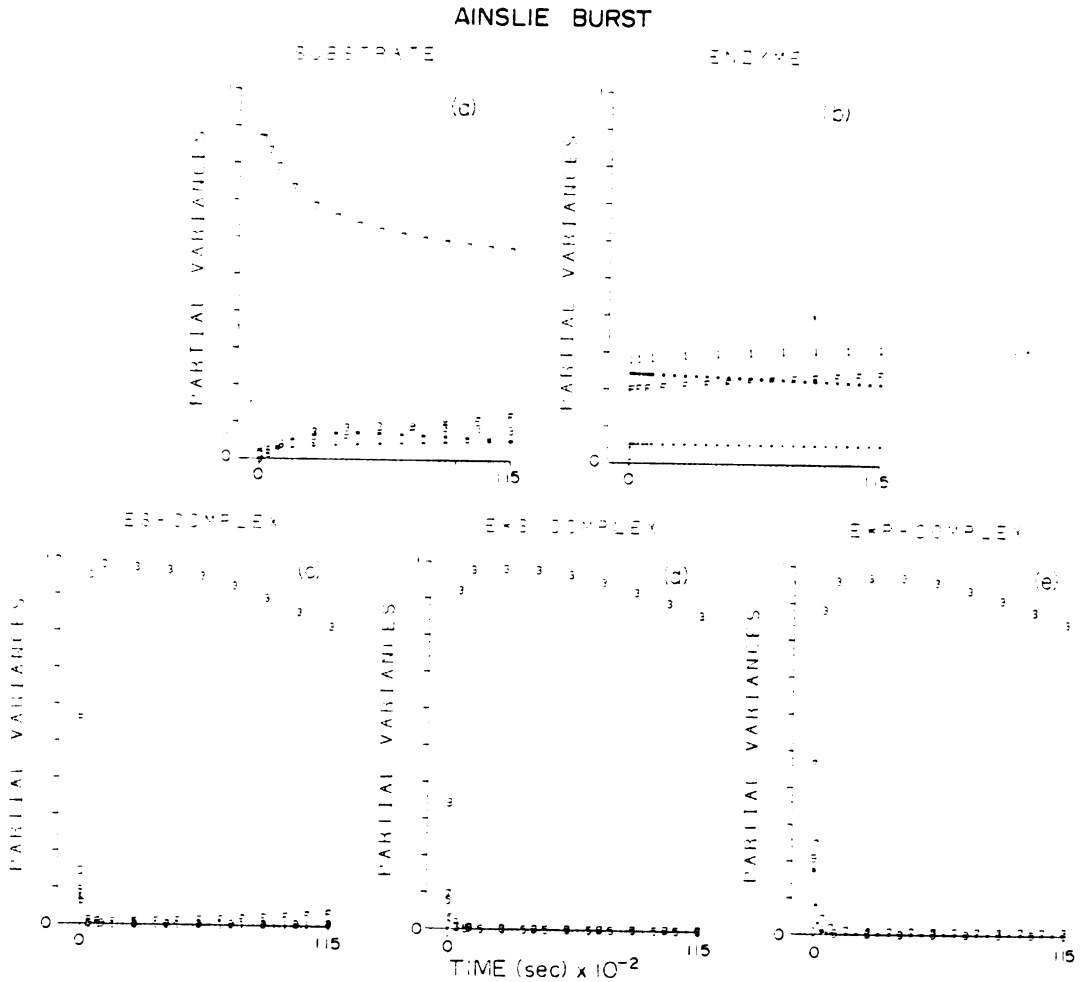


Figure 5.5. Partial variance plots for the Ainslie burst model. A number represents the partial variance for that rate constant. Other partial variances are represented by: B, S_{11}; D, S_{13}; F, S_{15}. Coupled partial variances are represented as follows: in (a) by *, $S_{11,7}$; in (b) by *, $S_{1,15}$; +, $S_{13,15}$. The symbol "s" represents the sum of the displayed partial variances.

negligible. Thus we may conclude that the enzyme concentration in the burst region of parameter space is not significantly affected by the rate constants. Of course, if the equilibrium constants were allowed to vary the results could be greatly altered.

The sensitivity plots of the intermediates, ES, E*S and E*P, shown in Figures 5.5c, 5.5d and 5.5e, respectively, are dominated by the sensitivity to the isomerization of ES to E*S. At very short times the top and bottom cycle rates have some sensitivity, but the total variance is very small here.

From the above detailed analysis a simple rationale of the operation of this model with regard to burst and lag behavior can be formulated. The initial relative concentrations of E and E* are determined by K_9. It is the relatively slow transformation of E*S to ES and vice-versa brought about by substrate binding which gives rise to the bursts and lags. The formation of E*S and ES from E* and E, respectively, is rapid compared with the rate of interconversion of these forms. Because of the importance of the step with rate constants k_3 and k_4 it is not surprising that the sensitivity to k_3 (fixing k_3 also determines k_4 through the constant value of the equilibrium constant, K_3) provides an important clue to the behavior of the model. If we wish to focus on bursts and lags in product production, then the most informative

sensitivities will be those which relate product (or substrate) concentration to the rate constants.

With this information we can now formulate a reduced model which exhibits essentially the same sensitivities as the complete Ainslie model:

$$\begin{array}{ccccccc} E + S & \rightleftarrows & ES & \rightleftarrows & EP & \rightleftarrows & E + P \\ & & \downarrow\uparrow & & & & \\ E^* + S & \rightleftarrows & E^*S & \rightleftarrows & E^* + P & & \end{array} \qquad (5.3)$$

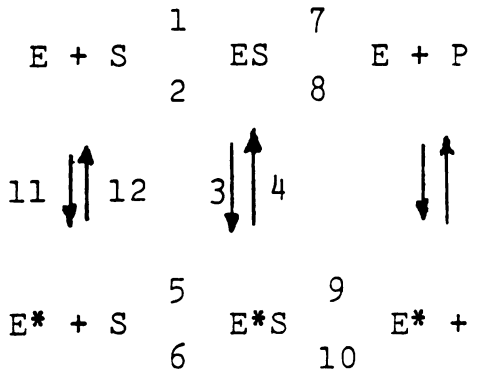
Of course to obtain the proper very long time behavior it would be necessary to incorporate the $E \rightleftarrows E^*$ step in the model to return E^* to E. However, the $E \rightleftarrows E^*$ step plays no significant role in the behavior of the model in the region of parameter space and the time range explored here. As long as the rate constant sets are chosen such that a pool of enzyme is bound up in the ES intermediate which is then slowly converted to E^*S, the lag behavior will result. For burst behavior one needs more E^* present initially to cycle through the bottom than the isomerization $ES \rightleftarrows E^*S$ would yield at equilibrium. Since the reduced model can exhibit these features, bursts and lags will result from this model.

Furthermore, the mechanistic steps by which EP is

created and destroyed are of secondary importance over this region of rate constant space. This suggests the possibility of still further reduction of the model.

Frieden Model

A model proposed by Bates and Frieden (1970) to account for time-lags in enzyme reactions and also studied by Ho (1976) may be represented by the scheme:



(5.4)

In order to permit comparison of this model with that of Ainslie *et al.* (1972) the same rate constants were used for equivalent steps. To evaluate k_7, k_8, k_9, and k_{10}, a steady-state approximation was applied to d[EP]/dt and d[E*P]/dt. This related the product release rate and equilibrium constants of the Frieden model to those of the Ainslie model by the following equations, in which the primed rate constants refer to the Frieden model,

$$k_7' = \frac{k_{16}k_{14}}{k_{14}+k_{15}}; \qquad k_8' = \frac{k_{15}k_{13}}{k_{14}+k_{15}}$$

$$k_9' = \frac{k_5k_7}{k_6+k_7}; \qquad k_{10}' = \frac{k_6k_8}{k_6+k_7} \; . \tag{5.5}$$

These rate constants, the other nominal rate constants, the equilibrium constants, and the initial conditions for the burst and lag runs of the Frieden model are given in Table 5.5, while the frequency sets and computer data are given in Table 5.4. As with the Ainslie model, the equilibrium constants were fixed and the rate constant ranges were $10^{\pm 1}$ times the nominal values.

Figure 5.6 shows the average concentrations and the standard deviations of the Frieden model. Both the lag and burst cases are similar to those of the Ainslie model. It is interesting to note that there is more "effective" enzyme in the Frieden model since there are fewer intermediate complexes.

The substrate sensitivity shown in Figure 5.7a is largest for k_9, the rate constant for release of product from the active form. Initially, the corresponding upper cycle rate constant k_7 for the inactive form contributes about 20% to the rate of product formation but this decreases to less than 5% as the isomerization step

Table 5.5. Parameter Values for the Frieden Model.

	LAG [a]		Burst [a]	
i	k_i^o	k_{eq}	$k_i^{(o)}$	K_{eq}
1	10 $(\mu M\ s)^{-1}$	10^{-2} $(\mu M)^{-1}$	10.0 $(\mu M\ s)^{-1}$	10^{-1} $(\mu M)^{-1}$
3	10^{-2} s^{-1}	1.0	3×10^{-4} s^{-1}	10^{-2}
5	10 $(\mu M\ s)^{-1}$	10^{-1} μM	10 $(\mu M\ s)^{-1}$	10^{-2}
7	30 s^{-1}	3.1×10^3 μM	10 s^{-1}	10.0 μM
9	750 s^{-1}	3.1xx 10^2 μM	9.9×10^3 s^{-1}	100 μM
11	10^{-2} s^{-1}	10^{-1}	10^{-3} s^{-1}	10^{-1}

$E_T = 0.05\ \mu M$ [b]

$S_o = 4000\ \mu M$

$$K_T = 31 = \frac{[P]_{eq}}{[S]_{eq}} \qquad K_T = 1.0 = \frac{[P]_{eq}}{[S]_{eq}}$$

[a] The range of rate constants was $10^{\pm 1}$ times the nominal value k_i^o.

[b] Initial distribution: 90% E, 10% E*.

Figure 5.6. Average concentrations and the standard deviations of the concentrations of the Frieden models. The symbols represent: ⓪, S; Δ, E; +, E*S; ◈, P; ⧖, E*; ⋈, ES.

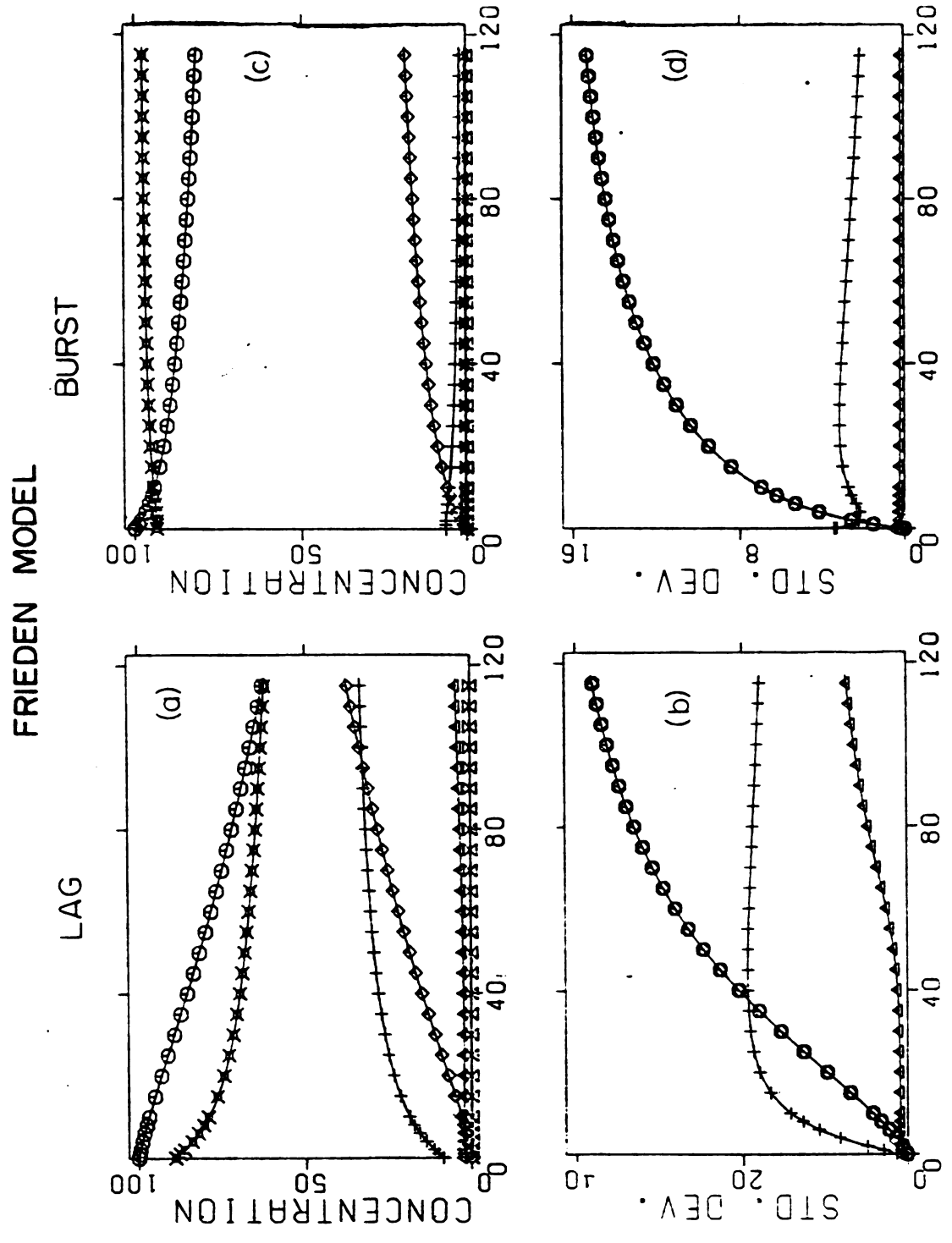

FRIEDEN MODEL
LAG
BURST
(a)
(b)
(c)
(d)
CONCENTRATION
STD. DEV.
TIME (sec)
0
50
100
40
80
120
20
16
8

Figure 5.6

proceeds and the isomerization rate constant, k_3, becomes more important.

The partial variances shown in Figure 5.7b are particularly revealing. The coupling between k_3 and k_9 grows and decays during the time range of the isomerication of ES to E*S. As this occurs the rate constant, k_5, for the binding of substrate to active enzyme grows in importance as does its coupling with k_9. By examining these time-developments, one can gain a rather clear picture of the lag behavior as product production shifts from the upper((slow) cycle to the lower (fast) cycle.

These dynamic effects are also mirrored in the sensitivities to the various enzyme forms. For example, the enzyme sensitivity shown in Figures 5.7c and 5.7d shifts with time from the E-cycle to the E* cycle. This shift is responsible for the lag behavior. Note the rapid growth and decay of the sensitivity of E to the coupling between k_1 and k_7, the slightly slower growth and decay of its sensitivity to k_3, and the slower growth in its sensitivity to k_5 and k_9 and to the coupling between k_5 and k_9. These plots show how the dependence of the concentration of free less active enzyme on the various rate constants changes with time.

The major route for the formation of E*S is the isomerization step ES $\rightleftarrows$ E*S. This is shown in Figure 5.7c by the dominance of the sensitivity of the E*S concentration

Figure 5.7. Partial variance plots for the Frieden lag model. A number represents the partial variance for that rate constant. Coupled partial variances are represented as follows: in (b) by +, $S_{5,9}$; in (c) by +, $S_{5,9}$; *, $S_{1,7}$.

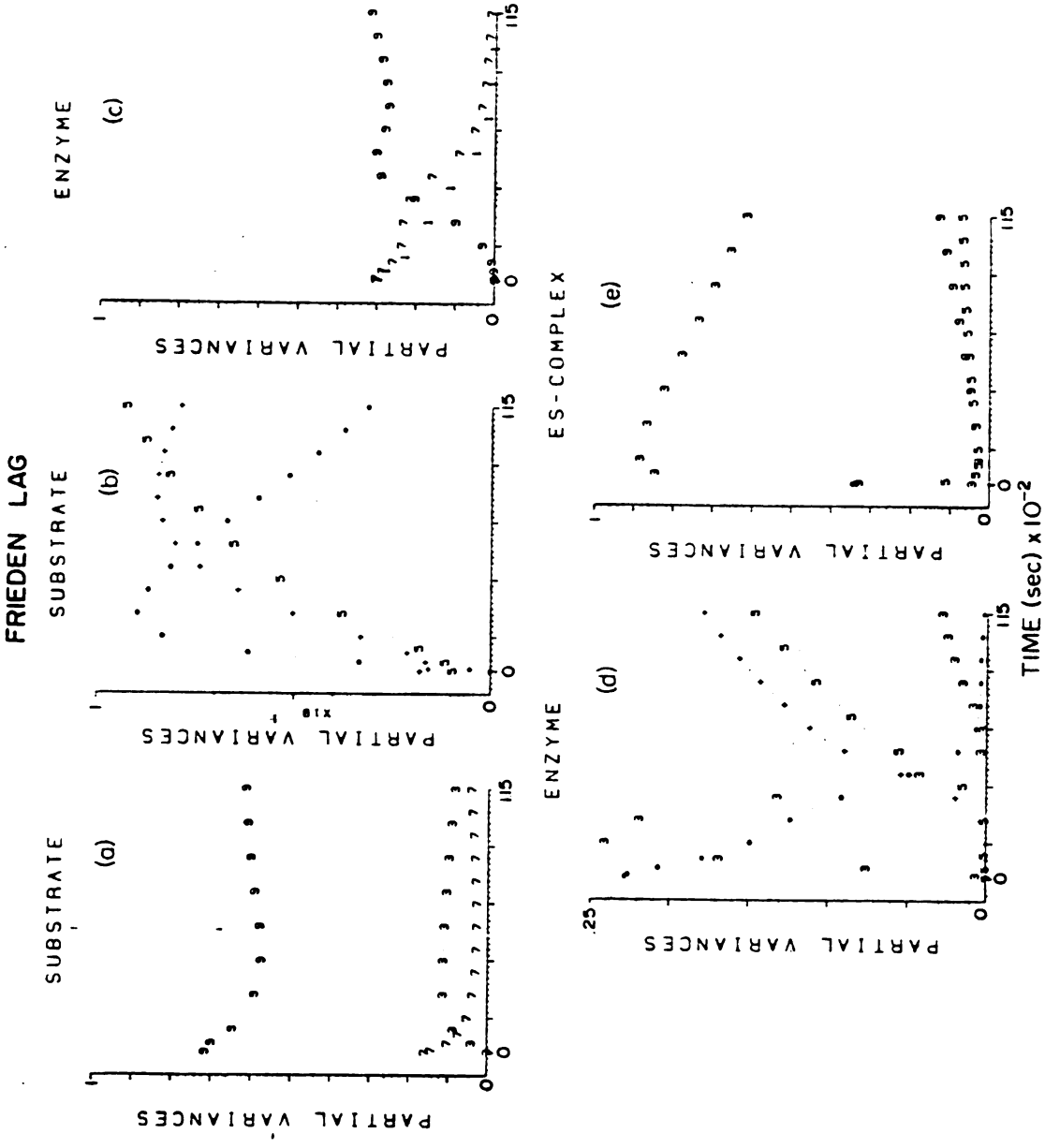

FRIEDEN LAG
SUBSTRATE
(a)
(b)
ENZYME
(c)
ES-COMPLEX
(d)
(e)
PARTIAL VARIANCES
TIME (sec) x10⁻²

Figure 5.7

to the isomerization rate constant k_3. The growth in sensitivity to k_3 is accompanied by a rapid decrease in the sensitivity to k_5. At longer times, some sensitivity to k_5 and k_9 accumulates.

The burst set of rate constants gives a reversal of this behavior pattern as shown in Figure 5.8. Again, the E* cycle initially controls the rate of product production. For this range of rate and equilibrium constants, however, E*S is converted to ES which is less active with the result that the overall rate of product production is decreased.

These simulations clearly show that the simpler Frieden scheme can give both bursts and lags. In fact, the insensitivity to k_{11} and k_{12} shows that an even simpler model without the E $\rightleftarrows$ E* step would also describe the time-behavior of these systems, <u>provided one started with an equilibrium distribution of E and E*</u>. This is because in the models studied here, the direct interconversion of E and E* is slow enough that it cannot compete with the ES $\rightleftarrows$ E*S isomerization.

Since one of the "bonuses" of the Ainslie model was its ability to describe allosteric behavior without the need for cooperative subunits it was of interest to see whether the simpler Frieden model could also give apparent cooperativity. In order to do this, two simulations were performed with the Frieden model which were designed to yield "initial velocities" at various substrate

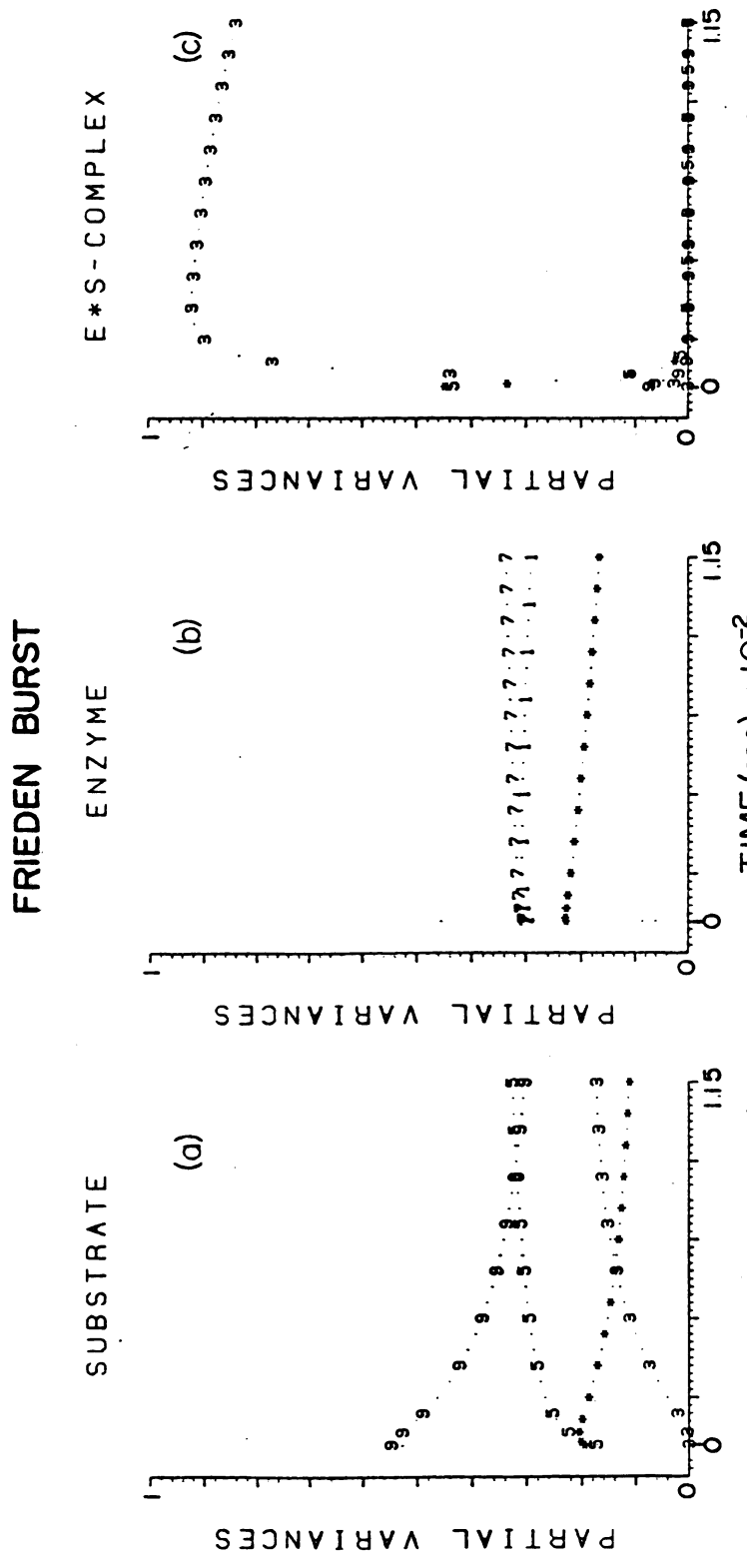


Figure 5.8. Partial variance plots for the Frieden burst model. A number represents the partial variance for that rate constant. Coupled partial variances are represented as follows: in (a) by *, $S_{5,9}$; in (b) by *, $S_{1,7}$.

levels under steady-state conditions. In these simulations, the rate constants k_8 and k_{10}, which yield overall reversibility in the Frieden model, were set equal to zero. After steady-state had been achieved, the reaction velocity dP/dt was evaluated as a function of substrate concentration. This is equivalent to the evaluation of initial steady-state velocities appropriate to separate assays.

It was possible in this way to find sets of rate constants and concentrations which gave Hill coefficients which vary from 0.125 (rate constant set 1 in Table 5.6) to 2.645 (rate constant set 2). These results mimic the behavior usually attributed to negative and positive cooperativity, respectively, and show that even a model as simple as that of Frieden can be made, with a suitable choice of rate constants and initial conditions, to exhibit allosteric behavior.

Summary

Model reduction is an important goal of sensitivity analysis. By applying sensitivity analysis to complex mechanistic schemes, one is able to determine which steps in a reaction are essential to the behavior being examined and which are not; perhaps permitting a simple model to be formed as a subset of the more complex scheme.

Another major aim of sensitivity analysis is to determine which rate constants are most likely to be

Table 5.6. Rate Constants for Allosteric Test.

Set #1	Set #2
$k_1 = 100\ (\mu M\ s)^{-1}$	$k_1 = 10\ (\mu M\ s)^{-1}$
$k_2 = 1000\ s^{-1}$	$k_2 = 1.0\ s^{-1}$
$k_3 = 10^{-4}\ s^{-1}$	$k_3 = 3 \times 10^{-4}\ s^{-1}$
$k_4 = 10^{-2}\ s^{-1}$	$k_4 = 3 \times 10^{-2}\ s^{-1}$
$k_5 = 10^{2}\ s^{-1}$	$k_5 = 10\ s^{-1}$
$k_6 = 10^{4}\ s^{-1}$	$k_6 = 10^{-1}\ s^{-1}$
$k_7 = 10\ s^{-1}$	$k_7 = 10\ s^{-1}$
$k_8 = 0$	$k_8 = 0$
$k_9 = 10^{4}\ s^{-1}$	$k_9 = 9900\ s^{-1}$
$k_{10} = 0$	$k_{10} = 0$
$k_{11} = 10^{-2}\ (\mu M\ s)^{-1}$	$k_{11} = 10^{-3}\ (\mu M\ s)^{-1}$
$k_{12} = 10^{-1}\ s^{-1}$	$k_{12} = 10^{-2}\ s^{-1}$
$S_o = 15000\ \mu M$	$S_o = 4000\ \mu M$
$E_o = 0.05\ \mu M$	$E_o = 0.05\ \mu M$
time range of test	
(1200 sec to 1750 sec)	(85 sec to 145 sec)
substrate decay over this time range	
(5550 μM to 3150 μM)	(2440 to 2160) μM
Hill coefficient=0.125	Hill coefficient=2.645
correlation coefficient $\rho = .999$	correlation coefficient $\rho = 0.997$

measurable and which have so little effect on the concentration-time curves that they cannot be accurately determined. For example, by following the full time course of product growth for a system which follows the Frieden mechanism and exhibits a lag one might expect, if the equilibrium constants are known, to be able to determine k_9, k_7, k_3 and k_5. However, k_3 and k_9 would be strongly coupled as would k_5 and k_9. The rate constant k_7 would be measurable primarily from the behavior at short times but this time period contains essentially no information about k_3 and k_5.

Our primary motivation for implementing these techniques was to ultimately apply them to the study of transients in enzyme kinetics. Under conditions where one can follow the production and disappearance of intermediates, it should be much easier to distinguish among various mechanisms. The application of sensitivity analysis should provide very useful information about the sensitivity of the various concentrations to the rate constants so that the latter can be arranged in order of their accessibility of measurement.

VI. SENSITIVITY ANALYSIS OF A TRYPTOPHANASE KINETIC MODEL

Recently the mechanism of Tryptophanase catalysis has been under investigation in our laboratories. Of particular interest is the variation of the ultra-violet-visible absorbance spectrum of Tryptophanase with pH. As the pH is changed the enzyme apparently changes its conformation which results in changes in the spectral shape (June et al. 1979). The model proposed (June et al. 1980) for this is

$$E_\beta H^+ \overset{K_{\beta H}}{\rightleftarrows} E_\beta + H^+$$

$$k^H_{\beta\gamma} \downarrow\uparrow K^H_{\beta\gamma} \quad \downarrow\uparrow k_{\beta\gamma},\ K_{\beta\gamma}$$

$$E_\gamma H^+ \overset{K_{\gamma H}}{\rightleftarrows} E_\gamma + H^+$$

$$k_{\delta\gamma} \downarrow\uparrow k_{\gamma\delta}$$

$$E_\delta$$

Scheme 1

which is composed of three interconvertible manifolds designated β, γ, and δ. During 1979-1980, two types of incremental pH change experiments were done to probe the absorbance changes which accompany a change in pH. Incremental pH jump experiments were done to examine the conversion of the low pH form of the enzyme to the high pH form. The reverse reaction was also tested with incremental pH drop experiments. The results of the rapid changes ($t < 10$ seconds) were analyzed in terms of a reduced model using only the β and γ manifolds since the growth or decay of form δ is slow. This simplification, along with the restriction that the protonation-deprotonation reactions occur within the mixing time of the stopped-flow experiment allowed the mechanism to be reduced to an apparent first order scheme with the apparent first order rate constant, k', and a model output function ΔA_{obs}. The equations used to fit the data (in a least squares sense) are given in Table 6.1. The program KINFIT4 (Dye, Nicely, 1971) was used (June, Dye, Suelter 1980) to obtain the parameters and their standard deviations, shown in Table 6.2. To clarify these results and to propose further experiments a sensitivity analysis of the model was undertaken.

The sensitivity analysis of this model can be separated into two regions, the investigation of the general sensitivity of the model with respect to the parameters, and

Table 6.1. Parameters for Tryptophanase Model

$$A_{obs} = A_\infty + (\Delta A)\exp(-k't)$$

$$\Delta A_{obs} = \frac{\Delta A_\infty + \Delta A_o \cdot K_a/(H^+)}{1 + K_a/(H^+)}$$

$$\Delta A_\infty = \frac{(\varepsilon_\beta \beta_t^o)[K_{\gamma H} = K_{\beta H}]}{[1 + \frac{K_{\gamma H}}{K_{\beta H}K_{\beta\gamma}}][K_{\beta H} + (H^+)_o]}$$

$$\Delta A_o = \frac{(\varepsilon_\beta \beta_t^o)(H^+)_o[K_{\beta H}/K_{\gamma H} - 1]K_{\beta\gamma}}{[1 + K_{\beta\gamma}][K_{\beta H} + (H^+)_o]}$$

$$K_a = \frac{K_{\beta H}[1 + K_{\beta\gamma}]}{1 + K_{\beta H}K_{\beta\gamma}/K_{\gamma H}}$$

$$k_1' = \frac{k_{\beta\gamma}K_{\beta H} + k_{\beta\gamma}^H(H^+)}{K_{\beta H}K_{\beta\gamma}[1 + (H^+)/K_{\gamma H}]} + \frac{k_{\beta\gamma}^H + k_{\beta\gamma}K_{\beta H}/(H^+)}{1 + K_{\beta H}/(H^+)}!$$

Thermodynamic constraints require that $K_{\beta\gamma}^H = K_{\beta H}K_{\beta\gamma}/K_{\gamma H}$ and, of course $k_{\gamma\beta} = k_{\beta\gamma}/K_{\beta\gamma}$; and $k_{\gamma\beta}^H = k_{\beta\gamma}^H/K_{\beta\gamma}^H$

Table 6.2. Best-fit Parameters Based Upon Scheme 1 and Their Marginal Standard Deviation Estimates.

Parameters	#	Value	Standard Deviation Value	Standard Deviation Percent
$(\varepsilon_\beta \beta_t^o)_{jump}$	1	0.080 cm^{-1}	0.001	1.4
$(\varepsilon_\beta \beta_t^o)_{drop}$		0.0175 cm^{-1}	0.0008	4.6
$K_{\beta H}$	2	2.0×10^{-10} M	0.4×10^{-10}	20
$K_{\beta\gamma}$	3	39	8	20
$K_{\gamma H}$	4	1.7×10^{-7} M	0.3×10^{-7}	18
$K_{\beta\gamma}$	5	8.3 sec^{-1}	1.6	18
$K^H_{\beta\gamma}$	6	0.0297 sec^{-1}	0.0045	15
K_a		7.7×10^{-9} M	0.4×10^{-9}	5
$K_{\gamma\beta}$		0.212 sec^{-1}	0.012	6
$k^H_{\beta\gamma}$		0.045 sec^{-1}	0.010	20
$k^H_{\gamma\beta}$		0.66 sec^{-1}	0.04	6

Table 6.3. Legend for Figures 6.5-6.11.

$K_{\gamma H}$ = ⬡	$k_{\beta\gamma}$ = *
$K_{\beta H}$ = △	$k^H_{\beta\gamma}$ = ⧖
$K_{\beta\gamma}$ = ◇	$\varepsilon_\beta \beta_t^o$ = ⋈

the investigation of the actual "fitted" parameters along with their estimated standard deviations. This procedure should permit one to see in which regions the measurements might provide better estimates of the parameters. The investigation of the general sensitivity of the model was done by varying each parameter over a fixed relative range. A 10% variation, i.e., within 10% of the nominal values, was chosen for this technique. Since each parameter is varied over an equal range the sensitivity of the model to its parameters enables us to rank order them in terms of their relative effects on the output function.

A second sensitivity analysis with the parameters varied only over the estimated standard deviations (as determined by KINFIT4) was performed. This type of analysis indicates which regions should be studied in order to refine the estimates of the parameters.

Sensitivity analysis for each different experiment was done, one for the pH drop and one for the pH jump. Each sensitivity analysis had two output functions, k' and ΔA_{obs}. Six parameters were varied in each analysis using 99 simulations with a fourth-order accurate Fourier frequency set. The value of $[H^+]_o$ was set equal to the initial H^+ concentration used in each experiment. For the pH drop $[H^+]_o = 10^{-8.7}$ M and for the pH jump $[H^+]_o = 10^{-7.0}$ M.

Figure 6.1 shows that the average value of ΔA_{obs} for

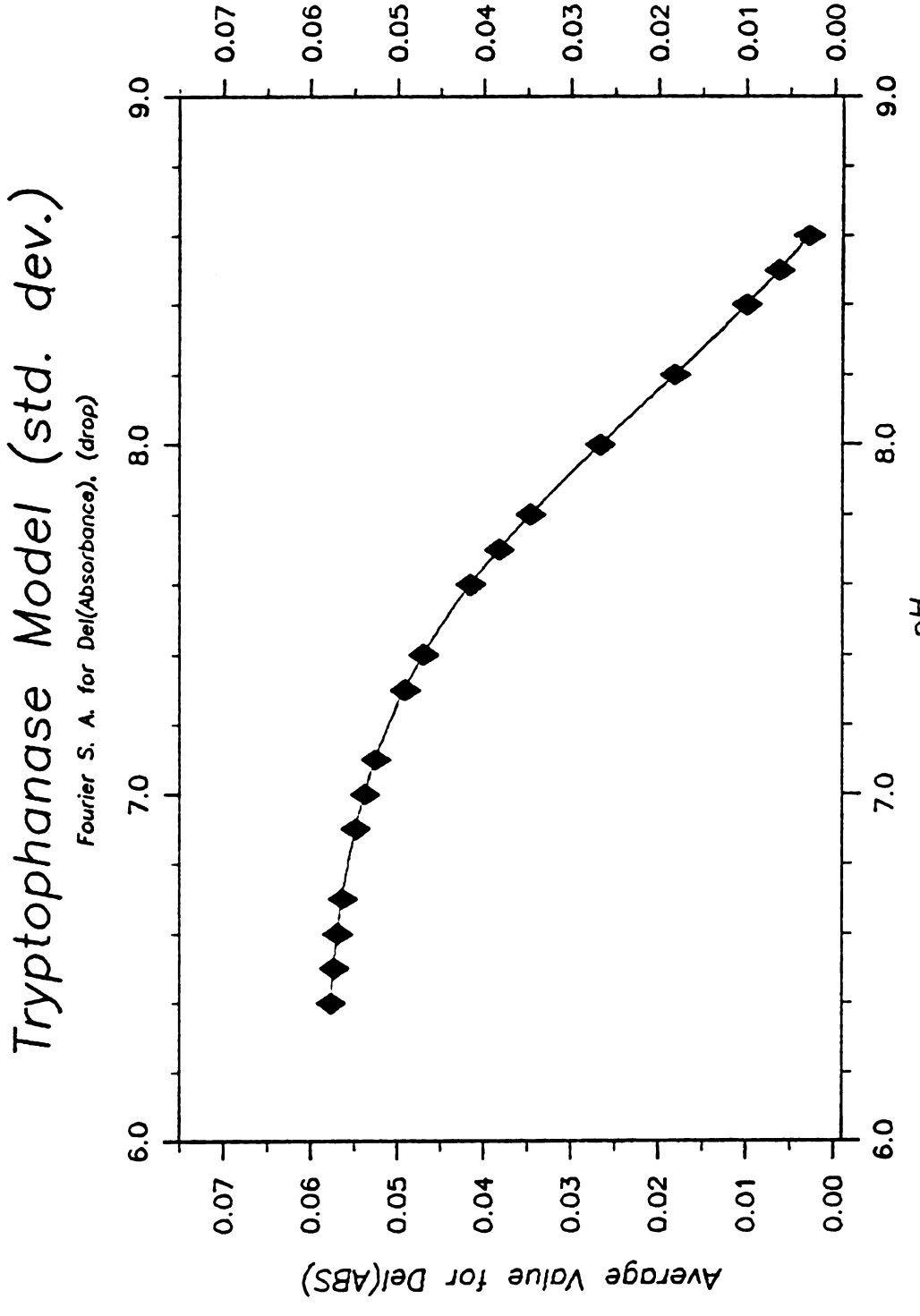


Figure 6.1 Averaged value of Del(ABS) from the pH drop Tryptophanase Model (standard deviation variation).

the pH drop analysis grows as the pH decreases from 8.7 to 6.4. Both the 10% parameter variation and the standard deviation variation give the same averaged nominal value for ΔA_{obs}. Similarly, Figure 6.2 shows the average value of ΔA_{obs} for the pH jump analysis. Since the range of pH covered in the pH jump analysis is more symmetric about the apparent pK_a value of 8.1 than is the pH drop analysis, Figure 6.1 is more like a complete titration curve than is Figure 6.2.

Figure 6.3 displays the averaged value of k' as a function of pH. The same averaged values were obtained for both the pH drop and the pH jump analysis as well as for both sets of parameter variations. This is, of course, expected from the functional form of k' which contains only rate constants, equilibrium constants and $[H^+]$.

The partial variances of ΔA_{obs} for the standard deviation (std. dev.) pH jump sensitivity analysis are shown in Figure 6.4. This output function is only sensitive to $\varepsilon_\beta \beta_t^o$, $K_{\beta H}$, and $K_{\beta\gamma}$. Note that there is no large variation of sensitivity in the pH region 7.0-8.5. The sensitivities only differ at the ends of the pH region.

Figure 6.5 gives the partial variances for ΔA_{obs} where the range of variation was 10%. This plot shows that the most important parameter is $\varepsilon_\beta \beta_t^o$. Note that the sensitivity to $\varepsilon_\beta \beta_t^o$ is much lower in Figure 6.4, where the standard deviation variation was used. Since the model is so

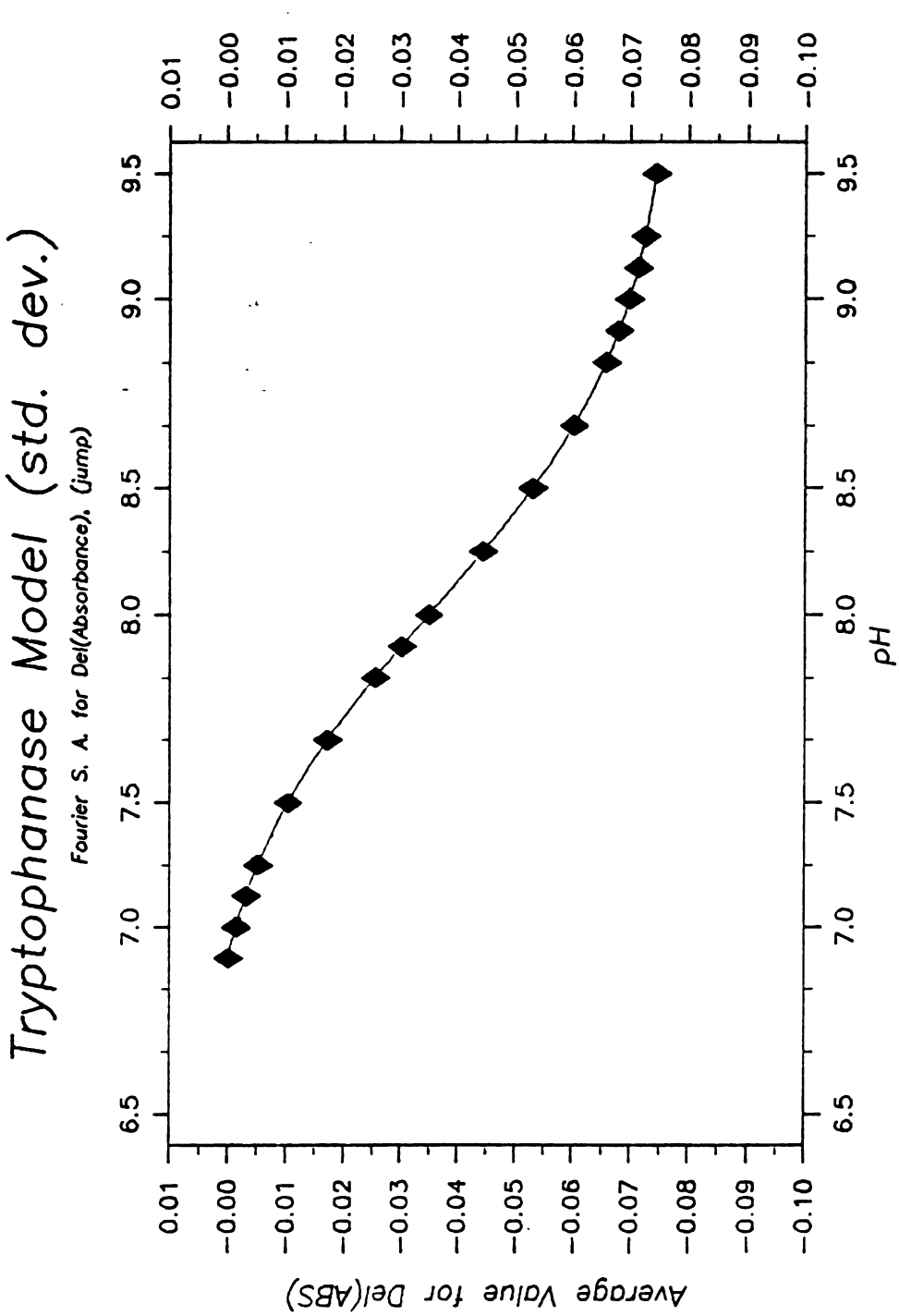


Figure 6.2 Averaged value of Del(ABS) from the pH jump Tryptophanase Model (standard deviation variation).

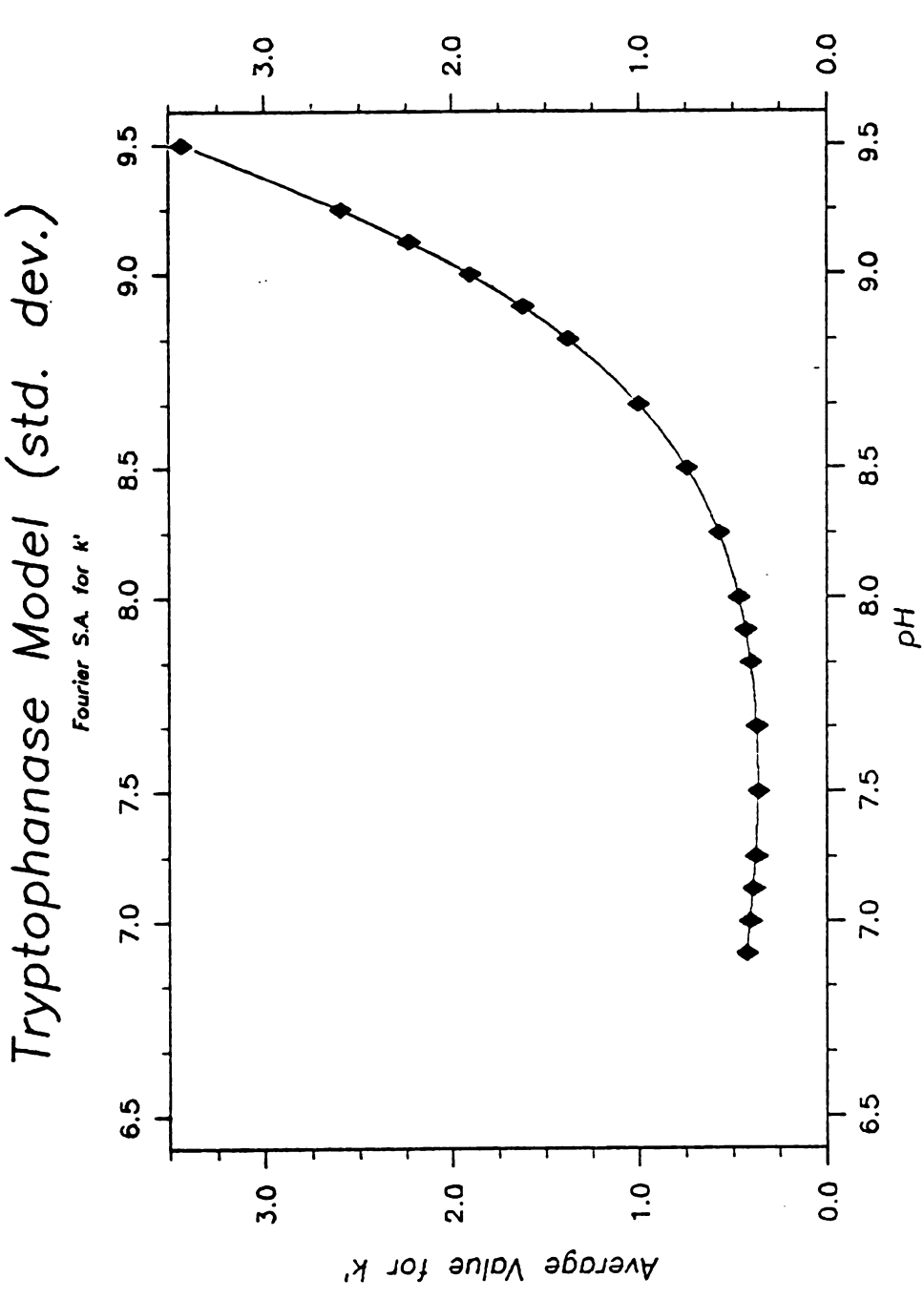


Figure 6.3 Averaged value of k' from the Tryptophanase Model (standard deviation variation).

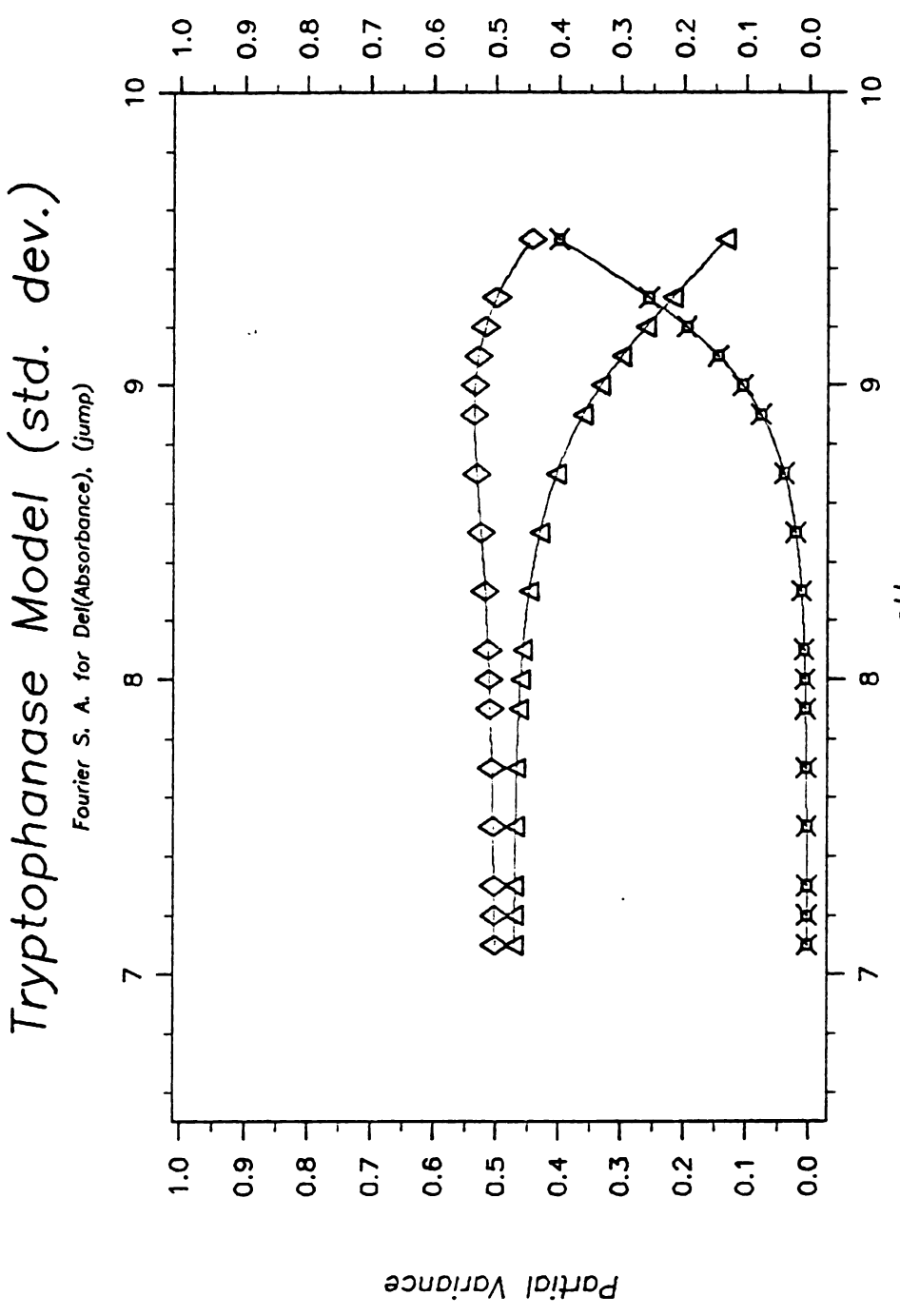


Figure 6.4 Partial variances of Del(ABS) from the pH jump Tryptophanase Model (standard deviation variation).

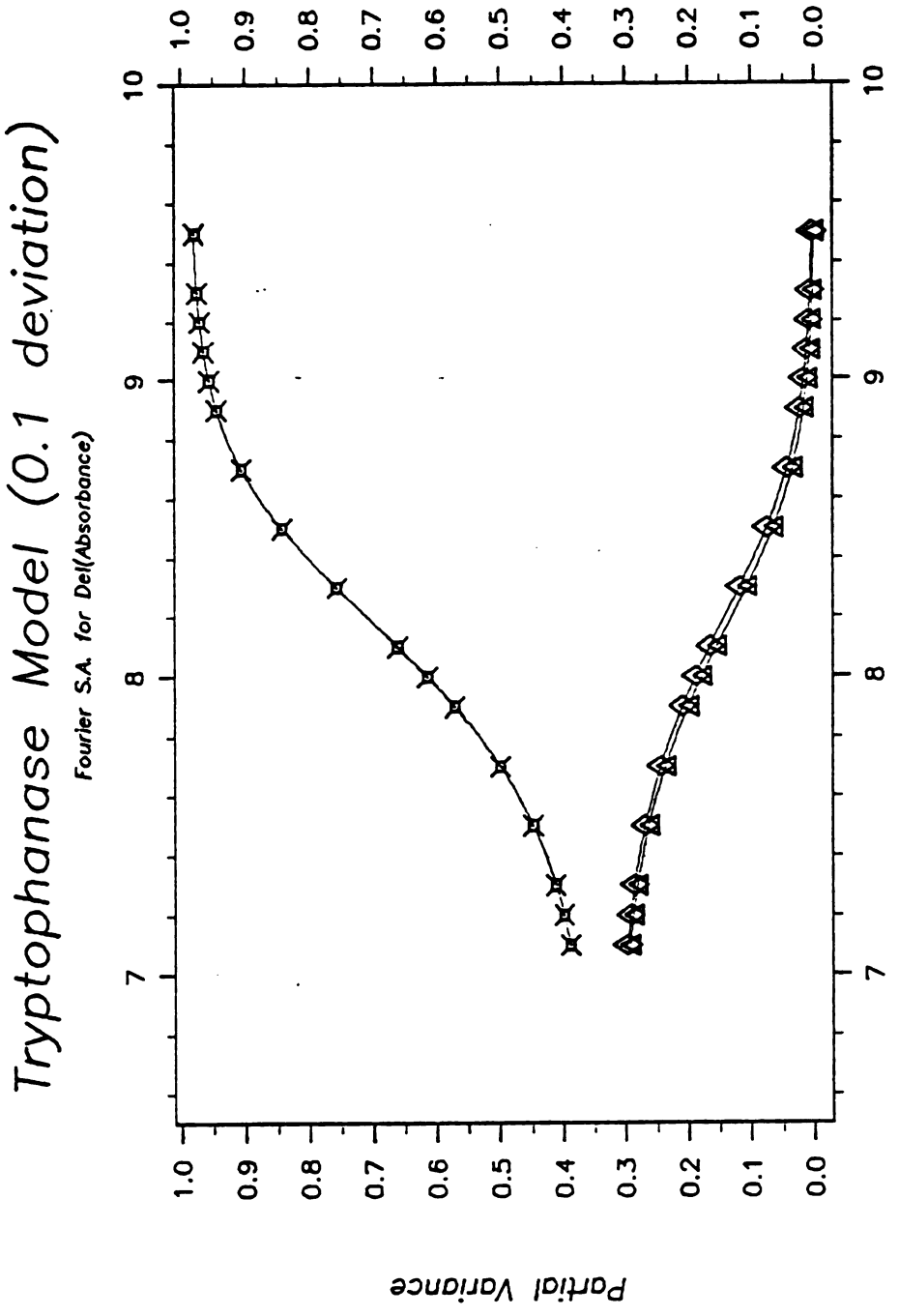


Figure 6.5 Partial variances of Del(ABS) from the Tryptophanase Model (10% variation).

sensitive to this parameter June et al. (1980) were able to fit it to less than 5% error. Both Figure 6.4 and Figure 6.5 show that the measurement of $\varepsilon_\beta \beta_t^o$ can be more accurately made at high pH values in pH jump experiments. The essentially equal sensitivities to $K_{\beta H}$ and $K_{\beta\gamma}$ reflect the fact that the amplitudes are most sensitive to $K_a \propto K_{\beta H} K_{\beta\gamma}$ as indicated by the computed standard deviation of K_a in Table 6.2.

Figure 6.6 is the standard deviation analysis for the apparent first order rate constant k'. Here at low pH, $K_{\beta\gamma}$ is the most important parameter. At higher pH, say 9, $k_{\beta\gamma}$ and $K_{\beta H}$ are the most important parameters. The partial variances for the 10% parameter variation, Figure 6.7, are not much different than those derived from the fitted standard deviation analysis. The overall shapes of the sensitivity curves are the same but the magnitude differ slightly. The maximum sensitivities of the parameters are grouped in two regions, $K_{\beta H}$ and $k_{\beta\gamma}$ are large at a pH of 9, while the other parameters reach their peak in the pH range of 7.0-7.4. The sensitivity to $k^H_{\beta\gamma}$ in the pH jump analysis is significant only at low pH values and neither the amplitude nor the rate constants in the pH jump analysis show appreciable sensitivity to $K_{\gamma H}$.

Figures 6.8 and 6.9 are the partial variances of the parameters in the pH drop model. The amplitude parameter $\varepsilon_\beta \beta_t^o$ is the most sensitive in the 10% deviation

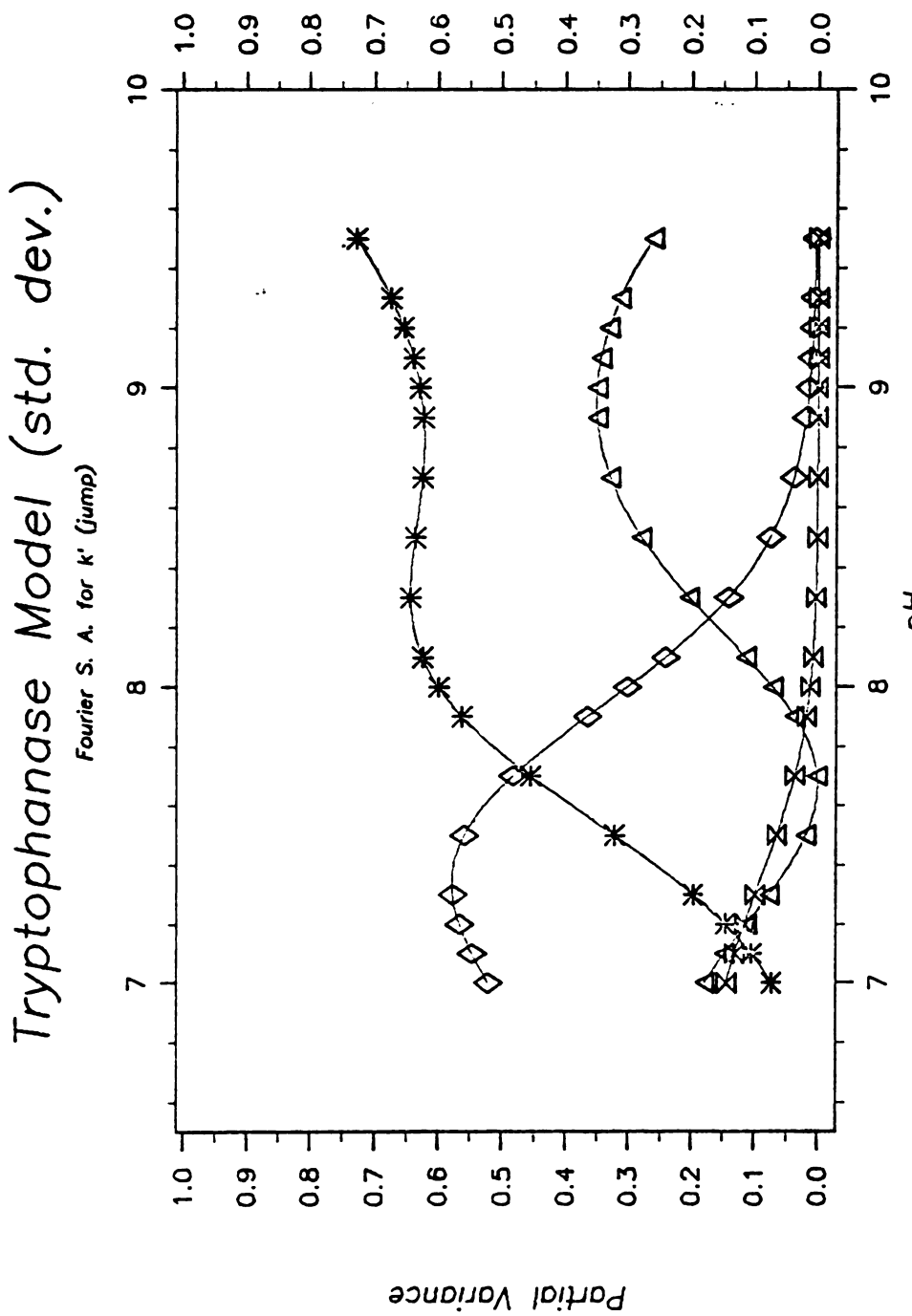


Figure 6.6 Partial variances of k' from the pH jump Tryptophanase Model (standard deviation variation).

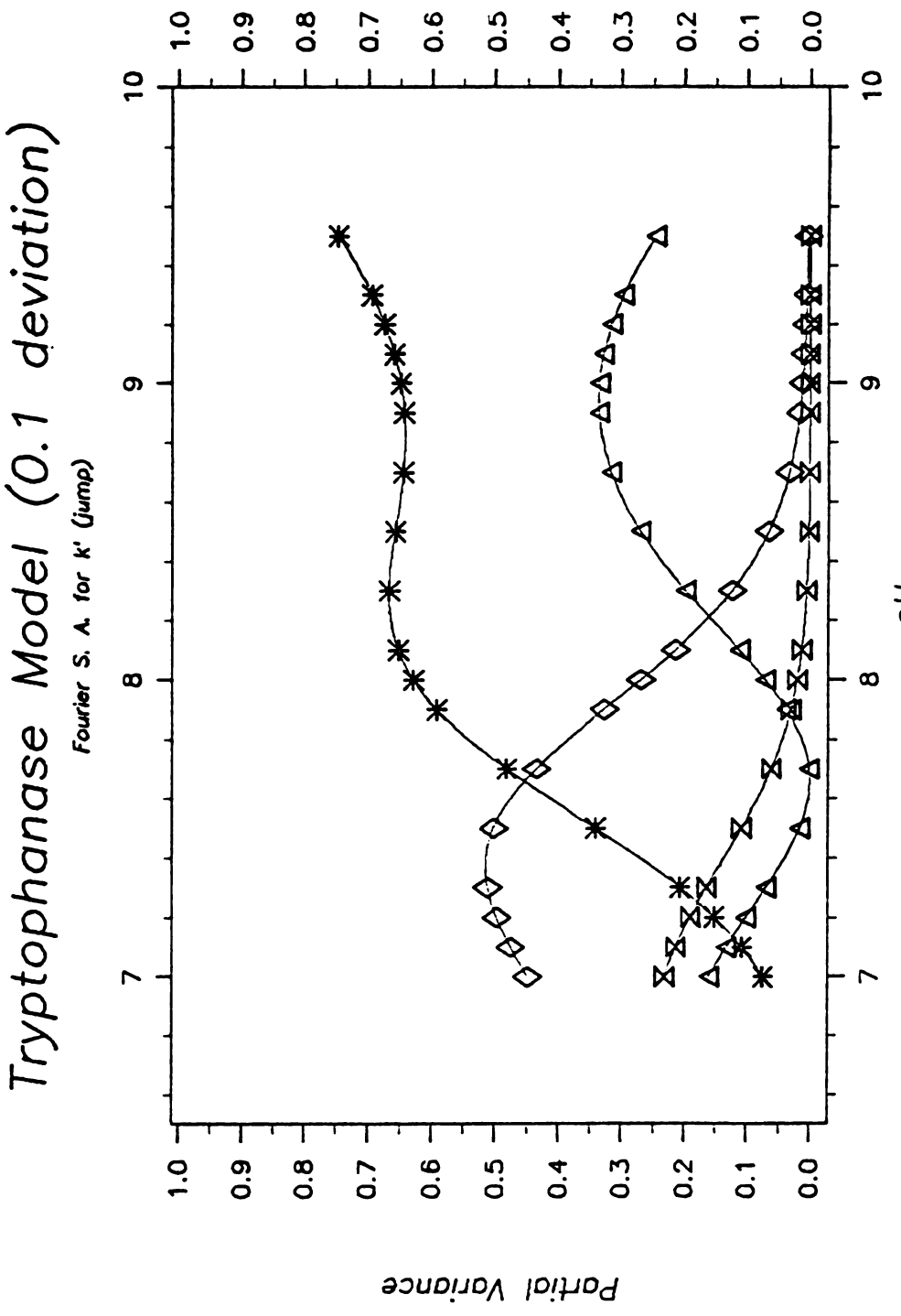


Figure 6.7 Partial variances of k' from the pH jump Tryptophanase Model (10% variation).

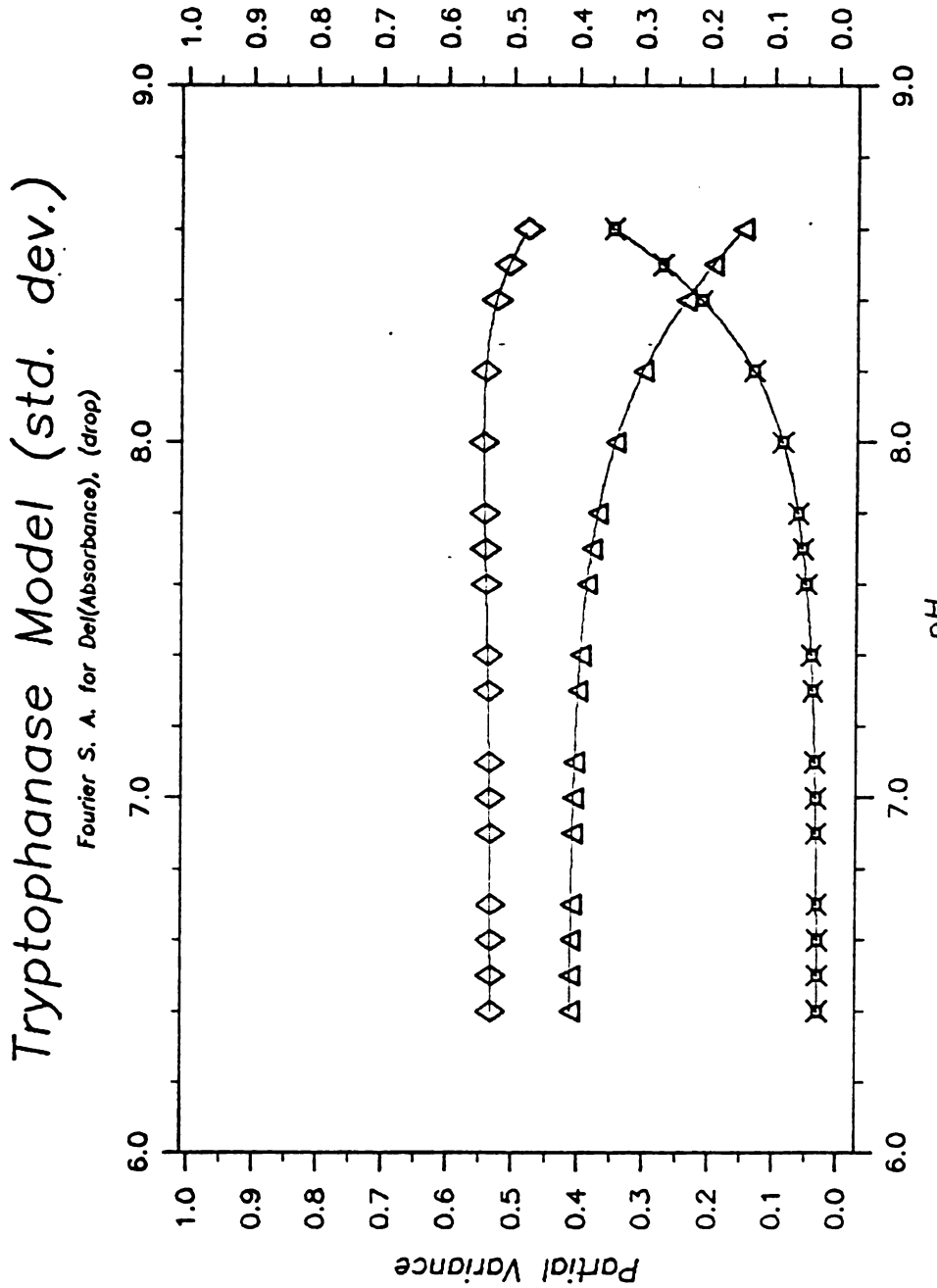


Figure 6.8 Partial variances of Del(ABS) from the pH drop Tryptophanase Model (standard deviation variation).

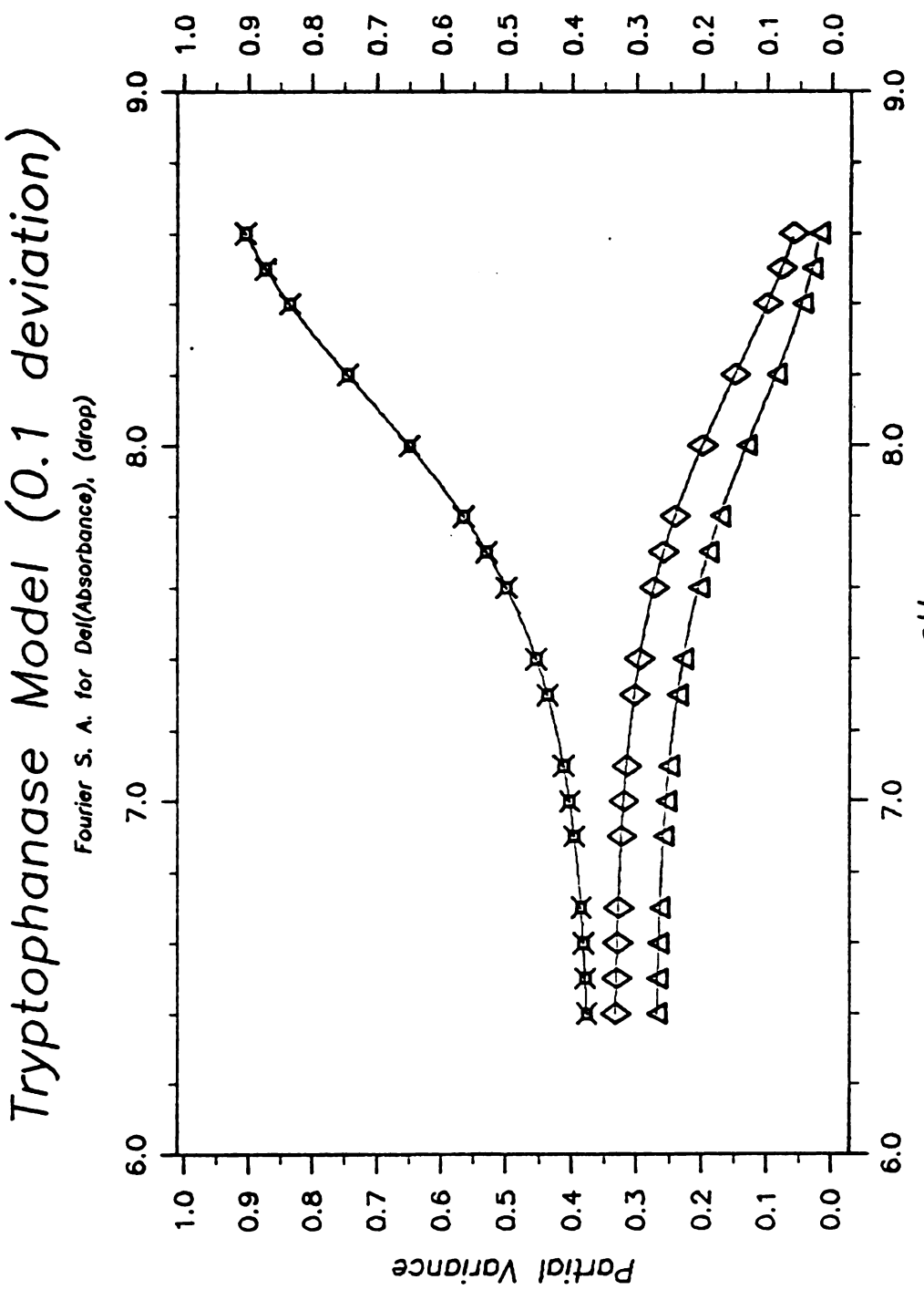


Figure 6.9 Partial variances of Del(ABS) from the pH drop Tryptophanse Model (10% variation).

analysis, but since it was accurately measured, the standard deviation analysis shows that the equilibrium constant $K_{\beta\gamma}$ is the largest. There are only two different regions of sensitivity in these analysis, a high pH set ($K_{\beta\gamma} > \varepsilon_\beta \beta_t^o > K_{\beta H}$) and a low pH set ($K_{\beta\gamma} > K_{\beta H} >> \varepsilon_\beta \beta_t^o$).

Figures 6.10 and 6.11 show the partial variances of the k' output function in the pH drop analysis. Here, for the first time, some sensitivity to $K_{\gamma H}$ appears at low pH values.

From the partial variance plots for the Tryptophanase model we see that the parameters can be grouped according to the pH dependence of their effect on the output functions. $K_{\beta H}$, $K_{\beta\gamma}$, and $\varepsilon_\beta \beta_t^o$ determine the value of ΔA_{obs} with reasonably uniform sensitivities at pH values below 9 when the standard deviations are used. Since $\varepsilon_\beta \beta_t^o$ is by far the most important parameter, allowing the same relative deviation for it as for the other parameters causes it to take most of the partial variance, from 40% at low pH values to over 95% at higher pH values. By restricting the ranges to the standard deviation values, $K_{\beta H}$ and $K_{\beta\gamma}$ become the dominant parameters except at high pH values.

Partial variances obtained when k' is the output function show that $K_{\beta\gamma}$ dominates at pH values around 7 but becomes third in importance above pH = 8.2. The parameters $k_{\beta\gamma}$ and $K_{\beta H}$ become the most important at high pH values

and maintain substantial sensitivity down to pH values of about 7.4. Only at pH values below about 7.6 do the sensitivities to $k^H_{\beta\gamma}$ and $K_{\gamma H}$ begin to become important. This reflects the fact that these parameters refer to a low pH pathway for the interconversion of the β and γ manifolds. To determine $k^H_{\beta\gamma}$ and $K_{\gamma H}$ with greater precision, the measurements should be extended to lower pH values if possible.

The sensitivities of k' to its parameters does not change much when the parameters are allowed to vary over equal relative intervals instead of over the estimated standard deviations. However, examination of Table 6.2 shows that the relative standard deviations for the most important parameters are not very different. Therefore, a change from ±10% relative deviation to ±σ is nearly the same as a change from ±10% on all parameters to ±20% on all parameters so that we would not expect much difference.

The sensitivity analysis applied here suggests what experiments should be done to further refine the parameters. Incremental pH jump experiments to higher pH values than the limit of 9.3 used to date would probably result in better estimates of $k_{\beta\gamma}$ and $K_{\beta H}$ while pH drop experiments to lower final pH values than 6.7 would greatly improve the determination of $k^H_{\beta\gamma}$ and $K_{\gamma H}$.

This application of sensitivity analysis to the Tryptophanase model was made after the model had been

developed and the parameters fit to the data. The marginal standard deviation estimates given by KINFIT4 for all of the parameters gave us an indication of their reliability. However, sensitivity analysis, not only confirmed these ideas, but also clearly delineated the regions of pH in which the absorbance changes and rate constants are most affected by particular parameters. Thus the major goals of sensitivity analysis, to rank the parameters in order of their importance to the output functions, and to assist in the design of future experiments, were both realized in this example.

VII. FUTURE WORK AND DEVELOPMENT

The previous chapters examined the theory and applications of Sensitivity Analysis. This chapter reviews those areas which should be profitable fields of research for further development of sensitivity analysis.

The most useful theoretical development would be in the relationship between the Walsh and Fourier methods. Christenson (1952) has laid the groundwork for this problem. He noted that Walsh functions may be generalized to sets of orthogonal functions with more than two values. This is done by relating the Walsh function to powers of $(\exp(2i\pi/N)$, where the two-value Walsh functions are obtained by letting $N = 2$, thereby giving powers of (-1). The generalized Walsh function may then take on N different values.

This relationship suggests that the N-point discrete Fourier transform may be totally developed from a discrete algebraic viewpoint without recourse to the continuous Fourier transform. If this were done, a clearer understanding of the errors involved in aliasing and choice of frequency sets should result. This would also lead to a more direct relationship between the linear sensitivity coefficients (Taylor series) and the Fourier expansion coefficients.

Choice of the frequency sets for sensitivity analysis has always been a limitation. In the Fourier method we may do sensitivity analysis with up to 50 parameters since their 4th order accurate frequency sets are known. However 6th order or higher accurate frequency sets are not known for an arbitrary number of parameters. It appears to be a difficult number - theoretic problem to even find a higher-order accurate set. However, finding higher-order accurate sets for arbitrary number of parameters would enable the computation of more accurate Fourier sensitivity analyses.

In Walsh analysis an arbitrary number of parameters may be evaluated. All the frequencies required for exact analysis are known (2^i). Unfortunately, the largest required frequency for a p-parameter set is 2^{p-1}. This requires 2^p simulations to compute the 2^{p-1} coefficient. For large values of p this becomes impractical. Analogous to the Fourier method we can develop approximate Walsh frequency sets to a required order of accuracy. Appendix 9 has an approximate Walsh frequency set which is 4th-order accurate. With this set of frequencies we can do approximate Walsh sensitivity analysis with up to 21 parameters using only 2^{12} simulations instead of 2^{21} which would be required for exact analysis.

This technique will work for any set of approximate frequencies, and with the apparent relationship of Fourier

and Walsh expansions we should be able to connect the approximate frequency sets from the two methods with each other. Unfortunately, an algorithm for finding approximate Walsh frequency sets has not been discovered, although it is easier to invent approximate Walsh sets than it is to invent approximate Fourier sets. The set given in Appendix 9 was chosen in an intuitive fashion. Obviously more work is required to develop a systematic method of finding approximate Walsh frequency sets for any desired accuracy. This should also clear up the problem of finding approximate Fourier frequency sets of arbitrary accuracy.

Another useful area of research is the connection of statistics and sensitivity analysis. Sensitivity analysis measures the effect on the output function of variations in the parameters. Statistics deals with the reverse problem, the effect on the parameters caused by variations, or errors, in the output function. Research in the relationships between sensitivity analysis and statistics would unite the more theoretical aspects of sensitivity analysis with the real world measurements used in statistics.

One direct approach is to "feed" the "answers" obtained from a least squares analysis of data directly into the sensitivity analysis programs. The least squares program delivers "best" estimates of the parameters and standard deviation estimates for each parameter. By using these values as the nominal parameters along with the standard

deviation as the range of variation a sensitivity analyis may be done on this model. From the partial variance curves obtained in this way one may determine whether the output function is sensitive to that particular parameter space. If there are maxima in the partial variance curves then one should make more measurements in that region to pin down the "best" value for the parameter in a least squares sense. Such an approach should be useful in both model reduction and experimental design.

The computer programs are well-designed. However, by examining the timing data printed by the programs it seems likely that improvement in the matrix transpose algorithm (SUBROUTINE TRANP) would decrease the amount of required computer time. Other than this, there are no new, faster algorithms (that I know of) which should be substituted for the ones presently used. However the programs were written to facilitate the replacement of sub-programs if better ones are developed.

One other place that the programs could be modified is in SUBROUTINE MODEL. It may cause a significant decrease in computer time if models written in terms of differential equations are recast into an integral equation form. Integral equations are usually more stable numerically than differential equations. This type of change could result in a decrease of orders of magnitude in the computer time spent computing the required simulations.

Applications of both the Fourier and Walsh sensitivity analysis should be straightforward. Interpretation of the results will, of course, depend on the problem. It is hoped that the applications and interpretations presented here are sufficiently detailed to enable interested researchers to perform sensitivity analysis on their own models. The insight available from sensitivity analysis is only realized after the model has been analyzed.

APPENDICES

APPENDIX 1

RELATIONSHIP OF FOURIER COEFFICIENTS TO TAYLOR SERIES COEFFICIENTS

If a function can be expanded in a Taylor series over an interval, it may also be expanded in terms of orthogonal polynomials over an equivalent interval. This may be written

$$f(x) = \sum_{j=0}^{\infty} a_j P_j(x) = \sum_{j=0}^{\infty} \frac{f^{(j)}_{(x_0)}}{j!} (x-x_0)^j$$

where $P_j(x)$ is an arbitrary orthogonal polynomial and $f^{(j)}_{(x_0)}$ is the jth derivative of f(x) with respect to 'x' evaluated at $x = x_0$.

By exploiting the orthogonality of the $P_j(x)$ polynomials we can relate the a_j expansion coefficients with the Taylor series coefficients, i.e.,

$$a_k = \sum_{j=0}^{\infty} \int w(x) P_j(x) P_k(x) dx = \sum_{j=0}^{\infty} \int w(x) \frac{f^{(j)}_{(x_0)}}{j!} (x-x_0)^j P_k(x) dx$$

$$= \sum_{j=0}^{\infty} \frac{f^{(j)}_{(x_0)}}{j!} w_{jk}$$

where

$$w_{jk} = \int w(x)\ (x-x_0)^j P_k(x)dx$$

From this equation we see that an orthogonal polynomial coefficient, a_k, is a weighted sum of all derivatives of the function evaluated at the nominal value, x_0. This implies that an orthogonal expansion coefficient is composed of the 'effects' of all the derivatives of the function.

We can specialize this result to the orthogonal series of sines and cosines. Expanding f(x) in terms of frequencies we obtain

$$f(x) = \sum_{j=1}^{\infty} \{a_j \cos(jx) + b_j \sin(jx)\} + \frac{a_0}{2}$$

where

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx)\ dx$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx)\ dx$$

Substituting in the Taylor series expansion for f(x) we obtain

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} (f(x_0) + f'(x_0)(x-x_0) + \ldots)\cos(kx)\, dx$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} (f(x_0) + f'(x_0)(x-x_0) + \ldots)\sin(kx)\, dx$$

Let $y = x-x_0$, then

$$a_k = \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} (f(x_0) + f'(x_0)y + \frac{1}{2} f''(x_0)y^2+\ldots)\cos(y+x_0)dy$$

$$b_k = \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} (f(x_0) + f'(x_0)y + \frac{1}{2} f''(x_0)y^2)\sin(y+x_0)dy$$

Using the expansion for $\sin(\alpha+\beta)$ and $\cos(\alpha+\beta)$ we obtain

$$a_k = \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} [f(x_0)\cos(y)\cos(x_0) - f(x_0)\sin(y)\sin(x_0)]dx$$

$$+ \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} [f'(x_0)(y)(\cos(y)\cos(x_0))-f'(x_0)y(\sin(y)\sin(x_0))]dx+\ldots$$

$$b_k = \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} [f(x_0)\sin(y)\cos(x_0) + f(x_0)\cos(y)\sin(x_0]dx$$

$$+ \frac{1}{\pi} \int_{-\pi+x_0}^{\pi+x_0} [f'(x_0)y\ \sin(y)\cos(x_0)+f'(x_0)y\ \cos(y)\sin(x_0]dx+\ldots$$

Setting the nominal value, x_0, to zero, we may reduce equations as follows

$$a_k = \sum_{j=0}^{\infty} \frac{1}{\pi} \int_{-\pi}^{\pi} f_{(0)}^{(2j)} y^{2j} \cos(ky)dy$$

$$+ \frac{1}{\pi} \int_{-\pi}^{\pi} f_{(0)}^{(2j+1)} y^{2j+1} \sin(ky)\, dy$$

$$= \sum_{j=0}^{\infty} \langle (y^{2j})(f_{(0)}^{(2j)})(\cos(ky))\rangle + \langle y^{2j+1} f_{(0)}^{(2j+1)} \sin(ky)\rangle$$

Similarly b_k may be reduced to

$$b_k = \sum_{j=0}^{\infty} \langle y^{2j+1} f_{(0)}^{(2j+1)} \sin(ky)\rangle - \langle y^{2j} f_{(0)}^{(2j)} \cos(ky)\rangle$$

This clearly shows that the Fourier coefficients are composed of all derivatives of the expansion function.

APPENDIX 2

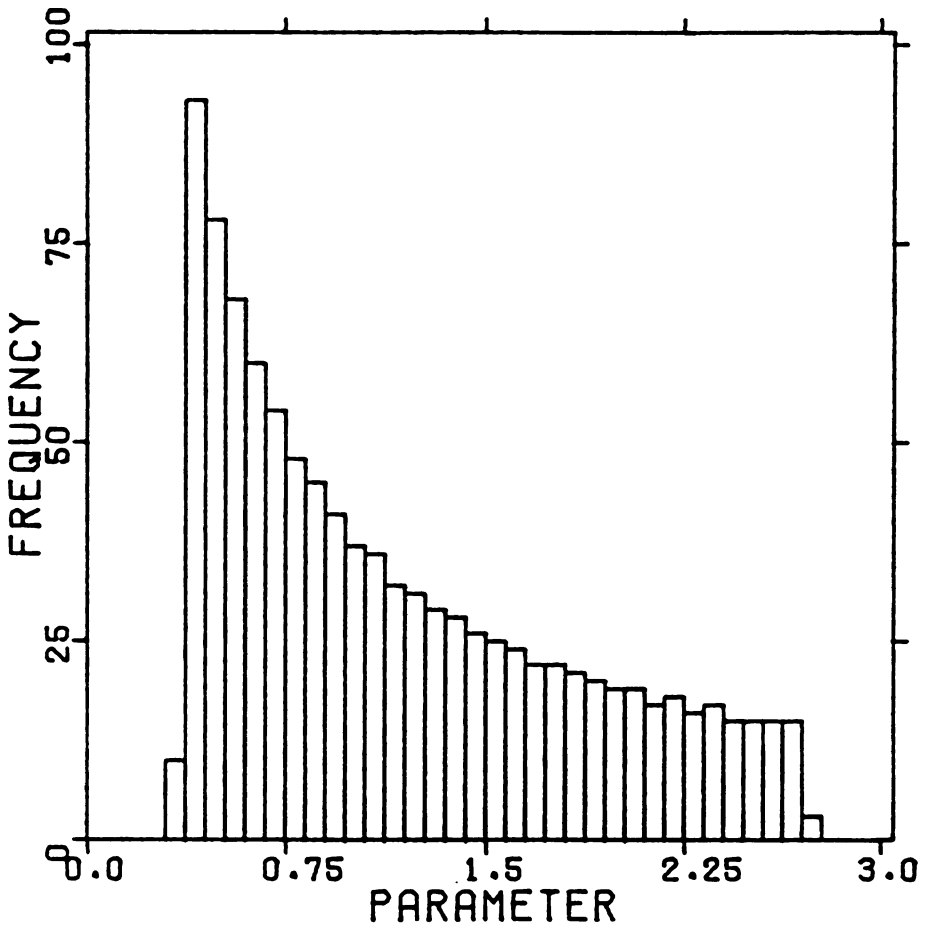


Figure A.1. Histogram of Log-uniform Distribution Function.

This function is given by:

$$\text{Parameter} - \text{nominal} * \exp\left(\Delta \frac{2}{\pi} \sin^{-1}(\sin(sq))\right)$$

where here

$$\text{nominal} = (P_{HI}/P_{LO})^{1/2}$$

$$= 1/2 \ \ell n(P_{HI} P_{LO}).$$

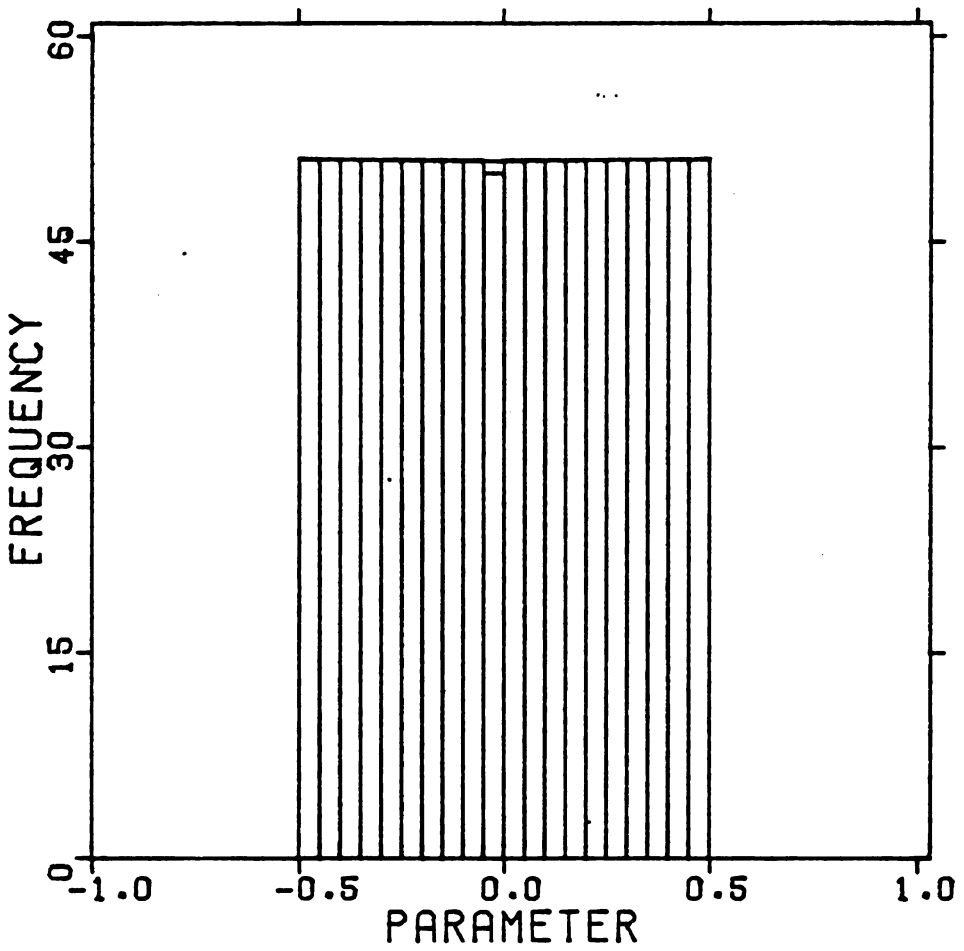


Figure A.2. Histogram of uniform distribution function.

This function is given by:

$$\text{Parameter} = \text{nominal} + \Delta\sin^{-1}(\sin(sq)).$$

where here

$$\text{nominal} = \frac{1}{2}(P_{HI}+P_{LO})$$

$$\Delta = \frac{1}{2}(P_{HI}-P_{LO}) \ .$$

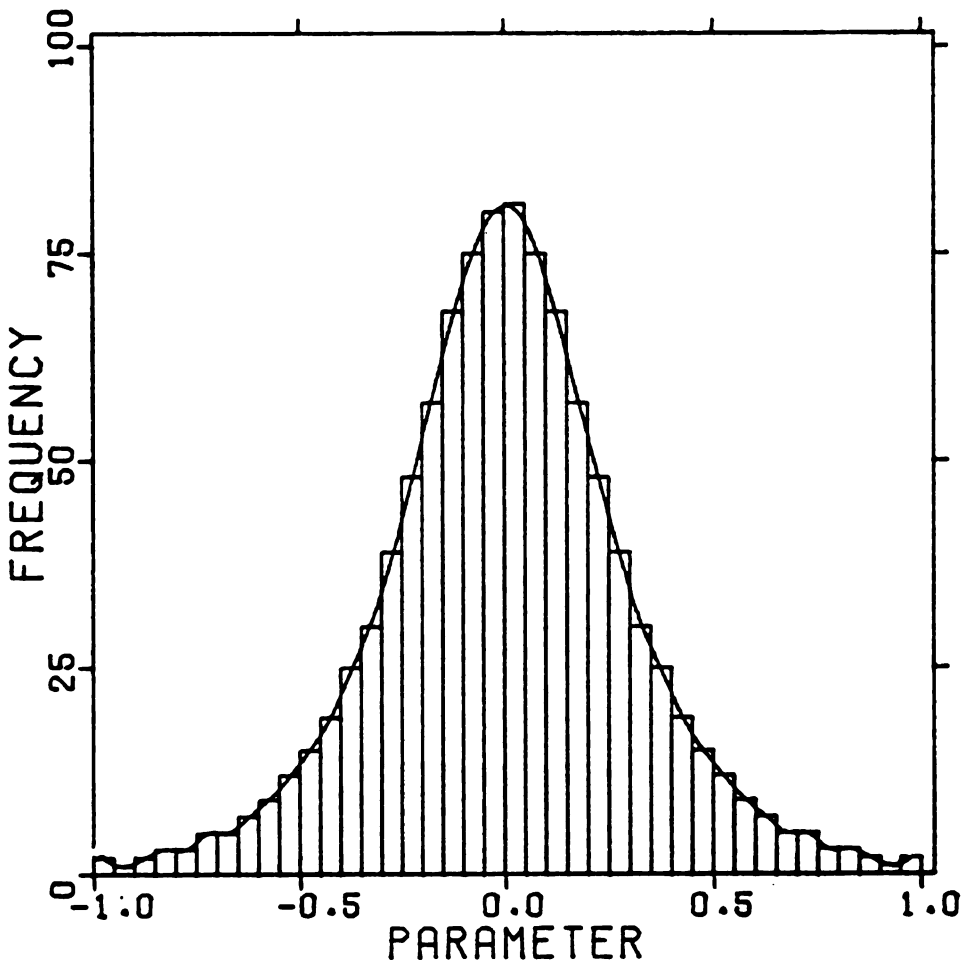


Figure A.3. Histogram of Gaussian-type distribution function.

This function is given by:

$$\text{Parameter} = \frac{2}{a} \log \left[\frac{1 + \sin(sq)}{1 - \sin(sq)}\right]$$

where

$$a = \frac{100}{P_{HI} - P_{LO}}$$

such that 90% of the samples are between P_{HI} and P_{LO}.

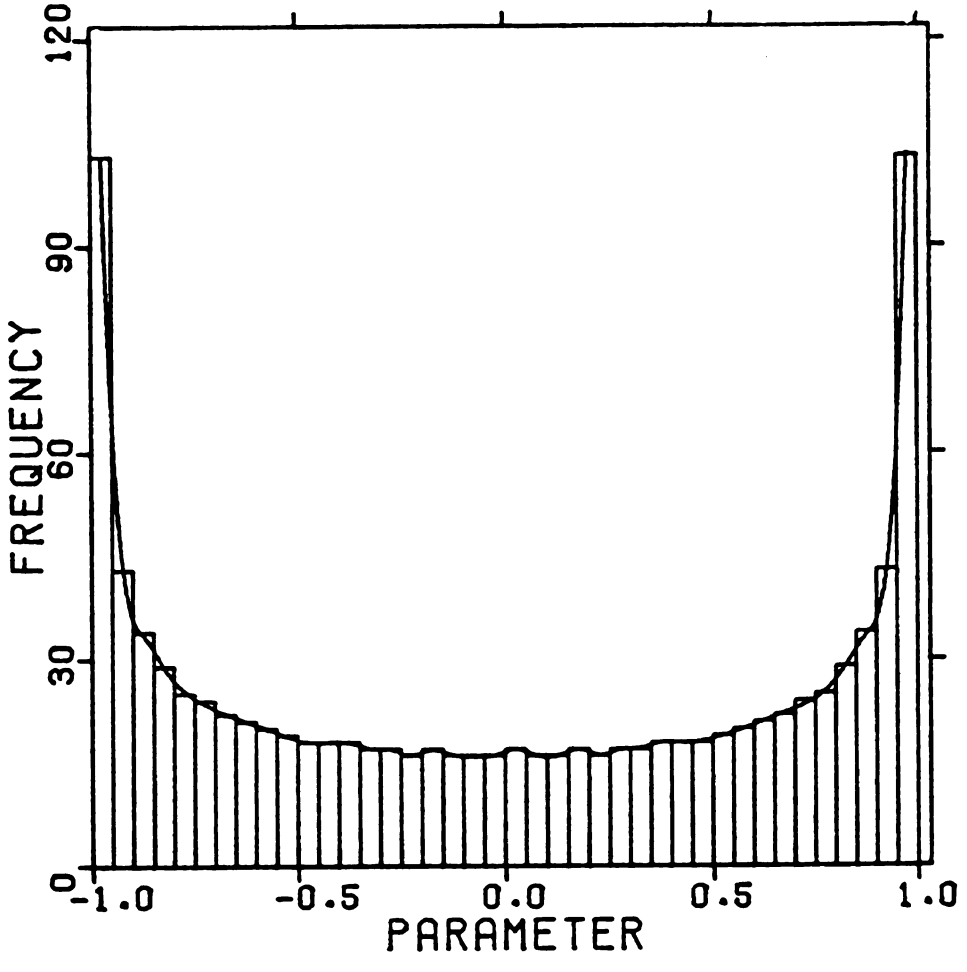


Figure A.4. Histogram plot of sin-transformation function.

This function is given by

$$\text{Parameter} = \text{nominal} + \Delta\sin(\text{sq})$$

where here

$$\text{nominal} = \frac{P_{HI}+P_{LO}}{2}$$

$$\Delta = \frac{P_{HI}-P_{LO}}{2} .$$

APPENDIX 3

This procedure follows the original Fast Fourier Transform, the Cooley-Tukey algorithm. In fact, some Fast Fourier Transform programs may be converted directly into Walsh transforms by simply setting all the trignometric values to ±1 and deleting the complex part of the transformation (since the Walsh transform is real).

The factorization of the transform relies on the lexicographic ordering of the sampled function values. Writing out the transform using binary representation for the time, $t = (t_1 t_2,\ldots,t_p)$, and for the sequency, $m = (m_1,m_2,\ldots,m_p)$

$$C_m = C(m_1 m_2 \ldots m_p) = \frac{1}{N} \sum_{n=0}^{N-1} f(t_n) WALH(n,t_n)$$

$$= \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} \cdots \sum_{m_p=0}^{1} f(t_1 t_2 \ldots t_p) (-1)^{\sum_{i=1}^{p} t_i m_i} \qquad (A\text{-}1)$$

The calculation of the transform is carried out in a series of stages. There is one stage for each power of two in the number of points, $2^p = N$. The first stage is to derive a partial transformation series, X_1, from the input series, f(t), by expanding the first sum in the equation (ignoring the scaling factor for now).

$$X_1(t_1 t_2 \ldots t_{p-1} m_p) = \sum_{t_p=0}^{1} (-1)^{t_p m_p} f(t_1 \ldots t_p)$$

$$= f(t_1 \ldots t_{p-1} 0) + (-1)^{m_p} f(t_1 \ldots t_{p-1} 1)$$

Now we pass through the data, either adding or subtracting adjacent function values. The second stage is constructed from the first by expanding the second sum. Then

$$X_2(t_1 t_2 \ldots t_{p-2} m_{p-1} m_p) = \sum_{t_{p-1}=0}^{1} X_1(t_1 \ldots t_{p-1} m_p)$$

This procedure is continued until all P-stages have been computed. The values of the last stage are the desired Walsh coefficients.

$$C_m = C(m_1 m_2 \ldots m_p) = X_p(m_1 m_2 \ldots m_p)$$

This is an extremely fast transform on a computer as only additions and subtractions are required.

APPENDIX 4

PARSEVAL'S FORMULA FOR WALSH FUNCTIONS

The total variance of a function may be expressed as the sum of its squared Walsh expansion coefficients. This may be easily seen by computing the variance for an arbitrary function. The defining equation for variance is

$$\sigma^2_{Total} = \langle (f(\underline{x}))^2 \rangle - \langle f(x) \rangle^2$$

where $\langle f(x) \rangle$ is defined as the average of the multidimensional function $f(\underline{x})$ with $\underline{x} = (x_1, x_2, \ldots x_p)$. $\langle (f(x)^2 \rangle$ is then the average of the square of the function $f(\underline{x})$.

Expanding $f(\underline{x})$ in a finite multidimensional Walsh series we obtain the following series:

$$f(\underline{x}) = \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \sum_{k_p=0}^{1} C_{k_1 k_2 \ldots k_p} \prod_{i=1}^{p} WALH(k_i, x_i)$$

$$= \sum_{\underline{k}} C_{\underline{k}} \prod_{i=1}^{p} WALH\,(k_i, x_i)$$

To compute the average of $f(\underline{x})$ we need only compute the average of the series expansion

$$\langle f(x) \rangle = \int_{\underline{x}} \sum_{\underline{k}} C_{\underline{k}} \prod_{i=1}^{p} \mathrm{WALH}\,(k_i, x_i)\, d\underline{x}$$

This equation must now be integrated over each dimension, x_i. However, since each dimension has only two values the multidimensional integral is equivalent to a multidimensional sum over these two values. This results in the following equation

$$\langle f(\underline{x}) \rangle = \frac{1}{2^p} \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_p=0}^{1} \left(\sum_{\underline{k}} C_{\underline{k}} \prod_{i=1}^{p} \mathrm{WALH}(k_i, x_i) \right)$$

$$= \frac{1}{2^p} \sum_{\underline{k}} C_{\underline{k}} \left\{ \sum_{x_p=0}^{1} \cdots \sum_{x_2=0}^{1} \sum_{x_1=0}^{1} \left(\prod_{i=1}^{p} \mathrm{WALH}(k_i, x_i) \right) \right\}$$

Substituting in the algebraic definition of the one digit Walsh function results in

$$\langle f(\underline{x}) \rangle = \frac{1}{2^p} \sum_{\underline{k}} C_{\underline{k}} \left\{ \sum_{x_p=0}^{1} \cdots \sum_{x_1=0}^{1} \left((-1)^{\sum_i k_i x_i} \right) \right\}$$

$$= \frac{1}{2^p} \sum_{\underline{k}} C_{\underline{k}} \left\{ \prod_{i=1}^{p} \left(1 + (-1)^{k_i x_i} \right) \right\}$$

The term in brackets is zero if any of the k_i's are nonzero. Hence it functions as a Kronecker delta and we may simplify the equation accordingly

$$\langle f(\underline{x})\rangle = \frac{1}{2^p} \sum_{\underline{k}} C_k \{2^p \delta_{\underline{k},\underline{o}}\} = C_{\underline{o}} = C_{oo\ldots o}$$

This shows that the average of an arbitrary function is the $C_{\underline{o}}$ coefficients of its Walsh expansion.

The computation of $\langle (f(\underline{x}))^2 \rangle$ is straightforward. First expressing the square of the function as a Cartesian product.

$$(f(x))^2 = \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} \prod_{i=1}^{p} \mathrm{WALH}(k_i, x_i) \prod_{j=1}^{p} \mathrm{WALH}(k_j, x_j)$$

which upon substitution of the definition of the Walsh function results in:

$$(f(\underline{x}))^2 = \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} (-1)^{\sum_{i=1}^{p} k_i x_i} (-1)^{\sum_{j} k'_j x_j}$$

This equation may be integrated over each dimension, x_i

$$\langle (f(\underline{x}))^2 \rangle = \int_{\underline{x}} \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} (-1)^{\sum_i (k_i + k_i')x_i}$$

$$= \frac{1}{2^p} \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} \{ \sum_{x_p=0} \cdots \sum_{x_2=0} [(1+(-1)^{(k_1+k_1')x_1})((-1)^{\sum_{i=2}^{p} (k_1+k_1')x_i} \}$$

$$= \frac{1}{2^p} \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} \{ \prod_{i=1}^{p} (1+(-1)^{(k_i+k_i')x_i}) \}$$

The term in brackets acts as a Kronecker delta, requiring $\underline{k} = \underline{k}'$. Hence the equation is now easily reduced

$$\langle (f(\underline{x}))^2 \rangle = \frac{1}{2^p} \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} \prod_{i=1}^{p} 2\delta_{k_i, k_i'}$$

$$= \frac{1}{2^p} \sum_{\underline{k}} \sum_{\underline{k}'} C_{\underline{k}} C_{\underline{k}'} 2^p \delta_{\underline{k}, \underline{k}'}$$

$$= \sum_{\underline{k}} (C_{\underline{k}})^2$$

giving a sum of the squared coefficients.

Combining the two results we may directly write the equation for the variance solely in terms of the expansion coefficients:

$$\sigma^2_{Total} = \langle (f(\underline{x}))^2 \rangle - \langle f(\underline{x}) \rangle = \sum_{\underline{k}} C^2_{\underline{k}} - C^2_{00\ldots0} = \sum'_{\underline{k}} C^2_{\underline{k}}$$

where the $\sum'_{\underline{k}}$ is a summation over all $\underline{k}$ except the $00\ldots0$ sequency term.

APPENDIX 5

THE CALCULATION OF WALSH PARTIAL VARIANCES

A partial variance is defined as the variance, or dispersion, in one dimension of a multidimensional function

$$\sigma_1^2 = \langle (f^*(x_1))^2 \rangle - \langle f^*(x_1) \rangle^2$$

where $f^*(x_1) = \langle f(x_1, x_2, x_3, \ldots, x_n) \rangle$, the multidimensional function averaged over all dimensions except the first. σ_1^2 is called the partial variance of variable, or parameter, x_1.

If $f(x_1, \ldots, x_n)$ is expanded in a multidimensional finite Walsh series with two points along each axis, then k_i and x_i may be represented in binary by one digit.

$$f(x_1, x_2 \ldots x_n) = \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} C_{k_1 \ldots k_n} \prod_{i=1}^{n} \mathrm{WALH}(k_i, x_i)$$

$$= \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} C_{k_1 \ldots k_n} (-1)^{\sum_{i=1}^{n} k_i x_i}$$

The average of the function with respect to $x_2,\ldots x_n$ is the sum of all those values divided by the number of samples

$$f^*(x_1) = \langle f(x_1 \ldots x_n) \rangle_{x_2 \ldots x_n}$$

$$= \frac{1}{2^{N-1}} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} \cdots \left[\sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} C_{k_1 \ldots k_n} (-1)^{\sum_{i=1}^{n} k_i x_i} \right]$$

Switching summations:

$$= \frac{1}{2^{N-1}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \left[\sum_{x_n=0}^{1} \cdots \sum_{x_2=0}^{1} C_{k_1 \ldots k_n} (-1)^{\sum_{i=1}^{n} k_i x_i} \right]$$

expanding the term in brackets:

$$f^*(x_1) =$$

$$= \frac{1}{2^{N-1}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} C_{k_1 \cdots k_n} (-1)^{k_1 x_1} \left[\sum_{x_n=0}^{1} \cdots \sum_{x_3=0}^{1} (1+(-1)^{k_2}) (-1)^{\sum_{i=3}^{n} k_i x_i} \right.$$

$$= \frac{1}{2^{n-1}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} C_{k_1 \cdots k_n} (-1)^{k_1 x_1} \left[\prod_{i=2}^{n} (1+(-1)^{k_i}) \right]$$

$$= \frac{1}{2^{n-1}} \sum_{k_1=0}^{1} C_{k_1 00 \cdots 0} (-1)^{k_1 x_1} 2^{n-1} = \sum_{k_1=0}^{1} C_{k_1 0 \cdots 0} (-1)^{k_1 x_1}$$

Now calculate the second moment, $\langle f^*(x_1)^2 \rangle$, using the function $f^*(x_1)$ and squaring the summing over x_1.

$$\langle (f^*(x_1))^2 \rangle = \frac{1}{2} \sum_{x_1=0}^{1} \left\{ \sum_{k_1=0}^{1} C_{k_1 0 \cdots 0} (-1)^{k_1 x_1} \sum_{k_1'=0}^{1} C_{k_1' 0 \cdots 0} (-1)^{k_1' x_1} \right\}$$

$$= \frac{1}{2} \sum_{k_1=0}^{1} \sum_{k_1'=0}^{1} C_{k_1 0 \cdots 0} C_{k_1' 00 \cdots 0} \sum_{x_1=0}^{1} (-1)^{(k_1+k_1')x_1}$$

$$= \frac{1}{2} \sum_{k_1=0}^{1} \sum_{k_1'=0}^{1} C_{k_1 0 \cdots 0} C_{k_1' 0 \cdots 0} (1 + (-1)^{k_1+k_1'})$$

$$= C_{10 \cdots 0}^2 + C_{0 \cdots 0}^2$$

To calculate the second required term, we need $\langle f^*(x_1)\rangle_{x_1}$

$$\langle f^*(x_1)\rangle_{x_1} = \frac{1}{2} \sum_{x_1=0}^{1} \sum_{k_1=0}^{1} C_{k_1 0..0} (-1)^{k_1 x_1}$$

$$= \frac{1}{2} \sum_{k_1=0}^{1} C_{k_1 0..0} \left(1 + (-1)^{k_1}\right)$$

$$= C_{0..0}$$

Hence

$$\langle f^*(x_1)\rangle_{x_1}^2 = C_{0..0}^2$$

Subtracting the first moment from the second, we obtain the desired partial variance

$$\sigma_1^2 = \langle f^*(x_1)^2\rangle_{x_1} - \langle f^*(x_1)\rangle_{x_1}^2 = C_{10..0}^2$$

APPENDIX 6

DERIVATION OF WALSH COUPLED PARTIAL VARIANCE FORMULAS

To measure the effect of coupled parameters, say "ℓ" coupled together at a time, we calculate a coupled partial variance.

$$\sigma^2_{1,2,..,\ell} = \langle f^*(x_{\ell+1}..x_\ell)^2 \rangle - \langle f(\underline{x}) \rangle^2_{\underline{x}}$$

where $f^*(x_{\ell+1}..x_\ell)$ is the function $f(\underline{x})$ averaged over the variables $x_{\ell+1}..x_n$, and $\langle f(\underline{x}) \rangle^2_{\underline{x}}$ is the average of the multi-dimensional function over all the variables $x_1..x_n$.

Expand $f(x_1..x_n)$ in an n-dimensional finite Walsh series with two points along each axis. In this case x_i and k_i may be represented by a one digit binary number.

$$f(x_1-x_n) = \sum_{\underline{k}} C_{\underline{k}} \prod_{j=1}^{n} \text{WALH}\ (k_i.x_i)$$

The average of $f(\underline{x})$ is the $C_{\underline{o}}$ term as previously shown

$$\langle f(\underline{x}) \rangle = C_{\underline{o}}$$

So we need only calculate the term $\langle f^*(x_1..x_\ell)^2 \rangle$.

First calculating $f^*(x_1..x_\ell)$

$$f^*(x_1..x_\ell) = \langle f(x_1..x_n) \rangle_{x_{\ell+1}..x_n}$$

$$= \frac{1}{2^{n-\ell}} \sum_{x_{\ell+1}=0}^{1} .. \sum_{x_n=0}^{1} \{ \sum_{\underline{k}} C_{\underline{k}} \prod_{i=1}^{n} WALH(k_i, x_i) \}$$

$$= \sum_{\underline{k}} C_{\underline{k}} \frac{1}{2^{n-\ell}} \sum_{x_{\ell+1}=0}^{1} \cdots \sum_{x_n=0}^{1} (-1)^{\sum_{i=1}^{n} k_i x_i}$$

$$= \sum_{\underline{k}} C_{\underline{k}} \frac{1}{2^{n-\ell}} [\prod_{i=\ell+1}^{n} (1+(-1)^{k_i})] \, (-1)^{\sum_{i=1}^{\ell} k_i x_i}$$

$$= \sum_{\underline{k}} C_{\underline{k}} \, (-1)^{\sum_{i=1}^{\ell} k_i x_i} \, \delta_{k_{\ell+1},0} \delta_{k_{\ell+2},0} \cdots \delta_{k_n,0}$$

$$f^*(x_1..x_\ell) = \sum_{k_1=0}^{1} \cdots \sum_{k_\ell=0}^{1} C_{k_1..k_\ell 0..0} (-1)^{\sum_{i=1}^{\ell} k_i x_i} = \sum_{\underline{k}^*} C_{\underline{k}^*} (-1)^{\sum_{i=1}^{\ell} k_i x_i}$$

We must now square this function $f^*(x_1...x_\ell)$ and average it over $x_1...x_\ell$.

$$f^*(x_1..x_\ell) f^*(x_1..x_\ell) = \sum_{\underline{k}^*} \sum_{\underline{n}^*} C_{\underline{k}^*} C_{\underline{n}^*} (-1)^{\sum_{i=1}^{\ell} k_i x_i} (-1)^{\sum_{i=1}^{\ell} n_i x_i}$$

$$\langle f^*(x_1..x_\ell)^2\rangle = \frac{1}{2^\ell} \sum_{x_1=0}^{1} .. \sum_{x_\ell=0}^{1} \{ \sum_{\underline{k}^*} \sum_{\underline{n}^*} C_{\underline{k}^*} C_{\underline{n}^*} (-1)^{\sum_{i=1}^{\ell} (k_i+n_i)x_i} \}$$

Rearranging the summations

$$\langle f^*(x_1..x_\ell)^2\rangle = \sum_{\underline{k}^*} \sum_{\underline{n}^*} C_{\underline{k}^*} C_{\underline{n}^*} \{ \frac{1}{2^\ell} \prod_{i=1}^{\ell} (1 + (-1)^{k_i n_i}) \}$$

$$= \sum_{\underline{k}^*} \sum_{\underline{n}^*} C_{\underline{k}^*} C_{\underline{n}^*} \prod_{i=1}^{\ell} \delta_{n_i,k_i}$$

$$= \sum_{\underline{k}^*} C^2_{\underline{k}^*} = \sum_{k_1=0}^{1} \cdots \sum_{k_\ell=0}^{1} C^2_{k_1..k_\ell 0..0}$$

Substituting the appropriate expressions for the two moments we obtain the ℓth coupled partial variance

$$\sigma^2_{1,2,..\ell} = \sum_{\underline{k}^*} C^2_{\underline{k}^*} = C_{\underline{o}} = \sum_{\underline{k}^*}{}' C^2_{\underline{k}^*}$$

where $\sum_{\underline{k}^*}{}'$ is a summation of all $\underline{k}^*$ vectors except the one equal to $\underline{0}$.

APPENDIX 7

RELATIONSHIP BETWEEN LINEARLY DEPENDENT EQUATIONS AND THEIR FOURIER COEFFICIENTS

In chemical kinetics mass balance equations often allow us to substitute an algebraic equation for a differential one. These mass balance equations are linear and in enzyme kinetics they are of the form:

$$E_o = \sum_{i=1}^{N} \nu_i X_i$$

where the X_i are the different types of enzyme-containing intermediates, and the ν_i is the number of enzyme molecules in specie X_i.

Given N-1 "X_i" expansions in fourier series, the fourier coefficients of the Nth species may be calculated. Using these fourier coefficients one can calculate the partial variances of the Nth specie.

This can be seen by inserting the N-1 fourier expansions into the mass balance equation

$$E_0 = \sum_{j=1}^{i-1} \nu_j \left[\sum_{L=0}^{M} a_L^{(j)} \cos LX + b_L^{(j)} \sin LX \right] + \nu_i X_i$$

$$+ \sum_{j=i+1}^{N} \nu_j \left[\sum_{L=0}^{M} a_L^{(j)} \cos LX + b_L^{(j)} \sin LX \right]$$

Solve for X_i

$$X_i = -\frac{1}{\nu_i} \left[\sum_{K=1}^{N-1} \nu_K \left\{ \sum_{L=0}^{M} a_L^{(K)} \cos LX + b_L^{(K)} \sin LX \right\} - E_0 \right]$$

$$= \left\{ \sum_{L=0}^{M} \left(\sum_{K=1}^{N-1} \frac{-\nu_K}{\nu_i} a_L^{(K)} \right) \cos LX + \left(\sum_{K=1}^{N-1} \left(\frac{-\nu_K}{\nu_i} \right) b_L^{(K)} \right) \sin LX \right\} - E_0$$

$$= \sum_{L=0}^{M} a'_L \cos LX + b'_L \sin LX$$

Hence the fourier coefficients of X_i are

$$a'_0 = \sum_{K=1}^{N-1} \frac{-\nu_K}{\nu_i} a_0^{(K)} - E_0$$

$$a'_L = \sum_{K=1}^{N-1} \frac{-\nu_K}{\nu_i} a_L^{(K)}$$

$$b'_L = \sum_{K=1}^{N-1} \frac{-\nu_K}{\nu_i} b_L^{(K)}$$

APPENDIX 8

SENSITIVITY ANALYSIS PROGRAMS

The use of the Sensitivity Analysis Programs at Michigan State University is a straightforward task. If the mathematical model is composed of ordinary differential equations or algebraic equations, no modifications to the programs are necessary. The equations only have to be coded into FORTRAN 66. After this is done, one has to decide: What kind of an analysis is desired (Walsh or Fourier), the parameters' nominal values and range of variation, the transformation function to be used, and the time points of interest. This data is read by the program SENANAL which does the required simulations. A second program, TRANS, reads the output from SENANAL, TAPE3, and computes the expansion coefficients and partial variances, both single and coupled. Since a Sensitivity Analysis may generate a large amount of data, depending on the number of output functions, parameters, and time points, an optional plotting program, PLTSEN, is provided. This program reads the output from TRANS, TAPE9, and plots four sets of curves for each output function. PLTSEN plots the average value of the output function, the single partial variances, the expansion coefficients, and the relative deviation.

The following cards execute the programs SENANAL, TRANS, AND PLTSEN.

```
 1.  PNC CARD
 2.  JOB CARD
 3.  PW CARD
 4.  ATTACH,MAIN,SENANALBINARYOPT2.
 5.  FTN,I=INPUT,B=SUB.
 6.  LOAD,MAIN.
 7.  LOAD,SUB.
 8.  EXECUTE.
 9.  RETURN,LGO.
10.  REWIND,TAPE3.
11.  ATTACH,TBIN,TRANSFORMBINARYOPT2.
12.  LOAD,TBIN.
13.  EXECUTE.
14.  CATALOG,TAPE9,SENSITIVITYANALYSISFILE,RP=30.
15.  ATTACH,PLT,PLTSENBINARYOPT2.
16.  RETURN,LGO.
17.  REWIND,TAPE9.
18.  PLT.
20.  (789)
```

```
SUBROUTINE MODEL (TIN, TOUT, YIN, NFUNC, TSTART)
COMMON /PARA/ P(50)
```

This is a subroutine which on input has TIN as the initial value at which the output functions have values (there are NFUNC output functions, YIN(1) is the first output function). TSTART is optional and tells MODEL when it is starting a new parameter vector (IF (TSTART .EQ. TIN)). The common block /PARA/ contains the parameters to be varied in the model.

On output from MODEL, the output functions, sometimes called object functions, have their values at 'TOUT', the time on returning from MODEL.

Note that this subroutine must change its FORTRAN code for each different mathematical model, but not for parameter variations.

(789)

Data cards for program SENANAL- see the comment cards in SUBROUTINE READIN.

(789)

Data cards for program PLTSEN, see the comment cards in the program PLTSEN

(789)

(6789)

The somewhat difficult part is to force SUBROUTINE MODEL to solve for the output functions given a parameter set, the initial values of the output functions and the time at which the solution is desired. The application to algebraic equations is straightforward. However, solving differential equations is more difficult. The use of the EPISODE package (Hindemarsh, 1977) for solving ordinary differential equations is recommended and is a part of the SENANAL package. These subprograms are extensively documented internally with comment cards.

Program SENANAL

```
      PROGRAM SENANAL(INPUT=65,OUTPUT=65,TAPE1=INPUT,TAPE2=OUTPUT,
     + TAPE3=513)
C*
C*************************************************************************
C*                                                                       *
C*    PROGRAM SENANAL IS THE DRIVER PROGRAM OF A SUITE OF CODES          *
C*    WHICH PREFORMS SENSITIVITY ANALYSIS ON A MODEL.  SENANAL           *
C*    READS INPUT AND BASED ON THAT INPUT CHOSES WALSH SENSITIVITY       *
C*    ANALYSIS METHOD OR FOURIER SENSITIVITY ANALYSIS METHOD.  IT THEN   *
C*    PROCEDES TO SOLVE THE MODEL EQUATIONS OVER THE DESIRED             *
C*    TIME POINTS WITH THE NECESSARY PAPAMETERS.  EACH                   *
C*    PARAMETER VECTOR WHICH IS TO BE SOLVED IS CALLED A SIMULATION.     *
C*    SENANAL SOLVES THE SIMULATIONS BY FIRST CREATING THE PARAMETER     *
C*    VECTOR AND THEN SOLVING THE MODEL EQUATIONS OVER ALL THE DESIRED   *
C*    TIME POINTS.  THE MODEL SOLUTIONS ( OBJECT FUNCTIONS ) ARE         *
C*    WRITTEN OUT TO TAPE3 AT EACH TIME POINT.                           *
C*                                                                       *
C*         AFTER A SIMULATION IS COMPLETED, SENANAL CREATES ANOTHER       *
C*    PARAMETER VECTOR AND SOLVES THE NEXT SIMULATION.  THIS IS          *
C*    REPEATED UNTIL ALL THE NECESSARY SIMULATIONS HAVE BEEN SOLVED.     *
C*                                                                       *
C*  VARIABLES                                                            *
C*                                                                       *
C*    BEGIN(NFUNC) = THE INITIAL CONDITIONS, OR EQUIVALENTLY             *
C*    (REAL)         THE VALUES OF THE OBJECT FUNCTIONS AT TSTART.       *
C*                                                                       *
C*                                                                       *
C*    IACCUR = ORDER OF ACCURACY OF THE FREQUENCY SET                    *
C*    (INTEGER)                                                          *
C*                                                                       *
C*                                                                       *
C*    IOMEGA =   0 IF FOURIER 4TH ORDER SET IS TO BE USED                *
C*    (INTEGER)  1 IF SPECIAL FREQUENCY SET IS TO BE USED                *
C*              -1 IF STANDARD WALSH FREQUENCY SET IS TO BE USED         *
C*                                                                       *
C*    IMETH = METHOD FLAG FOR SENSITIVITY ANALYSIS; = 1 FOR FOURIER      *
C*    (INTEGER)         ANALYSIS, = 2 FOR WALSH  ANALYSIS.               *
C*                                                                       *
C*    ITRANS = FLAG FOR TYPE OF TRANSFORMATION FUNCTION                  *
C*    (INTEGER)      SEE SUBROUTINE PARAM FOR DETAILS.                   *
C*                                                                       *
C*    IW(NPARA) = AN ARRAY CONTAINING THE FREQUENCY SET TO BE USED IN    *
C*    (INTEGER)      THE S. A. RUN.                                      *
C*                 IF 'IOMEGA' .EQ. 1, THIS ARRAY MUST BE READ IN FROM  *
C*                 CARDS.  OTHERWISE THE FREQUENCY SETS ARE CREATED     *
```

Program SENANAL, CONT'D.

```
C*                INTERNALLY.                                          *
C*                                                                     *
C*    NFUNC = NUMBER OF OBJECT FUNCTIONS WHICH WILL BE SAVED           *
C*    (INTEGER)   AT EACH TIME POINT.                                  *
C*                                                                     *
C*    NLABEL(NFUNC) = THE NAMES (LABELS) OF THE OBJECT FUNCTIONS       *
C*                    NLABEL(1) SHOULD BE THE NAME OF THE FIRST        *
C*                    OBJECT FUNCTION, ETC.                            *
C*                                                                     *
C*    NPARA = NUMBER OF PARAMETERS TO VARY                             *
C*    (INTEGER)                                                        *
C*                                                                     *
C*    NSIMUL = NUMBER OF SIMULATIONS                                   *
C*    (INTEGER)                                                        *
C*                                                                     *
C*    PHI(NPARA) = MAXIMUM VALUES OF THE PARAMETERS( OR ONE SIGMA MAX)*
C*    (REAL)                                                           *
C*                                                                     *
C*    PLO(NPARA) = MINIMUM VALUES OF THE PARAMETERS( OR ONE SIGMA MIN)*
C*    (REAL)                                                           *
C*                                                                     *
C*    TIME(TNPTS) = ARRAY CONTAINS THE TIME POINTS AT WHICH THE        *
C*    (REAL)        OUTPUT FUNCTIONS ARE TO BE SAVED AND THE           *
C*                  SENSITIVITY ANALYZED.                              *
C*                                                                     *
C*    TSTART = INITIAL TIME POINT, SO THERE ARE NO S. A. VALUES SAVED  *
C*    (REAL)   AT THIS POINT                                           *
C*                                                                     *
C*    TITLE(8) = A ONE CARD TITLE FOR S. A. RUN                        *
C*    (INTEGER)    (WRITTEN IN 8A10 FORMAT )                           *
C*                                                                     *
C*    TNPTS = NUMBER OF TIME POINTS                                    *
C*    (INTEGER)                                                        *
C*                                                                     *
C*    YIN(NFUNC) = AN ARRAY OF LENGTH NFUNC CONTAINING ON              *
C*    (REAL)     INPUT TO SUBROUTINE MODEL THE VALUES OF               *
C*               OBJECT FUNCTIONS AT  TIN  AND UPON OUTPUT FROM MODEL  *
C*               YIN( ) CONTAINS THE VALUES OF THE OBJECT FUNCTIONS    *
C*               AT TOUT.                                              *
C*                                                                     *
C*                                                                     *
C*                                                                     *
C*                                                                     *
C* ERROR CODES                                                         *
C*                                                                     *
C*    STOP "R1" OR STOP 1: IF IOMEGA WAS UNACCEPTABLE,EITHER NOT READ *
C*          CORRECTLY OR ABS(IOMEGA) .GT. 1                            *
C*                                                                     *
C*    STOP "R2" OR STOP 2: IMETH WAS UNACCEPTABLE (.LT.1 .OR. .GT. 2) *
```

Program SENANAL, CONT'D.

```
C*                                                                      *
C*       STOP 3: TNPTS WAS TOO LARGE OR TOO SMALL (0,150)               *
C*                                                                      *
C*       STOP 4: ITRANS WAS OUTSIDE DEFINED RANGE (1,5)                 *
C*                                                                      *
C*    STOP "R4" OR STOP 5: NFUNC HAS A VALUE OUTSIDE THE DEFINED RANGE  *
C*          ( 1,40)                                                     *
C*                                                                      *
C*    STOP "R5" OR STOP 6: NSIMUL HAS A VALUE OUTSIDE THE DEFINED       *
C*          RANGE ( .GE. 1)                                             *
C*                                                                      *
C*       STOP 7: PHI(J) .LE. PLO(J), THIS COULD CAUSE A DIVIDE          *
C*           BY ZERO IN SUBROUTINE PARAM.                               *
C*                                                                      *
C*       STOP 10: IW(J) .LE. ZERO, FREQUENCIES MUST BE  .GE. 1          *
C*                                                                      *
C*       STOP "R3": NPARA .LT. 1  NUMBER OF PARAMETERS MUST .GE. 1      *
C*                                                                      *
C*       STOP "F1": NPARA .GT. 50 , USING FOURIER METHOD, NPARA         *
C*                                                                      *
C*    STOP "ORDER" MEANS THAT THE ORDER OF ACCURACY VARIABLE            *
C*                    WAS LESS THAN 4.                                  *
C*                                                                      *
C*          MUST BE .LE. 50 ALSO                                        *
C*                                                                      *
C*    STOP "GETER": ERROR IN A FREE FORMAT READ, EITHER AN EOF          *
C*           OR AN ILLEGAL CHARACTER.                                   *
C*                                                                      *
C*                                                                      *
C*                                                                      *
C*    UPON SUCCESFUL COMPLETION OF SENANAL  TAPE3  HAS THE              *
C*    FOLLOWING FORMAT.                                                 *
C*                                                                      *
C*       1)  TITLE -- ( 8A10 FORMAT)                                    *
C*                                                                      *
C*       2)  METHOD,NPARA,TNPTS,NSIMUL,NFUNC                            *
C*           ( A10, 4I6 FORMAT; ' METHOD =1OHWALSH     , 1OHFOURIER     *
C*                                                                      *
C*       3)  FREQUENCY SET ( 15H  FORMAT, A LABEL FOR THE  FILE )       *
C*       AND IACCUR IN I3-FORMAT                                        *
C*                                                                      *
C*       4)  IW(1),IW(2),...,IW(NPARA)   ( 16I6 FORMAT )                *
C*                                                                      *
C*       5)  TIME POINTS   ( 15H FORMAT, A LABEL FOR THE FILE )         *
C*                                                                      *
C*       6)  TIME(1),TIME(2),...,TIME(TNPTS)    ( 7E12.6 FORMAT )       *
C*                                                                      *
C*       7)  NLABEL(1),...,NLABEL(NFUNC)   (8A10 FORMAT )               *
C*                                                                      *
```

Program SENANAL, CONT'D.

```
C*       8)  YIN(1),YIN(2),...,YIN(NFUNC)     UNFORMATTED WRITE       *
C*                                                                     *
C*       THERE ARE  TNPTS*NSIMUL  RECORDS OF TYPE 8, ONE FOR EACH      *
C*       TIME POINT IN A SIMULATION MULTIPLIED BY THE NUMBER OF        *
C*       SIMULATIONS.                                                  *
C*                                                                     *
C*            THESE SIMULATION VECTORS ARE IN AN UNSUITABLE FORM       *
C*       FOR SENSITIVITY ANALYSIS SINCE TO DO S. A. WE NEED            *
C*       ALL THE DIFFERENT SIMULATIONS OBJECT FUNCTIONS' VALUES        *
C*       AT THE SAME TIME POINT.                                       *
C*                                                                     *
C*           THE SUITE OF CODES RUN BY PROGRAM TRANS WILL REFORMAT     *
C*       TAPE3 AND WILL TRANSFORM THE SIMULATION CURVES INTO           *
C*       SEQUENCY VECTORS( WALSH OR FOURIER ) FOR WHICH PARTIAL        *
C*       VARIANCES WILL BE COMPUTED.                                   *
C*                                                                     *
C*********************************************************************
C*
      COMMON /PARA/ P(50)
C*
      REAL PMAX(50),PMIN(50),PAVE(50)
      REAL TIME(150),PHI(50),PLO(50)
      REAL YIN(40),BEGIN(40)
C*
      INTEGER IOMEGA,IMETH,NFUNC,ITRANS,NPARA,NSIMUL,TNPTS,LABEL(2)
      INTEGER TITLE(8),TRANS(5,2),IW(50)
      INTEGER NLABEL(40)
C*
      DATA PAVE/50*0./, PMIN/50*1.0E+99/, PMAX/50*0./
      DATA LABEL/ 10H FOURIER  ,10H WALSH    /
      DATA TRANS/10HLOGUNIFORM,10H UNIFORM  ,10H SINE TEST,
     +10HLOG(P)BELL,10HBELL-SHAPE,10HARITHMETIC,10HMULTPLIER ,
     +3*(10H          )/
C*
C*    SUBROUTINE TIMES IS A TOTALLY UNNECESSARY BUT SOMEWHAT USEFUL
C*    TIMING ROUTINE
C*
      CALL TIMES(1,0)
      CALL READIN(IOMEGA,TIME,TSTART,IMETH,NFUNC,ITRANS,PHI,PLO,
     +NPARA,NSIMUL,TNPTS,TITLE,BEGIN,IW,NLABEL,IACCUR)
      CALL TIMES(1,0)
C*
C*    WRITE OUT INPUT
C*
      WRITE(2,5) TITLE
5     FORMAT(1H1,8A10)
      WRITE(2,6)
6     FORMAT(1H )
      WRITE(2,10) LABEL(IMETH)
```

Program SENANAL, CONT'D.

```
10    FORMAT(1H ,* THIS IS A SENANAL RUN USING *,A10,* ANALYSIS*)
      WRITE(2,11) IACCUR,(IW(J),J=1,NPARA)
11    FORMAT(1H ,* WITH THE*,I3,*TH ORDER  FREQUENCY VECTOR=*,13(2X,I5)
     +,/,30X,
     +14(2X,I5))
      WRITE(2,12) NSIMUL,TNPTS
12    FORMAT(1H ,* THERE ARE *,I6,* SIMULATIONS IN THIS RUN WITH*,
     +,I5,* TIME POINTS*)
      WRITE(2,27)
27    FORMAT(/,/,6X,* FUNCTION *,5X,* INITIAL VALUE *,/)
      DO 31 J=1,NFUNC
      WRITE(2,28) NLABEL(J),BEGIN(J)
28    FORMAT(6X,A10,5X,1PE14.6)
31    CONTINUE
      WRITE(2,6)
      WRITE(2,15) TRANS(ITRANS,IMETH)
15    FORMAT(1H ,* THE PARAMETERS WERE CALCULATED USING *,A10,
     +* TYPE TRANSFORMATION FUNCTIONS*)
      WRITE(2,17)
17    FORMAT(* *,/,*  PARAMETER*,2X,* PHI(J) *,7X,* PLO(J)*)
      DO 19 J=1,NPARA
         WRITE(2,18) J,PHI(J),PLO(J)
18       FORMAT(3X,I5,2X,2X,1PE13.6,2X,E13.6)
19    CONTINUE
      WRITE(2,21)
21    FORMAT(* *,/,*  TIME POINTS *)
      WRITE(2,22)(TIME(J),J=1,TNPTS)
22    FORMAT(* *,10(1X,E12.6))
C*
C*       CHECK FOR ACCEPTABLE INPUT PARAMETERS
C*
C*

      IF(IOMEGA .LT. -1 .OR. IOMEGA .GT. 1 ) STOP 1
      IF(IMETH .LT. 1  .OR. IMETH  .GT. 2 )  STOP 2
      IF( TNPTS .LT. 1  .OR.  TNPTS .GT. 150 ) STOP 3
      IF(ITRANS .GT. 5  .OR. ITRANS  .LT. 1 )  STOP 4
      IF(NFUNC .LT. 1  .OR.  NFUNC .GT. 40 ) STOP 5
      IF( NSIMUL .LT. 1)  STOP 6
      IF( IACCUR .LT. 4 )  STOP "ORDER"
      DO 1 J=1,NPARA
         IF( PHI(J) .LE. PLO(J) ) STOP 7
         IF( IW(J) .LE. 0 )   STOP 10
1     CONTINUE
      J=1
      IF( TIME(1) .LE. TSTART ) WRITE(2,2) J,J
      IF( TNPTS .EQ. 1) GO TO 4
      DO 3 J=2,TNPTS
         IF( TIME(J) .LE. TIME(J-1) )  WRITE(2,2) J,J
2        FORMAT(1H ,/,*  TIME(*,I4,*) .LE. TIME(*,I4,*- 1)*)
```

Program SENANAL, CONT'D.

```
3        CONTINUE
4     CONTINUE
C*
C*
C*    WRITE THE OUTPUT TAPE LABELS
C*
      WRITE(3,23) TITLE
23    FORMAT(8A10)
      WRITE(3,20) LABEL(IMETH),NPARA,TNPTS,NSIMUL,NFUNC
20    FORMAT(A10,4I6)
      WRITE(3,25) IACCUR
25    FORMAT(* FREQUENCY SET *,I3)
      WRITE(3,26)(IW(J),J=1,NPARA)
26    FORMAT(16I6)
      WRITE(3,30)
30    FORMAT(* TIME POINTS   *)
      WRITE(3,35)(TIME(J),J=1,TNPTS)
35    FORMAT(7E12.6)
      WRITE(3,23)(NLABEL(J),J=1,NFUNC)
      CALL TIMES(2,0)
C*
C*    LOOP OVER THE DIFFERENTS SIMULATIONS
C*
      DO 1000 ISIMUL=1,NSIMUL
C*
      IQ = ISIMUL
C*
C*    CALCULATE THE PARAMETER VECTOR FOR THIS SIMULATION
C*
      CALL PARAM(IMETH,IQ,ITRANS,PHI,PLO,NPARA,P,NSIMUL,IW)
C*
C*    CALCULATE THE PARAMETER STATISTICS
C*
      DO 100 J=1,NPARA
         PMAX(J) = AMAX1(PMAX(J),P(J))
         PMIN(J) = AMIN1(PMIN(J),P(J))
         PAVE(J) = (P(J) + FLOAT(ISIMUL - 1)*PAVE(J))/FLOAT(ISIMUL)
100      CONTINUE
      CALL TIMES(3,0)
C*
C*    INITIALIZE THE FUNCTION WITH ITS INITIAL VALUE.
C*    (NECESSARY IF THE FUNCTIONS ARE ODE'S, OTHERWISE ONE CAN SET
C*     BEGIN TO ZERO )
C*
      DO 98 J = 1, NFUNC
         YIN(J) = BEGIN(J)
98    CONTINUE
C*    SET INITIAL TIME FOR SIMULATION RUN
      TIN = TSTART
```

Program SENANAL, CONT'D.

```
C*
C*     CALCULATE THE OUTPUT FUNCTIONS FOR THE "TNPTS" POINTS
C*
       DO 200 KOUNT = 1,TNPTS
          TOUT = TIME(KOUNT)
          CALL MODEL(TIN, TOUT, YIN, NFUNC, TSTART )
C*
C*        WRITE OUT THE SOLUTION AT TOUT
C*
          WRITE(3,10000) (YIN(J),J=1,NFUNC)
10000  FORMAT(4020)
C*
          CALL TIMES(4,0)
C*
          TIN = TOUT
200       CONTINUE
C*
C*
1000   CONTINUE
C*
C*     SIMULATIONS ARE OVER WITH
C*     WRITE OUT THE PARAMETER STATS
C*
       WRITE(2,6)
       WRITE(2,1500)
1500   FORMAT(1H ,1X,*    PARAMETER*,3X,* AVERAGE VALUE *,2X,
      +* MAXIMUM VALUE *,2X,* MINIMUM VALUE *)
       DO 1620 J=1,NPARA
          WRITE(2,1610) J,PAVE(J),PMAX(J),PMIN(J)
1610   FORMAT(1H ,5X,I5,5X,2X,3(1PE13.6,4X))
1620   CONTINUE
C*     PRINT TIMING DATA
       CALL TIMES(1,1)
       END
```

Program SENANAL, CONT'D.
SUBROUTINE READIN

```
      SUBROUTINE READIN(IOMEGA,TIME,TSTART,IMETH,NFUNC,ITRANS,PHI,
     +PLO,NPARA,NSIMUL,TNPTS,TITLE,BEGIN,IW,NLABEL,IACCUR)
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*    SUBROUTINE READIN READS THE INPUT FOR THE PROGRAM SENANAL.  ALL *
C*    VARIABLES ARE DEFINED IN THE SENANAL COMMENT CARDS.             *
C*                                                                    *
C*    THIS IS INSTALLATION DEPENDENT SECTION OF THE METHOD AS IT USES *
C*    FREE FORMAT INPUT. (VARIABLES SEPARATED BY A COMMA )            *
C*                                                                    *
C*    FORMAT OF INPUT CARDS                                           *
C*                                                                    *
C*       SET 1 = TITLE  (ONE CARD )                                   *
C*                                                                    *
C*       SET 2 =  IOMEGA,IMETH,NPARA  (INTEGERS)                      *
C*                                                                    *
C*       SET 3 IFF IOMEGA = 1                                         *
C*       SET 3A = IW(1),IW(2),IW(3),...,IW(NPARA),IACCUR    (INTEGERS)*
C*                                                                    *
C*       SET 4 = NSIMUL,TNPTS,ITRANS,NFUNC,TSTART (TSTART IS A REAL,  *
C*                                          THE OTHERS ARE INTEGERS*
C*                                                                    *
C*       SET 5 = TIME(1),TIME(2),...,TIME(TNPTS)   ( REALS )          *
C*                                                                    *
C*       SET 6 = PHI(1),PHI(2),...,PHI(NPARA)      (REALS)            *
C*                                                                    *
C*       SET 7 = PLO(1),PLO(2),...,PLO(NPARA)  (REALS)                *
C*                                                                    *
C*       SET 8 = BEGIN(1),BEGIN(2),...,BEGIN(NFUNC)   (REALS)         *
C*                                                                    *
C*       SET 9 = NLABEL(1),...,NLABEL(NFUNC)                          *
C*                TO BE READ IN ONE (1) VALUE TO A CARD  (A10 FORMAT) *
C*                                                                    *
C*                                                                    *
C*  VARIABLES                                                         *
C*                                                                    *
C*    BEGIN(NFUNC) = THE INITIAL CONDITIONS, OR EQUIVALENTLY          *
C*    (REAL)         THE VALUES OF THE OBJECT FUNCTIONS AT TSTART.    *
C*                                                                    *
C*                                                                    *
C*    IACCUR = ORDER OF ACCURACY OF THE FREQUENCY SET                 *
C*    (INTEGER)                                                       *
C*                                                                    *
C*                                                                    *
C*    IOMEGA =   0 IF FOURIER 4TH ORDER SET IS TO BE USED             *
C*    (INTEGER)  1 IF SPECIAL FREQUENCY SET IS TO BE USED             *
C*              -1 IF STANDARD WALSH FREQUENCY SET IS TO BE USED      *
```

Program SENANAL, CONT'D.
SUBROUTINE READIN

```
C*                                                                      *
C*    IMETH = METHOD FLAG FOR SENSITIVITY ANALYSIS; = 1 FOR FOURIER     *
C*    (INTEGER)         ANALYSIS, = 2 FOR WALSH  ANALYSIS.            *
C*                                                                      *
C*    ITRANS = FLAG FOR TYPE OF TRANSFORMATION FUNCTION                 *
C*    (INTEGER)      SEE SUBROUTINE PARAM FOR DETAILS.                *
C*                                                                      *
C*    IW(NPARA) = AN ARRAY CONTAINING THE FREQUENCY SET TO BE USED IN   *
C*    (INTEGER)      THE S. A. RUN.                                    *
C*                  IF 'IOMEGA' .EQ. 1, THIS ARRAY MUST BE READ IN FROM *
C*                  CARDS.  OTHERWISE THE FREQUENCY SETS ARE CREATED   *
C*                  INTERNALLY.                                       *
C*                                                                      *
C*    NFUNC = NUMBER OF OBJECT FUNCTIONS WHICH WILL BE SAVED            *
C*    (INTEGER)    AT EACH TIME POINT.                                  *
C*                                                                      *
C*    NLABEL(NFUNC) = THE NAMES (LABELS) OF THE OBJECT FUNCTIONS        *
C*                    NLABEL(1) SHOULD BE THE NAME OF THE FIRST         *
C*                    OBJECT FUNCTION, ETC.                             *
C*                                                                      *
C*    NPARA = NUMBER OF PARAMETERS TO VARY                              *
C*    (INTEGER)                                                         *
C*                                                                      *
C*    NSIMUL = NUMBER OF SIMULATIONS                                    *
C*    (INTEGER)                                                         *
C*                                                                      *
C*    PHI(NPARA) = MAXIMUM VALUES OF THE PARAMETERS( OR ONE SIGMA MAX)* 
C*    (REAL)                                                            *
C*                                                                      *
C*    PLO(NPARA) = MINIMUM VALUES OF THE PARAMETERS( OR ONE SIGMA MIN)*
C*    (REAL)                                                            *
C*                                                                      *
C*    TIME(TNPTS) = ARRAY CONTAINS THE TIME POINTS AT WHICH THE         *
C*    (REAL)         OUTPUT FUNCTIONS ARE TO BE SAVED AND THE           *
C*                   SENSITIVITY ANALYZED.                              *
C*                                                                      *
C*    TSTART = INITIAL TIME POINT, SO THERE ARE NO S. A. VALUES SAVED   *
C*    (REAL)   AT THIS POINT                                            *
C*                                                                      *
C*    TITLE(8) = A ONE CARD TITLE FOR S. A. RUN                         *
C*    (INTEGER)    (WRITTEN IN 8A10 FORMAT )                            *
C*                                                                      *
C*    TNPTS = NUMBER OF TIME POINTS                                     *
C*    (INTEGER)                                                         *
C*                                                                      *
C**********************************************************************
C*
      REAL TIME(150),BEGIN(40)
```

Program SENANAL, CONT'D.
SUBROUTINE READIN

```
C*
      INTEGER TITLE(8),IOMEGA,IMETH,NFUNC,ITRANS,TNPTS,NPARA,IW(NPARA)
      INTEGER NLABEL(40)
C*
C*
      COMMON /GETERR/ IFLAG,NUMVAR,RABC(40)
      EQUIVALENCE (RABC(1),IRABC(1))
      INTEGER IRABC(40)
C*
C*    IO IS THE INPUT UNIT  ( TAPE1 )
      DATA IO/1/
C*
C*
C*    READ IN SET 1
C*
      READ(IO,10) (TITLE(J),J=1,8)
10    FORMAT(8A10)
C*
C*    READ IN SET 2
      IACCUR = 4
      ICARD = 2
      CALL GETNUM(IO)
      IF( IFLAG .GE. 0 ) CALL GETERR(IFLAG, ICARD, NUMVAR)
      IOMEGA = IRABC(1)
      IMETH = IRABC(2)
      NPARA = IRABC(3)
      IF(IOMEGA .LT. -1 .OR. IOMEGA .GT. 1 ) STOP "R1"
      IF(IMETH .LT. 1  .OR. IMETH  .GT. 2 )  STOP "R2"
      IF( NPARA .LT. 1 ) STOP "R3"
C*    OBTAIN FREQUENCY SET
      IF(IOMEGA .EQ. 0) CALL FOURST(IW,NPARA)
      IF(IOMEGA .EQ. -1) CALL WALSET(IW,NPARA)
      IF(IOMEGA .NE. 1) GO TO 100
C*    READ IN SPECIAL FREQUENCY SET       (SET 3 )
C*    READ IN THE ACCURACY OF THE SET ALSO
      NP1 = NPARA + 1
      CALL IREAD(IW,NP1,IO, ICARD)
      IACCUR = IW(NP1)
100   CONTINUE
C*    READ IN SET 4
      ICARD = ICARD + 1
      CALL GETNUM(IO)
      IF( IFLAG .GE. 0 ) CALL GETERR(IFLAG, ICARD, NUMVAR)
      NSIMUL = IRABC(1)
      TNPTS = IRABC(2)
      ITRANS = IRABC(3)
      NFUNC = IRABC(4)
      TSTART = RABC(5)
```

Program SENANAL, CONT'D.
SUBROUTINE READIN

```
      IF(NFUNC .LT. 1  .OR.  NFUNC .GT. 40 ) STOP "R4"
      IF( NSIMUL .LT. 1)  STOP "R5"
C*
C*    READ IN THE TIME POINTS    ( SET 5)
C*
      CALL RREAD(TIME,TNPTS,IO, ICARD)
C*
C*    READ IN PHI(J)      ( SET 6 )
C*
      CALL RREAD(PHI,NPARA,IO, ICARD)
C*
C*    READ IN PLO(J)      ( SET 7 )
C*
      CALL RREAD(PLO,NPARA,IO, ICARD)
C*
C*    READ IN INITIAL VALUES      ( SET 8 )
C*
      CALL RREAD(BEGIN,NFUNC,IO, ICARD)
C*
      DO 175 J=1,NFUNC
      READ(IO,150) NLABEL(J)
      ICARD = ICARD + 1
150   FORMAT(A10)
175   CONTINUE
      WRITE(2,101) ICARD
101   FORMAT(1H ,/,I6,*  DATA CARDS READ IN *)
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE FOURST

```
      SUBROUTINE FOURST(IW,NPARA)
C*
C*************************************************************************
C*                                                                       *
C*    SUBROUTINE FOURST CALCULATE THE STANDARD 4TH ORDER CORRECT         *
C*    FOURIER FREQUENCY SET FOR "NPARA" PARAMETERS.                      *
C*                                                                       *
C*    REFERENCE:  CUKIER,SHAILBY,SHULER. JOURNAL OF CHEMICAL PHYSICS,    *
C*                  VOL 63, NO. 3,  (1975) PP 1140-1149.                 *
C*                                                                       *
C*************************************************************************
C*
      INTEGER IW(NPARA),IOMEGA(50),IDN(49)
C*
      DATA IOMEGA/0,0,1,5,11,1,17,23,19,25,41,31,23,87,67,
     + 73,85,143,149,99 ,119,237,267,283,151,385,
     + 157,215,449,163,337,253,375,441,673,773,875,873,587,849,
     + 623,637,891,943,1171,1225,1335,1725,1663,2019/
      DATA IDN/ 4,8,6,10,20,22,32,40,38,26,56,62,46,76,96,
     + 60,86,126,134,112,92,128,154,196,34,416,106,
     + 208,328,198,382,88,348,186,140,170,284,568,302,438,
     + 410,248,448,388,596,216,100,488,166/
C*
      IF(NPARA .GT. 50 ) STOP "F1"
      IW(1) = IOMEGA(NPARA)
      DO 100 J=2,NPARA
      IW(J) = IW(J-1) + IDN(NPARA + 1 - J)
100   CONTINUE
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE WALSET

```
      SUBROUTINE WALSET(IW,NPARA)
C*********************************************************************
C*                                                                   *
C*    SUBROUTINE WALSET CALCULATES THE FREQUENCY SET FOR EXACT WALSH  *
C*    ANALYSIS FOR 'NPARA' PARAMETERS.                               *
C*                                                                   *
C*    REFERENCE: T.H. PIERCE, PHD THESIS (1980)                      *
C*                                                                   *
C*********************************************************************
C*
      INTEGER IW(NPARA)
C*
      DO 100 J=1,NPARA
      IW(J) = 2**(J-1)
100   CONTINUE
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE IREAD

```
      SUBROUTINE IREAD(IRRAY,LAST,IO, ICARD)
C*
C*************************************************************************
C*                                                                       *
C*    SUBROUTINE IREAD READS IN A VARIABLE LENGTH (LAST) INTEGER         *
C*    ARRAY USING FREE FORMAT INPUT.                                     *
C*                                                                       *
C*************************************************************************
      INTEGER IRRAY(LAST)
C*
C*
      COMMON /GETERR/ IFLAG,NUMVAR,RABC(40)
      EQUIVALENCE (RABC(1),IRABC(1))
      INTEGER IRABC(40)
C*
      KOUNT = 1
10    CONTINUE
      ICARD = ICARD + 1
      CALL GETNUM(IO)
      IF( IFLAG .GE. 0 ) CALL GETERR(IFLAG, ICARD, NUMVAR)
C*    IF THE CARD READ WAS BLANK  NUMVAR = 0.
      IF( NUMVAR .LT. 1 ) GO TO 10
      DO 20 J=1,NUMVAR
      IRRAY(KOUNT) = IRABC(J)
      KOUNT = KOUNT + 1
      IF(KOUNT .GT. LAST) GO TO 25
20    CONTINUE
C*
C*    RETURN FOR ANOTHER CARD FULL OF VARIABLES
      GO TO 10
25    CONTINUE
C*    ALL DONE SO STOP
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE RREAD

```
      SUBROUTINE RREAD(ARRAY,LAST,IO, ICARD)
C*
C*************************************************************************
C*                                                                       *
C*    SUBROUTINE RREAD READS IN A VARIABLE LENGTH (LAST) REAL            *
C*    ARRAY USING THE FREE FORMAT ROUTINE GETNUM.                        *
C*                                                                       *
C*************************************************************************
      REAL ARRAY(LAST)
C*
C*
      COMMON /GETERR/ IFLAG,NUMVAR,RABC(40)
      EQUIVALENCE (RABC(1),IRABC(1))
      INTEGER IRABC(40)
C*
      KOUNT = 1
10    CONTINUE
      ICARD = ICARD + 1
      CALL GETNUM(IO)
      IF( IFLAG .GE. 0 ) CALL GETERR(IFLAG, ICARD, NUMVAR)
C*    IF THE CARD READ WAS BLANK  NUMVAR = 0.
      IF( NUMVAR .LT. 1) GO TO 10
      DO 20 J=1,NUMVAR
      ARRAY(KOUNT) = RABC(J)
      KOUNT = KOUNT + 1
      IF(KOUNT .GT. LAST) GO TO 25
20    CONTINUE
C*
C*    RETURN FOR ANOTHER CARD FULL OF VARIABLES
      GO TO 10
25    CONTINUE
C*    ALL DONE SO STOP
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE PARAM

```
      SUBROUTINE PARAM(METH,IQ,TRANS,PHI,PLO,NPARA,P,NSIMUL,IW)
C*************************************************************************
C*                                                                       *
C*    SUBROUTINE PARAM COMPUTES THE IQ"TH PARAMETER                      *
C*    VECTOR.  THIS IS DONE USING A PRESELECTED ( TRANS,METH )           *
C*    TRANSFORMATION FUNCTION.                                           *
C*                                                                       *
C*  METH = 1 ---- FOURIER METHOD                                         *
C*                                                                       *
C*      TRANS                                                            *
C*        1 => USE FOURIER LOG-UNIFORM TRANSFORMATION                    *
C*             PHI = NOMINAL*EXP(DELTA)                                  *
C*             PLO = NOMINAL*EXP( -DELTA )                               *
C*             WITH LN(P) SPREAD UNIFORMLY OVER @LN(PHI) , LN(PLO)@      *
C*                                                                       *
C*        2 => USE FOURIER UNIFORM TRANSFORMATION                        *
C*             PHI = NOMINAL + DELTA                                     *
C*             PLO = NOMINAL - DELTA                                     *
C*             P IS UNIFORMLY SPREAD OVER @ PLO , PHI @                  *
C*                                                                       *
C*        3 => USE THE FOURIER TEST FUNCTION                             *
C*             PHI = NOMINAL*( 1 + DELTA )                               *
C*             PLO = NOMINAL*( 1 - DELTA)                                *
C*             P(SQ) = NOMINAL*(1. + DELTA*SIN(W*SQ) )                   *
C*                                                                       *
C*        4 => USE THE FOURIER COSH DISTRIBUTION FUNCTION                *
C*             IN LOG(P)-SPACE                                           *
C*             HERE LN(PHI)-LN(PLO) = 4.0/A                              *
C*             WHERE 82.87 OF THE SAMPLES ARE BETWEEN PHI AND PLO        *
C*                                                                       *
C*        5 => USE THE FOURIER COSH DISTRIBUTION FUNCTION                *
C*             IN P-SPACE                                                *
C*             HERE (PHI)-(PLO) = 4.0/A                                  *
C*             WHERE 82.87 OF THE SAMPLES ARE BETWEEN PHI AND PLO        *
C*                                                                       *
C*                                                                       *
C*  METH = 2 ---- WALSH METHOD                                           *
C*                                                                       *
C*      TRANS                                                            *
C*                                                                       *
C*        1 => USE ARITHMETIC WALSH TRANSFORMATION                       *
C*             PHI = NOMINAL + DELTA                                     *
C*             PLO = NOMINAL - DELTA                                     *
C*             P IS EITHER PHI OR PLO                                    *
C*                                                                       *
C*        2 => USE MULTIPLICATIVE WALSH TRANSFORM                        *
C*             PHI = NOMINAL*10**( DELTA )                               *
C*             PLO = NOMINAL*10**(-DELTA )                               *
```

Program SENANAL, CONT'D.
SUBROUTINE PARAM

```
C*              P IS EITHER PHI OR PLO                                  *
C*                                                                      *
C*                                                                      *
C************************************************************************
C*
      REAL NOMINAL,DELTA,PHI(NPARA),PLO(NPARA),P(NPARA)
C*
      INTEGER TRANS,IQ,METH,NSIMUL,IW(NPARA)
      INTEGER WALH
C*
      EXTERNAL WALH
C*
      IF ( METH .EQ. 2 )  GO TO 1000
      TWODPI = 2.0/ACOS(-1.0)
      R = FLOAT(NSIMUL)
      SQ = FLOAT(2*IQ -NSIMUL- 1)/(R*TWODPI)
C*
      GO TO (100,200,300,400,500)TRANS
C*
100   CONTINUE
C*    FOURIER METHOD WITH LOG-UNIFORM TRANSFORMATION FUNCTION
C*
      DO 110 J=1,NPARA
      DELTA = 0.5*ALOG(PHI(J)/PLO(J))
      NOMINAL = SQRT(PHI(J)*PLO(J))
      P(J) = NOMINAL*EXP(DELTA*TWODPI*ASIN(SIN(SQ*FLOAT(IW(J)))))
110   CONTINUE
      RETURN
C*
200   CONTINUE
C*    FOURIER METHOD USING UNIFORM TRANSFORMATION FUNCTION
      DO 210 J=1,NPARA
      NOMINAL = 0.5*(PHI(J)+PLO(J))
      DELTA = ( PHI(J) - PLO(J) )*0.5
      P(J) = NOMINAL + DELTA*TWODPI*ASIN(SIN(SQ*FLOAT(IW(J))))
210   CONTINUE
      RETURN
C*
300   CONTINUE
C*    FOURIER METHOD WITH TEST TRANSFORMATION FUNCTION
      DO 310 J=1,NPARA
      NOMINAL = 0.5*(PHI(J)+PLO(J))
      DELTA = (PHI(J)-PLO(J))/(PHI(J)+PLO(J))
      P(J) = NOMINAL*(1.0 + DELTA*SIN(FLOAT(IW(J))*SQ))
310   CONTINUE
      RETURN
C*
400   CONTINUE
```

Program SENANAL, CONT'D.
SUBROUTINE PARAM

```
C*     COSH-DISTRIBUTION FUNCTION
C*      BELL-SHAPE IN LOG(K)-SPACE
C*
       DO 410 J=1,NPARA
       AJ = 4.0/(ALOG(PHI(J)) - ALOG(PLO(J)))
       THETA = SQ * FLOAT(IW(J))
       UJ = (1.0/(2.*AJ))*ALOG((1. + SIN(THETA))/(1. - SIN(THETA)))
       NOMINAL = SQRT(PHI(J)/PLO(J))
       P(J) = NOMINAL * EXP(UJ)
410    CONTINUE
       RETURN
C*
500    CONTINUE
C*
C*     COSH-DISTRIBUTION FUNCTION
C*      BELL-SHAPED IN K-SPACE
C*
       DO 510 J=1,NPARA
       AJ = 4.0/(PHI(J) - PLO(J))
       THETA = SQ*FLOAT(IW(J))
       UJ = (1.0/AJ)*ALOG((1. + SIN(THETA))/(1. - SIN(THETA)))
       NOMINAL = (PHI(J)+PLO(J))*0.5
       P(J) = NOMINAL + UJ
510    CONTINUE
       RETURN
C*
C*
1000   CONTINUE
C*     ENTRY INTO HERE FOR WALSH ANALYSIS
C*
       ISQ = IQ - 1
       GO TO (1100,1200) TRANS
C*
1100   CONTINUE
C*     ARITHMETIC WALSH TRANSFORMATION FUNCTION
       DO 1110 J=1,NPARA
       NOMINAL = 0.5*(PHI(J)+PLO(J))
       DELTA = ( PHI(J) - PLO(J) )*0.5
       P(J) = NOMINAL + DELTA*FLOAT(WALH(IW(J),ISQ))
1110   CONTINUE
       RETURN
C*
1200   CONTINUE
C*     MULTIPLICATIVE WALSH TRANSFORMATION FUNCTION
       DO 1210 J=1,NPARA
       DELTA = 0.5*ALOG10(PHI(J)/PLO(J))
       NOMINAL = SQRT(PHI(J)*PLO(J))
       P(J) = NOMINAL*10.0**(DELTA*FLOAT(WALH(IW(J),ISQ)))
```

Program SENANAL, CONT'D.
SUBROUTINE PARAM

```
1210  CONTINUE
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE GETNUM

```
C****
      SUBROUTINE GETNUM(LINPUT)
      COMMON /GETERR/ ICRK,JREAD,RABC(40)
      DIMENSION IRABC(40)
      EQUIVALENCE (RABC(1),IRABC(1))
      COMMON /EL/ L(80)
      LOGICAL INTR
C*************************************************************************
C****                                                                   **
C****         *** FREE FORM VARIABLE INPUT ROUTINE. ***                 **
C****         ROUTINE ACCEPTS A,F,E AND I FORMAT INPUT.                 **
C****         ALL BLANKS EXCEPT IN HOLLERITH STRINGS ARE IGNORED.       **
C****         THE ONLY LEGAL DELIMITER IS COMMA (,), ANY                **
C****         OTHER RESULTS IN ERROR TERMINATION.                       **
C****         KL IS THE MAXIMUM COLUMN WIDTH COUNTER.                   **
C****         ONLY 80 COLUMNS ARE READ, SO ONLY 40 VARIABLES            **
C****         CAN BE RETURNED.  TO ENLARGE THIS, CHANGE THE DATA        **
C****         AND COMMON STATEMENTS TO REFLECT THE SIZE YOU WISH.       **
C****                                                                   **
C****         --- INPUT ---                                             **
C****         LINPUT -- THE TAPE UNIT BEING READ FROM                   **
C****                                                                   **
C****         --- OUTPUT ---                                            **
C****         JREAD IS THE NUMBER OF VARIABLES RETURNED IN              **
C****               COMMON /GETERR/                                     **
C****         RABC(=IRABC) CONTAINS THE READ VARIABLES.                 **
C****         COMMON /EL/ CONTAINS THE LINE AS READ IN 80R1             **
C****              ERROR CODES: (STANDARD IF(UNIT) VALUES)              **
C****         ICRK=1    ILLEGAL CHARACTER (MSG PRINTED)                 **
C****                   VARIABLES TO POINT OF ERROR RETURNED            **
C****         ICRK=0    EOF ON READ, JREAD=0                            **
C****         ICRK=-1   NORMAL TERMINATION                              **
C****                                                                   **
C****         --- INTERNAL VARIABLES ---                                **
C****         S IS SIGN OF VARIABLE, IFA THE SIGN OF THE EXPONENT       **
C****         NUM IS THE MANTISSA, IE THE EXPONENT                      **
C****         IP IS THE NUMBER OF DECIMAL PLACES INPUT.                 **
C****         I IS THE CHARACTER COUNTER (1-80)                         **
C****         INTR -- REAL VARIABLE(FALSE)/INTEGER(TRUE) FLAG           **
C****                                                                   **
C****         HOLLERITH STRINGS OF 10 OR MORE CHARACTERS MAY            **
C****         BE INPUT WITHOUT COMMAS EVERY 10 CHARACTERS AND WILL      **
C****         BE INSERTED 10 CHARACTERS PER WORD WITH BLANK FILL        **
C****         (STANDARD A FORMAT).  ANY COMMAS FOUND IN THE HOLLER-     **
C****         ITH STRING END THE STRING AT THAT POINT.  THE FIRST       **
C****         CHARACTER OF THE STRING MAY NOT BE A COMMA(,)             **
C****         PERIOD(.) PLUS(+) OR MINUS(-) OR DIGIT(0-9) OR BLANK.     **
C****                                                                   **
```

Program SENANAL, CONT'D.
SUBROUTINE GETNUM

```
C****         HOLLERITH STRINGS WHICH ARE THE LAST INPUT ON A          **
C****         LINE, BUT NOT ENDING WITH A COMMA, WILL BE ASSUMED       **
C****         TO CONTINUE RIGHT OUT TO COLUMN 80(OR KL) AND BLANK      **
C****         FILLED VARIABLES WILL THUS BE RETURNED.                  **
C****                                                                  **
C**************************************************************************
      INTEGER PERIOD,COMMA,BLANK,ZERO,PLUS,EE,DD
      DATA PERIOD,COMMA,BLANK,ZERO /1R., 1R,, 1R , 1RO/
      DATA NINE,PLUS,MINUS,EE,DD /1R9, 1R+, 1R-, 1RE, 1RD/
      DATA KL /80/
C****
C****         READ THE INPUT LINE FROM UNIT LINPUT
      READ (LINPUT,1000) (L(I),I=1,80)
      JREAD=0
      IF (EOF(LINPUT) .NE. 0.) GO TO 998
      ICRK=-1
      I=1
C****         PREPARE FOR A NEW VARIABLE
10    NUM=IE=IFA=IP=0
      INTR=.TRUE.
      S=1.0
C****         DECODE THE FIRST CHARACTER IN THE VARIABLE
      IF (I .GT. KL) RETURN
      IF (L(I) .LE. NINE  .AND.  L(I) .GE. ZERO) GO TO 35
      IF (L(I) .EQ. PLUS) GO TO 30
      IF (L(I) .NE. MINUS) GO TO 25
      S=-1.0
      GO TO 30
25    IF (L(I) .EQ. PERIOD) GO TO 39
      IF (L(I) .EQ. COMMA) GO TO 60
      IF (L(I) .EQ. BLANK) GO TO 291
C****         HOLLERITH VARIABLE (A FORMAT)
251   DO 26 LL=1,10
         JL=LL
         IF (I .GT. KL) GO TO 27
         IF (L(I) .EQ. COMMA) GO TO 27
         ISH=60-6*LL
         IE=SHIFT(L(I),ISH)
         NUM=OR(NUM,IE)
         I=I+1
26       CONTINUE
C****         FULL WORD(10 CHARS) FILLED IF YOU FALL
C****         THRU HERE.
C****         STORE THE HOLLERITH VARIABLE
      JREAD=JREAD+1
      IRABC(JREAD)=NUM
C****         SKIP THE TRAILING COMMA, OTHERWISE ASSUME
C****         THE HOLLERITH STRING CONTINUES
```

Program SENANAL, CONT'D.
SUBROUTINE GETNUM

```
      IF (I .GT. KL) RETURN
      IF (L(I) .EQ. COMMA) GO TO 291
      NUM=0
      GO TO 251
C**** BLANK FILL WORD
27    DO 29 LL=JL,10
         ISH=60-6*LL
         IE=SHIFT(BLANK,ISH)
29       NUM=OR(NUM,IE)
C**** STORE THE PARTIAL HOLLERITH VARIABLE
      JREAD=JREAD+1
      IRABC(JREAD) = NUM
291   I=I+1
      GO TO 10
C**** INTEGER PORTION OF VARIABLE
30    I=I+1
35    IF (L(I) .EQ. PERIOD) GO TO 39
      IF (I .GT. KL) GO TO 60
      IF (L(I) .EQ. BLANK) GO TO 30
      IF (L(I) .EQ. EE) GO TO 50
      IF (L(I) .EQ. COMMA) GO TO 60
      IF (L(I) .LT. ZERO  .OR.  L(I) .GT. NINE) GO TO 999
      NUM = NUM*10 + (L(I)-27)
      GO TO 30
C**** EVALUATE DECIMAL PORTION
39    INTR=.FALSE.
40    I=I+1
      IF (I .GT. KL) GO TO 60
      IF (L(I) .EQ. BLANK) GO TO 40
      IF (L(I) .EQ. EE) GO TO 50
      IF (L(I) .EQ. COMMA) GO TO 60
      IF (L(I) .LT. ZERO  .OR. L(I) .GT. NINE) GO TO 999
C**** INCREMENT THE DECIMAL COUNT
      IP=IP+1
      NUM = NUM*10 + (L(I)-27)
      GO TO 40
C**** EVALUATE EXPONENT
50    IFA=1
      INTR=.FALSE.
      I=I+1
      IF (L(I) .EQ. PLUS) GO TO 51
      IF (L(I) .NE. MINUS) GO TO 52
      IFA=-1
51    I=I+1
      IF (I .GT. KL) GO TO 60
52    IF (L(I) .EQ. COMMA) GO TO 60
      IF (L(I) .EQ. BLANK) GO TO 51
      IF (L(I) .LT. ZERO  .OR.  L(I) .GT. NINE) GO TO 999
```

Program SENANAL, CONT'D.
SUBROUTINE GETNUM

```
      IE = IE*10 + (L(I)-27)
      GO TO 51
C****    STORE THE FINISHED VARIABLE (EXCEPT HOLLERITH)
60    CONTINUE
C****          CHECK ILLEGAL EXPONENT RANGE
C****          THE CHECK IS NOT PERFECT:  DIGITS BEFORE
C****          THE MANTISSA PERIOD ARE NOT CONSIDERED.
      IEX=IE*IFA-IP
      IF (IEX .GT. 322) GO TO 995
      IF (IEX .LT. -294) GO TO 995
      I=I+1
      JREAD=JREAD+1
      IF (INTR) GO TO 62
      RABC(JREAD) = S*(FLOAT(NUM)*10.**IEX)
      GO TO 64
62    IRABC(JREAD) = S*NUM
64    IF (I .GT. KL) RETURN
      GO TO 10
C****    ERROR CONDITION CODE
995   I=I-1
999   CONTINUE
      ICRK=1
      JM=I-1
      IF (JM .LE. 0) JM=1
      PRINT 1010, L,(BLANK,LL=1,JM),PLUS
      RETURN
C****          EOF ENCOUNTERED, JREAD ALREADY ZEROED, SET
C****          ICRK AND RETURN.
998   ICRK=0
      RETURN
C*************************************************************************
C****          **FORMATS**
1000  FORMAT (80R1)
1010  FORMAT (*OILLEGAL CHARACTER FOUND AT PLUS(+) */1X,80R1/1X,80R1)
C****
      END
```

Program SENANAL, CONT'D.
FUNCTION WALH

```
      INTEGER FUNCTION WALH(N,IT)
C*****************************************************************************
C*                                                                           *
C*    SUBROUTINE WALH(N,IT) COMPUTES THE HADAMARD-ORDERED                    *
C*    WALSH FUNCTION OF SEQUENCY ' N ' AT TIME POINT ' IT '.                 *
C*                                                                           *
C*       WHERE  N IS OF THE RANGE ( 0,1,2,3,...(2**(LENGTH+1) - 1)  )        *
C*       AND IT IS OF THE RANGE ( 0,1,2,3,4,...,(2**(LENGTH+1) - 1) )        *
C*                                                                           *
C*****************************************************************************
      INTEGER TBIT(60),NBIT(60),M,I
      REAL OLDN,FRAC
      DATA LENGTH /15/
C* DECODE N  INTO ITS BINARY REPRESENTATION
      OLDN = FLOAT(N)
      DO 10 I=1,LENGTH
      M=OLDN/2.0
      FRAC = OLDN/2.0 - FLOAT(M)
      NBIT(I) = FRAC*2.0
      OLDN = FLOAT(M)
10    CONTINUE
C* DECODE IT  INTO ITS BINARY REPRESENTATION
      TOLD = IT
      DO 27 I=1,LENGTH
      M=TOLD/2.0
      FRAC = TOLD/2.0 - FLOAT(M)
      TBIT(I) = FRAC*2.0
      TOLD = FLOAT(M)
27    CONTINUE
C* WE NOW KNOW THE BINARY REP FOR T AND N
C*  CALCULATE THE EXPONENT
      NSUM = NBIT(1)*TBIT(1)
      DO 30 I=2,LENGTH
      NSUM = NSUM + NBIT(I)*TBIT(I)
30    CONTINUE
C     WRITE(2,2)(((NBIT(L),L=1,LENGTH),(TBIT(K),K=1,LENGTH)),NSUM)
C 2    FORMAT(* *,*NBIT=*,15I1,*  TBIT=*,15I1,*  NSUM =*,I4)
      WALH = (-1)**NSUM
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE TIMES

```
      SUBROUTINE TIMES(ISUB,ITYPE)
C*****************************************************************************
C*                                                                           *
C*    SUBROUTINE TIMES COMPUTES THE CPU TIME SPENT BETWEEN CALLS.            *
C*    THIS IS INSTALLATION DEPENDENT.                                        *
C*    ITS USE IS FOR DOCUMENTATION PURPOSES ONLY                             *
C*                                                                           *
C*       ISUB = THE PROCEDURE TO BE TIMED.                                   *
C*                                                                           *
C*       ITYPE = FLAG: .LT. 1 FOR TIMING                                     *
C*                     .GE. 1 FOR FINAL PRINT                                *
C*                                                                           *
C*****************************************************************************
C*                                                                           *
C*
      REAL TIMS(15),NEW,LAST
      INTEGER NAME(15)
C*
      DATA TIMS/15*0./
      DATA LAST /0./
      DATA NAME/10HREAD INPUT,10HSIMULATION,10HPARAM CALC,10HMODEL CALC
     +,10HWRITE OUTP/
      DATA LENGTH /5/
C*
      IF(ITYPE .GE. 1) GO TO 5
      NEW = SECOND(CPU)
C*     INITIALIZE THE VALUES IN FIRST ENTRY
      IF( LAST .EQ. 0. ) LAST = NEW
      TIMS(ISUB) = TIMS(ISUB) + NEW - LAST
      LAST = NEW
      RETURN
5     CONTINUE
      WRITE(2,100)
100   FORMAT(* *,/,/,/,5X,*  SECONDS*,4X,* PROCEDURE *,/)
      TSUM = 0.
      DO 10 J=1,LENGTH
      TSUM = TSUM + TIMS(J)
      WRITE(2,150) TIMS(J),NAME(J)
150   FORMAT(* *,5X,F7.3,5X,A10)
10    CONTINUE
      WRITE(2,160) TSUM
160   FORMAT(* *,/,/,5X,F7.3,5X,* TOTAL TIME *)
      RETURN
      END
```

Program SENANAL, CONT'D.
SUBROUTINE GETERR

```
      SUBROUTINE GETERR(IFLAG,ICARD,NUMVAR)
C*
C*************************************************************************
C*    SUBROUTINE GETERR IS AN ERROR MESSAGE SUBROUTINE WHICH             *
C*    TERMINATES SENANAL IF INPUT WAS NOT CORRECTLY                      *
C*    READ IN.                                                           *
C*                                                                       *
C*       IFLAG IS THE FLAG FROM SUBROUTINE GETNUM                        *
C*                                                                       *
C*       ICARD IS THE INPUT CARD NUMBER OF THE ERROR                     *
C*                                                                       *
C*       NUMVAR IS THE NUMBER OF VARIABLES ON CARD ICARD                 *
C*************************************************************************
      INTEGER IFLAG,ICARD,NUMVAR
      WRITE(2,11) IFLAG,ICARD,NUMVAR
11    FORMAT(1H ,5H*****,* ERROR IN GETNUM *,/,15X,* ICRK =*,I5,
     +/,15X,* ICARD =*,I5,/,15X,* NUMVAR =*,I5)
      STOP "GETER"
      END
```

Program TRANS

```
      PROGRAM TRANS(OUTPUT=65,TAPE2=OUTPUT,TAPE3=513,TAPE4=513,
     + TAPE5=513,TAPE6=513,TAPE7=513,TAPE8=65,TAPE9=65)
C*
C*************************************************************************
C*       PROGRAM TRANS IS THE FOLLOW-UP PROGRAM TO PROGRAM 'SENANAL'.  *
C*    IN THIS PROGRAM 'TAPE3', THE OUTPUT FILE FROM 'SENANAL' IS       *
C*    READ IN AND OPERATED ON.                                         *
C*                                                                     *
C*       FIRST THE SIMULATION RUNS ARE TRANSPOSED INTO SENSITIVITY-    *
C*    ANALYSIS POINTS. (INSTEAD OF ALL THE TIME POINTS OF ONE FUNCTION *
C*    A SIMULATION, WE HAVE ALL THE FUNCTIONS AT ONE TIME POINT,       *
C*    A SENSITIVITY-ANALYSIS POINT )  UPON SUCCESSFUL TRANSPOSITION    *
C*    WE ITERATE THRU THE TIME POINTS.  EACH S.A. POINT IS THEN        *
C*    TRANSFORMED INTO SEQUENCY SPACE ( FOURIER OR WALSH ).            *
C*    THIS TRANSFORMATION GIVES US THE EXPANSION COEFFICIENTS          *
C*    FROM WHICH WE COMPUTE THE PARTIAL VARIANCES OF THE OBJECT        *
C*    FUNCTIONS.                                                       *
C*                                                                     *
C*       AFTER THE PARTIAL VARIANCES ARE CALCULATED THEY ARE           *
C*    WRITTEN OUT ONTO TAPE9.  THE EXPANSION COEFFICIENTS ARE WRITTEN  *
C*    OUT ONTO TAPE8.  THE TRANSPOSED MATRIX IS AVAILABLE ON TAPE7,    *
C*    LOGICAL UNIT 'WRITEUP'.                                          *
C*                                                                     *
C*   VARIABLES                                                         *
C*                                                                     *
C*       F( ) = AN ARRAY WHICH HOLDS ONE OBJECT FUNCTION               *
C*              IE AT LEAST OF LENGTH  'NSIMUL'                        *
C*                                                                     *
C*                                                                     *
C*       A( ) = A REAL ARRAY WHICH WILL HOLD THE COSINE                *
C*              COEFFICIENTS IN THE FOURIER METHOD.  THEREFORE IT      *
C*              MUST BE OF LENGTH (NSIMUL + 1)                         *
C*                                                                     *
C*       B( ) = A REAL ARRAY WHICH WILL HOLD THE SINE COEFFICIENTS      *
C*              IN THE FOURIER METHOD.  IT ALSO MUST BE OF LENGTH      *
C*              ( NSIMUL + 1)                                          *
C*                                                                     *
C*       IWK( ) = THE WORKING STORAGE OF THIS PROGRAM.  THIS ARRAY IS   *
C*                USED AS STORAGE FOR DIFFERENT TEMPORARY VARIABLES.   *
C*                                                                     *
C*                MINIMALLY IT MUST BE DIMENSIONED FOR THE FFT ROUTINES*
C*                IF NSIMUL IS THE NUMBER OF SIMULATIONS IN THE FOURIER*
C*                METHOD THEN THRU SYMMETRY WE HAVE 2*NSIMUL POINTS    *
C*                TO FOURIER TRANSFORM.                                *
C*                   THE EQUATION FOR IWK DIMENSIONS IS:               *
```

Program TRANS CONT'D.

```
C*              MINIMUM LENGTH = 3*( F + N ) + 26                      *
C*                                                                     *
C*              WHERE F IS THE NUMBER OF THE PRIME FACTORS OF NSIMUL   *
C*              EXCLUDING THE TRIVIAL  FACTOR 1 .                      *
C*              N = 2*NSIMUL                                           *
C*                                                                     *
C*              IE IF NSIMUL = 21 ( HAS TO BE ODD )                    *
C*                THEN 21 = 3*7 => F = 2                               *
C*                MINIMUM LENGTH = 3*( 2 + 42 ) + 26 = 158             *
C*                                                                     *
C*                                                                     *
C*   ERROR CODES:                                                      *
C*                                                                     *
C*    STOP "WALPR" , STOPS EXECUTION IF TWO FREQUENCIES ARE EQUAL      *
C*       SEE SUBROUTINE WALPR.                                         *
C*                                                                     *
C*    STOP "MTH" , THE METHOD READ IN ON TAPE3 WAS SOMETHING OTHER     *
C*       THAN WALSH OR FOURIER.  SEE PROGRAM TRANS                     *
C*                                                                     *
C*    STOP 3 , ERROR IN MATRIX TRANSPOSITION.  SEE SUBROUTINE TRANP.   *
C*                                                                     *
C*                                                                     *
C*   OUTPUT FORMAT FOR  TAPE7 -PARTIAL VARIANCES-                      *
C*                                                                     *
C*    1) LABEL,TIME,AVE,STDDEV,RELDEV,LENGTH,NPARA                     *
C*       ( A8,4(2X,E15.7),2I6 -FORMAT )                                *
C*                                                                     *
C*       LABEL = NAME OF THE OBJECT FUNCTION                           *
C*                                                                     *
C*       TIME = TIME VALUE OF OBJECT FUNCTION                          *
C*                                                                     *
C*       AVE = AVERAGE VALUE OF OBJECT FUNCTION AT THIS TIME           *
C*                                                                     *
C*       STDDEV = SQUARE ROOT OF TOTAL VARIANCE OF OBJECT FUNCTION     *
C*                AT THIS TIME                                         *
C*                                                                     *
C*       RELDEV = STDDEV/AVE: STANDARD DEVIATION DIVIDED BY AVERAGE    *
C*                VALUE                                                *
C*                                                                     *
C*       LENGTH = NUMBER OF PARTIAL VARIANCES TO WRITE OUT             *
C*           ( NPARA*(NPARA + 1))/2                                    *
C*                                                                     *
C*       NPARA = NUMBER OF PARAMETERS ANALYZED.                        *
C*                                                                     *
C*                                                                     *
C*    2) ( SWLJ(K),K=1,LENGTH )                                        *
C*        ( 5(2X,E15.7) -FORMAT )                                      *
C*                                                                     *
C*       SWLJ(K) = A SINGLE OR COUPLED PARTIAL VARIANCE, WHERE         *
```

Program TRANS CONT'D.

```
C*                  K = NPARA*(L-1) - (L*(L-1))/2 + J                  *
C*                                                                     *
C*                                                                     *
C*    3) (B(IW(L)+1),L=1,NPARA)                                        *
C*      ( 5(2X,E15.7) -FORMAT )                                        *
C*                                                                     *
C*       B(IW(L)+1) = IN FOURIER ANALYSIS, IT IS THE SINE              *
C*                    EXPANSION COEFFICIENT FOR THE IW(L)TH            *
C*                    FREQUENCY.                                       *
C*                    IN WALSH ANALYSIS, IT IS THE EXPANSION           *
C*                    COEFFICIENT FOR THE IW(L)TH FREQUENCY.           *
C*                                                                     *
C*    CARDS 1,2 AND 3 ARE REPEATED FOR EACH LABEL-TIME POINT.          *
C*                                                                     *
C*                                                                     *
C*   OUTPUT FORMAT FOR  TAPE8 -EXPANSION COEFFICIENTS-                 *
C*                                                                     *
C*    FOURIER ANALYSIS                                                 *
C*                                                                     *
C*    1) LABEL,TIME,N2  ( A8,E15.7,I6 -FORMAT)                         *
C*                                                                     *
C*    2) ((A(K),B(K)),K=1,N2)   ( 4020  -FORMAT )                      *
C*                                                                     *
C*    CARDS 1 AND 2 ARE REPEATED FOR EACH LABEL-TIME POINT             *
C*                                                                     *
C*    WALSH ANALYSIS                                                   *
C*                                                                     *
C*    1) LABEL,TIME,NCOEFF  ( A8,E15.7,I6 -FORMAT )                    *
C*                                                                     *
C*    2) (F(K),K=1,NCOEFF )   ( 4020  -FORMAT )                        *
C*                                                                     *
C*    CARDS 1 AND 2 ARE REPEATED FOR EACH DIFFERENT LABEL-TIME         *
C*    POINT.                                                           *
C*                                                                     *
C*                                                                     *
C*********************************************************************
      REAL TIME(150)
      REAL MINSW(10,55),MAXSW(10,55),AVESW(10,55)
      REAL A(2048),B(2048)
      REAL F(2048)
      REAL X(2048)
C*
      INTEGER TITLE(8),METHOD,NPARA,TNPTS,NSIMUL,IW(50),NFUNC
      INTEGER WRITEUP,WRITED,READUP,READOWN
      INTEGER IWK(5000)
      INTEGER ITYP(2)
      INTEGER ILABEL(10)
C*
      LOGICAL TEST
```

Program TRANS CONT'D.

```
C*
      EQUIVALENCE (IWK(1),F(1))
C*
      COMMON SWLJ(210)
C*
      DATA IUNIT/3/, IHALF/2048/
      DATA IFLAG/0/
      DATA READUP/4/, READOWN/5/, WRITEUP/6/, WRITED/7/
      DATA MAXSW/550*0.0/, MINSW/550*1.0/, AVESW/550*0./
      DATA ITYP/10H FOURIER   ,10H WALSH     /
      DATA TEST/ .FALSE. /
C*
C*    INITIALIZE THE TIMING ROUTINE
      CALL TIMES(1,0)
C*
C*
C*    READ TAPE3
C*
      READ(3,10) (TITLE(J),J=1,8)
10    FORMAT(8A10)
      WRITE( 2,11) (TITLE(J),J=1,8)
11    FORMAT(1H ,8A10)
      READ(3,20) METHOD,NPARA,TNPTS,NSIMUL,NFUNC
20    FORMAT(A10,4I6)
      WRITE( 2,21)METHOD,NPARA,NFUNC,NSIMUL,TNPTS
21    FORMAT(1H ,/,1H ,A10,* SENSITIVITY ANALYSIS USING :*,/,
     +*  NUMBER OF PARAMETERS =*,I6,/,*  NUMBER OF OBJECT FUNCTIONS =*,
     + I6,/,*  NUMBER OF SIMULATIONS =*,I6,/,*  NUMBER OF TIME POINTS
=*
     +,I6)
C
      READ(3,30) JUNK,IACCUR
30    FORMAT(A10,5X,I3)
      READ(3,40)(IW(J),J=1,NPARA)
40    FORMAT(16I6)
      WRITE( 2,41)
41    FORMAT(1H ,* FREQUENCY SET * )
      WRITE( 2,42) (IW(J),J=1,NPARA)
42    FORMAT(15X,16I6)
      READ(3,30) JUNK
      READ(3,50) (TIME(J),J=1,TNPTS)
50    FORMAT(7E12.6)
C*
      READ(3,60)(ILABEL(J),J=1,NFUNC)
60    FORMAT(8(A8,2X))
C*    DETERMINE THE TYPE OF ANALYSIS
      IF( METHOD .EQ. ITYP(1) ) TEST = .TRUE.
      IF( METHOD .EQ. ITYP(2) ) TEST = .TRUE.
      IF(.NOT. TEST ) STOP "MTH"
```

Program TRANS CONT'D.

```
C*
C*     TIME THE INPUT
       CALL TIMES(1,0)
C*
C*
C*     NOW WE ARE READY TO TRANSPOSE THE MATRIX
C
       CALL TRANP( IUNIT, READUP, READOWN, WRITEUP, WRITED,
      + TNPTS, NFUNC, NSIMUL, IWK(1), IWK(1), IWK(IHALF+1),IHALF,NUMROW)
C
C*     TIME THE TRANSPOSE OPERATION
       CALL TIMES(2,0)
C*************************************************************************
C*
C*     INITIALIZE N2,LENGTH
C*
C*     N2 = LENGTH OF A FOURIER TRANSFORM COEFFICIENT VECTOR
C*
C*     LENGTH = LENGTH OF THE PARTIAL VARIANCE MATRIX WHEN FOLDED
C*              INTO A LINEAR ARRAY
C*
C*************************************************************************
       N2 = NSIMUL + 1
       LENGTH = ((NPARA*(NPARA+1))/2)
C*
C*
       DO 1000 ITIME=1,TNPTS
C*
C*
C*     NOW TRANSFORM EACH OBJECT FUNCTION AT THE TIME POINT
C*     'TIME(ITIME)'
C*
       DO 900 NF = 1, NFUNC
C*
C*
       READ(WRITEUP)(F(K),K=1,NUMROW)
C*
C*     NOW HAVING SET UP THE ARRAY F TRANSFORM IT
C
       IF(METHOD .EQ. ITYP(2)) CALL WHT( NSIMUL, F, IFLAG,IWK(IHALF+1) )
       IF( METHOD.EQ.ITYP(1)) CALL FFAST(F,NSIMUL,X,IWK,A,B)
C*
C*     CALCULATE THE TIME SPENT IN TRANSFORMATION
       CALL TIMES(3,0)
C
C*     NOW CALCULATE THE PARTIAL VARIANCES
C*
       IF(METHOD.EQ.ITYP(2))CALL WALPAR(F,NSIMUL,IW,NPARA,SWLJ,TOTVAR)
       IF(METHOD.EQ.ITYP(1))CALL FORPAR(A,B,NSIMUL,IW,NPARA,SWLJ,
```

Program TRANS CONT'D.

```
C*
C*     CALC THE TIME SPENT IN PARTIAL VARIANCES CALCULATIONS
       CALL TIMES(4,0)
C*
C*
C*     CALC THE PARTIAL VARIANCE STATISTICS
C*
       DO 300 L=1,NPARA
       DO 250 J=L,NPARA
       INDEX = NPARA*(L-1) - (L*(L-1))/2 + J
       MINSW(NF,INDEX) = AMIN1( MINSW(NF,INDEX), SWLJ(INDEX) )
       MAXSW(NF,INDEX) = AMAX1( MAXSW(NF,INDEX), SWLJ(INDEX) )
       AVESW(NF,INDEX)=((ITIME-1.)*AVESW(NF,INDEX)+SWLJ(INDEX))/FLOAT(ITI
      +ME)
250    CONTINUE
300    CONTINUE
C
C*     WRITE OUT THE EXPANSION COEFFICIENTS
C*
       IF( METHOD .EQ. ITYP(2) )  GO TO 375
C*     FOURIER METHOD
       CALL OUTCF( A, B,  N2, TIME(ITIME), ILABEL(NF) )
       CALL OUTP( SWLJ, A(1), TOTVAR,TIME(ITIME),LENGTH,ILABEL(NF),B,IW,
      +NPARA)
       GO TO 400
C*
375    CONTINUE
C*     WALSH METHOD
       CALL OUTCW(F, NSIMUL, TIME(ITIME), ILABEL(NF) )
       CALL OUTP( SWLJ, F(1), TOTVAR, TIME(ITIME), LENGTH,ILABEL(NF),F,IW
      +,NPARA)
400    CONTINUE
C*
C*
C*     COMPUTE TIME SPENT IN WRITING OUTPUT
       CALL TIMES(5,0)
C*
C*
900    CONTINUE
C*
1000   CONTINUE
C*
C*
C*     WRITE OUT DIAGNOSTICS
C*
       DO 1100 NF=1,NFUNC
       SUM = 0.0
       WRITE( 2,1400) ILABEL(NF)
       DO 1050 L=1,NPARA
```

Program TRANS CONT'D.

```
      DO 1050 J=L,NPARA
      INDEX = NPARA*(L-1) - (L*(L-1))/2 + J
      WRITE( 2,1500) L,J,AVESW(NF,INDEX),MINSW(NF,INDEX),MAXSW(NF,INDEX)
      SUM = SUM + AVESW(NF,INDEX)
1050  CONTINUE
      WRITE( 2,1600) SUM
1100  CONTINUE
1400  FORMAT(1H1,10X,A10,*  CONCENTRATION STATISTICS *,/)
1500  FORMAT(1H ,* (*,I2,*,*,I2,*)  *,*  AVESW =*,1PE14.6,
     +3X,* MIN =*,E14.6,3X,* MAX = *,E14.6)
1600  FORMAT(/,/,/,10X,* SUM OF AVERAGES =*,G14.6)
      CALL TIMES(1,1)
      END
```

Program TRANS CONT'D.
SUBROUTINE OUTP

```
      SUBROUTINE OUTP(SWLJ,AVE,TOTVAR,TIME,LENGTH,LABEL,B,IW,NPARA)
C*
C*************************************************************************
C*    SUBROUTINE OUTP WRITES OUT THE PARTIAL VARIANCES ON LOGICAL      *
C*    UNIT 'IUNIT'.                                                    *
C*************************************************************************
C*
      REAL SWLJ(LENGTH)
      REAL B(1)
      INTEGER IW(1)
      DATA IUNIT/9/
C*
      STDDEV = SQRT(TOTVAR)
       RELDEV = 0.0
       IF( AVE .EQ. 0.0 ) GO TO 5
      RELDEV = STDDEV/AVE
5      CONTINUE
      WRITE(IUNIT,10) LABEL,TIME,AVE,STDDEV,RELDEV,LENGTH,NPARA
10    FORMAT(A8,4(2X,E15.7),2I6)
      WRITE(IUNIT,20)(SWLJ(L),L=1,LENGTH)
20    FORMAT(5(2X,E15.7))
      WRITE(IUNIT,20)(B(IW(L)+1),L=1,NPARA)
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE OUTCW

```
      SUBROUTINE OUTCW(F,NCOEFF,TIME,LABEL)
C*************************************************************************
C*    SUBROUTINE OUTCW WRITES OUT THE WALSH EXPANSION COEFFICIENTS      *
C*    TO LOGICAL UNIT 'IUNIT'                                           *
C*************************************************************************
      REAL F(NCOEFF)
      DATA IUNIT/8/
      WRITE(IUNIT,10) LABEL,TIME,NCOEFF
10    FORMAT(A8,E15.7,I6)
      WRITE(IUNIT,20)(F(K),K=1,NCOEFF)
20    FORMAT(4020)
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE OUTCF

```
      SUBROUTINE OUTCF(A,B,N2,TIME,LABEL)
C*************************************************************************
C*                                                                       *
C*    SUBROUTINE OUTCF WRITES OUT THE FOURIER COEFFICIENTS               *
C*                                                                       *
C*************************************************************************
      REAL A(N2),B(N2)
      DATA IUNIT/8/
C*
      WRITE(IUNIT,10) LABEL,TIME,N2
10    FORMAT(A8,E15.7,I6)
      WRITE(IUNIT,20)((A(K),B(K)),K=1,N2)
20    FORMAT(4020)
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE WALPAR

```
      SUBROUTINE WALPAR(A,N,IW,NPARA,SWLJ,TOTVAR)
C*
C*************************************************************************
C*    SUBROUTINE WALPAR CALCULATES THE TOTAL VARIANCE AND                 *
C*    PARTIAL VARIANCES GIVEN THE WALSH EXPANSIONS COEFFICIENTS           *
C*    AND THE FREQUENCY SET.  ONLY THE  SINGLE PARTIAL VARIANCES AND      *
C*    COUPLED  PARTIAL VARIANCES,  S(L,J) ARE COMPUTED.                   *
C*                                                                        *
C*       ALL PARTIAL VARIANCES ARE STORED IN A LINEAR ARRAY               *
C*    (SWLJ( ) )., WHICH IS AN UNFOLDED UPPER TRIANGULAR MATRIX.          *
C*    THE DIAGONAL ELEMENTS, SWLJ( I,I ), ARE THE I'TH ISINGLE            *
C*    PARTIAL VARIANCES, AND THE (L,J)'TH ELEMENT IS THE COUPLED          *
C*    PARTIAL VARIANCE OF THE L'TH ND J'TH PARAMETERS.  THIS IS           *
C*    LINEARLY FOLDED BY                                                  *
C*                                                                        *
C*      INDEX = NPARA*(L - 1) - (L*(L - 1))/2 + J                         *
C*                                                                        *
C*    REFERENCE: T.H. PIERCE PHD. THESIS, M.S.U. 1980                     *
C*                                                                        *
C*   INPUT                                                                *
C*                                                                        *
C*       A = AN ARRAY OF THE WALSH EXPANSION COEFFICIENTS                 *
C*                                                                        *
C*       N = THE NUMBER OF EXPANSION COEFFICIENTS IN 'A'                  *
C*                                                                        *
C*       IW = AN INTEGER ARRAY OF THE FREQUENCY SET USED.                 *
C*                                                                        *
C*       NPARA = THE NUMBER OF PARAMETERS TO BE ANALYZED                  *
C*                                                                        *
C*   OUTPUT                                                               *
C*                                                                        *
C*       SWLJ = AN ARRAY OF THE SINGLE AND COUPLED PARTIAL VARIANCES      *
C*                                                                        *
C*       TOTVAR = THE TOTAL VARIANCE OF THE EXPANDED FUNCTION             *
C*                PARSEVAL'S FORMULA - AO**2                              *
C*                                                                        *
C*                                                                        *
C*   RESTRICTIONS                                                         *
C*                                                                        *
C*       1) IW(J) MUST NEVER BE EQUAL TO IW(K) FOR ANY J,K                *
C*                                                                        *
C*       2) SWLJ MUST BE DIMENSIONED AT LEAST  (NPARA*(NPARA+1))/2        *
C*                                                                        *
C*************************************************************************
C*
      REAL A(N)
      REAL SWLJ(1)
C*
```

Program TRANS CONT'D.
SUBROUTINE WALPAR

```
      INTEGER IW(NPARA)
C*************************************************************************
C*    CALCULATE THE TOTAL VARIANCE
C*    REMEMBER TO ADD ONE(1) TO THE FREQUENCIES TO ACCOUNT FOR
C*    THE FREQUENCY AO STORED AS A(1)
C*************************************************************************
      TOTVAR = 0.
C*    SKIP A(1) AS THIS IS THE AVERAGE VALUE
      DO 100 J=2,N
      TOTVAR = TOTVAR + A(J)*A(J)
100   CONTINUE
C*
C*    CALCULATE THE SINGLE PARTIAL VARIANCES
C
      DO 200 J=1,NPARA
      INDEX = NPARA*(J-1) - (J*(J-1))/2 + J
      SWLJ(INDEX) = (A(IW(J)+1)*A(IW(J)+1))/TOTVAR
200   CONTINUE
C*
C*    CALCULATE THE COUPLED PARTIAL VARIANCES
      NPARM1 = NPARA - 1
      DO 400 L=1, NPARM1
      JSTART = L + 1
      DO 300 J = JSTART, NPARA
C*    IF THE FREQUENCIES ARE EQUAL; IVAL=0 (MISTAKE)
      IF(IW(L) .EQ. IW(J) )  STOP  "WALPR"
      IVAL = XOR(IW(L),IW(J))
      INDEX = NPARA*(L-1) - (L*(L-1))/2 + J
      SWLJ(INDEX) = (A(IVAL + 1)*A(IVAL + 1))/TOTVAR
300   CONTINUE
400   CONTINUE
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE WHT

```
      SUBROUTINE WHT(NUM,X,II,Y)
C*************************************************************************
C****     II = 0 HADAMARD-ORDERED WHT                                    **
C****     II = 1 INVERSE HADAMARD-ORDERED WHT                            **
C****     II = 2 WALSH-ORDERED WHT                                       **
C****     II = 3 INVERSE WALSH-ORDERED WHT                               **
C****                                                                    **
C****     THIS ROUTINE CALCULATES THE FAST WALSH-HADAMARD                **
C****     TRANSFORMS (WHT)  FOR ANY GIVEN NUMBER WHICH                   **
C****     IS A POWER OF TWO.                                             **
C****                                                                    **
C****     NUM = NUMBER OF POINTS                                         **
C****     X(NUM) = ARRAY OF DATA TO BE TRANSFORMED                       **
C****              ON OUTPUT X(NUM) IS THE TRANSFORMED                   **
C****              EXPANSION COEFFICIENTS                                **
C*                                                                      *
C*   REFERENCE:   AHMED AND RAO, "ORTHOGONAL TRANSFORMS FOR             *
C*                  DIGITAL SIGNAL PROCESSING ", SPRINGER-              *
C*                  VERLAG, (1975).                                     *
C*                                                                      *
C*************************************************************************
      DIMENSION IPOWER(20),X(NUM),Y(NUM)
      IF(II.LE.1) GO TO 14
C****    BIT REVERSE THE INPUT
      DO 11 I=1,NUM
      IB = I - 1
      IL = 1
9     IBD = IB/2
      IPOWER(IL) = 1
      IF(IB.EQ.(IBD*2)) IPOWER(IL) = 0
      IF(IBD.EQ. 0) GO TO 10
      IB = IBD
      IL = IL + 1
      GO TO 9
10    CONTINUE
      IP = 1
      IFAC = NUM
      DO 12 I1 = 1,IL
      IFAC = IFAC/2
12    IP = IP + IFAC*IPOWER(I1)
11    Y(IP) = X(I)
      DO 13 I = 1,NUM
13    X(I) = Y(I)
14    CONTINUE
C****    CALCULATE THE NUMBER OF ITERATIONS
65    ITER = 0
```

Program TRANS CONT'D.
SUBROUTINE WHT

```
      IREM = NUM
1     IREM = IREM/2
      IF(IREM.EQ.0) GO TO 2
      ITER=ITER + 1
      GO TO 1
2     CONTINUE
C****     BEGIN A LOOP FOR (LOG TO BASE TWO OF NUM) ITERATIONS
      DO 50 M=1,ITER
C****
C****     CALCULATE THE NUMBER OF PARTIONS
      IF(M.EQ.1) NUMP = 1
      IF(M.NE.1) NUMP = NUMP*2
      MNUM = NUM/NUMP
      MNUM2 = MNUM/2
C****
C****     BEGIN A LOOP FOR THE NUMBER OF PARTITIONS.
C****
      ALPH = 1.
      DO 49 MP = 1,NUMP
      IB = (MP-1)*MNUM
C****
C****     BEGIN A LOOP THROUGH THIS PARTITION.
C****
      DO 48 MP2 = 1,MNUM2
      MNUM21 = MNUM2 + MP2 + IB
      IBA = IB +MP2
      Y(IBA) = X(IBA) + ALPH*X(MNUM21)
      Y(MNUM21) = X(IBA) - ALPH*X(MNUM21)
48    CONTINUE
      IF(II.GE.2) ALPH = -ALPH
49    CONTINUE
C****
C****
      DO 7 I=1,NUM
7     X(I) = Y(I)
50    CONTINUE
      IF(II.EQ.1 .OR. II.EQ.3) RETURN
      R=1./FLOAT(NUM)
      DO 15 I=1,NUM
15    X(I) = X(I)*R
      RETURN
      END
```

Program TRANS CONT'D.
TRANP

```
      SUBROUTINE TRANP( IUNIT, READUP, READOWN, WRITEUP, WRITED,
     + TNPTS, NFUNC, NSIMUL, C, UP, DOWN, LENGTH,NUMROW)
C*
C*************************************************************************
C*    SUBROUTINE TRANP TAKES THE MATRIX STORED ON 'IUNIT' AND            *
C*    TRANSPOSES IT. ( A(I,J) => A(J,I) ) RETURNING THE                  *
C*    TRANSPOSED MATRIX ON LOGICAL UNIT 'WRITEUP'.                       *
C*                                                                       *
C* NOTES                                                                 *
C*                                                                       *
C*    1) THE ARRAY 'C' MAY BE EQUIVALENCED TO EITHER THE ARRAY           *
C*       'UP' OR THE ARRAY 'DOWN'.                                       *
C*                                                                       *
C*    2) LENGTH MUST BE ONE POWER OF TWO GREATER THAN NROW, UNLESS       *
C*       NROW IS A POWER OF TWO.                                         *
C*************************************************************************
      INTEGER READUP,READOWN,WRITEUP,WRITED
      INTEGER TNPTS,NSIMUL,NFUNC,IUNIT
C*************************************************************************
C*   THE VECTOR 'C' SHOULD BE OF DIMENSION ONE POWER OF TWO
C*    GREATER THAN NROW( UNLESS NROW IS A POWER OF TWO )
      REAL C(LENGTH)
      REAL UP(LENGTH),DOWN(LENGTH)
C*
      DATA KOUNT/0/, ZERO/0./, LCOUNT/0/, NUMADD/0/
      DATA NUMADD2/0/
      NCOL = TNPTS*NFUNC
      NROW = NSIMUL
C*   READ IN THE MATRIX
1     CONTINUE
      DO 1000 J=1, TNPTS
      ISTR = (J-1)*NFUNC + 1
      ISTOP = ISTR + NFUNC - 1
      READ(IUNIT,10000)(C(K),K=ISTR,ISTOP)
10000 FORMAT(4020)
1000  CONTINUE
C*   IF KOUNT=0 ,THEN WE NEED TO WRITE THE UP-TAPE(TAPE1 INITIALLY)
C*   IF KOUNT = 1 , THEN WE WRITE THE DOWN-TAPE( TAPE2 INITIALLY)
C*
      IF(KOUNT .NE. 0 ) GO TO 5
C*   WRITE THE ODD ROWS
      DO 500 K=1,NCOL
500   WRITE(READUP) C(K)
      KOUNT = 1
C*
C*   LCOUNT COUNTS THE NUMBER OF ROWS WRITTEN, BOTH UP AND DOWN
C*
      LCOUNT = LCOUNT + 1
```

Program TRANS CONT'D.
TRANP

```
C*
C*    IF DONE, IE LCOUNT = NUMBER OF ROWS, THEN GO TO NEXT TASK
C*    IF NOT DONE, THEN CONTINUE READ-WRITE
C*
      IF( LCOUNT .EQ. NROW ) GO TO 9
      GO TO 1
5     CONTINUE
C*    WRITE THE EVEN ROWS
      DO 510 K=1,NCOL
510   WRITE(READOWN) C(K)
C*    SET KOUNT=0 SO NEXT WRITE IS 'DOWN'
      KOUNT = 0
      LCOUNT = LCOUNT + 1
C*    CHECK FOR END OF DATA
      IF(LCOUNT .EQ. NROW) GO TO
      GO TO
9     CONTINUE
C*    TAPE 1,2 ARE WRITTEN WITH THE MATRIX NOW WE NEED TO MAKE SURE
C*    THAT THE ROW-DIMENSION IS A POWER OF TWO,AND IF NOT THEN WE
C*    MUST ADD SUFFICIENT ZEROS TO MAKE THE ROW-DIMENSION A POWER OF 2
C*
C*
C*  CHECK FOR HAVING WRIITEN AN EVEN NUMBER OF ROWS
      IF(KOUNT .EQ. 0 ) GO TO 120
C*  ODD NUMBER OF ROWS WERE WRITTEN
C*  ADD ONE ROW OF ZEROS
      DO 115 K=1,NCOL
      WRITE(READOWN ) ZERO
115   CONTINUE
      LCOUNT = LCOUNT + 1
C*  EVEN NUMBER OF ROWS WRITTEN
120   CONTINUE
C*  FIGURE OUT EXPONENT OF NEAREST POWER OF TWO LARGER
      DO 116 M=1,50
      MDIVID = M
      RTEST = FLOAT(LCOUNT)/(2.**M)
      IF ( RTEST .LE. 1 ) GO TO 118
116   CONTINUE
      STOP 2
118   CONTINUE
C*    CHECK FOR EXACT POWER OF TWO
      IF (RTEST .EQ. 1. ) GO TO 200
C*    NOW CALCULATE THE NUMBER OF ROWS WE MUST ADD
      NUMADD = 2**MDIVID - LCOUNT
C*    NUMADD SHOULD ALSO BE DIVISIBLE BY TWO
      IF(NUMADD .NE. 2*(NUMADD/2) ) STOP 3
C*    WRITE ZEROS INTO DUMMY ROWS
      NUMADD2 = NUMADD/2
```

Program TRANS CONT'D.
TRANP

```
      DO 130 L=1,NUMADD2
      DO 130 K=1,NCOL
      WRITE(READUP ) ZERO
      WRITE(READOWN ) ZERO
130   CONTINUE
200   CONTINUE
C*
C*   LET NUMADD BE THE TOTAL NUMBER OF ADDED ROWS
C*   REMEMBER WE MAY HAVE ADDED A ROW EARLIER
      IF( NROW .NE. LCOUNT )NUMADD = NUMADD + 1
C* THE FINAL CHECK
      NUMROW = NUMADD + NROW
      IF(NUMROW .NE. (2**MDIVID)) STOP 4
C*    EVEN AND ODD TAPE WRITTEN
      LOOP = 1
      REWIND READUP
      REWIND READOWN
      REWIND WRITEUP
      REWIND WRITED
C*
C$    DIAGNOSTICS
      WRITE( 2,101)
101   FORMAT(/,/,/,*     SUBROUTINE TRANP STATISTICS *,/)
      WRITE( 2,112) NROW,NCOL,NUMADD,NUMADD2
112   FORMAT(* *,* NROW=*,I5,* NCOL=*,I4,*  NUMADD=*,I6,
     +*   NUMADD2=*,I5)
      NINSERT = NCOL
      NCHECK = NUMROW/2
10    CONTINUE
C*    INITIALIZE FOR THE READ-WRITE
C*      KOUNT = NUMBER OF INSERTS DONE
C*     LCOUNT = TOTAL NUMBER OF READ-WRITES DONE THIS
C*     ITERATION
      KOUNT = 0
      LCOUNT = 0
20    CONTINUE
      READ(READUP )(UP(L),L=1,LOOP)
      READ(READOWN)(DOWN(L),L=1,LOOP)
      WRITE(WRITEUP)(UP(L),L=1,LOOP),(DOWN(M),M=1,LOOP)
      KOUNT = KOUNT + 1
      IF( KOUNT .NE. NINSERT ) GO TO 20
      LCOUNT = LCOUNT + 1
C*     LCOUNT SHOULD EQUALNCHECK HERE
C*     ONLY IF NCHECK = 1 AND IT IS THE LAST MIX
      IF( LCOUNT .EQ. NCHECK ) GO TO 65
      KOUNT = 0
30    CONTINUE
      READ(READUP )(UP(L),L=1,LOOP)
```

Program TRANS CONT'D.
TRANP

```
      READ(READOWN )(DOWN(L),L=1,LOOP)
      WRITE(WRITED)(UP(L),L=1,LOOP),(DOWN(M),M=1,LOOP)
      KOUNT = KOUNT + 1
      IF( KOUNT .NE. NINSERT ) GO TO 30
      LCOUNT = LCOUNT + 1
      IF( LCOUNT .EQ. NCHECK ) GO TO 50
      KOUNT = 0
      GO TO 20
50    CONTINUE
      LOOP = LOOP*2
      NCHECK = NCHECK/2
C*    ERROR CHECKING
      IF( NCHECK .LE. 0 ) STOP 2
      ISAVUP = READUP
      ISAVD =READOWN
      READUP = WRITEUP
      READOWN = WRITED
      WRITEUP=ISAVUP
      WRITED= ISAVD
      REWIND READUP
      REWIND READOWN
      REWIND WRITEUP
      REWIND WRITED
      GO TO 10
65    CONTINUE
      REWIND WRITEUP
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE TIMES

```
      SUBROUTINE TIMES(ISUB,ITYPE)
C*****************************************************************************
C*                                                                           *
C*    SUBROUTINE TIMES COMPUTES THE CPU TIME SPENT BETWEEN CALLS.            *
C*    THIS IS INSTALLATION DEPENDENT.                                        *
C*    ITS USE IS FOR DOCUMENTATION PURPOSES ONLY                             *
C*                                                                           *
C*       ISUB = THE PROCEDURE TO BE TIMED.                                   *
C*                                                                           *
C*       ITYPE = FLAG: .LT. 1 FOR TIMING                                     *
C*                     .GE. 1 FOR FINAL PRINT                                *
C*                                                                           *
C*****************************************************************************
C*
      REAL TIMS(15),NEW,LAST
      INTEGER NAME(15)
C*
      DATA TIMS/15*0./
      DATA LAST /0./
      DATA NAME/10HREAD INPUT,10HTRANSPOSE ,10HTRANSFORM ,10HPARTIALVAR
     +,10HWRITE OUTP/
      DATA LENGTH /5/
C*
      IF(ITYPE .GE. 1) GO TO 5
      NEW = SECOND(CPU)
C*     INITIALIZE THE VALUES IN FIRST ENTRY
      IF( LAST .EQ. 0. ) LAST = NEW
      TIMS(ISUB) = TIMS(ISUB) + NEW - LAST
      LAST = NEW
      RETURN
5     CONTINUE
      WRITE( 2,100)
100   FORMAT(* *,/,/,/,5X,*  SECONDS*,4X,* PROCEDURE *,/)
      TSUM = 0.
      DO 10 J=1,LENGTH
      TSUM = TSUM + TIMS(J)
      WRITE( 2,150) TIMS(J),NAME(J)
150   FORMAT(* *,5X,F7.3,5X,A10)
10    CONTINUE
      WRITE( 2,160) TSUM
160   FORMAT(* *,/,/,6X,F7.3,4X,* TOTAL TIME *)
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE FFAST

```
      SUBROUTINE FFAST(F,NPTS,X,IWK,A,B)
C*
C*************************************************************************
C*    SUBROUTINE FFAST COMPUTES THE FOURIER TRANSFORM OF A VECTOR      *
C*    IN THIS CASE 'F' IS LENGTHENED FROM ( -PI/2 , PI/2 ) TO          *
C*    ( 0 , 2PI ) AND THEN FOURIER-TRANSFORMED INTO COSINE AND SINE    *
C*    COEFFICIENTS, A AND B RESPECTIVELY.                              *
C*                                                                     *
C*  NOTE                                                               *
C*                                                                     *
C*    1) SINCE 'F' IS NEVER USED IN FFCSIN IT MAY BE EQUIVALENCED      *
C*       TO 'IWK'.                                                     *
C*                                                                     *
C*    2) THE ROUTINES ALSO ALLOW THE EQUIVALENCING OF 'A' AND          *
C*       'X'.                                                          *
C*                                                                     *
C*    3) A0, THE AVERAGE VALUE, IS STORED AS 'A(1)'.                   *
C*************************************************************************
      REAL F(NPTS)
      REAL X(1),A(1),B(1)
C*
      INTEGER IWK(1)
C*   NPTS MUST BE AN ODD INTEGER AND NOTE WE ARE GOING FROM
C*      -PI/2 TO  PI/2  AND TRANSFORMING TO (0,2*PI)
C******
C*
      NPTSP1=NPTS + 1
      NPTS2=NPTS*2
      N2=NPTS+1
      RNPTS2=(1.0/FLOAT(NPTS2))
      IQ=(NPTS-1)/2
      IQP1=IQ+1
C*
C
      L=0
C
C*   TRANSFORM F(-PI/2 , PI/2) TO X(0 , 2*PI)
C
      DO 1000 J=IQP1,NPTS
      L=L+1
1000  X(L)=F(J)
      DO 2000 J=1,IQP1
      L=L+1
      JJ=NPTSP1-J
      X(L)=F(JJ)
2000  CONTINUE
      DO 3000 J=1,IQ
      L=L+1
```

Program TRANS CONT'D.
SUBROUTINE FFAST

```
      JJ=IQP1-J
      X(L)=F(JJ)
3000  CONTINUE
      DO 4000 J=1,IQ
      L=L+1
      X(L)=F(J)
4000  CONTINUE
C
C
C*    CALL IMSL ROUTINES TO CALCULATE FOURIER COEFFICIENTS
C
      CALL FFCSIN(X,NPTS2,A,B,IWK)
C*
C*    SCALE THE COEFFICIENTS TO THEIR CORRECT VALUES.
      DO 350 J=1,N2
      A(J)=RNPTS2*A(J)
      B(J)=RNPTS2*B(J)
350   CONTINUE
      A(1)=A(1)/2.0
      A(N2)=A(N2)/2.0
      B(1)=0.0
      B(N2)=0.0
C
C*
      RETURN
      END
```

Program TRANS CONT'D.
SUBROUTINE FORPAR

```
      SUBROUTINE FORPAR(A, B, NPTS, IW, NPARA, SWLJ, TOTVAR,IACCUR)
C*****************************************************************
C*    CALCULATE THE PARTIAL VARIANCES                             *
C*     THIS ROUTINE IS SET UP FOR 4TH ORDER ACCURATE FREQUENCY    *
C*     SETS                                                       *
C*    FIRST CALCULATE THE VARIANCE ( PARSEVAL"S FORMULA - AO**2   *
C*    THEN CALCULATE THE SUM OF HARMONICS NOTING THAT ALL BUT THE *
C*    NPARA'TH ARE SUMMED TO THE FIRST HARMONIC AND THE NPARA'TH ONLY T*
C*      THE FUNDAMENTAL HARMONIC                                  *
C*                                                                *
C*        ALL PARTIAL VARIANCES ARE STORED IN A LINEAR ARRAY      *
C*     (SWLJ( ) ), WHICH IS AN UNFOLDED UPPER TRIANGULAR MATRIX.  *
C*     THE DIAGONAL ELEMENTS, SWLJ( I,I ), ARE THE I'TH ISINGLE   *
C*     PARTIAL VARIANCES, AND THE (L,J)'TH ELEMENT IS THE COUPLED *
C*     PARTIAL VARIANCE OF THE L'TH ND J'TH PARAMETERS.  THIS IS  *
C*     LINEARLY FOLDED BY                                         *
C*                                                                *
C*       INDEX = NPARA*(L - 1) - (L*(L - 1))/2 + J                *
C*                                                                *
C*     REFERENCE: T.H. PIERCE PHD. THESIS, M.S.U. 1980            *
C*                                                                *
C*    INPUT                                                       *
C*                                                                *
C*        A = AN ARRAY OF THE COSINE EXPANSION COEFFICIENTS       *
C*                                                                *
C*        B = AN ARRAY OF THE SINE EXPANSION COEFFICIENTS         *
C*                                                                *
C*        NPTS = NUMBER OF SIMULATIONS                            *
C*                                                                *
C*        IW = AN INTEGER ARRAY OF THE FREQUENCY SET USED.        *
C*                                                                *
C*        NPARA = THE NUMBER OF PARAMETERS TO BE ANALYZED         *
C*                                                                *
C*    OUTPUT                                                      *
C*                                                                *
C*        SWLJ = AN ARRAY OF THE SINGLE AND COUPLED PARTIAL VARIANCES *
C*                                                                *
C*        TOTVAR = THE TOTAL VARIANCE OF THE EXPANDED FUNCTION    *
C*                 PARSEVAL'S FORMULA - AO**2                     *
C*                                                                *
C*                                                                *
C*    RESTRICTIONS                                                *
C*                                                                *
C*        1) IW(J) MUST NEVER BE EQUAL TO IW(K) FOR ANY J,K       *
C*                                                                *
C*        2) SWLJ MUST BE DIMENSIONED AT LEAST  (NPARA*(NPARA+1))/2 *
C*****************************************************************
      REAL A(1),B(1),SWLJ(1)
```

Program TRANS CONT'D.
SUBROUTINE FORPAR

```
C*
C*
      INTEGER IW(NPARA)
C*
      IRANGE = IACCUR - 1
      IMID = IACCUR/2
      N2 = NPTS + 1
      NPARM1 = NPARA - 1
      TOTVAR = 0.0
C*    SKIP A(1) AND B(1) AS AO, THE AVERAGE VALUE, IS
C*    STORED AS A(1); B(1) = 0.
      DO 600 JJ=2,N2
      TOTVAR = A(JJ)*A(JJ) + B(JJ)*B(JJ) +TOTVAR
600   CONTINUE
C*  LP IS THE HARMONICS
      DO 650 L=1,NPARM1
      SUM = 0.0
      DO 625 LP = 1,2
      LPWP1=LP*IW(L) + 1
      SUM = A(LPWP1)*A(LPWP1) + B(LPWP1)*B(LPWP1) + SUM
625   CONTINUE
      INDEX = NPARA*(L-1) - (L*(L-1))/2 + L
      SWLJ(INDEX) = SUM/TOTVAR
650   CONTINUE
C*
      SUM=0.0
      DO 675 L=1,1
      LPWP1 = L*IW(NPARA) + 1
      SUM = SUM + A(LPWP1)*A(LPWP1) + B(LPWP1)*B(LPWP1)
675   CONTINUE
      INDEX = (NPARA*(NPARA+1))/2
      SWLJ(INDEX) = SUM/TOTVAR
C*
C*
      DO 900 L=1, NPARM1
      JSTART = L + 1
      DO 800 J=JSTART, NPARA
      SUM = 0.
      DO 750 KP=1,IRANGE
      IP = IMID - KP
      DO 700 IK = 1, IRANGE
      K = IMID - IK
      IF(IP .EQ. 0) GO TO 750
      IF( K .EQ. 0) GO TO 700
C*    ADD ONE (1) TO THE FREQUENCY COUNT TO ACCOUNT FOR AO  BO
                                                          e
      IFREQ = IP*IW(L) + K*IW(J) + 1
      IF(IFREQ .LE. 1 ) GO TO 700
      IF(IFREQ .GE. N2) GO TO 700
```

Program TRANS CONT'D.
SUBROUTINE FORPAR

```
      SUM = A(IFREQ)*A(IFREQ) + B(IFREQ)*B(IFREQ) + SUM
700   CONTINUE
750   CONTINUE
      INDEX = NPARA*(L-1) - (L*(L-1))/2 + J
      SWLJ(INDEX) = SUM/TOTVAR
800   CONTINUE
900   CONTINUE
      RETURN
      END
```

Program PLOTSEN

```
      PROGRAM PLTSEN(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT,
     +            TAPE9)
C*****************************************************************************
C*    THIS PROGRAM PLOTS RESULTS OF  TAPE9 SENSITIVITY ANALYSIS FILE.        *
C*   THE PROGRAM READS CARDS FOR INFORMATION ON WHAT TO PLOT.                *
C*    IT THEN SEARCHES THE FILE (TAPE9) FOR THE DESIRED VALUES,              *
C*   AND PLOTS IT ON A LINE PRINTER.                                         *
C*   IF MORE THAN ONE PLOT IS DESIRED, IT REWINDS THE FILE                   *
C*   AND REPEATS.                                                            *
C*                                                                           *
C*    INPUT                                                                  *
C*          CARD 1                                                           *
C*                                                                           *
C*                NPLOT, NCONC (2I5 FORMAT)                                  *
C*                                                                           *
C*                     NPLOT = THE TOTAL NUMBER OF PLOTS                     *
C*                               DESIRED.                                    *
C*                                                                           *
C*                     NCONC= THE NUMBER OF DIFFERENT OUTPUT                 *
C*                              FUNCTIONS IN  TAPE9.                         *
C*                                                                           *
C*  NOTE  CARDS 2-6 ARE TO BE REPEATED FOR ALL THE DESIRED OBJECT            *
C*          FUNCTIONS.                                                       *
C*                                                                           *
C*           CARD 2                                                          *
C*                   ITEST  ( A10 FORMAT)                                    *
C*                                                                           *
C*                       ITEST = THE LABEL OF THE OBJECT (OR                 *
C*                           OUTPUT) FUNCTION TO BE PLOTTED                  *
C*                                                                           *
C*          CARD 3                                                           *
C*                                                                           *
C*                    NFUNC,NPOINT                                           *
C*                                                                           *
C*                       NFUNC = THE NUMBER OF FUNCTIONS TO PLOT             *
C*                             FOR THE LPT'TH PLOT.                          *
C*                                                                           *
C*                     NPOINT = THE NUMBER OF POINTS TO BE PLOTTED           *
C*                               IN THE LPT'TH PLOT (X-AXIS)                 *
C*                                                                           *
C*           CARD 4                                                          *
C*                                                                           *
C*                  ITITLE(8)   (8A10 FORMAT )                               *
C*                                                                           *
C*                              ITITLE = THE PLOT TITLE FOR THE LPT'TH       *
```

Program PLOTSEN CONT'D.

```
C*                                        PLOT.                                 *
C*                                                                              *
C*         CARD 5                                                               *
C*                                                                              *
C*                  SYM(K)  (10A1 FORMAT)                                       *
C*                                                                              *
C*                      SYM(K) = THE SYMBOLS TO BE USED IN THE PLOT.            *
C*                               ( IE THE SYMBOL FOR THE KTH FUNCTION).         *
C*                                                                              *
C*          CARD 6 - CARD(5+NFUNC)                                              *
C*                                                                              *
C*              NAME(K)  (A10 FORMAT)                                           *
C*                                                                              *
C*                       NAME(K) = THE NAME (LABEL) OF THE KTH FUNCTION         *
C*                                 TO BE PLOTTED.                               *
C*                                                                              *
C*******************************************************************************
C*
      REAL TIME(100)
      REAL PLT(100,10)
      REAL DELX(100)
C*
      CHARACTER*10 JNAME(10),NAME(10),ITITLE(8)
       CHARACTER*10 ITEST
C*
      COMMON /PLTPTS/ SYM(10)
C*
      DATA IO/2/, IS/1/, DELX/100*0./
      DATA MAX/100/
C*
C*    READ IN CARD INPUT
C*
C*
C*    READ IN:
C*           NFUNC = NUMBER OF FUNCTION TO PLOT
C*           NPOINT = NUMBER OF POINTS PER PLOT
C*           NPLOT = NUMBER OF PLOTS
C*
      READ(1,20) NPLOT,NCONC
20    FORMAT(3I5)
      WRITE(2,30) NPLOT
30    FORMAT('1',' THERE ARE ',I5,' PLOTS')
C*
C*
      DO 1000 LPT=1,NPLOT
C*
C* READ IN CORRECT OBJECT FUNCTIONS
      READ(1,35) ITEST
35    FORMAT(A10)
```

Program PLOTSEN CONT'D.

```
C*    READ IN THE NUMBER OF FUNCTIONS TO BE PLOTTED AND
C*     THE NUMBER OF POINTS TO PLOT PER FUNCTION
C*
      READ(1,*) NFUNC,NPOINT
45    FORMAT(2I5)
C*
C*     READ IN PLOT TITLE
      READ(1,10)(ITITLE(K),K=1,8)
10    FORMAT(8A10)
C*    READ IN PLOT SYMBOLS
      READ(1,55)(SYM(K),K=1,NFUNC)
55    FORMAT(10A1)
C*
C*    READ IN THE NAME OF THE PLOTTED FUNCTIONS
      DO 80 K=1,NFUNC
      READ(1,75) NAME(K)
75    FORMAT(A10)
80    CONTINUE
      NKOUNT = NCONC*NPOINT
C*
      LPLOT=LPT
      CALL READ9(NKOUNT, NFUNC,TIME,ITEST,NCONC,LPLOT,PLT,ITPTS)
      IFLAG = 0
      IF( ITPTS .NE. NPOINT ) IFLAG = 1
      IF ( IFLAG .EQ. 1) WRITE(2,310) NPOINT,ITPTS
310   FORMAT('1','  NUMBER OF POINTS EXPECTED =',I6,/,5X,
     + ' NUMBER OF POINTS READ =',I6)
      IF( IFLAG .EQ. 1 ) NPOINT = ITPTS
C*
      CALL PLOT(IO,PLT,DELX,IS,ITITLE,NAME,TIME(1),TIME(NPOINT),NPOINT,
     + NFUNC,MAX)
C*
C*    WRITE OUT THE PLOTTED POINTS.
C*
      WRITE(2,300)(NAME(K),K=1,NFUNC)
300   FORMAT('1',' POINT',10(1X,A10,1X))
      DO 350 K=1,NPOINT
      WRITE(2,325)( K,(PLT(K,L),L=1,NFUNC))
325   FORMAT(' ',I4,1X,10(1X,1PE10.3,1X))
350   CONTINUE
      REWIND 9
1000  CONTINUE
      END
```

Program PLOTSEN CONT'D.
SUBROUTINE READ9

```
      SUBROUTINE READ9(NKOUNT,NFUNC,TIM,ITEST,NCONC,LPLOT,PLT,ITIME)
C*****************************************************************
C*                                                               *
C*         SUBROUTINE READ READS IN THE OUTPUT TAPE7 FROM PROGRAM *
C*         TRANS.  THIS IS READ IN SO THAT IT MAY BE PLOTTED.    *
C*                                                               *
C*    INPUT UNIT = 9                                             *
C*                                                               *
C*    OUTPUT UNIT = 2                                            *
C*                                                               *
C*                                                               *
C*    VARIABLES:                                                 *
C*       NKOUNT = NUMBER OF TOTAL SENSITIVITY POINTS             *
C*                                                               *
C*       NCONC = NUMBER OF CONCENTRATIONS                        *
C*                                                               *
C*       TIM(70) = TIME POINT OF S. A. POINT                     *
C*                                                               *
C*****************************************************************
C*                                                               *
      CHARACTER*10 LABEL,ITEST
C*
       REAL PLT(100,10)
      REAL TIM(100),SWLK(50),B(50)
      DATA IIN/9/
C*
      ITIME = 0
      ISCALE = 1
C
      DO 1000 KOUNT = 1,NKOUNT
C
      READ(IIN,10) LABEL,TIME,AVE,STDDEV,RELDEV,LENGTH,NPARA
10    FORMAT(A10,4(2X,E15.7),2I6)
C
      READ(IIN,20)(SWLK(K),K=1,LENGTH)
20    FORMAT(5(2X,E15.7))
C
      READ(IIN,20)(B(L),L=1,NPARA)
C
C
C*    FINDS CORRECT CONCENTRATION LABEL
      DO 200 I=1,NCONC
      IVAL=I
      IF( LABEL .EQ. ITEST) GO TO 350
200   CONTINUE
      GO TO 1000
350   CONTINUE
C             NORMILIZE LPLOT TO ( 1,2,3,4)
```

Program PLOTSEN CONT'D.
SUBROUTINE READ9

```
C*
      IF(LPLOT .LE. 4) GO TO 400
      LPLOT = LPLOT - 4
      GO TO 350
400    CONTINUE
C*  FOUND THE CORRECT OBJECT FUNCTION
C*   SAVE THE DESIRED VALUES.
      ITIME = ITIME + 1
      TIM(ITIME) = TIME
C
      GO TO ( 500, 550, 600, 650 ) LPLOT
500   CONTINUE
C       SAVE THE AVERAGE VALUE
C
      PLT(ITIME,1) = AVE
      GO TO 1000
C
 550   CONTINUE
C          SAVE THE  RELATIVE DEVIATION CURVE
       PLT(ITIME,1) = RELDEV
      GO TO 1000
C
600    CONTINUE
C          SAVE THE SINGLE PARTIAL VARIANCES
      DO 610 NP=1,NPARA
       INDEX = NPARA*(NP-1) - (NP*(NP-1))/2 +  NP
       PLT(ITIME,NP) = SWLK(INDEX)
610    CONTINUE
      GO TO 1000
C
650     CONTINUE
C
C            SAVE THE EXPANSION COEFFICIENTS
      DO 660 NP = 1, NPARA
       PLT(ITIME,NP) = B(NP)
660      CONTINUE
C
1000   CONTINUE
C*
      RETURN
      END
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
      SUBROUTINE PLOT(IO,ARAY,XARAY,ISCALE,JNAME,NAME,BLOW,BHI,NPT,NF,
     + MAX)
C*
C*************************************************************************
C*    IO = THE OUTPUT UNIT                                               *
C*                                                                       *
C*    ISCALE = TYPE OF PLOT DESIRED, 1= LINEAR SCALE ,2= SEMILOG,        *
C*         3= LOG-LOG.  FOR LOG-LOG READ IN EQUAL INTERVALS ON A LOG     *
C          SCALE.                                                       *
C*         NO MIXING OF 1,2,OR 3 ALLOWED IN THE SAME PLOT.               *
C*                                                                       *
C*    BLOW= THE LOWER BOUND OF THE PLOT FOR THE X-AXIS                   *
C*    BHI = THE UPPER BOUND OF THE PLOT FOR THE X-AXIS                   *
C*                                                                       *
C*    NPT = THE NUMBER OF POINTS PER FUNCTION TO BE PLOTTED              *
C*                                                                       *
C*    MAX = THE INNER DIMENSION OF THE ARAY DEFINED IN THE MAIN PROGRAM  *
C*                                                                       *
C*    NF = THE NUMBER OF FUNCTIONS TO BE PLOTTED                         *
C*                                                                       *
C*    ARAY(MAX,NF) = ARRAY OF POINTS TO BE PLOTTED                       *
C*                                                                       *
C*    XARAY(MAX) = THE ERRORS ASSOCIATED WITH THE POINTS FOR THE FIRST   *
C*         FUNCTION. ONLY THE FIRST FUNCTION WILL BE PLOTTED WITH ERROR  *
C          BARS                                                         *
C*                                                                       *
C*    JNAME(8) = THE TITLE OF THE PLOT. THIS WILL BE PRINTED AT THE TOP  *
C*         OF THE PLOT (8A10 FORMAT )                                    *
C*                                                                       *
C*    NAME(10) = THE NAME OF EACH FUNCTION TO BE PLOTTED(USE A10 FORMAT  *
C*         OR 10H )                                                      *
C*                                                                       *
C*  A LABELED COMMON BLOCK IS ALSO REQUIRED                              *
C*  THIS BLOCK CONTAINS THE SYMBOLS TO BE USED IN THE                    *
C*  PLOT FOR EACH FUNCTION                                               *
C*     USE                                                               *
C*         COMMON /PLTPTS/ POINT(10)                                     *
C*  WHERE POINT(I) IS THE SYMBOL FOR THE ITH                             *
C*   FUNCTION TO BE READ IN USING A1 FORMAT OR DEFINED WITH 1H FORMAT    *
C*************************************************************************
C*
      DIMENSION ARAY(MAX,NF),XARAY(MAX)
      REAL  XMAX(10),XMIN(10)
      CHARACTER*7 SC(3)
      CHARACTER*10 NAME(10),JNAME(10)
C*
      DIMENSION VAL(106)
      DIMENSION XDIV(3),XMIN1(11)
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
C*
      COMMON /PLTPTS/ POINT(10)
C*
      DATA BLANK/1H /,
     *  DASH/1H-/,STAR/1H*/
      DATA XDIV/1.,2.,5./
      DATA SC/'LINEAR','LOG','LOG LOG'/
C*
      IF( NPT .LE. MAX ) GO TO 5
      WRITE(IO,1)(JNAME(K),K=1,8)
1     FORMAT('  ARRAY SIZE TOO LARGE IN PLOT OF',/,3X,8A10)
      RETURN
5     CONTINUE
      IF( BLOW .LT. BHI) GO TO 50
      WRITE(IO,160) BLOW,BHI
160   FORMAT(' X-AXIS MINIMUM AND MAXIMUM ARE NOT',E15.7,'.GE.',E15.7)
50    CONTINUE
C*  TO PAGE OR NOT TO PAGE
      IF(NPT.LE.40) GO TO 8
      WRITE(IO,6)
6     FORMAT('1')
      GO TO 9
8     WRITE(IO,7)
7     FORMAT(/////)
9     CONTINUE
C*       WRITE HEADER FOR PLOTS
      WRITE(IO,10)((JNAME(K),K=1,8),SC(ISCALE))
10    FORMAT('   PLOT OF ',8A10,5X,A10,' SCALE',/)
      WRITE(IO,13)
      WRITE(IO,12)(POINT(I),NAME(I),I=1,NF)
12    FORMAT(10X,A2,5X,A10)
13    FORMAT(30H ---------------------------- ,/)
      WRITE(IO,13)
C
C        FIND MAXIMUM AND MINIMUM
C
      M=0
C        INITIALIZE XMAX AND XMIN
      DO 630 LD=1,NF
      XMAX(LD)=ARAY(1,LD)
      XMIN(LD)=ARAY(1,LD)
630   CONTINUE
      DO 20 LDS=1,NF
      DO 20 L=1,NPT
      IF(ARAY(L,LDS).GT.XMAX(LDS)) XMAX(LDS)=ARAY(L,LDS)
      IF(ARAY(L,LDS).LT.XMIN(LDS)) XMIN(LDS)=ARAY(L,LDS)
20    CONTINUE
C        CHECK FOR ZERO ARRAYS
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
      DO 640 NSF=1,NF
      IF(XMAX(NSF).EQ.0..AND.XMAX(NSF).EQ.XMIN(NSF)) WRITE(IO,21)NSF
640   CONTINUE
21    FORMAT(' ALL POINTS IN THE',I3,' GRAPH OF THIS PLOT ARE ZERO')
      IF(ISCALE.EQ.3) GO TO 2
156   DIV=(BHI-BLOW)/(NPT-1)
      IF(ISCALE.NE.1) GO TO 3
      FAC=1
      XMIN2=2.**40
      XMAX1=-XMIN2
      DO 650 JL=1,NF
      IF(XMAX(JL).GT.XMAX1)XMAX1=XMAX(JL)
      IF(XMIN(JL).LT.XMIN2)XMIN2=XMIN(JL)
650   CONTINUE
      XDIF=XMAX1-XMIN2
25    DO 22 L=1,3
      IF((XDIF/XDIV(L)).LE.100.) GO TO 23
22    CONTINUE
      FAC=FAC*10.
      XDIF=XDIF/10.
      GO TO 25
23    XSCALE=FAC*XDIV(L)
      IF(XSCALE.GT.1.) GO TO 28
      IF(XSCALE.EQ.1..AND.XDIF.GT.50.) GO TO 28
      DO 29 LL=1,7
      DO 26 L=1,3
      IF(XDIF*XDIV(L).GT.100.) GO TO 27
26    CONTINUE
      FAC=FAC*10.
29    XDIF=XDIF*10.
27    IF(L.EQ.1) L=4
      IF(L.EQ.4.AND.FAC.NE.1.) FAC=FAC/10.
      XSCALE=1./(FAC*XDIV(L-1))
28    CONTINUE
      XMIN2=XMIN2/XSCALE
      XMIN2=INT(XMIN2/10.+(SIGN(1.,XMIN2)-1.)/2.)*10.*XSCALE
      DO 30 L=1,11
30    XMIN1(L)=XMIN2+FLOAT(L-1)*10.*XSCALE
      WRITE(IO,31) XMIN1
31    FORMAT(5X,G12.5,2X,9(G9.2,1X),G12.5)
      WRITE(IO,32)
32    FORMAT(13X,21('I....'),'I.I')
      XVAL=BLOW-DIV
      JO=1
C*
C*       HERE WE LOOP OVER THE POINT PLOTTING ONE LINE AT A TIME
C*       IARAY IS THE ARRAY INDEX OF VAL( ) WHERE A SYMBOL SHOULD BE
80    DO 40 L=1,NPT
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
      DO 800 LS=1,106
800   VAL(LS)=BLANK
      IF(ISCALE.EQ.3) GO TO 500
      XVAL=XVAL+DIV
      GO TO 510
500   VALOG=VALOG+DIV
      XVAL=10.**(VALOG)
510   CONTINUE
      DO 700 JZ=1,NF
      IARAY=(ARAY(L,JZ) - XMIN2)/XSCALE + 0.5
      IF(IARAY.GT.106) GO TO 700
      IF(IARAY.LT.0) IARAY = 0
      IF(JZ.GT.1)GO TO 730
      IERR=ABS(XARAY(L)/XSCALE)+0.5
      JERR=106-IERR
      KERR=IARAY-1
      IF(IARAY.NE.0)GO TO 42
      IF(IERR.EQ.0)GO TO 700
      GO TO 710
730   IF(IARAY.EQ.0)GO TO 700
42    VAL(IARAY)=POINT(JZ)
      IF(IERR.EQ.0.OR.JZ.GT.1)GO TO 700
710   LERR=IARAY-IERR
      IF(IERR.GE.IARAY)LERR=1
      JOHN=IARAY+IERR
      IF(JOHN.GT.106)JOHN=106
      KERR1=IARAY+1
      DO 720 JTZ=LERR,KERR
      VAL(JTZ)=DASH
720   CONTINUE
      DO 760 JTY=KERR1,JOHN
      VAL(JTY)=DASH
760   CONTINUE
700   CONTINUE
      WRITE(IO,44)XVAL,VAL,ARAY(L,1)
44    FORMAT(1X,E11.4,1X,'I',106A1,'I',1X,E11.4)
40    CONTINUE
      GO TO (501,502,503,504,505),JO
501   WRITE(IO,32)
      GO TO 810
502   WRITE(IO,106)
      GO TO 810
503   WRITE(IO,109)
      GO TO 810
504   WRITE(IO,113)
      GO TO 810
505   WRITE(IO,116)
810   CONTINUE
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
      IF( ISCALE .EQ. 1) GO TO 1000
C*       RETURN THE VALUES TO NORMAL SPACE
      DO 965 K=1,NPT
        XARAY(K) = 10.0**(XARAY(K))
965    CONTINUE
      DO 975 L=1,NF
        DO 975 K=1,NPT
         ARAY(K,L) = 10.0**(ARAY(K,L))
975      CONTINUE
1000  CONTINUE
      RETURN
C
C     LOG SCALE
C
3     DO 900 MAA=1,NF
      IF(XMIN(MAA).LE.0.) GO TO 110
900   CONTINUE
      XVAL=BLOW-DIV
150   CONTINUE
      DO 100 L=1,NPT
       IF( XARAY(L) .LE. 0 ) XARAY(L) = 1.0
       XARAY(L) = ALOG10(XARAY(L))
      DO 100 LL=1,NF
100   ARAY(L,LL)=ALOG10(ARAY(L,LL))
      XMIN2=2.**40
      XMAX1=-XMIN2
      DO 910 JL=1,NF
      IF(XMAX(JL).GT.XMAX1)XMAX1=XMAX(JL)
      IF(XMIN(JL).LT.XMIN2)XMIN2=XMIN(JL)
910   CONTINUE
      XMAX1=ALOG10(XMAX1)
      XMIN2=ALOG10(XMIN2)
      XDIF=XMAX1-XMIN2
      IXDIF=INT(XDIF)+1
      IXMIN=INT(XMIN2+(SIGN(1.,XMIN2)-1.)/2.)
      IF( IXDIF .GT. 5 ) GO TO 10000
      GO TO (101,102,103,103,104),IXDIF
10000 CONTINUE
      IF (IXDIF.LE.10) GO TO 1500
C*       TO LARGE A RANGE OF Y VALUES
      WRITE(IO,45) IXDIF
45    FORMAT('  TO LARGE A RANGE ON THE Y AXIS, MAGNITUDE=',I3,'.GT.10')
1500  CONTINUE
      DO 105 L=1,11
105   XMIN1(L)=10.**(IXMIN+L-1)
      WRITE(IO,31) XMIN1
      WRITE(IO,32)
      XSCALE=0.1
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
      IXMIN=IXMIN*10
      J0=1
      GO TO 80
101   XMIN1(1)=10.**IXMIN
      XMIN1(11)=10.**(IXMIN+1)
      WRITE(I0,250) XMIN1(1),XMIN1(11)
250   FORMAT(9X,E8.1,26X,'2',19X,'3',11X,'4',9X,'5',6X,'6',
     *5X,'7',5X,'8',4X,'9',2X,E8.1)
      WRITE(I0,106)
106   FORMAT(13X,'I',29('.'),'I',19('.'),'I',11('.'),'I',
     *9('.'),'I',6('.'),'I',5('.'),'I',5('.'),
     *'I....I...I',5('.'),'I')
      XSCALE=0.01
      IXMIN=IXMIN*100
      J0=2
      GO TO 80
102   DO 107 L=1,3
107   XMIN1(L)=10.**(IXMIN+L-1)
      WRITE(I0,108)(XMIN1(L),L=1,3)
108   FORMAT(9X,E8.1,30X,E8.1,42X,E8.1)
      WRITE(I0,109)
109   FORMAT(13X,2('I.............I.........I.....I....I',
     *'...I..I..I.I.'),'I.....I')
      XSCALE=1./50.
      IXMIN=IXMIN*50
      J0=3
      GO TO 80
103   DO 111 L=1,4
111   XMIN1(L)=10.**(IXMIN+L-1)
      WRITE(I0,112)(XMIN1(L),L=1,4)
112   FORMAT(9X,E8.1,3(18X,E8.1))
      J0=4
      WRITE(I0,113)
113   FORMAT(13X,4('I',24('.')),'I.....I')
      XSCALE=1./25.
      IXMIN=IXMIN*25.
      GO TO 80
104   DO 114 L=1,5
114   XMIN1(L)=10.**(IXMIN+L-1)
      WRITE(I0,115)(XMIN1(L),L=1,5)
115   FORMAT(9X,E8.1,4(13X,E8.1))
      J0=5
      WRITE(I0,116)
116   FORMAT(13X,5('I',19('.')),'I.....I')
      XSCALE=1./20.
      IXMIN=IXMIN*20
      GO TO 80
C
```

Program PLOTSEN CONT'D.
SUBROUTINE PLOT

```
C     LOG-LOG SCALE
C
2     DO 950 ND=1,NF
      IVAL = ND
      IF(XMIN(ND).LE.0.) GO TO 110
950   CONTINUE
      IF(BLOW.LE.0.) GO TO 120
      VALOG=ALOG10(BLOW)
      VBLOG=ALOG10(BHI)
      DIV=(VBLOG-VALOG)/NPT
      VALOG=VALOG-DIV
      GO TO 150
110   WRITE(IO,155) NAME(IVAL)
155   FORMAT(' PLOT OF ',A10,' CONTAINS NEGATIVE VALUES',/,
     *' AND WILL BE DONE WITH A LINEAR SCALE')
      ISCALE=1
      GO TO 156
120   WRITE(IO,159) BLOW
159   FORMAT(' THE LOWER BOUND =',E15.7,' IS NEGATIVE' )
      RETURN
      END
```

APPENDIX 9

4th Order Accurate WALSH Sequency Set

W	BINARY EXPANSION	No.
1	1	1
2	10	2
4	100	3
8	1000	4
16	10000	5
31	11111	6
32	100000	7
64	1000000	8
124	1111100	0
128	10000000	10
256	100000000	11
496	111110000	12
512	1000000000	13
1024	10000000000	14
1849	11100111001	15
1984	11110000000	16
2048	100000000000	17
2341	100100100101	18
2730	101010101010	19
3699	111001110011	20
4004	111110100100	21

REFERENCES

REFERENCES

ACM Signum, (1979), 14 (2), 22.

Acton, Forman S., (1966), Analysis of Straight Lines, Dover, New York.

Ahmed, N. and Rao, K. R. (1975), Transforms for Digital Signal Processing, Springer, New York.

Ainslie, G. R., Shill, J. P. and Neet, K. E., (1972), J. Biol. Chem., 247, 7080.

Andrews, H. C. and Caspari, K. L. (1970), IEEE Trans. Comp. 10, 16.

Bates, D. J. and Frieden, C. (1973), J. Biol. Chem. 248, 7878.

Beauchamp, K. G. (1975), Walsh Functions and Their Applications, Academic Press, New York.

Beck, J. V. and Arnold, K. J. (1977), Parameter Estimation in Engineering and Science, J. Wiley and Sons, New York.

Benacerraf, P. and Putnam, H. eds. (1964), Philosophy of Mathematics, Selected Readings, Prentice-Hall, New Jersey.

Carslaw, H. S. (1950), Introduction to the Theory of Fourier's Series and Integrals 3rd ed., Dover, New York.

Chrestenson, H. E. (1952), Pac. J. Math. 5, 17.

Cooley, J. W. and Tukey, J. W. (1965), Math. Comp. 19, 297.

Cukier, R. I., Fortuin, C. M., Schuler, K. E., Perschek, A. G. and Schaibly, J. H. (1973). J. Chem. Phys. 59, 3843.

Cukier, R. I., Schailby, J. H., Schuler, K. E. (1975), J. Chem. Phys. 63, 1140.

Cukier, R. I., Levine, H. B. and Schuler, K. E. (1978), J. Comp. Phys. 26, 1.

Dahlquist, G. and Bjorck, A. (1974), Numerical Methods, Prentice-Hall, New York.

De Boor, Carl (1978), A Practical Guide to Splines, Springer, New York.

Dickenson, R. P. and Gelinas, R. J. (1976), J. Comp. Phys. 21, 123.

Dye, J. L. and Nicely, V. A. (1971), J. Chem. Ed. 48, 443-448.

Fall, A., McRae, G., and Seinfeld, J. (1979), Int. J. Chem. Kinetics 11, 1137.

Fersht, A. (1977), Enzyme Structure and Mechanism, Freeman, New York.

Fine, N. J. (1949), Trans. Amer. Math. Soc., 65, 372.

Fine, N. J. (1955), Pacific J. Math. 5, 51.

Fine, N. J. (1955), Pacific J. Math. 5, 61.

Garfinkel D. and Marbach, C. B. (1977), Ann. Rev. Biophys. Bioeng. 6, 525.

Gear, C. W. (1971), Numerical Initial Value Problem in Ordinary Differential Equation, Prentice-Hall, New Jersey.

Hamming, R. W. (1973), Numerical Methods for Scientists and Engineers, McGraw-Hill, New York.

Hill, R. (1925), Proc. R. Soc. B100, 419.

Hindmarsh, A. C. (1977), L. L. L. Report, UCRL-51186, Rev. 1.

Ho Guan (1976), Ph.D. Thesis, Michigan State University.

International Mathematical and Statistical Libraries (1977), Houston, Texas.

Jerri, A. J. (1977), Proceedings of the IEEE 65, 1565.

June, D. S., Kennedy, ., Pierce, T. H., Elias, S. V., Halaka, F., Behbahani, Nejad, I., El-Bayoumi, A., Suelter, C. H., and Dye, J. L. (1979), J. Amer. Chem. Soc., 101, 2218-2219.

June, D. S., Dye, J. L. and Suelter, C. H. (1980), Biochemistry.

Kak, S. C. (1970), Electronic Letters, 6.

Kanyar, B. (1978), Acta Biochem. et Biophys. Acad. Hung. 13, 153.

Knuth, Donald (1973), The Art of Computer Programming, Sorting and Searching, Vol. 3, Addison-Wesley, San Francisco.

Koshland, D. E., Nemethy, G. and Filmer, D. (1966), Biochemistry 5, 365.

Kunz, H. O. (1979), IEEE Trans. on Comp. C-28,

Lanczos, C. (1955), Applied Analysis, Prentice-Hall, New York.

Lackey, R. B. and Meltzer, D. (1971), IEEE Trans. on Comp. C-20, 211.

Levine, H. (1974), "Sensitivity Analysis of a Linear Programming Model of Petroleum Economics", Systems, Science, and Software Report SSS-R-722526.

Michaelis, L. and Menten, M. L. (1913), Biochem. Z. 49, 333.

Monod, J., Wyman, J. and Changeux, J. P. (1965), J. Molec. Biol. 12, 88.

Nordsieck, A. (1962), Math. of Computation 16, 22.

Polge, R. J. and Bhagavan, B. K. (1976), IEEE Trans. on Comp. , 534.

Redlich, O. (1972), J. Chem. Ed., 49, 222.

Schailby, J. H., and Schuler, K. E. (1973), J. Chem. Phys. 59, 3879.

Segel, I. (1975), Enzyme Kinetics, J. Wiley and Sons, New York.

Shampine, L. F. and Gear, C. W. (1979), SIAM Review 21, 1-17.

Singleton, R. C. (1967), Comm. of the ACM 10, 647-654.

Singleton, R. C. (1968)., Comm. of the ACM 11, 773-779.

Tomovic, R. and Vukobratovic, M. (1972), General Sensitivity Theory, American Elsevier, New York.

Walsh, J. L. (1923), Amer. J. Math. 45, 5-24.

Weyl, Hermann, (1938), Amer. J. Math. 60, 887.

Whitehead, E. (1970), Prog. in Biophys. and Mol. Biol. 21, 321.

Zygmund, A. (1959), Trignometric Series 2ed Vol. 1, Cambridge University Press, New York.