SOME TECHNIQUES OF DATA ANALYSIS

by

JEREMY HOWARD NUSSBAUM

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1979

Signature of Author.....
Department of Electrical Engineering and Computer Science,
May 24, 1979

Certified by.....
Thesis Supervisor

Accepted by.....
Chairman, Departmental Committee on Theses

SOME TECHNIQUES OF DATA ANALYSIS

by

Jeremy Howard Nussbaum

Submitted to the Department of Electrical Engineering and Computer Science on My 24, 1979 in partial fulfillment of the requirements for the degree of Bachelor of Science.

ABSTRACT

Most theories are tested by experimentation. Often statistical analysis must be applied in order to fully understand the fit between theory and experiment. This thesis examines a number of techniques, including a grid search, gradient search and linearization approach, that can be used to examine the fit between theory and experiment. A case history involving experimental time constants from a series of electromechanochemical step-response experiments is examined in detail. These time constants were then compared to a non-linear diffusion-reaction theory. A powerful generalized curve fitting subsystem written in APL that was developed do to much of the curve fitting is described and documented in an appendix.

Thesis Supervisor: Alan J. Grodzinsky

Title: Associate Professor of Electrical and Bioengineering

ACKNOWLEDGEMENTS

Without the help of many people, I would not have finished this thesis. These people, both those named here and those who remain anonymous, desired my whole-hearted thanks and appreciation.

Professor Alan J. Grodzinsky stands out as the primary mover of this thesis Valuable time and effort were taken from his many other pursuits and duties and given to me to help put together this thesis. I would have been at a loss without the guidance given to me by Professor Al. I enjoyed some enlightening discussions with my office-mate, Sol Eisenberg. The modified linear regression technique received its severest test at his hands. I had some interesting discussions with Dr. Raphael Lee, who also helped out on the entire project that I worked on.

While the theoretical and experimental aspects of this thesis were done with the assistance of those mentioned above, the actual work and effort to type up and proofread this thesis required additional assistance. Aisha Haneef efficiently typed up this thesis. Much of the proof-reading was done by Andy Schwartz.

Of course, a special mention is due my parents, without whom I'd never have worked on this thesis. Their patience with me over the years has enabled me to get as far as I've gotten, and their willingness to let me try to work on my own is greatly appreciated. They have certainly put a tremendous amount of work into me over the years.

Table of Contents

List of Figures

List of Tables

I. Introduction

There are two phases to most scientific research. The more glamorous one involves formulating new and novel theories to explain observed phenomena. The more painstaking one involves the verification of the new theory through experiments and statistical analysis. This thesis tries to characterize certain aspects of the second phase of research. Several methods of data collection will be discussed, and the advantages and shortcomings of each method will be presented. Sources of error will be discussed, and ways of dealing with them will be presented.

In chapter two, a detailed discussion of statistical methods ensues. Standard curve-fitting techniques are explained, and more sophisticated and general methods are presented and discussed. An adaption of an existing set of programs into an APL subsystem and their use is presented in appendix C. Chapter three is a case history which illustrates the ideas discussed in the first two chapters. Data from electromechanical experiments done on protein membranes [1] was collected using different methods and compared. Different data analysis techniques were applied to sample sets of data. The accuracy and expense of these different techniques were then compared.

Finally, suggestions are made for subsystems on mini-

computer systems to make data analysis and data collection easier. These include hardware and software features, and the interface between the two. These features will enable one to more easily collect digitized data from experiments in a controlled (e.g. timed or event triggered) manner, and to analyze it (e.g. curve fit or match to an expected distribution) in a statistically sound and consistent manner.

In most experiments, some form of data which is a function of one or more independent variables is obtained. This data can be recorded continuously (e.g. output to a strip chart recorder), or sampled at discrete times (e.g. read off of a meter every 5 seconds). In the experiments of interest, the independent variable is time, but it could just as easily be voltage or concentration.

Each method of recording data has its advantages and disadvantages. By the continuous recording of data, no information is lost, and one gets a very accurate picture of what is occurring. On the other hand, it is difficult to further analyze the data. Analog electronics can be used in certain specific instances, but not in the general case. Noise especially complicates the interpretation of continuous data.

Discrete data, which is the more common form, loses a certain amount of information. In theory, if the data is sampled faster than the Nyquist rate ($F_{sample} > 2 F_{max}$) then no information is lost.

There is often noise that appears in the data along with the phenomenon being measured. The many sources of this noise include,

1. Uncertainty in the independent variable (e.g. time).
2. Random noise in the measured quantity (e.g. voltage).
3. Non-linearities in the measuring system.
4. Finite resolution in the recording instrument (e.g. digital voltmeter).
5. Transients in the experiments (i.e. short-time artifacts of the experiment).
6. Drift (i.e. long-time artifacts of the experiment).

As a case study, a set of experiments were done to determine the 1/e time constant of the change in force exerted by a collagen membrane associated with a step jump in bath pH. During the experiment, a voltage corresponding to force was sampled at regular intervals.

A number of sources of noise were found in the data collected from these experiments. The membrane itself had a certain amount of fluctuation. The non-linearities and sources of noise in the measuring apparatus, particularly the amplification system, made it difficult to extract a time con-

stant from the raw data. During the first seconds of the experiment, many non-equilibrium processes, such as solution mixing, were occurring. This seemed to cause the first few data points on many experiments to deviate from simple exponential curve. In addition, long-term baseline drift was often observed.

Many techniques have been devised to combat problems such as these. In the analog realm, filtering is widely used. This can attenuate selected frequency ranges. High frequency noise is particularly susceptible to filtering.

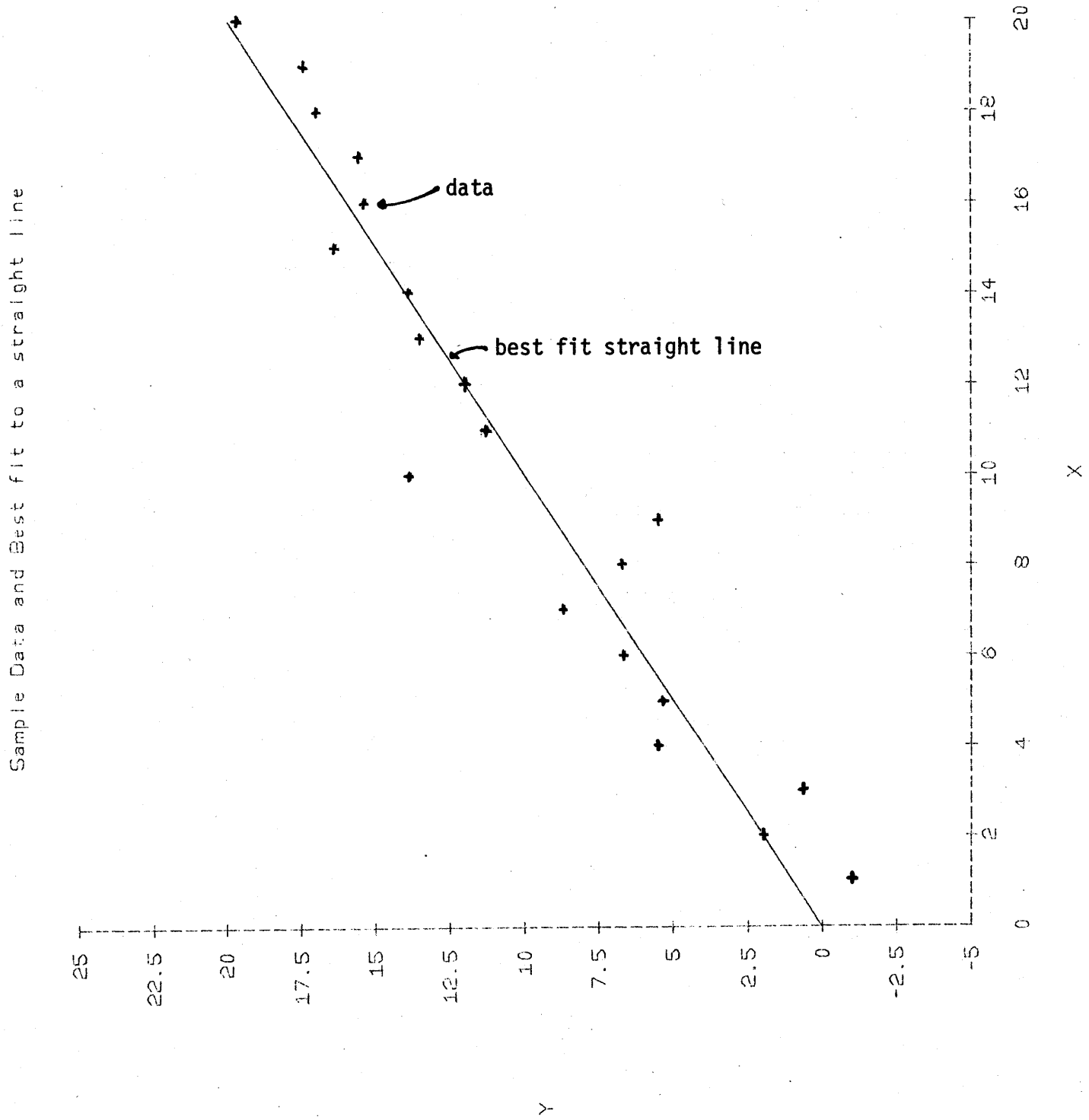
Filtering can also be done with discrete data. However, even more powerful techniques can be used to extract the desired result from the raw data. Statistical analysis can be used both to obtain the desired result, and to estimate its reliability.

Some experiments and studies yield data that can be interpreted only through the use of statistics. For example, the desired result might be a probability density function, with the values of the parameters in the function to be determined from the experiment. The probability of getting that particular set of results, given the calculated probability density function can also be determined.

Most experiments have noise that can be modelled as "random", i.e. following a normal distribution. The desired output is the total signal minus the noise. Since the noise is randomly distributed, the desired output (e.g. an exponen-

tial curve) is in some sense the mean, or average of the data (see figure 1). In the next chapter this model will be developed, and some methods of curve fitting will be presented.

Figure 1 Sample data and best fit straight line. While the sum of the deviations from the straight line is about 0, the sum of the squared deviations is about 1.5.



Chapter II

Curve Fitting Techniques

Without sources of experimental error, there would be very little need for sophisticated techniques of data analysis. A few calculations would have been sufficient to produce a perfect fit of data to theory. Unfortunately, the comparison of experimental data and theoretical curves often requires statistical analysis to better interpret the data.

All of the techniques of curve fitting described here rely on one major assumption: the data deviates from its expected value due to normally distributed random fluctuations. In other words, the errors have a normal or Gaussian distribution, and each error is independent of the other errors.

The probability of the i^{th} observation is [2]

$$P_i = \frac{1}{\sigma_i \sqrt{2\pi}} \; e^{-\frac{1}{2}\left(\frac{y_i - \bar{y}_i}{\sigma_i}\right)^2} \qquad (1)$$

where

σ_i = the standard deviation of the i^{th} observation

y_i = the actual i^{th} observation

$\bar{y}_i$ = predicted value for the i^{th} observation.

In general, σ_i will not be known exactly. σ_i may be

due to a lack of precision in the recording device, a particularly sloppy experiment, or problems with a piece of equipment. It can usually be estimated. The importance of σ_i is its relative value from observation to observation.

We now want a relation for the probability of the entire set of observations. Since the errors are independent of each other, we can take the product of the probability of each observation to obtain the desired probability.

$$P(\bar{x}) = \prod_{i=1}^{N} P_i = \frac{1}{\sqrt{2\pi}} \prod_{i=1}^{N} \frac{1}{\sigma_i} e^{-\frac{1}{2}\left(\frac{y_i-\bar{y}_i}{\sigma_i}\right)^2} \qquad (2a)$$

$$= \frac{1}{\sqrt{2\pi}} e^{\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i-\bar{y}_i}{\sigma_i}\right)^2\right]} \prod_{i=1}^{N}\left(\frac{1}{\sigma_i}\right) \qquad (2b)$$

Note that $\bar{y}_i$ need not be a constant. It can be a function of an independent variable (e.g. time). We are interested only in deviations from $\bar{y}_i$, the predicted value of each observation.

The goal is to maximize the probability of the set of observations. This is done by minimizing the argument of the exponential in equation (2b). It is usual to call this expression, which represents the goodness of the fit, χ^2.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - \bar{y}_i}{\sigma_i} \right)^2 \qquad (3)$$

Calculating the average

To illustrate the meaning and usage of equation (3), we will look at a simple case. Let $\bar{y}$ be a single constant, $\bar{Y}$ (also known as the mean or average). To maximize the probability, we minimize χ^2 in equation (3). To do this, we take the derivative of χ^2 with respect to $\bar{Y}$, which is the only quantity which we can vary, and set it to 0.

$$\frac{\partial \chi^2}{\partial \bar{Y}} = -\sum_i 2 \frac{(y_i - \bar{Y})}{\sigma_i^2} = 0 \qquad (4)$$

Solving for $\bar{Y}$,

$$\bar{Y} = \frac{\sum \frac{y_i}{\sigma_i^2}}{\sum \frac{1}{\sigma_i^2}} \qquad (5)$$

In the special case of a constant standard deviation from experiment to experiment, equation (5) reduces to the common expression for the arithmetic mean.

$$\bar{Y} = \frac{\sum y_i}{N} \qquad (6)$$

Linear Regression

We can now look at a more complicated case, in which $\bar{y}$ is a "linear" function of some independent variable x. (Technically, $\bar{y}$ is not a linear function of x because $\bar{y}(x_1 + x_2) \neq \bar{y}(x_1) + \bar{y}(x_2)$, but the result of this analysis is commonly called linear regression.) Equation (7) describes such a relationship. While x_i can assume different values for different i, it must be known.

$$\bar{y}_i = a + bx_i \tag{7}$$

We now proceed to minimize χ_2, defined in equation (3), using $\bar{y}$ as defined in equation (7). Since both a and b are to be determined, the partial derivatives of χ^2 with respect to both a and b are set to 0.

$$\frac{\partial x^2}{\partial a} = -2\Sigma\left(\frac{y_i - (a + bx_i)}{\sigma_i^2}\right) = 0 \tag{8a}$$

$$\frac{\partial x^2}{\partial b} = -2\Sigma\left(\frac{(y_i - a + bx_i)\,x_i}{\sigma_i^2}\right) = 0 \tag{8b}$$

In matrix form, these equations can be rewritten

$$\begin{bmatrix} \sum \frac{1}{\sigma_i^2} & \sum \frac{x_i}{\sigma_i^2} \\ \sum \frac{x_i}{\sigma_i^2} & \frac{\sum x_i^2}{\sigma_i^2} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum \frac{y_i}{\sigma_i^2} \\ \sum \frac{x_i y_i}{\sigma_i^2} \end{bmatrix} \qquad (9a\text{-}b)$$

with the solutions

$$a = \frac{1}{\Delta}\left(\sum \frac{x_i^2}{\sigma_i^2} \sum \frac{y_i}{\sigma_i^2} - \sum \frac{x_i}{\sigma_i^2} \sum \frac{x_i y_i}{\sigma_i^2} \right) \qquad (10a)$$

$$b = \frac{1}{\Delta}\left(\sum \frac{1}{\sigma_i^2} \sum \frac{x_i y_i}{\sigma_i^2} - \sum \frac{x_i}{\sigma_i^2} \sum \frac{y_i}{\sigma_i^2} \right) \qquad (10b)$$

$$\Delta = \sum \frac{1}{\sigma_i^2} \sum \frac{x_i^2}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i^2} \right)^2 \qquad (10c)$$

In the special case of equal σ_i's, equations (10a-c) reduce to

$$b = \frac{N \Sigma Y_i X_i - \Sigma Y_i \, \Sigma X_i}{N \Sigma X_i^2 - (\Sigma X_i)^2} \qquad (11a)$$

$$a = \frac{\Sigma Y_i - b\ \Sigma X_i}{N} \tag{11b}$$

We would like to know how reliable the assumption that $\bar{y}_i = a + bx_i$ is, and also how accurate a and b are. The correlation coefficient tells us how strong the correlation between x and y actually is.

If y is "linearly" related to x, then x should be linearly related to y.

$$x = a' + b'y \tag{12}$$

The values of a' and b' will be different from a and b in equation (7). However, if x and y are indeed related, then a' and b' are related to a and b. First, solve (12) for y.

$$y = \frac{x}{b'} - \frac{a'}{b'} \tag{13}$$

and relate the coefficients to equation (7)

$$a = -\frac{a'}{b'} \tag{14a}$$

$$b = \frac{1}{b'} \tag{14b}$$

In the ideal case, then, $bb' = 1$. If x and y are not correlated, then $bb' = 0$. The correlation coefficient is

defined to be $\sqrt{bb'}$. First, calculate b' from equations (12) and (10a)

$$b' = \frac{1}{\Delta'} \left(\sum \frac{1}{\sigma_i^2} \sum \frac{x_i y_i}{\sigma_i^2} - \sum \frac{y_i}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2} \right) \tag{15a}$$

$$\Delta' = \sum \frac{1}{\sigma_i^2} \sum \frac{y_i^2}{\sigma_i^2} - \left(\sum \frac{y_i}{\sigma_i} \right)^2 \tag{15b}$$

The correlation coefficient is then

$$r = \sqrt{bb'} = \frac{\sum \frac{1}{\sigma_i^2} \sum \frac{x_i y_i}{\sigma_i^2} - \sum \frac{y_i}{\sigma_i^2} \sum \frac{x_i}{\sigma_i^2}}{\left(\left[\sum \frac{1}{\sigma_i^2} \sum x_i^2 - \left(\sum x_i^2 \right)^2 \right] \left[\sum \frac{1}{\sigma_i^2} \sum y_i^2 - \left(\sum y_i^2 \right)^2 \right] \right)^{1/2}} \tag{16}$$

The closer the absolute value of r is to 1, the better the degree of correlation. The probability density function of r is [3]

$$P_r(r,\nu) = \frac{1}{\sqrt{\pi}} \frac{\Gamma\left[(\nu+1)/2\right]}{\Gamma\left[(\nu/2)\right]} (1-r^2)^{(\nu-2)/2} \tag{17}$$

where ν = number of degrees of freedom for the data, which is equal to the number of data points minus the number of parameters in the fit-

ting function, determined from the data.

and the probability that a sample of N uncorrelated data points would yield a correlation coefficient as large or larger than $|r|$ is

$$P_c(r\nu) = 2 \int_{|r|}^{1} P_r(r', \nu)\, dr' \qquad (18)$$

where $\nu = N-2$.

A small value for P_c means that x and y are actually correlated.

The variance, which is the square of the standard deviation, σ_i, can be approximated by equation (19) for normal distributions

$$\sigma^2 = \frac{1}{\nu} \Sigma (y_i - \bar{y}) \qquad (19)$$

It appears in the normal distribution probability density function, equation (1). The larger the standard deviation, the larger the spread of values to be expected. The probable error is .6745 σ; half of the observations are expected to be within $\bar{y} \pm 0.6745\ \sigma$. Over two thirds (68.3%) of the observations should be within a single standard deviation of the expected value. [2]

To characterize the uncertainties in the parameters a

and b determined in equations (10a)-(10c), we can derive expressions for σ_a and σ_b, defined as

$$\sigma_a^2 = \frac{\Sigma(a_i-\bar{a})^2}{N} \approx \Sigma \frac{\left[(y_i-\bar{y}_i)\frac{\partial a}{\partial y}\right]^2}{N} = \Sigma\sigma_{yi}^2\left(\frac{\partial a}{\partial y}\right)^2 \quad (20a)$$

$$\sigma_b^2 = \Sigma\sigma_{yi}^2\left(\frac{\partial b}{\partial y}\right)^2 \quad (20b)$$

Equations (10a) and (10b) are used to determine the partial derivatives.

$$\frac{\partial a}{\partial y_j} = \frac{1}{\Delta}\left(\frac{1}{\sigma_j^2}\sum_i \frac{x_i^2}{\sigma_i^2} - \frac{x_j}{\sigma_j^2}\sum_i \frac{x_i}{\sigma_i^2}\right) \quad (21a)$$

$$\frac{\partial b}{\partial y_j} = \frac{1}{\Delta}\left(\frac{x_i}{\sigma_j^2}\sum_i \frac{1}{\sigma_i^2} - \frac{1}{\sigma_j^2}\sum_i \frac{x_i}{\sigma_i^2}\right) \quad (21b)$$

Using equations (21) in (20), we get

$$\sigma_a^2 = \frac{1}{\Delta^2}\sum_j \sigma_j^2\left[\frac{1}{\sigma_j^2}\left(\sum_i \frac{x_i^2}{\sigma_j^2}\right) - \frac{x_j}{\sigma_j^2}\sum_i \frac{x_i}{\sigma_i^2}\right]^2 \quad (22a)$$

$$= \frac{1}{\Delta^2}\sum_j \sigma_j^2\left[\frac{1}{\sigma_j^4}\left(\sum \frac{x_i^2}{\sigma_i^2}\right)^2 - 2\frac{x_j}{\sigma_j^4}\sum_i \frac{x_i^2}{\sigma_i^2}\sum_i \frac{x_i}{\sigma_i^2} + \right.$$

$$\frac{x_j^2}{\sigma_j^4} \left(\sum_i \frac{x_i}{\sigma_i^2} \right)^2 \Bigg] \tag{22b}$$

$$= \frac{1}{\Delta^2} \left[\sum_i \frac{1}{\sigma_i^2} \left(\sum \frac{x_i^2}{\sigma_i^2} \right)^2 - 2 \sum_i \frac{x_i}{\sigma_i^2} \sum_i \frac{x_i^2}{\sigma_i^2} \sum_i \frac{x_i}{\sigma_i^2} + \right.$$

$$\left. \sum_i \frac{x_i^2}{\sigma_i^2} \left(\sum_i \frac{x_i}{\sigma_i^2} \right)^2 \right] \tag{22c}$$

$$= \frac{1}{\Delta^2} \left[\sum \frac{x_i^2}{\sigma_i^2} \left(\sum \frac{x_i^2}{\sigma_i^2} \sum \frac{1}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i^2} \right)^2 \right) \right] \tag{22d}$$

$$\sigma_a^2 = \frac{1}{\Delta} \sum \frac{x_i^2}{\sigma_i^2} \tag{22e}$$

Similarly,

$$\sigma_b^2 = \frac{1}{\Delta^2} \sum_j \sigma_j^2 \left[\frac{x_j}{\sigma_j^2} \sum \frac{1}{\sigma_i^2} - \frac{1}{\sigma_j^2} \sum \frac{x_i}{\sigma_i^2} \right]^2 \tag{23a}$$

$$= \frac{1}{\Delta^2} \sum_j \left[\frac{x_j^2}{\sigma_j^2} \left(\sum_i \frac{1}{\sigma_i^2} \right)^2 - 2 \frac{x_j}{\sigma_j^2} \sum_i \frac{1}{\sigma_i^2} \sum_i \frac{x_i}{\sigma_i^2} + \right.$$

$$\frac{1}{\sigma_j^2}\left(\sum \frac{x_i}{\sigma_i^2}\right)^2\Bigg] \tag{23b}$$

$$= \frac{1}{\Delta^2}\left[\sum \frac{x_i^2}{\sigma_i^2}\left(\sum \frac{1}{\sigma_i^2}\right)^2 - 2\sum \frac{x_i}{\sigma_i^2}\sum \frac{1}{\sigma_i^2}\sum \frac{x_i}{\sigma_i^2} +\right.$$

$$\left.\sum \frac{1}{\sigma_i^2}\left(\sum \frac{x_i}{\sigma_i^2}\right)^2\right] \tag{23c}$$

$$= \frac{1}{\Delta^2}\sum \frac{1}{\sigma_i^2}\left[\sum \frac{1}{\sigma_i^2}\sum \frac{x_i^2}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i^2}\right)^2\right] \tag{23d}$$

$$\sigma_b^2 = \frac{1}{\Delta}\sum \frac{1}{\sigma_i^2} \tag{23e}$$

The determination of a and b in the above analysis is known as "linear regression." It enables one to fit data to a linear function of an independent variable plus a constant. It can readily be extended to many independent variables or functions of those independent variables, as long as they are linear functions of the coefficients to be determined, i.e.

$$y = a_0 + a_1 f_1(x_1, x_2\ldots) + a_2 f_2(x_1, x_2\ldots)\ldots \tag{24}$$

We can then take the partial derivatives of χ^2 with respect to all of the a_i's and set them to 0. A set of linear equations are the result, and these can always be solved.

Modified Linear Regression

Not all fitting functions are linear. In fact, in the analysis of experimental data very few fitting functions are linear. In a non-linear case, we can still proceed by taking the partial derivatives of χ^2 with respect to each of the parameters and setting them to 0, but in general we will not be able to solve the resulting set of non-linear equations. There are times, however, that the fitting function can be recast into a linear form.

Equation (25) is a simple case.

$$y = a_1^2 x + (a_2 - a_1) x^2 \tag{25}$$

A simple set of transformations enables linear regression techniques to be applied

$$a_1' = a_1^2 \tag{26a}$$

$$a_2' = a_2 - a_1 \tag{26b}$$

$$y = a_1' x + a_2' x^2 \tag{27}$$

After the primed coefficients are determined, the original

coefficients can easily be obtained. The uncertainty in the original coefficients can be approximated by equation (28)

$$\sigma^2_{a_1} \approx \sigma^2_{a_i} \left(\frac{\partial a_1}{\partial a_1'} \right)^2 \tag{28}$$

A more complicated example is shown in equation (29).

$$f = f_2 + (f_1 - f_2)\, e^{-t/\tau} \tag{29}$$

where f_1, f_2 and τ are constants to be determined.

By subtracting f_2 from both sides, multiplying by -1, and taking the natural log of both sides we get

$$\ell n(f_2 - f_1) = \ell n(f_2 - f_1)^{-t_i/\tau} \tag{30}$$

Let

$$y_i = \ell n(f_2 - f_i) \tag{31a}$$

$$x_i = t_i \tag{31b}$$

$$a = \ell n(f_2 - f_1) \tag{31c}$$

$$b = -1/\tau \tag{31d}$$

Then equation (30) becomes

$$y_i = a + bx_i \tag{32}$$

The standard deviations also must be transformed

$$\sigma_{yi}^2 = \sigma_{fi}^2\left(\frac{\partial y}{\partial f}\right)^2 = \sigma_{fi}^2\left(\frac{1}{f_2-f_i}\right)^2 \tag{33}$$

At this stage we are ready to apply standard linear regression techniques to the problem. Equations (10a-c) give us values for a and b, and equations (22e) and (23e) give us the uncertainty in a and b. Equations (31c-d) relate (f_2-f_1) to a, and τ to b. In a manner similar to equation (33), $\sigma_{f_2-f_1}$ can be related to σ_a, and σ_τ can be related to σ_b,

$$\sigma_{(f_2-f_1)} = \sigma_a e^a \tag{34a}$$

$$\sigma_\tau = \frac{\sigma_b}{b^2} \tag{34b}$$

This curve fitting technique was tried with an experimental case study (see Chapter 3) involving a set of data fit to a curve of the form (29). However, the dependent variable in equation (31) or (33) is $\ell n(f_2-f_i)$. This means that f_2 must be known a priori to use linear regression on this problem. In practice, however, knowledge of f_2 would necessitate an infinitely long-time experiment with no drift. Therefore, a first estimate of f_2 was used to determine f_1 and τ. Chi

squared was then calculated using equation (35):

$$\chi^2 = \sum_i \frac{\left[f_2+(f_1-f_2)e^{-t_i/\tau} - f_i\right]^2}{\sigma_{f_i}^2} \qquad (35)$$

using a value of $\sigma_{fi}^2 = 1$. A specified increment was added to f_2 and χ^2 was again calculated. If χ^2 increased the first time, the increment was multiplied by -1, and added to the original f_2. The increment was kept the same until χ^2 started to increase. At that point the increment was multiplied by -1/2. When the absolute value of the increment falls below a specified value, this binary search is halted (see appendix A.)

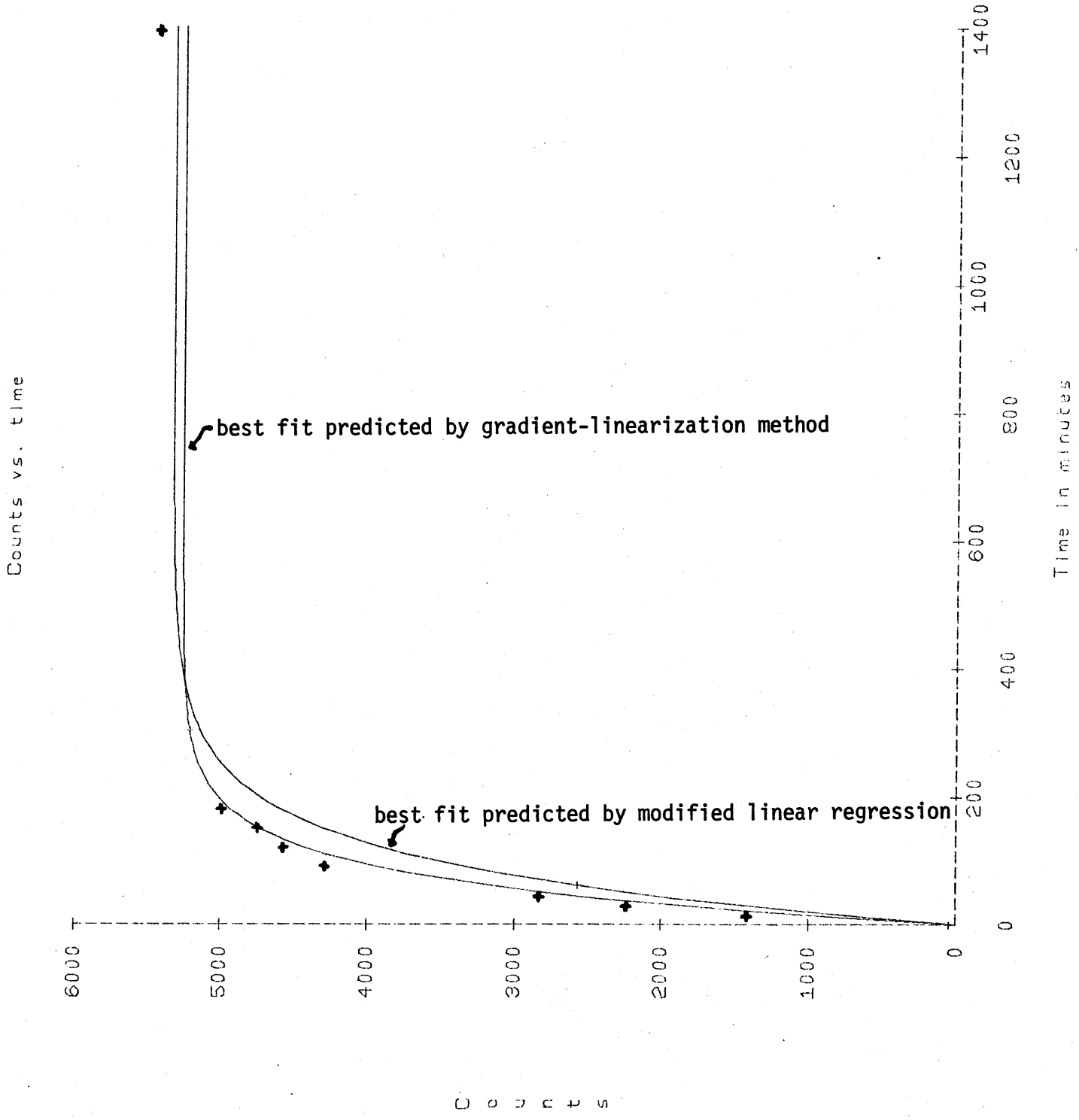
There are times that linear regression techniques can be unsatisfactory. Figure 2 compares experimental data with a best fit to equation (29) predicted by the modified linear regression techniques described above, and with a best fit predicted by the gradient-linearization technique described below.

Optimization of Each Parameter in Turn

Let us define the fitting function to be an arbitrary function of n parameters, a_1-a_n, and any number of independent parameters, $\vec{x}$.

$$y = y(a_1 \ldots a_n, \vec{x}) \qquad (36)$$

Figure 2 Data from a radioactive tracer study vs. time, along with the best fit predicted by the modified linear regression program and the best fit predicted by the gradient-linearization method.



We want to choose $a_1 \ldots a_n$ to minimize χ^2. We can take the partial derivative of χ^2 with respect to each parameter a_i, and set it to 0. In general, though, we will not be able to solve the resulting set of n non-linear equations.

A very simple approach is to vary one parameter until χ^2 is minimized with respect to it, given the values of the other parameters. This is done to each parameter in turn, and repeated until it can no longer reduce χ^2 very much. Simplicity is the virtue of this method. It requires many computations of χ^2 to converge, however. It will generally have difficulty zeroing in on the minimum of χ^2, because the best value of a parameter calculated each time is dependent on the current values of the other parameters rather than on the actual best values of the other parameters. Far away from the minimum, this method zig-zags back and forth as the parameters are slowly optimized.

Gradient Search

A more powerful approach is to calculate the gradient of χ^2 in n-space, and to travel down the gradient until a trough is reached. The gradient is then recalculated. The process is continued until χ^2 no longer changes.

The gradient of χ^2 in n space is the partial derivative of χ^2 with respect to each parameter a_j, in the $\hat{a}_j$ direction.

$$\nabla\chi^2 = \sum_{j=1}^{n} \frac{\partial \chi^2}{\partial a_j} \hat{a}_j \qquad (37)$$

We can make this quantity dimensionless by normalizing each component to the current magnitude of the parameter a_j.

$$\gamma \equiv \nabla(\chi^2) a_j = \Sigma \frac{\partial \chi^2}{\partial a_j} a_j \hat{a}_j \tag{38}$$

The normalized increment of $\Delta\chi^2$ in any direction, b_j, is then equal to the component of the normalized gradient in that direction, divided by the magnitude of the normalized gradient

$$b_j = \frac{-\frac{\partial \chi^2}{\partial a_j} a_j}{\left[\Sigma\left(\frac{\partial \chi^2}{\partial aj} a_j\right)^2\right]^{1/2}} \tag{39}$$

The actual increment to move in each direction will be the total increment times the factor calculated in (39).

$$\Delta a_j = b_j \delta_j \tag{40}$$

The program that implements this procedure moves in small increments along the gradient until χ^2 no longer diminishes. (See appendix B)

Far away from the minimum in χ^2, this method works quite well. Close to the actual minimum, however, the gradient becomes very small. Calculations involve subtracting almost equal numbers to determine the derivatives with respect to χ^2, so a good deal of precision is lost. Convergence near

the minimum is therefore very slow.

Linearization of the Fitting Function

A method which works well near the minimum of χ^2 involves linearizing the fitting function around the current set of parameters $a_1, a_2 \ldots a_n$.

$$y(a_1 + \delta_{a_1}, a_2 + \delta_{a_2} \ldots, \vec{x}) \approx y_0(a_1 \ldots a_n, \vec{x}) + \sum_j \delta_{a_j} \frac{\partial y(a_1 \ldots, \vec{x})}{\partial a_j} \qquad (41)$$

where y_0 is the current value of the fitting function. In this case, χ^2 can also be approximated as

$$\chi^2 \approx \sum_i \frac{1}{\sigma_i^2} \left[y_i - y_0(a_1 \ldots, \vec{x}_i) - \sum \frac{\partial y_0}{\partial a_j} (a_1 \ldots, \vec{x}_i) \delta a_j \right]^2 \qquad (42)$$

Let

$$y_i' = y_i - y_0(a_1 \ldots, x_i) \qquad (43a)$$

$$X'_{ji} = \frac{\partial y_0}{\partial a_j} (a_1 \ldots \vec{x}_i) \qquad (43b)$$

$$a_j' = \delta a_j \qquad (43c)$$

With these substitutions, equation (42) becomes

$$\chi^2 \approx \sum_i \frac{1}{\sigma_i^2} \left[y'_i - \sum_j a'_j x'_{ji} \right]^2 \qquad (44)$$

This is an example of multiple linear regression; the function, $y'_i(\vec{x}) = \sum_j a'_j x'_{ji}$, is linear in a'_j, the parameters to be determined. We can differentiate (44) with respect to each a'_j and set each derivative to 0. The resulting set of simultaneous linear equations can be solved for a'_j, giving us the desired result, δa_j.

$$\frac{\partial \chi^2}{\partial a'_k} = -2 \sum_i \left(\frac{1}{\sigma_i^2} \left[y'_i - \sum_j a'_j x'_{ji} \right] x'_{ki} \right) = 0 \qquad (45a)$$

or

$$\sum_i \frac{1}{\sigma_i^2} x'_{ki} \sum_j a'_j x'_{ji} = \sum_i \frac{y'_i}{\sigma_i^2} x_{ki} \qquad (45b)$$

Using matrix notation, we can rewrite (45b) as

$$a\alpha = \beta \qquad (46)$$

where

$$a = [a'_1 \quad a'_2 \ldots a'_n] = [\delta a_1 \quad \delta a_2 \quad \ldots \delta a_n] \qquad (47a)$$

$$\alpha_{jk} = \sum_i x'_{ki} x'_{ji} = \sum_i \frac{\partial y_{oi}}{\partial a_k} \frac{\partial y_{oi}}{\partial a_j} \qquad (47b)$$

$$\beta = \begin{bmatrix} \sum_i \frac{1}{\sigma_i^2} \, y_i' \, x_{1i}' \\ \vdots \\ \sum_i \frac{1}{\sigma_i^2} \, y_i' \, x_{ni} \end{bmatrix} = \begin{bmatrix} \sum_i \frac{1}{\sigma_i^2} \, (y_i - y_{0i}) \, \frac{\partial y_{0i}}{\partial a_1} \\ \vdots \\ \sum_i \frac{1}{\sigma_i^2} \, (y_i - y_{0i}) \, \frac{\partial y_{0i}}{\partial a_n} \end{bmatrix} \tag{47c}$$

The solution for a can be obtained by inverting the α matrix

$$a = \beta \alpha^{-1} \tag{48a}$$

or

$$a_j = \sum_k \beta_k \, \alpha_{jk}^{-1} \tag{48b}$$

Close to the minimum in x^2, this method converges both rapidly and accurately. Further away, it suffers from the inaccuracies resulting from the linearization of the fitting function. There, the gradient method works better.

An approach proposed by Marquardt [4] involves using a combination of the gradient method and the linearization approach. The normalized diagonal terms of the α matrix defined in equation (49b) are all 1's. Add a constant, which we will call λ, to the diagonal terms. If the constant is small, the lin-

earization equations are hardly affected. If λ is large, the equations decouple.

$$a'_j \lambda \alpha_{ij} \approx \beta_j \tag{49a}$$

or

$$\delta a_j \approx \frac{\sum_i \frac{1}{\sigma_i^2} (y_i - y_{0i}) \frac{\partial y_{0i}}{\partial a_j}}{\lambda \sum_i \left(\frac{\partial y_{0i}}{\partial a_j} \right)^2} = -\frac{\partial \chi^2}{\partial a_j} C \tag{49b}$$

where $C = \frac{1}{2\lambda\alpha_{ji}}$

and so δa_j points down the gradient, with a magnitude scaled by $2\lambda\alpha_{jj}$. The larger λ, the more of the gradient method that is included in the calculations.

$$\alpha'_{jk} = \begin{cases} \alpha_{jk} & k \neq j \\ \alpha_{jk}(1+\lambda) & k = j \end{cases} \tag{50}$$

The suggested approach was to initialize λ to 0.001. Each time the increments are chosen, χ^2 is checked to see if it decreased. If it did, indicating that the proportion of gradient search is at least sifficient, λ is decreased by a factor of 10, reducing the proportion of gradient search.

If χ^2 increased, indicating that more gradient search is needed, λ is increased by a factor of 10. Each time χ^2 decreases, the prescribed direction is followed until χ^2 no longer decreases. The entire process is repeated until χ^2 no longer gets smaller.

The uncertainty in the parameters cannot be calculated, since an analytical solution was not obtained to minimize χ^2. However, near the minimum in χ, the uncertainty can be approximated by

$$\sigma_{aj}^2 \approx \frac{1}{\alpha_{jj}} = \frac{2}{\frac{\partial^2 \chi^2}{\partial a_j^2}} \qquad (51)$$

This technique was seen to be extremely versatile and powerful. Unlike the gradient method alone, it converged to the same minimum for all reasonable starting values of $a_1 \ldots a_n$. Appendix C contains both a listing of the APL subsystem written to implement this algorithm and to make it easy to set up and use, and a description of how it was used.

Table 1 compares the different techniques of curve fitting. The asterisks denote methods which are easily implementable on a programmable calculator.

Technique	Advantages	Disadvantages
*Iteration	Simple to implement. Applicable to every case	Requires a good guess and the proper range to ever converge.
*Optimize each parameter in turn	Simple to implement Applicable to every case	Slow convergence Very slow if the parameters are not independent.
*Linear Regression	Quick Fairly accurate	Slight complication in programming. Works only on linear problems. Bombs out on certain cases.
*Modified Linear gression	Quick, Fairly accurate, works on some non-linear problems	Doesn't work on everything. The transformed standard deviations are not too accurate for relatively large errors.
Gradient search	Applicable to every case Converges very well from far away	More complicated to program. Strains the capability of a programmable calculator. Cannot quite "zero in" on the minimum.
Linearization of the fitting function.	Zeroes quickly into the minimum	Very complicated to program. Has trouble finding the right path from far away.
Combination of gradient search and linearization	Works on every problem. Converges well both from far away and close up. Finds the minimum independent of starting position	Very complicated to program.

*Implementable on a programmable pocket calculator.

Table 1

Chapter III

A Case History

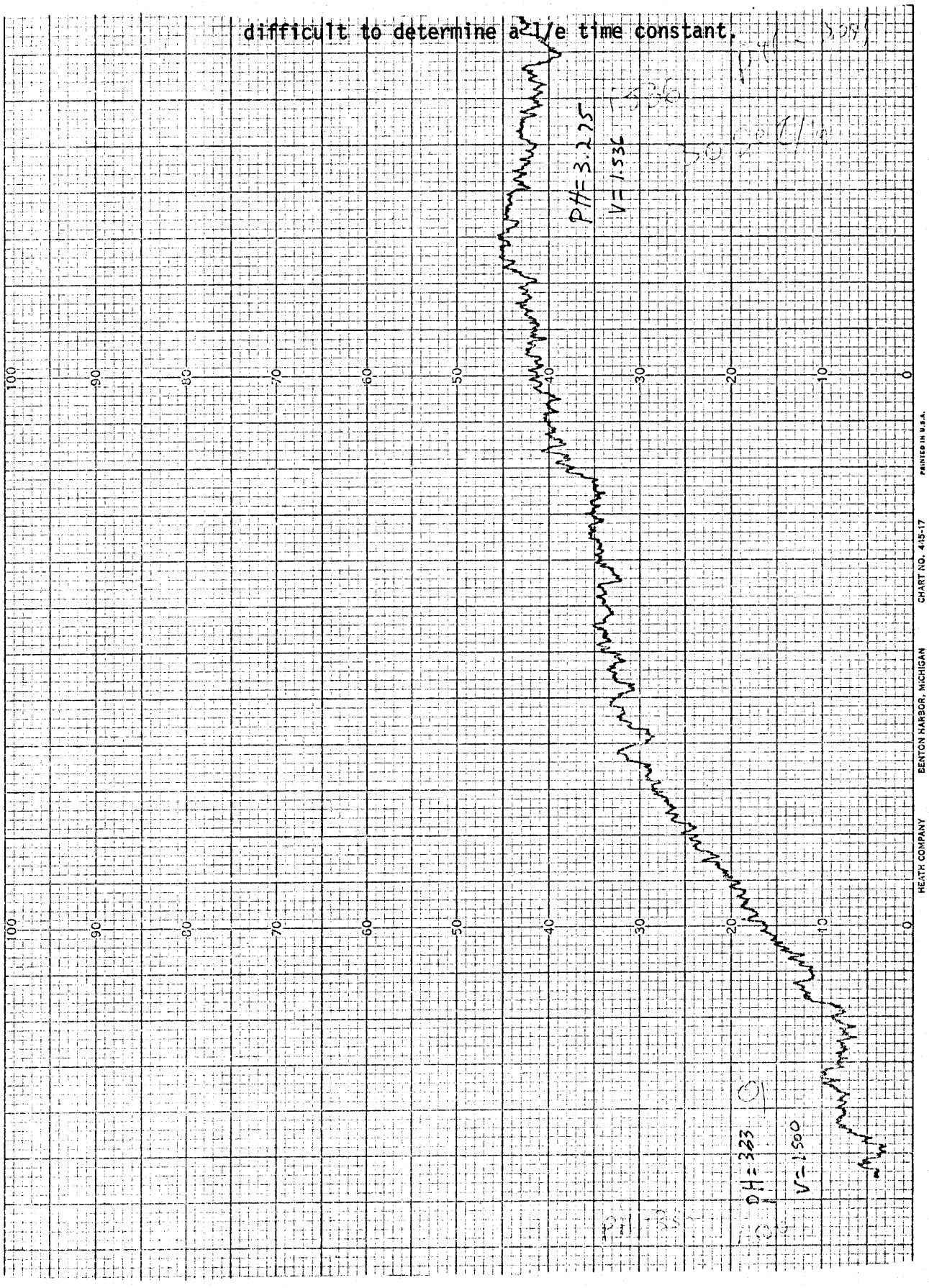
A case history illustrating the previous two chapters will now be presented. The methods used to analyze the data collected from some electromechanochemical experiments will be examined in detail.

The experiments were done to determine the 1/e time constants of the change in isometric force generated by a collagen membrane associated with a step jump in pH. They are described in detail elsewhere.[1] The force was converted into a proportional voltage by a load cell.

Collecting the Data

Originally, the output of the load cell was amplified and recorded by a strip chart recorder. It was found that too much noise was introduced by the apparatus (see figure [3]), and it was very difficult to determine a time constant. In order for the second level of curve fitting (which involved fitting the time constants of the experiments to a theory) to be meaningful, the time constants had to be reasonably accurate. A simple, passive low-pass filter was added, but did not filter out enough of the noise. Discrete sampling and data analysis techniques were then tried. These techniques may be useful for the analysis of data from many experiments attempting to characterize the chemical kinetics or the dynamics of

Figure 3 Sample data collected using an amplifier and strip chart recorder. The noise and drift make it very difficult to determine a 1/e time constant.



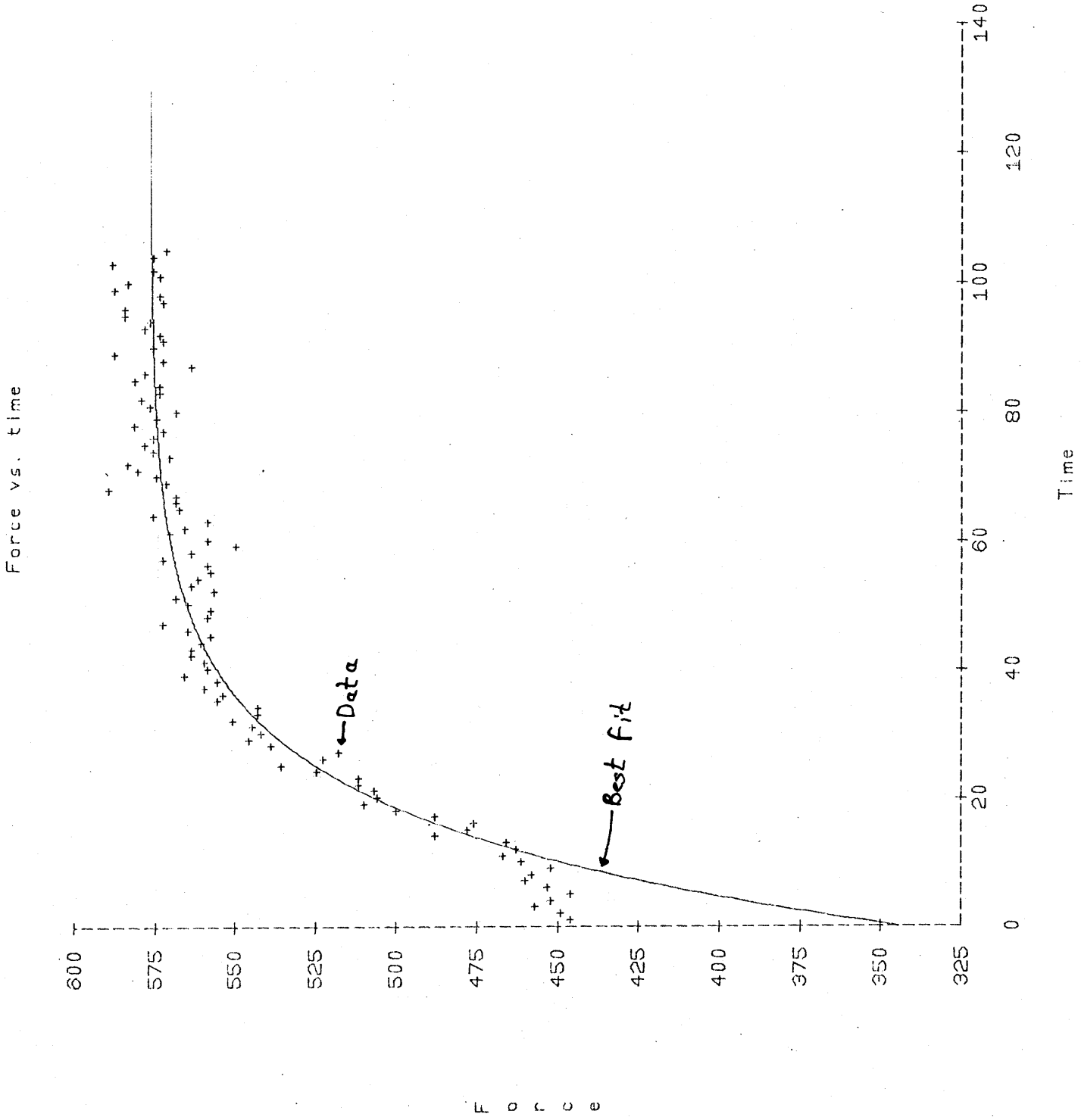
some process.

The most direct method of sampling the data was to manually read the voltage from a digital voltmeter. In a typical experiment, the voltage increased 40μV. The resolution of the voltmeter was 1 μV. Due to the physical limitations, the maximum sampling rate was .2 Hz, or 1 sample every 5 seconds. The actual time constants associated with the experiments ranged from 15 seconds to over 5 minutes.

To increase the sampling, and also to increase the degree of resolution, a General Automation SPC-12 mini computer with 8k of 8 bit bytes, an 8 channel A/D converter, and tape cartridge mass storage system was used. After much pain and sorrow, it was programmed to take data at a rate determined by a square wave generator (see appendix D), and to print it out on the teletype for use in further processing. It was found that between the amplifier and the A/D converter, a significant amount of noise was introduced into the signal. (See figure [4]). Due to this fact, it was more difficult to determine time constants from this data, even with the most sophisticated curve fitting techniques. The best time constant was strongly affected by truncation of the data, a phenomenon which will be discussed in detail later. This made a final accurate determination of the time constant nearly impossible.

The ground connections of the A/D converter were very temperamental. After 30 experiments, the wires were moved and the A/D converted ceased to work properly. Subsequent

Figure 4 Sample data collected using the A/D converter. Due to the noise and uncertainty in starting time, it is very difficult to determine a 1/e time constant.



reconnection attempts did not work, so this method of data collection was discontinued.

Analyzing the Data

As mentioned in chapter two, the objective in curve fitting is to minimize χ^2, which was defined in equation (2-3). The first level of curve fitting involved fitting the raw data that was collected to a function of the form

$$f_p = f_2 + (f_1 - f_2)\ e^{-t/\tau} \qquad (1)$$

The standard deviation for each reading was chosen to be 1. The σ_i's certainly should be equal, since the same scale was used for all of the readings. The constant 1 was chosen because the resolution of the voltmeter was 1 unit. This is certainly not rigorous, but it is a reasonable approximation.

An SR-60 programmable desk calculator was obtained to speed up the data analysis. It was first programmed to calculate the weighted average error for an arbitrary choice of f_1, f_2 and τ. The weighted average error is defined in equation (2),

$$W.A.E. = \left[\frac{\sum_i \frac{(f_i - f_{pi})^2}{\sigma_i^2}}{\sum_i \frac{1}{\sigma_i^2}} \right]^{1/2} = \left[\frac{\chi^2}{\sum_i \frac{1}{\sigma_i^2}} \right]^{1/2} \qquad (2)$$

where f_{pi} is the value of f predicted for the i^{th} observation.

The program was extended to iterate on all of the parameters, and to remember the set of parameters with the lowest w.a.e. Table [2] gives an example of such calculations. This method was very simple, but also very inefficient and wasteful in terms of the calculations necessary to zero in on the best fit. In fact, it is necessary to choose the right range of values for f_1, f_2 and τ in order to have any chance of minimizing χ^2.

As mentioned in chapter 2, by manipulating equation (1), one can recast it in a form amenable to linear regression techniques (see equations (2-29) - (2-34).)

Equations (2-10a) - (2-10c) can be applied to solve for $\ell n(f_2-f_1)$ and $-1/\tau$, yielding (f_2-f_1) and τ. In order to use this technique, f_2 must be known. A binary search was used to find the best value to choose for f_2, while the modified linear regression yielded the best values of f_1 and τ for a given f_2. (see appendix A)

Using this method, it took about 5 minutes to determine the best values of f_1 and τ given f_2, and on the order of an hour to find the best integer value of f_2. Table [3] illustrates the results generated by the program. In at least one instance, strange behavior of the fitting program was noted,

f_1	f_2	Best τ	Average error = σ
16	47	295	0.868
17	"	310	0.964
18	"	320	1.135
19	"	335	1.357
16	48	315	0.667
17	"	330	0.742
18	"	340	0.927
19	"	355	1.172
16	49	335	0.549
17	"	350	0.578
18	"	365	0.764
19	"	386	1.023
16	56	360	0.512
17	"	370	0.482
18	"	385	0.645
19	"	405	0.905
16	51	380	0.564
17	"	395	0.467
18	"	410	0.578
19	"	425	0.816

Table 2 Sample set of calculations done on an SR-60 calculator using the exhaustive search algorithm to search for the best fit of f_1, f_2 and τ. Each calculation took ~2 minutes.

f_1	f_2	Best τ	Average Error	Number of Points
68.5	100	144	0.492	59
68.8	101	154	0.476	"
69.1	102	164	0.522	"
69.3	103	175	0.591	"
0.6	23	67	0.87	36
0.8	24	74	0.60	"
1.1	25	81	0.414	"
1.5	26	89	0.366	"
1.9	27	99	0.438	"
59.1	95	442	0.626	53
59.2	96	464	0.597	"
59.4	97	486	0.575	"
59.5	98	509	0.561	"
59.6	99	532	0.553	
59.7	100	555	0.553	"

Table 3 Sample set of calculations done on an SR-60 calculator using the modified linear regression algorithm to find the best fit for f_1, f_2 and τ. Each calculation took ~5 minutes.

and other techniques gave a much better fit. For this and other reasons, a much more powerful computer system was used for subsequent data analysis.

The first method tried on the MULTICS computer system was the variation of each parameter in turn. Starting values for each of the parameters were chosen and then each parameter was varied so that χ^2 was minimized given the current values of all the other parameters. This was actually used for fitting the calculated time constants to a theory predicting them, so it will be discussed later. The gradient method was also used only for the second stage of curve fitting.

When difficulty was encountered in fitting the calculated time constants to the theory predicting them, even using the gradient-linearization technique, the original raw data was re-analyzed using the more sophisticated gradient-linearization techniques.

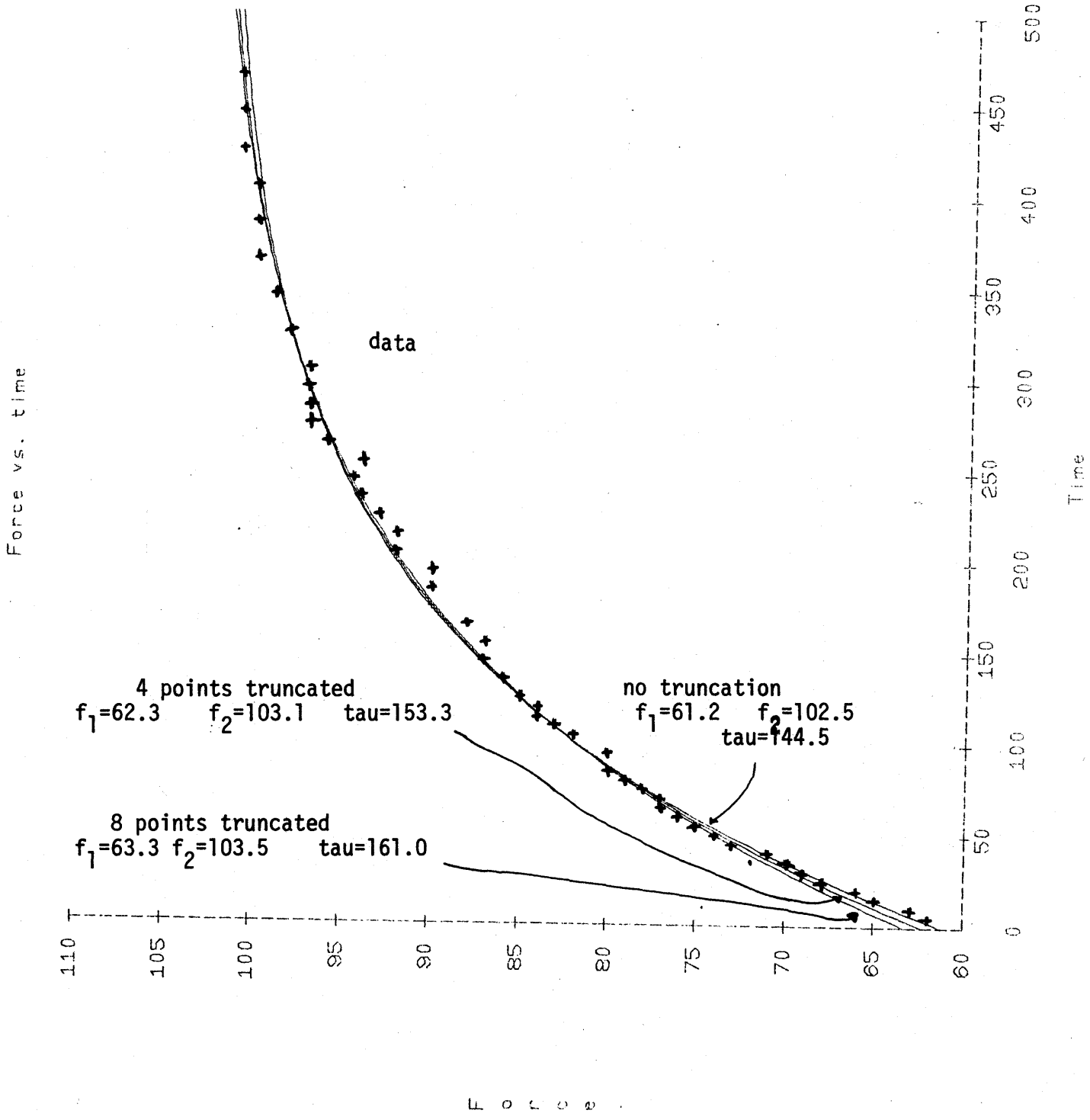
An interesting phenomenon was observed during this round of data analysis. The full set of data from each experiment was analyzed, and then data points were truncated from the beginning. Some sets of data were relatively insensitive to truncation; i.e. the values of the best fit parameters did not vary as a function of truncation. Other sets of data were very sensitive to truncation. Table [4], shows the effects of truncation on sample data sets. Figure [5] has graphs of the different best fits of one data set along with the actual data.

This phenomenon made it difficult to decide upon the

f_2	τ	average error	# of points truncated
57.5	398	0.495	0
57.7	421	0.461	4
57.8	425	0.460	8
103.0	176	0.507	0
103.1	179	0.496	4
103.3	189	0.428	8
120.0	93	0.594	0
120.0	93	0.593	1
120.0	92	0.599	2
120.7	90	0.587	3
120.6	88	0.584	4
120.8	90	0.579	5
120.8	91	0.586	6
120.9	93	0.584	7
120.9	93	0.592	8
120.9	93	0.601	9

Table 4 Effect of truncation on the best fit values of f_1, f_2 and τ found by the gradient-linearization method.

Figure 5 Sample data vs. time and the best fit exponential curves predicted by the gradient-linearization method with all the data, with the first 4 points truncated, and with the first 8 points truncated.



most likely time constant for an experiment. There were many possible causes of this phenomenon. Most of these stem from the short transients that occur at the beginning of the experiment (e.g. mixing time), while others stem from the nature of the approximations made in deriving the theory.

In practice, there usually was a region in which truncation no longer affected the values of the best-fit parameters. The average value of τ in that region was chosen to be the time constant for the experiment. The statistical uncertainty in the time constant was smaller than the fluctuation of the time constant with truncation. Therefore, the standard deviation of each time constant was chosen to reflect its sensitivity to the truncation of the data.

Analyzing the Reduced Data

Once the time constants and their associated standard deviations were determined, they were used as data to be fit to a non-linear diffusion-reaction theory. The independent parameter was the initial pH at which the experiment was done. The fitting function was

$$\tau = \frac{L^2}{\pi^2 a_4}\left(1 + \frac{a_1 a_3}{(a_3+c)^2}\right) \qquad (5)$$

where

$$c^3[c_0 + c_{10}] + c^2[(c_0+c_{10})a_3 + c_0 a_2]$$

$$+ c[c_0 a_2 a_3 - a_1 c_0 a_3 - c_{20} c_0^2] - c_{20} c_0^2 a_3 = 0 \qquad (6)$$

where

c_0, c_{10}, c_{20} are the independent parameters of the experiment

L is determined by independent experiments

a_3 is a constant parameter to be determined

a_1, a_2, a_4 are constants scaled by quantities determined by independent experiments. The constants are to be determined.

(Eq. (5) is the linearized diffusion/reaction time constant and equation (6) represents a Donnan equilibrium relation for internal H^+ ion concentration of a collagen membrane. See reference [1] for details). This is clearly not amenable to linear regression techniques. The exhaustive search technique was tried as a start, but took too much time and paper to look over variations in four parameters.

The next technique tried was the variation of each parameter in turn. This was more efficient, but was not able to converge upon a minimum in χ^2. It seems that the parameters were too interdependent.

A gradient search was then implemented. This worked fairly well on simple functions. However, with the actual fitting function the gradient search stopped short of the minimum. In fact, when the starting points were chosen on oppo-

site sides of the minimum, the gradient search stopped on opposite sides of the minimum with the minimum clearly in the middle. (See Table [5].)

At this stage, the combination gradient linearization method was implemented and used. This provided the same best fit almost regardless of starting point. This proved to be very useful, because it allowed full attention to be focused on the theory and data, rather than on weaknesses of the curve fitting techniques. After re-analyzing the original raw data with this method, a best fit to the experimental τ's was found which had reasonable values for the independent parameters (see reference [1] for complete details of the theory and experiment).

Analysis of Auxiliary Experiments

In order to complete the theoretical analysis of the time constants, two separate experiments had to be done. The thickness and fractional water content of the collagen membrane varied as a function of the independent parameters, and had to be characterized.

Membrane thickness and weight were measured at different bath pH's. The results fit very well to a curve of the form

$$Q = a_1 + a_2 \frac{c}{c+a_3} \qquad (7)$$

where c = bath H^+ concentration

Starting Points			Ending Points			
C'_{at-}	pK	τ'	C'_{at-}	pK	τ'	Average error
1.2	3.5	0.4	1.19	3.70	0.495	10.00
1.2	3.7	0.4	1.20	3.70	0.490	10.00
1.4	3.5	0.4	1.39	3.76	0.381	10.03
1.4	3.7	0.4	1.40	3.77	0.378	10.003
1.2	3.5	0.4	1.20	3.40	0.538	8.0
1.2	3.7	0.4	1.20	3.40	0.536	8.0
1.4	3.5	0.4	1.40	3.46	0.416	8.0
1.4	3.7	0.4	1.40	3.46	0.414	8.0
1.2	3.5	0.4	1.21	3.38	0.543	8.1
1.2	3.7	0.4	1.21	3.39	0.541	8.1
1.4	3.5	0.4	1.40	3.44	0.420	8.1
1.4	3.7	0.4	1.41	3.44	0.419	8.1

Table 5 Best fit values determined using the gradient method. The parameters are for a simplified version of equation (3-5,6), with a different breakdown of parameters. Note the dependence of the final value of C_{at-} on the initial value.

and it was decided to use equation (7) to model both thickness and weight. Using the APL subsystem described in appendix C, the setup of equation (7) and the final determination of a_1, a_2 and a_3 took under a half hour.

Chapter IV

Conclusions and Discussion

Two basic types of data collection techniques were used. Continuous data collection was implemented using a strip chart recorder. Noise was very difficult to eliminate. Discrete data collection also had its share of problems. However, with the use of powerful curve fitting techniques, many of those problems were overcome.

A number of different curve fitting techniques were used to analyze the data. In general, there was a tradeoff between ease of programming on the one hand, and generality, power and speed on the other. Table [6] compares the different techniques in terms of their complexity, relative speed of execution, and number of computations required.

Often, other factors had to be taken into account during the data analysis phase. In the case history presented, there were often transients in the beginning of the data, which had to be truncated in order to accurately determine the time constant.

A question which was touched upon in chapter two involves the validity of the assumption that the fitting function does indeed represent the experimental data. There are a number of statistical tests which focus on the probability distribution of χ^2. The χ^2 test looks at the probability of the

Method	Steps per Single Application of Method	Typical Number of Applications Required for Convergence
Exhaustive or iterative	1 calculation of $\chi^2 \approx (n+2)N$ Total=$(n+2)N$	Many (e.g. 100)
Optimize each parameter in turn	~3 calculations of χ^2 per parameter = $3pN(n+2)$ Total$\approx 3pN(n+2)$	5-15
Linear regression	$\Sigma \frac{x^2}{\sigma^2} \approx 3N$ $\Sigma \frac{xy}{\sigma^2} = 3N$ $\Sigma \frac{y}{\sigma^2} = 2N$ $\Sigma \frac{x}{\sigma^2} = 2N$ $\Sigma \frac{1}{\sigma^2} = 2N$ calculation of a,b,Δ=11 Total = 12N+11	1
Modified linear regression	Transformations=NT Regression=12N+11 Total=$(12+T)N+11$	1-5

Table 6

Gradient	2p calculation of χ^2 for gradient=2pN(n+3) sum of squares of gradient=2p calculation of increments=2p ~10 steps along path, calculating χ^2=10N(n+3) Total=p(4+2Nn+6N)+10Nn +30N =N(2np+6p+10n+30) +4p	3-10
Combine linearization gradient	partial derivatives= N[n(2p+5)] β = Np(n+4) α = $Np^2$3 temporary array=$p^2$10 matrix inversion≈$p^2$3 increments ≈ 11p Total ≈ p^2(13+3N)+p(11+3Nn+4N)+5Nn ≈ N(3np+5n+4p+3p^2)+13p^2	3-10

Table 6 Approximate relative computational demands of the different curve fitting techniques. There are N data point and p parameters to be fit. It takes n steps to evaluate the fitting function. To transform the non-linear function to the linear one for the modified linear regression it takes T steps per data point.

reduced value of χ^2 defined by

$$\chi^2_\nu = \frac{\chi^2}{\nu} \tag{1}$$

where ν is the number of degrees of freedom.

The probability of χ^2 being that high should usually be $\sim$50% for one to have confidence in the fitting function. If it is unusually small, there is reason to doubt the validity of the fitting function.

The χ^2 test is ambiguous, because χ^2 measures both the deviations of the fitting function from the expected values of the data, but also the scatter in the data. The F test looks at the ratio of two different reduced χ^2 distributions, whose probability density function is called the F distribution.

$$F_{12} = \frac{\chi^2_{\nu 1}}{\chi^2_{\nu 2}} \tag{2}$$

Two types of F tests are normally done. One tests the entire fit, and is related to the multiple correlation coefficient R defined as

$$R^2 = \sum_{j=1}^{n} b_j \frac{s_{jy}^2}{s_y^2} \tag{3}$$

where the fitting function is

$$y = a + \Sigma b_j x_j \tag{4}$$

and

$$s_{jy}^2 = \frac{1}{N-1} \left(\Sigma_i x_{ij} y_i - \frac{1}{N} \Sigma x_{ij} \Sigma y_i \right) \tag{5a}$$

$$s_y^2 = \frac{1}{N-1} \left(\Sigma y_i^2 - \frac{1}{N} (\Sigma y_i)^2 \right) \tag{5b}$$

A quantity, F_R can be derived which is distributed according to the F distribution, with $\nu_1 = n$ and $\nu_2 = N-n-1$. [5]

$$F_R = \frac{R^2 (N-n-1)}{(1-R^2)\ n} \tag{6}$$

A large value for F_R corresponds to a good fit. A small value for F_R means that at least one term in the fitting function is probably not valid, and is decreasing the multiple correlation coefficient R by its inclusion.

We can also test for validity of adding another term to the fitting function. The addition of a new term reduces χ^2 by a certain amount $\Delta\chi^2$. The ratio of $\Delta\chi^2$ to the new χ^2 should follow the F distribution with $\nu_1 = 1$ and $\nu_2 = N-n-1$

$$F_\chi = \frac{\Delta\chi^2}{\chi_\nu^2} \tag{7}$$

It is important to be able to put all of this statistical theory into practice with a minimum of trouble for the

varied settings that the experimenter faces.

Stages of an Experiment

Most experiments involve a certain amount of time spent debugging the experimental apparatus and techniques. Once this is done, the experiment is run once, or more probably many times. For the case study of chapter 3, running the experiment involved

1. Setup and calibration.
2. Start of excitation or initialization of independent test parameter.
3. Data collection and storage.
4. Equilibration.
5. Variation of excitation or test parameter.

Steps 3-5 were repeated until the experiments of the current run were done. Step 3, data collection, may include some real-time data reduction, in which case the reduced data (e.g. amplitude of harmonics) might be stored instead of the raw data.

Types of Data Analysis

In this thesis, the type of data analysis focused on involved fitting the data to a function predicted by theory. There are three major types of data analysis, in general. They involve,

1. Fit to a probability density function.
2. Fit to a function of one or many independent parameters.
3. Straightforward calculation of some quantity.

An example of the first might be the fit of test grades to a normal distribution and calculating the mean, standard deviation and skewness of the fit. The second has been extensively discussed. An example of the third might involve the determination of growth rates of bacteria under different conditions.

A major goal to work for is to make all the phases of experiments (e.g. running, data collection and data analysis) as fast and accurate as possible. To that end, a generalized, easy to use set of programs was developed to aid in the analysis of data. Attempts were made to design a general-purpose data collection system, but were unsuccessful. Better hardware and software tools should enable such a system to be constructed.

As digital hardware becomes more affordable to many research groups, we will see more and more experiments run and analyzed using dedicated mini-computer systems. Hopefully, general purpose data collection and data analysis systems will come into use, eliminating the need for many different people to "reinvent the wheel" each time they wish to run an experiment and analyze the data.

Bibliography

1. Nussbaum, J.H., "H^+ Reaction and Diffusion in Collagen Membranes", S.M. Thesis, Department of Electrical Engineering, M.I.T., May, 1979.

2. Bevington, P.R., Data Reduction and Error Analysis for the Physical Sciences, McGraw Hill, New York, 1969, pp. 43.

3. Pugh, E.M., Winslow, G.H., The Analysis of Physical Measurements, Addison-Wesley, Reading, Mass., 1966, sec. 12-8.

4. Marquardt, D.W., "An Algorithm for Least Squares Estimation of Nonlinear Parameters", J. Soc. Ind. Appl. Math, 1963, 11, 2, p. 431.

5. Bevington, P.R., op. cit., p. 198.

Appendix A

Usage of Programs on An SR-60 Calculator to Read In and Analyze Data

Two sets of programs were used on an SR-60 calculator, one to input data, edit it and store it on magnetic cards, and the other to analyze it. The exhaustive search method and modified linear regression method were implemented to find the parameters f_1, f_2 and τ in the fitting function $f=f_2+(f_1-f_2)e^{-t/\tau}$. Up to 60 data points could be stored and analyzed. The entire set of programs consisted of over 2500 keystrokes and were stored on 3 magnetic cards. Each set of stored data resided on 1 side of a magnetic card. Results were printed out on a 20 character wide printer. A sample data input session and sample data analysis section follow.

```
**DATA ENTRY AND
 EDIT

FOR MORE INFO PRESS
ΠSIN

FUNCTION OF THE KEYS

A - WRITE DATA ONTO
 MAGNETIC CARDS
B - PRINT DATA SET
 NUMBER
C - PRINT OUT DATA
D - PRINT OUT LAST
 REGISTER
E - LIST REGISTERS

ΠD/R - INPUT DATA
ΠARC - DELETE LAST
 ENTRY
ΠHYP - INSERT
ΠDMS - ZERO A
 REGISTER
ΠX≠R - DELETE
DATA SET NUMBER=
          10.
END
LAST REGISTER=
          71.

DATA SET NUMBER
          10.

           9.0
          0.05

          11.0
          0.10

          13.0
          0.15

          15.0
          0.20

          17.0
          0.25

          19.0
          0.30

          20.0
          0.35

          21.0
          0.40

          22.0
          0.45

          23.0
          0.50

          23.0
          0.55

          24.0
          1.00

          25.0
          1.05

          25.0
          1.10

          26.0
          1.15

          27.0
          1.20

          28.0
          1.25
           .
          28.0
          1.30

          29.0
          1.35

          29.0
          1.40

          29.0
          1.45

          29.0
          1.50

          29.0
          1.55

          29.0
          2.00

          29.0
          2.05
          2.10

          30.0
          2.15

          30.0
          2.20

          30.0
          2.25
 ***
INSERT AFTER
          65.00
INSERTION
          30.00
LAST REGISTER=
          72.00
WRITE
DATA WRITTEN
```

Data entered.

Data

Time in M.SS

Sample run of the data entry and edit program.

ΠHYP pressed.

Data written onto a magnetic card.

```
**DATA ANALYSIS

READ
DATA READ

DATASET NUMBER=
          10.

B1
          30.
B2
          32.
DELB
          0.5

B=
          30.0
TAU=
          40.
          0.40
A-B=
         24.14
A=
           5.9
SIGMA TAU=
           2.0
SIGMA A-B=
           0.8
 ***



TAU=
          40.
          0.40

N=
          29.
AVERAGE ERROR=
       0.58700
 ***


B=
          30.5
TAU=
          42.
          0.42
A-B=
         24.46
A=
           6.0
SIGMA TAU=
           1.9
SIGMA A-B=
           0.8
 ***



TAU=
          42.
          0.42

N=
          29.
AVERAGE ERROR=
       0.48549
 ***
```

Sample run of the data analysis program

```
B=
          31.0
TAU=
          45.
          0.45
A-B=
         24.66
A=
           6.3
SIGMA TAU=
           1.9
SIGMA A-B=
           0.8
 ***



TAU=
          45.
          0.45

N=
          29.
AVERAGE ERROR=
       0.44790
 ***


B=
          31.5
TAU=
          48.
          0.48
A-B=
         24.80
A=
           6.7
SIGMA TAU=
           2.0
SIGMA A-B=
           0.7
 ***


TAU=
          48.
          0.48

N=
          29.
AVERAGE ERROR=
       0.46383
 ***


B=
          32.0
TAU=
          51.
          0.51
A-B=
         24.91
A=
           7.1
SIGMA TAU=
           2.0
SIGMA A-B=
           0.7
 ***



TAU=
          51.
          0.51

N=
          29.
AVERAGE ERROR=
       0.51555
 ***

 ***
```

Best fit values

Estimated errors

Appendix B

APL Programs That Implement the Variation of Each Parameter in Turn and the Gradient Methods

The following two programs in APL, grdls and gradls implement the variation of each parameter in turn and the gradient methods respectively. Each expects a vector "A" to contain the parameters and the variable "NTERMS' to contain the number of terms in "A". "DELTAA" is expected to contain the increments to use for advancing or taking the partial derivatives. The function CHIS takes one argument and returns the weighted average error.

```
      ∇GRDLS[□]∇
    ∇ GRDLS;I;UPTO;CHI1;CHI2;CHI3;TEMP;DELTA;FN
[1]   I←UPTO←1    Initialize I
[2]  ALOOP:FN←0  Initialize FN
[3]   DELTA←DELTAA[I]   Delta is the current increment
[4]   CHI1←CHIS A     CHIl is the first chi-squared
[5]  EQ:A[I]←A[I]+DELTA    Increment A by DELTA
[6]   CHI2←CHIS A     Calculate new chi-squared
[7]   →EQ×⍳CHI2=CHI1  If they are equal, increment A again
[8]   →LOOP1×⍳CHI2<CHI1    ; If it's decreasing, we're set
[9]   DELTA←-DELTA          ; If it's increasing, reverse direction
[10]  A[I]←A[I]+DELTA
[11]  TEMP←CHI1
[12]  CHI1←CHI2
[13]  CHI2←TEMP
[14] LOOP1:FN←FN+1          ; Keep track of the number of times A was
[15]  A[I]←A[I]+DELTA       ; incremented
[16]  CHI3←CHIS A           ; CHI3 is the last chi-squared
[17]  →(DONE+NOPARABOLIC)×⍳CHI3>CHI2   ;If chi=squared increases,
[18]  CHI1←CHI2             ; go on to the next parameter.
[19]  CHI2←CHI3             ; Otherwise, update the chi-squared variables
[20]  →LOOP1
[21] DONE:DELTA←DELTA×(1÷((1+(CHI1-CHI2)÷(CHI3-CHI2)))+.5)
[22]  A[I]←A[I]-DELTA       ; Use a parabolic approximation for the actual
[23]  DELTAA[I]←DELTAA[I]×FN÷3; minimum.  Rescale DELTAA-If many steps
[24]  →ALOOP×⍳NTERMS≥I←UPTO←I+1  ; were taken, make it larger.
[25]  'GRIDLS DONE'
    ∇
```

APL program implementing the variation of each parameter in turn method.

```
      ∇GRADLS[□]∇
    ∇ GRADLS;DELTA;CHI1;CHI2;CHI3;GRAD;SUM;CHI
[1]   LASTCHI←CHI1←CHIS A      ; Initialize CHI1
[2]   DELTA←.1×DELTAA
[3]   GRAD←NTERMS⍴0
[4]   J←1
[5]  GRADLOOP:A[J]←A[J]+DELTA[J] ; Find the gradient
[6]   GRAD[J]←CHI1-CHIS A
[7]   A[J]←A[J]-DELTA[J]
[8]   →GRADLOOP×⍳NTERMS≥J←J+1
[9]   SUM←(+/GRAD*2)*.5        ; Get the sum of the squares of the gradient
[10]  GRAD←DELTA×GRAD÷SUM      ; Normalize the gradient to it
[11] SMALLER:A←A+GRAD          ; Move along the gradient
[12]  CHI2←CHIS A
[13]  →0×⍳CHI2=LASTCHI         ; If nothing happened, quit
[14]  →LOOP×⍳CHI1>CHI2         ; If it decreased, good.
[15]  LASTCHI←CHI2             ; If not, go a shorter distance along
[16]  A←A-GRAD                 ; the gradient
[17]  GRAD←GRAD÷2
[18]  →SMALLER
[19] LOOP:A←A+GRAD             ; Go along gradient until chi-squared
[20]  CHI3←CHIS A              ; increases
[21]  →(DONE+NOPARABOLIC)÷⍳CHI3≥CHI2
[22]  CHI1←CHI2
[23]  CHI2←CHI3
[24]  →LOOP
[25] DONE:DELTA←PARAB DELTA,CHI1,CHI2,CHI3
[26]  A←A-DELTA×GRAD
[27]  CHI←CHIS A
[28]  A←A+(DELTA-1)×GRAD×CHI2<CHI
[29]  'GRADLS DONE'
    ∇


      ∇PARAB[□]∇
    ∇ A←PARAB B
[1]   A←B[1]×(1÷((1+(B[2]-B[3])÷(B[4]-B[3])))+.5)
    ∇
```

Implementation of the gradient search method in APL.

Appendix C

An APL Subsystem to Curve Fit Data to an Arbitrary Function

The set of APL programs that follow consist of a number of general programs to implement the linearization-gradient method of curve fitting. Also included are a number of programs to implement the function $f=f_2+(f_2-f_1)e^{-(t+t_0)/\tau}$ and the function that predicts the diffusion/reaction time constants (see equations (3-5)-(3-6)).

Usage of the subsystem

Once the subsystem is entered, it is necessary to define the function that you want the curve fit to. It takes two parameters. The left side parameter is the set of subscripts that indicate which of the data points are to be used. Normally it is set to ⍳NPTS. The right side argument is the vector "A", which contains the parameters that are to be determined. For example, to implement $y=a_1+a_2x^{a_3}$ we can type:

The functions must be called "F1", "F2"... until "F10" in the current implementation. Data is stored in the "X" and "Y" vectors. In the simple case, an assignment of values to "X" and "Y" will suffice. In more involved cases, specialized input functions can be written.

To set the function number, one types "SFN <n>", where <n> is the number form 1-10. "F<n>" is then the function used. Typing "FIT" starts the actual curve fitting programs. The programs ask for starting values for the vector "A", and increments ot be used for taking partial derivatives, which are stored in "DELTAA". Variance is then asked for. The reply can either be a vector of length "NPTS", the number of data points, or else a single constant, indicating that all variances are equal. The program then runs, printing out the path that it's following. When χ^2 is no longer changing very much, the program prints out a table of independent parameter, observed dependent parameter, predicted dependent parameters and deviation. If so desired, the user can implement a function to print out the final results in neater form.

Both the force vs. time data and the time constant vs. pH data were stored as tables. Special functions were written to input data and to convert the data to "X", "Y" and "VARIANCE" vectors. Special functions were also written to print the final fit in neater form. The sensitivity of

the force vs. time data to truncation necessitated a special function which called the fitting program, printed out the results, and truncated the data. This was done a specified number of times, after which the results were printed out in tabular form.

The time constant vs. pH curve fitting occassionally used some of the parameters as constants. Special support functions were written to implement this. A vector called "USE" controlled which parameters were to be constants. It had as many elements as there were parameters. A 0 signified that the corresponding paramter was to be a constant, and a 1 signified that the corresponding parameter was to be a constant, and a 1 signified that it was to be fit by the fitting function.

The following is a list of functions and a brief description of what they do.

General Curve Fitting Subsystem

Name	Purpose
CHIS	calculate χ^2
COMPXY	print out a table comparing the data to the predicted fit.
CURFIT	does the actual curve fitting
F	calls the correct fitting function
FDERIV	calculate the partial derivatives of the function with respect to each parameter
FIT	calls the programs necessary to actually set up and do the curve fitting
GETA	initializes variables, gets necessary input and checks for errors in input

Name	Purpose
RAISED	Implement raising a negative number to the 1/3 power
SFN	sets the function_number
WORK	does some initialization, and keeps calling CURFIT until χ^2 is no longer significantly decreased by CURFIT

Special Functions for Fitting Force vs. Time

F9	implements $f_2+)f_1-f_2)e^{-(t+t_0)/\tau}$ where t_0 is a constant
F10	implements same equation where t_0 is a parameter
GETTAU	function and truncating function a specified number of times, and accumulates a table of the results
INPUT	inputs prints out and stores the force vs. time data in "TDATA"
PRINT	actually prints out and stores the data read in by INPUT
PTABLE	prints out the table accumulated by GETTAU
PTAUFIT	prints out the best fit in a special format
REDO	helps to recover from errors in typing data into INPUT
SE TUPTAU	sets up the "X" and "Y" vectors from the force vs. time data stored in "TDATA"
TRUNC	truncates a specified number of points from the beginning of the data.

Special Functions for Fitting τ vs. pH

CALC	solves the cubic equation of
COMP	prints out on a table comparing predicted and observed values of τ
DATAIN	inputs τ vs. pH data nad stoves it in "DATA"
F2	implements the calcualtion of τ
GETINC	reads in the values for "DELTAA" for the parameters being varied
GETSTART	reads in the initial values for "A" for the parameters being varied
GETVAL	returns a vector which includes the constants and the current values of the parameters being varied
GETVALUES	does the actual I/O for GETSTART GETVAL and INIT
GO	starts the τ vs pH curve fitting
GRIND	calculate τ, membrane charge, and internal salt concentrations from the internal H^+ concentrations.
INIT	does most of the initialization, and reads in the values of the constants
P2	formats the values printed out each step in the path
SETA	maps "A" onto the mnemonic variables used by CALC and GRIND

A sample session follows the program listings, illustrating the use of the general subsystem to fit some perfect data to the function $y=A_1 + A_2 X^{A_3}$.

```
      ∇CHIS[□]∇
    ∇ B←CHIS A
[1]   B←(+/((Y-A F ⍳NPTS)*2)÷VARIANCE)÷WEIGHTSUM  ; calculate the weighted
    ∇                                              ; average error
      ∇COMPXY[□]∇
    ∇ COMPXY;T;YP;FMT
[1]   T←(⍴Y),1                          ; print out a table
[2]   '       X           Y     Y PREDICTED      DELTA'
[3]   FMT←(10 1),(10 2),(10 2),(10 2)
[4]   YP←A F ⍳NPTS
[5]   FMT ⍪ (T⍴X),(T⍴Y),(T⍴YP),T⍴Y-YP
    ∇
      ∇CURFIT[□]∇
    ∇ CURFIT;ALPHA;DERIV;CHI1;CHI0;ARRAY;BETA;T1;T2;B;T3;I
[1]   ALPHA←(NTERMS,NTERMS)⍴0           ; Initialize
[2]   DERIV←A FDERIV ⍳NPTS              ; Take partial derivatives
[3]   BETA←(WEIGHTx(Y-A F ⍳NPTS))+.xDERIV  ; Calculate the beta matrix
[4]   I←1
[5]  LOOP:ALPHA←ALPHA+WEIGHT[I]xDERIV[I;]∘.xDERIV[I;]
[6]   →LOOPx⍳NPTS≥I←I+1                 ; Accumulate the alpha matrix
[7]   CHI1←CHIS A                       ; The initial chi-squared
[8]  BACK:T1←1 1 ⌹ALPHA
[9]   T2←ALPHA÷T3←(T1∘.xT1)*.5          ; Normalize  the alpha matrix
[10]  ARRAY←⌹T2+IDENxFLAMDA             ; Add lamda to the diagonal terms and invert
[11]  B←A+BETA+.xARRAY÷T3               ; Calculate new  A
[12]  →OKx⍳CHI1≥CHI0←CHIS B             ; See if it's in the right direction
[13]  FLAMDA←FLAMDAx10                  ; If not, increase lamda.
[14]  →BACK                             ; And try again
[15] OK:A←B
[16]  SIGMAA←((1 1⌹ARRAY)÷T1)*.5        ; Approximate uncertainty
[17]  FLAMDA←FLAMDA÷10                  ; Reduce lamda by a factor of 10.
[18]  ' CURFIT DONE'
    ∇
      ∇F[□]∇
    ∇ Y←A F I
[1]   →2xFUNCTION_NUMBER                ; Dispatch to proper function call
[2]   Y←A F1 I
[3]   →0
[4]   Y←A F2 I
[5]   →0
[6]   Y←A F3 I
[7]   →0
[8]   Y←A F I
[9]   →0
[10]  Y←A F5 I
[11]  →0
[12]  Y←A F6 I
[13]  →0
[14]  Y←A F7 I
[15]  →0
[16]  Y←A F8 I
[17]  →0
[18]  Y←A F9 I
[19]  →0
[20]  Y←A F10 I
[21]  →0
    ∇
```

APL subsystem to fit data to theory.

```
      ∇FDERIV[⎕]∇
    ∇ D←A FDERIV I;J;AJ;DELTA;YFIT
[1]   D←((⍴,I),NTERMS)⍴0              ; Initialize the partial derivatives
[2]   J←1
[3]  DLOOP:DELTA←DELTAA[J]            ; Loop through all of the parameters
[4]   A[J]←DELTA+AJ←A[J]
[5]   YFIT←A F I
[6]   A[J]←AJ-DELTA
[7]   D[;J]←(YFIT-A F I)÷2×DELTA      ; Calculate the partial derivative
[8]   A[J]←AJ
[9]   →DLOOP×⍳NTERMS≥J←J+1
    ∇
      ∇FIT[⎕]∇
    ∇ FIT
[1]   GETA                            ; Input the starting points and initialize
[2]   WORK                            ; Curve fit
[3]   ' SIGMAA= ';SIGMAA              ; Print out the uncertainties
[4]   ' FIT DONE'                     ; Print out an ending message
    ∇
      ∇GETA[⎕]∇
    ∇ GETA;T
[1]   →ERR0×⍳(⍴T)≠NPTS←⍴X             ; Check that there is the same number of
[2]   ' FUNCTION NUMBER= ';FUNCTION_NUMBER    ; X and Y numbers.
[3]   ' FUZZ= ';FUZZ
[4]   FLAMDA←.001                     ; Initialize flamda
[5]   'A'                             ; Input the starting points
[6]   A←,⎕
[7]   'DELTAA'                        ; Input the increments used for taking
[8]   DELTAA←,⎕                       ; partial derivatives
[9]   →ERR1×⍳(⍴DELTAA)≠NTERMS←⍴A      ; Make sure both A and DELTAA have the
[10] VARQUERY:'VARIANCE'              ; same number of elements.
[11]  T←⍴VARIANCE←,⎕                  ; Input the variance.
[12]  →SKIP×⍳T=1                      ; If it's a single number, it's constant
[13]  →ERR2×⍳T≠NPTS                   ; Otherwise make sure it's the same length
[14]  →MORE                           ; as the data
[15] SKIP:VARIANCE←NPTS⍴VARIANCE
[16] MORE:WEIGHTSUM←+/WEIGHT←1÷VARIANCE       ; calculate the weightings
[17]  →0
[18] ERR0:'X AND Y ARE NOT OF THE SAME LENGTH'
[19]  →0
[20] ERR1:'A AND DELTAA ARE NOT OF THE SAME LENGTH'
[21]  →0
[22] ERR2:'VARIANCE AND X/Y ARE NOT OF THE SAME LENGTH'
[23]  →VARQUERY
    ∇
      ∇RAISED[⎕]
    ∇ AN←A RAISED B
[1]   AN←(×A)×(|A) *B                 ; allows raising a negative number to  a
    ∇                                 ; 1/3 power.
```

APL Subsystem to fit theory to data

```
      ∇P[□]∇
    ∇ B←P A
[1]   B←A
[2]   →0×⍳FUNCTION_NUMBER≠2          ; unless FN=2, just print out A
[3]   B←P2 A                         ; If FN=2, then call a special formatter
    ∇
      ∇SFN[□]∇
    ∇ SFN N
[1]   FUNCTION_NUMBER←N              ; sets the function number
    ∇
      ∇WORK[□]∇
    ∇ WORK;CHI1;CHI2
[1]   IDEN←(⍳NTERMS)∘.=⍳NTERMS       ; Set up an identity matrix
[2]   FLAMDA←.001                    ; Initialize flamda
[3]   CURFIT                         ; Call the curve-fitting program
[4]   CHI1←CHIS A                    ; Calculate chi-squared
[5]   (CHI1*.5),P A                  ; Print out the weighted average error and
[6]  LOOP:CURFIT                     ; the current value of A
[7]   CHI2←CHI1                      ; Call the curve-fitting program and
[8]   CHI1←CHIS A                    ; print out the results
[9]   (CHI1*.5),P A
[10]  →LOOP×⍳FUZZ<|(CHI2-CHI1)÷CHI2; Continue until chi-squared no longer
[11]  'DONE'                         ; decreases by more than "FUZZ" %.
    ∇
```

APL Subsystem to fit theory to data

```
      ∇CALC[□]∇
    ∇ CALC;A1;A2;A3;A;B;D;E;G;H;P;Q;DET;P3;Q2;A1N;PHI;APHI;C2TOT;CQ1;CQ2
[1]   C2TOT←C0+C20                    ; Calculate the boundary condition
[2]   E←C0+C10                        ; using Donnan equilibrium and
[3]   D←C2TOT×C0*2                    ; a Langmuir isotherm
[4]   G←C0×PM0
[5]   H←CAT×C0×K
[6]   A←K+(G÷E)
[7]   B←((G×K)-(H+D))÷E
[8]   C←- D×K÷E
[9]   P←B-(A*2)÷3
[10]  Q←((2×A*3)+(27×C)-9×A×B)÷27
[11]  P3←(P*3)÷27
[12]  Q2←(Q*2)÷4
[13]  DET←Q2+P3
[14]  A1←DET≥0
[15]  A1N←1-A1
[16]  A2←Q<0
[17]  A3←(A1×DET)*.5
[18]  APHI←(- A1N×Q2÷P3)*.5
[19]  PHI←((1-A2)×¯2○-APHI)+A2×¯2○APHI
[20]  CQ1←((A1×A3-Q÷2) RAISED  THIRD)-(A1×A3+Q÷2) RAISED THIRD
[21]  CQ2←C+2×(2○PHI÷3)×(-A1N×P÷3    )*.5
[22]  C←CQ1+CQ2-A÷3
    ∇

      ∇COMP[□]∇
    ∇ COMP;A
[1]   A←(⍴TAUO),1                     ; Print out a table
[2]   TAUP←⌊TAUP+.5
[3]   (A⍴PHO),(A⍴TAUO),(A⍴TAUP),(A⍴TAUP-TAUO)
[4]   (2,1)⍴' '
    ∇

      ∇DATAIN[□]∇
    ∇ DATAIN;DATANAME;IN              ; Function for inputting data
[1]   DATANAME←(1,2)⍴0
[2]  LOOP:IN←□
[3]   →END×⍳+/(IN=0)
[4]   →ERR×⍳(⍴,IN)≠2
[5]   DATANAME←DATANAME,[1]IN
[6]   →LOOP
[7]  END:' DONE'
[8]   DATA←DATANAME[1+⍳((⍴DATANAME)[1] -1)  ;]
[9]   →0
[10] ERR:' NOT RIGHT - TRY AGAIN'
[11]  →LOOP
    ∇

      ∇F2[□]∇
    ∇ Y←A F2 I
[1]   SETA A                          ; Set up the paramaters for CALC
[2]   CALC                            ; Calculate the boundary condition
[3]   GRIND                           ; Calculate the reaction-diffusion time constant
[4]   Y←TAUP[I]                       ; Set Y to those time constants
    ∇
```

APL programs written to fit the calculated time constants from the experiments to those predicted by a non-linear diffusion-reaction theory.

```
      ∇GETINC[□]∇
    ∇ GETINC
[1]   'INCREMENTS'                    ; Read in the increments for taking
[2]   DELTAA←GETVALUES 1              ; partial derivatives
    ∇


      ∇GETSTART[□]∇
    ∇ GETSTART
[1]   'INITIAL POINTS:'               ; Read in the starting points
[2]   A←GETVALUES 1
    ∇
      ∇GETVAL[□]∇
    ∇ T←GETVAL A
[1]   T←(USE×USE\A)+(~USE)×CONSTANTS  ;get a string of both constants
    ∇                                 ; and variable parameters
      ∇GETVALUES[□]∇
    ∇ T←GETVALUES M
[1]   T←5ρ0                           ; Read in the values for the various
[2]   →SCAT×\USE[1]≠M                 ; parameters.  Depending on M, they
[3]   'CAT'                           ; are for constants (M=0) or for the
[4]   T[1]←□÷AVO                      ; variable parameters (M=1).
[5]  SCAT:→SPK×\USE[2]≠M
[6]   'PK'
[7]   T[2]←10*-□
[8]  SPK:→SRAT×\USE[3]≠M
[9]   'RAT'
[10]  T[3]←□÷AVO
[11] SRAT:→SD×\USE[4]≠M
[12]  'DBAR'
[13]  T[4]←□
[14] SD:→SFILM×\USE[5]≠M
[15]  'FILM THICKNESS'
[16]  T[5]←□
[17] SFILM:→0×\~M
[18]  T←(USE)/T                       ; For the variable parameters, condense
    ∇                                 ; T.
      ∇GO[□]∇
    ∇ GO
[1]   INIT                            ; Initialize
[2]   GETSTART                        ; Get the starting points
[3]   GETINC
[4]   WORK                            ; Fit
[5]   PPFIT                           ; Pretty print the results
[6]   COMP                            : Print out a table comparing fit and data
    ∇
```

APL programs written to fit the time constants calculated from experiments to those predicted by theory.

```
      ▽GRIND[□]▽
    ▽ GRIND
[1]   PHI←(×C)×-10*|C                    ; Calculate time constants
[2]   FM←FM0-CAT×K÷(K+C)
[3]   TAUF←(FILM*2)÷PI2×DINF
[4]   TAUF←TAUF+((THICKNESS*2)÷(PI2×DBAR×DFACT))×RFACT←1+CAT×K÷(K+C)*2
[5]   C1←C10×C÷C0
[6]   C2←C0×C20÷C
    ▽
      ▽INIT[□]▽
    ▽ INIT
[1]   FUNCTION_NUMBER←2                  ; Initialize
[2]   Y←TAU0←DATA[;2]
[3]   C0←X←10*-PH0←DATA[;1]
[4]   NPTS←⍴TAU0
[5]   NTERMS←+/USE
[6]   IDEN←(⍳NTERMS)∘.=⍳NTERMS
[7]   VARIANCE←(STATWEIGHT×TAU0*POWER)+VARIANCE×~STATWEIGHT
[8]   WEIGHTSUM←+/WEIGHT←1÷VARIANCE
[9]   FLAMDA←.001
[10]  PI2←3.14159*2
[11]  W←W0+W1×C0÷KVOL+C0                 ; Set up the weight, volume of
[12]  VW←(W-WC)÷PW                       ; the membrane.
[13]  THICKNESS←DELTA0+DELTA1×C0÷KTHICKNESS+C0 ;set up the thickness
[14]  DFACT←((1-VC÷VW+VC)÷(1+VC÷VW+VC))*2
[15]  VOLFACT←VC÷VW
[16]  CONSTANTS←GETVALUES 0              ; Input the constants
    ▽
      ▽F2[□]▽
    ▽ B←F2 A;VAL
[1]   VAL←GETVAL A                       ; Format the current values for printing
[2]   B←(VAL[1]×(AVO,1)),((×VAL[2])×-10*|VAL[2]),(VAL[3]×AVO),VAL[4],VAL[5]
    ▽
      ▽SETA[□]▽
    ▽ SETA A;VAL
[1]   VAL←GETVAL A                       ; Set up the current value of A for use
[2]   CAT←VAL[1]×VOLFACT                 ; by CALC
[3]   K←VAL[2]
[4]   FM0←CAT+RAT←VAL[3]×VOLFACT
[5]   DBAR←VAL[4]
[6]   FILM←VAL[5]
    ▽
```

APL programs written to fit the time constants calculated from experiments to those predicted by theory.

```
      ∇F9[□]∇
    ∇ Y←A F9 I
[1]   Y←A[1]+(A[2]-A[1])×*-(X[I]+T0)÷A[3]    ; calculate the predicted force
    ∇
      ∇F10[□]∇
    ∇ Y←A F10 I
[1]   Y←A[1]+(A[2]-A[1])×*-(X[I]+A[4])÷A[3]
    ∇
      ∇GETTAU[□]∇
    ∇ GETTAU N
[1]   SETUPTAU                              ; Initialize
[2]   SFN 10-USET0
[3]   GETA
[4]   TABLE←''
[5]  LOOP:WORK
[6]   TABLE←TABLE,A[3],A[1],((CHIS A)*.5),Y[1],Y[NPTS]
[7]   PTAUFIT
[8]   TRUNC TRUNCNUM                        ; Truncate
[9]   →LOOP×⍳0<N←N-1
[10]  COMPXY
[11]  PTABLE                                ; Print a table summarizing the results
    ∇
      ∇INPUT[□]∇
    ∇ INPUT;N;F;INC;T;UPTO
[1]   Y1←''                                 ; Input the data
[2]  LOOP1:→SKIP×⍳1≥⍴LINE←,□
[3]   Y1←Y1,LINE
[4]   →LOOP1
[5]  SKIP:N←1
[6]   UPTO←⍴Y1
[7]   Y2←T1←''
[8]   T←0
[9]  LOOP2:→SINC×⍳0>F←Y1[N]
[10]  T1←T1,T←T+INC
[11]  Y2←Y2,F
[12] BACK:→LOOP2×⍳UPTO≥N←N+1
[13]  PRINT
[14]  →0
[15] SINC:INC←-F                            ; Print out the data
[16]  →BACK
    ∇
      ∇PRINT[□]∇
    ∇ PRINT;T
[1]   T←((⍴Y2),1)
[2]   TDATA←(T⍴Y2),T⍴T1                     ; Format and print out
[3]   TDATA
    ∇
```

APL programs written to determine the 1/e time constant from experimental data.

```
      ∇PTABLE[□]∇
    ∇ PTABLE;LEN;BEST;TTABLE;FMT
[1]   ' '                                  ; Print out a formatted table summarizing
[2]   LEN←(⍴TABLE)÷5                       ; the results of truncating the data
[3]   TTABLE←(LEN,5)⍴TABLE
[4]   '       TAU          B       A,E,          VAL1        VAL2'
[5]   FMT←(10 0),(10 2),(10 4),(10 1),(10 1)
[6]   FMT ⍪TTABLE
[7]   BEST←TTABLE[;3]⍳L/TTABLE[;3]
[8]   ' '
[9]   'BEST SET OF VALUES='
[10]  FMT⍪TTABLE[BEST;]
[11]  ' '
    ∇

      ∇REDO[□]∇
    ∇ REDO;N;F;INC;T;UPTO
[1]  SKIP:N←1                              ; A function to help correct input errors
[2]   UPTO←⍴Y1                             ; from INPUT
[3]   Y2←T1←''
[4]   T←0
[5]  LOOP2:→SINC×⍳0>F←Y1[N]
[6]   T1←T1,T←T+INC
[7]   Y2←Y2,F
[8]  BACK:→LOOP2×⍳UPTO≥N←N+1
[9]   PRINT
[10]  →0
[11] SINC:INC←-F
[12]  →BACK
    ∇

      ∇SETUPTAU[□]∇
    ∇ SETUPTAU
[1]   X←TDATA[;2]                          ; Initialize
[2]   Y←TDATA[;1]
    ∇



      ∇TRUNC[□]∇
    ∇ TRUNC N
[1]   T0←T0+N×T0INC
[2]   NPTS←NPTS-N                          ; Truncate N points from the start of
[3]   Y←Y[N+⍳NPTS]                         ; the data
[4]   X←X[N+⍳NPTS]
[5]   WEIGHTSUM←+/WEIGHT←WEIGHT[N+⍳NPTS]
[6]   VARIANCE←VARIANCE[N+⍳NPTS]
    ∇
```

APL programs written to determine the 1/e time constant from experimental data.

```
      ∇Y←A F1 I
[1]   Y←A[1]+A[2]×X[I]*A[3]∇ ; Definition of function

      X←1 3 5 6 7                ; X data
      Y←1.3+2.41×X*3.48          ; Y data (perfect data)
      Y
3.71 111.5551816 653.57789 1231.52173 2104.876071 ; actual Y values
      SFN 1                      : use F1
      FIT                        ; start the curve-fitting programs
 FUNCTION NUMBER= 1
 FUZZ= 1E¯7
A
□:
      1 2 3
DELTAA
□:
      .0000001 .000000001 .000000001
VARIANCE
□:
      1
 CURFIT DONE
92.73450958 103.6055399 3.168112326 3.318401715
 CURFIT DONE
20.68362981 6.652979869 3.193563916 3.328371516
 CURFIT DONE
12.44876943 ¯13.65546387 3.137892903 3.345016511
 CURFIT DONE
11.38990688 ¯6.671105593 2.749131082 3.408468669
 CURFIT DONE
8.485469537 0.5809976662 2.424879259 3.47299233
 CURFIT DONE
0.008089956925 1.294434695 2.410069935 3.479989994
 CURFIT DONE
1.369105913E¯6 1.299998368 2.410000068 3.479999986
 CURFIT DONE
1.498639302E¯10 1.3 2.41 3.48
 CURFIT DONE
2.281000425E¯15 1.3 2.41 3.48
 CURFIT DONE
3.763918735E¯16 1.3 2.41 3.48
 CURFIT DONE
3.077029793E¯16 1.3 2.41 3.48
 CURFIT DONE
3.077029793E¯16 1.3 2.41 3.48
DONE
 SIGMAA= 0.01413507702 3.01663358E¯5 6.629367578E¯6
 FIT DONE
```

Example of the use of the APL curve-fitting subsystem to define a function, input data, and to determine the best fit of the data to that function.

Appendix D

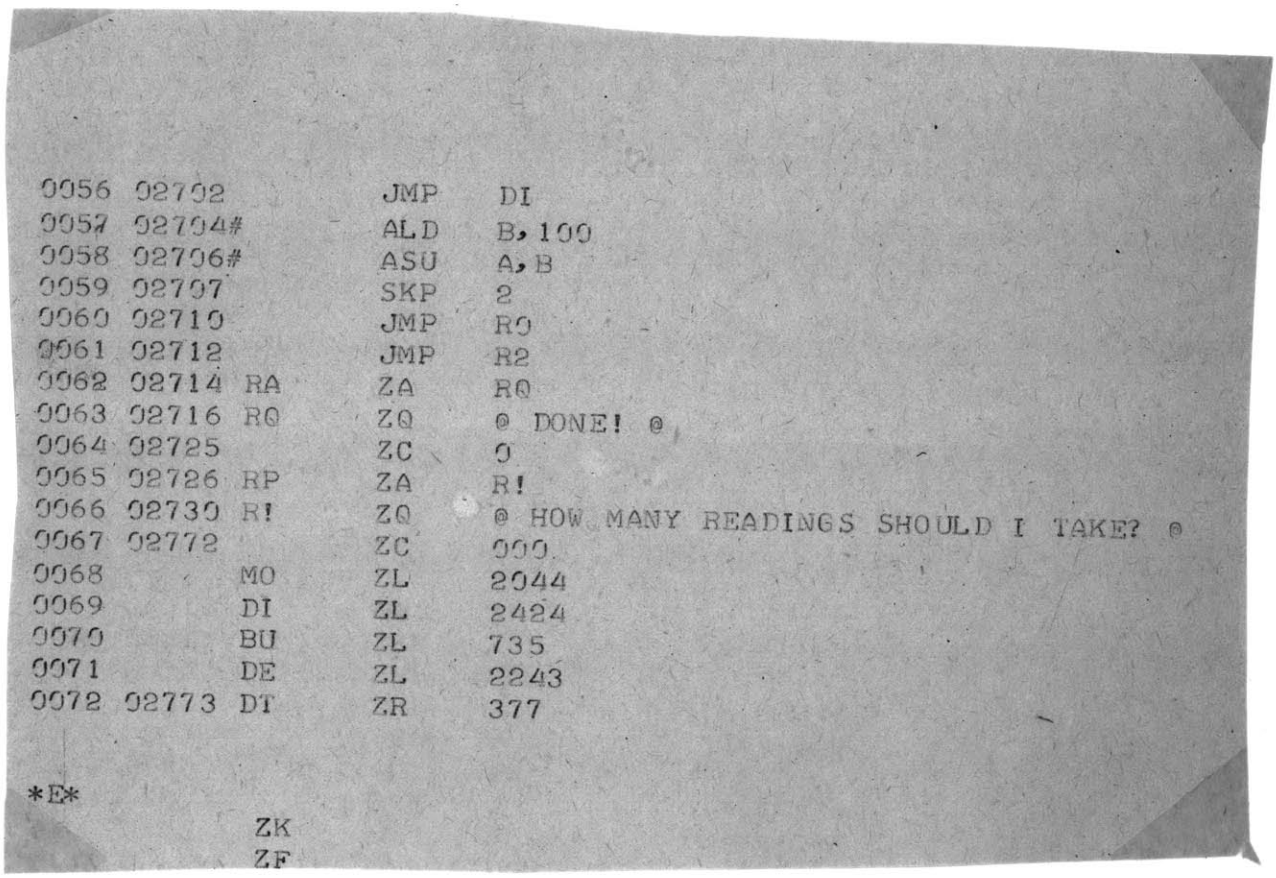
Programs for an SPC-12 Minicomputer to Read In and Print Out Data

An SPC-12 minicomputer was programmed to read in data at a rate specified by a function generator attached to channel 2 of the A/D converter. Data was actually input on channel 0. The number of datums to read in is input at the start of the program. At the end of data collection, the program prints "done!"

After the data was read in, the output program was loaded and run. It printed the index and the value of the data point. Decimal input and output routines had to be written, as well as a few other support programs.

```
TING(Y OR N)?
PLY: Y                ZO 2570
0001        *READ
0002        *          X=CHANNEL
0003        *          A=DATA
0004        *          Z=DATUM NUMBER
0005        *          Y=MAXIMUM # OF DATA TO READ IN
0006        *
0007        ************************************************
0008 02570             ELB   RP          ASK HOW MANY READINGS TO TAKE
0009 02572#            RTR   B,Y
0010 02573#            RIC   P,B
0011 02574             JMP   MO
0012 02576#            RIC   P,B
0013 02577             JMP   DE          READ IN MAXIMUM NUMBER
0014 02601             RTR   X,Y         STORE IN Y
0015 02603             RDC   Y,B         MULTIP # READINGS BY 2-1
0016 02605             AAD   Y,B         BECAUSE EACH READING USES 2 LOCATION
0017 02607             AZE   Z
0018 02611'R0          ALD   X,002       WAIT FOR A 1 ON CHANNEL 2
0019 02613#            RIC   P,B
0020 02614             JMP   DI
0021 02616#            ALD   B,100       SUBTRACT 100 OCTAL FROM INPUT
0022 026R0#            ASU   A,B
0023 02621             SKP   2
0024 02622             JMP   R0
0025 02624#R1          AZE   X           READ IN DATA FROM CHANNEL 0
0026 02625#            RIC   P,B
0027 02626             JMP   DI
0028 02630#            RTR   A,B
0029 02631             RTR   Z,X         SET UP FOR INDEXED STORE
0030 02633             STB   DT,X
0031 02635#            RIC   X,X
0032 02636#            TBB
0033 02637             STB   DT,X
0034 02641             RIC   X,Z
0035 02643#            RTR   Y,B
0036 02644#            ASU   X,B         PRESENT #-MAXIMUM #
0037 02645#            RTR   X,B
0038 02646#            TBB
0039 02647#            RTR   B,X
0040 02650             AND   X,010
0041 02653             SKZ   2
0042 02654             JMP   R2
0043 02656             RTR   Z,X         SET UP FOR INDEXED STORE
0044 P2660#            TBB               ZERO THE B REGISTER
0045 02661#            TBB
0046 02662             STB   DT,X
0047 02664#            RIC   X,X
0048 02665             STB   DT,X
0049 02667             ELB   RA          GO BACK TO BUS'
0050 02671#            RTR   B,Y
0051 02672#            RIC   P,B
0052 02673             JMP   MO
0053 02675             JMP   BU
0054 02677#R2          ALD   X,002       WAIT FOR A 0 ON CHANNEL 2
0055 02701#            RIC   P,B
```

Program written in SPC-12 assembly language to read in data from an A/D converter at a rate controlled by a signal at channel 2.



```
0056 02702          JMP   DI
0057 02704#         ALD   B,100
0058 02706#         ASU   A,B
0059 02707          SKP   2
0060 02710          JMP   R0
0061 02712          JMP   R2
0062 02714 RA       ZA    R0
0063 02716 R0       ZQ    @ DONE! @
0064 02725          ZC    0
0065 02726 RP       ZA    R!
0066 02730 R!       ZQ    @ HOW MANY READINGS SHOULD I TAKE? @
0067 02772          ZC    000
0068       MO       ZL    2044
0069       DI       ZL    2424
0070       BU       ZL    735
0071       DE       ZL    2243
0072 02773 DT       ZR    377

*E*
           ZK
           ZF
```

Program written in SPC-12 assembly language to read in data (continued).

```
REPLY: Y            ZO 2250
0001        * PRINT - PRINTS OUT DATA READ IN BY READ
0002        *
0003        *       Y=INDEX REGISTER
0004        *       X=ADDRESS OF INDIRECT WORD
0005        *       A=DATA
0006        *
0007        ****************************************
0008 02250#P0       RIC   P,B          Get ready for a subroutine call
0009 02251          JMP   NL           <CR>,<LF>
0010 02253          AZE   Y            INITIALIZE
0011 02255          ELB   PA
0012 02257#         RTR   B,X
0013 02260#P1       RIC   P,B          SAVE REGISTERS
0014 02261          JMP   SR
0015 02263          RTR   Y,Z
0016 02265          RTR   YY,Y         GET 4 MSB'S
0017 02267#         SHR   Y            DIVIDE BY 2, SENDING THE LAST BIT TO
0018 02270          SHL   Z            THE Z REGISTER
0019 02272#         AZE   A
0020 02273#         RIC   P,B          PRINT OUT DATUM NUMBER
0021 02274          JMP   DP
0022 02276          ALD   Y,5          PRINT 5 SPACES
0023 02301          ELB   PP           PRINT SPACES TO GET TO POSITION 7
0024 02303          AAD   Y,B          Y POINTS TO START OF PROPER NUMBER
0025 02305#         RIC   P,B          OF SPACES
0026 02306          JMP   MO           PRINT SPACES
0027 02310#         RIC   P,B          RESTORE REGISTERS
0028 02311          JMP   RR
0029 02313#         GOL   A,X,I        LOAD FIRST WORD INTO A
0030 02314          RTR   A,Z          AND THEN INTO Z
0031 02316#         GOL   A,X,I        LOAD SECOND WORD INTO A
0032 02317#         RIC   P,B          SAVE REGISTERS
0033 02320          JMP   SR
0034 02322          RTR   A,Y          NOW MOVE SECOND WORD INTO Y
0035 02324#         AND   A,010        4TH BIT 1(NEGATIVE)
0036 02326          SKN   2            IF NOT, CONTINUE
0037 02327          JMP   P2
0038 02331#         ALD   A,255        PRINT A MINUS SIGN
0039 02333#         RIC   P,B
0040 02334          JMP   TO
0041 02336#         RDC   B,B          SET B TO ALL ONES
0042 02337          AXR   Y,B          COMPLEMENT Y
0043 02341          RTR   Z,Z          CHECK IF Z=0
0044 02343          SKZ   2            IF NOT, THERE WILL NOT BE A CARRY
0045 02344          JMP   P3           WHEN 1 IS ADDED TO Z
0046 02346#         RIC   Y,Y          OTHERWISE ADD 1 TO Y
0047 02347          JMP   P2           AND WE HAVE THE 2'S COMPLEMENT
0048 02351  P3      AXR   Z,B          COMPLEMENT Z
0049 02353          RIC   Z,Z          ADD 1
0050 02355#P2       AZE   A
0051 02356          AND   Y,017        TAKE ONLY 4 LSB'S OF Y
0052 02361#         RIC   P,B          PRINT OUT DATUM
0053 02362          JMP   DP
0054 02364#         RIC   P,B
0055 02365          JMP   NL           PRINT OUT NEWLINE
```

Program written in SPC-12 assembly language to print out the data read in by the previous program.

```
0056 02367#        RIC   P,B        RESTORE REGISTERS
0057 02370         JMP   RR
0058 02372         RTR   Z,Z
0059 02374         SKZ   2          IF Z<>0 THEN CONTINUE
0060 02375         JMP   P1
0061 02377         RTR   A,A
0062 02401         SKZ   2          IF A<>0 THEN CONTINUE
0063 02402         JMP   P1
0064 02404         JMP   BU
0065 02406#NL      RTR   B,A        KEEP RETURN ADDRESS AROUND
0066 02407         ELB   NP         LOAD ADDRESS OF NEWLINE
0067 02411#        RTR   B,Y        INTO Y
0068 02412#        RTR   A,B        RESTORE RETURN ADDRESS
0069 02413         JMP   MO         PRINT NEWLINE
0070 02415 NP      ZA    NM
0071 02417 NM      ZC    240
0072 02420         ZC    215
0073 02421         ZC    212
0074 02422         ZC    000
0075 02423 PA      ZA    PB         POINTER ADDRESS
0076 02425 PB      ZA    DT-1,Y,I   POINTER TO DATA
0077       SR      ZL    1043       SAVE REGISTERS
0078       DP      ZL    2071
0079       MO      ZL    2044
0080       RR      ZL    1441
0081       BU      ZL    735
0082       TO      ZL    120
0083 02427 PP      ZA    PS
0084 02431 PS      ZC    240
0085 02432         ZC    240
0086 02433         ZC    240
0087 02434         ZC    240
0088 02435         ZC    240
0089 02436         ZC    240
0090 02437         ZC    000

*E*
```

SPC-12 program written to print out data (continued).