NUMERICAL PARAMETER ESTIMATION IN DIFFERENTIAL EQUATIONS

by

MAURICE W. BENSON

B.Sc., Lakehead University 1971
M.Sc., Lakehead University 1973

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science and The Institute for Applied Mathematics and Statistics)

We accept this thesis as conforming
to the required standard.

THE UNIVERSITY OF BRITISH COLUMBIA

December, 1977

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date December 12, 1977

ABSTRACT

The problem of numerical least squares parameter estimation in differential equations is considered. Several new algorithms that pay particular attention to the differential equation aspect of the problem are presented. These reduce some of the difficulties encountered when the problem is treated solely as a question of nonlinear optimization. The extremely powerful interactive approach is considered and an interactive package incorporating standard techniques using sensitivity equations along with a selection of our special algorithms is presented. We consider methods involving the fitting of integrals and derivatives using piecewise polynomial approximations to the observations. Continuation methods with a quasi multiple shooting technique to bridge the gap between these coarse but well behaved methods and the full least squares problem are explored.

Special methods are developed for the important case of two state variables with observations available on only one of them. In particular we consider algorithms which use an initial guess at the behavior of the unobserved state variable and then iteratively improve this guess.

The need for effective algorithms for fitting population growth models in ecology is one motivation for this thesis. We devote a chapter to an important predator-prey model of population dynamics and extensive experiments are presented

which demonstrate some of the typical difficulties which can arise and which illustrate the ability of our algorithms to overcome some of these difficulties.

Some special problems involving jumps from one equilibrium to another (loosely referred to as catastrophes) are examined. This type of model has important applications in ecology. Models involving stiff differential equations are also considered.

A short chapter is devoted to the use of sequential reestimation techniques. Experiments indicate that such methods can be useful for improving a crude initial guess at the parameters and this improvement can be crucial for the successful solution of the problem.

Finally a chapter is devoted to a selection of "real world" problems. It is on such problems that the true value of an algorithm is determined.

CONTENTS

FIGURES

TABLES

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor J. M. Varah, for his advice and encouragement during the preparation of this thesis. I would also like to thank Dr. C. J. Walters for keeping me aware of the "real world" aspect of the problem.

Finally, I would like to express my appreciation for National Research Council of Canada Scholarship support and Killam Pre-Doctoral Fellowship support during the preparation of this thesis.

INTRODUCTION

Parameter fitting in dynamic models occurs in a wide variety of fields (see for example Section 1.2). In many cases, standard procedures (employing the sensitivity equations as outlined in Chapter 2) suffice. However, in a substantial number of cases, this approach is extremely sensitive to poor initial guesses at the optimal parameters (see for example Chapter 4). The main purpose of this thesis is to develop algorithms and strategies designed to overcome poor or absent initial approximations to the optimal parameters. Our basic philosophy for attaining this goal is to avoid the full nonlinear optimization problem as much as possible during the early stages of parameter estimation. The interactive approach is ideal for addressing this problem and the algorithms developed in this thesis are designed with user intervention in mind. Indeed there are cases, such as the one in Section 6.3, where an interactive approach appears to be the only way to obtain certain solutions.

Chapters 1 and 2 establish our notation and provide a background for the numerical integration and optimization procedures employed throughout the thesis. In Chapter 3, we begin our development of special procedures.

We start Chapter 3 with the standard approaches of derivative and integral fitting and then we expand on these techniques later in the chapter. These techniques, which

involve the use of smoothed approximations to the observations on the state variables, are obvious candidates for an interactive approach. These methods are compared on a stiff problem and on a problem involving a change in equilibrium. In Section 3.4, we develop techniques (which employ guessed observations and iterative improvement of guessed observations) for extending the methods of Sections 3.2 and 3.3 to the important case when observations are not available on all state variables. As illustrated in Sections 4.4 and 4.5, these iterated methods provide a powerful tool (especially in an interactive environment) for handling the case where some state variables are unobserved. Again, our basic philosophy, of avoiding the full nonlinearity that arises with the direct approach, lies behind the success of these methods.

In Section 3.5, we present systematic approaches using break points and continuation parameters for bridging the gap between the relatively coarse integral fitting technique and the full nonlinear problem. These methods are highly interactive by nature, and again, they attempt to ease the approach to the full nonlinear problem. As shown in Section 3.5, these methods can be useful for overcoming instabilities.

We end Chapter 3 with a discussion on how the various methods developed should be incorporated into an interactive package. A discussion of effective strategies employing our special techniques in an interactive environment is also presented at the end of Chapter 3.

Chapter 4 gives extensive experiments comparing the techniques of Chapters 2 and 3 on a specific problem.

In Chapter 5, we continue our search for methods which reduce the effect of the nonlinearity associated with a direct approach. In particular a sequential approach is shown to be effective in several cases where a direct method encountered difficulties.

When solving a parameter fitting problem involving a dynamic model, the particular strategy employed can be as important as the choice of algorithms. This is especially the case when an interactive approach is used. In Chapter 6 we present details of successful strategies on four "real world" problems where the observations were obtained from physical experiments and not computer simulations. (A condensation of our experience with parameter fitting in dynamic models is contained in the flow charts at the end of Chapter 3.) The details of such strategies of course vary from problem to problem; however, as experience with such problems grows, certain strategies emerge as being more effective than others. One such strategy that has proved highly effective involves the temporary freezing of parameters.

CHAPTER 1

NOTATION AND BACKGROUND

1.1 INTRODUCTION AND BASIC NOTATION

We are interested in the problem of fitting dynamic models to observations and wish to pay special attention to the differential equation aspect of the problem. Our dynamic models are of the form

$$\begin{aligned} y' &= g(t,y,p) \\ y(t_0) &= y_0(p) \end{aligned} \tag{1.1.1}$$

where y is an n-vector of state variables, p is an m vector of parameters, t is the independent variable which we call time for convenience, and ' denotes differentiation with respect to time. Along with the above initial value problem we have a set of observations $v_1,\ldots,v_k$ taken at distinct times $t_1,\ldots,t_k$ respectively where $t_\ell \geq t_0$, $\ell=1,\ldots,k$. Each v_ℓ is an r-vector where $r \leq n$. That is, not all components of y need be observed.

Define the weighted residual vector f of length kr by

$$f_{r(\ell-1)+s} = w_{r(\ell-1)+s}\,(y_{i(s)}(t_\ell) - v_{\ell s}) \tag{1.1.2}$$

for $s=1,\ldots,r$; $\ell=1,\ldots,k$; where $v_{\ell s}$ is component s of v_ℓ and $y_{i(s)}(t_\ell)$ is the corresponding element of the vector $y(t_\ell)$. $w_{r(\ell-1)+s}$ is a weighting factor. The weighted least squares problem is to find p to minimize

$$F(p) = f^T(p)\,f(p). \tag{1.1.3}$$

The use of weights in the above function allows us to handle some maximum likelihood problems (see Section 2.4). Adapting the notation in Bard[5], let

$$e_\ell(p)=(f_{n(\ell-1)+1},\ldots,f_{n\ell})^T, \tag{1.1.4}$$

$$M(p)=\sum_{l=1}^{k} e_\ell(p)e_\ell^T(p). \tag{1.1.5}$$

where, for the moment, we are taking all weights equal to one. Further let the $\{v_\ell\}$ have normally distributed measurement errors with zero mean and covariance matrix V. When there are no errors in the t_ℓ, the maximum likelihood estimate of p is found by minimizing

$$.5\mathrm{Tr}(V^{-1}M(p)) \tag{1.1.6}$$

where Tr denotes the trace operator. (We are assuming the errors in the observations taken at different times are uncorrelated.) When V is a diagonal matrix, (1.1.6) reduces to the form of the function in (1.1.3). It is this special case we consider in Section 2.4. For a list of some other objective functions we refer the reader to Bard[5].

The dynamic model considered above is a special case of the standard dynamic model described by Bard[6,p221]. He considers problems where g and y of (1.1.1) are functions of a vector, x, of independent variables in addition to the above arguments and where the observations available are on variables which are given functions of t, y, p, and x. One of our aims in this

thesis is to investigate special methods which handle some of the difficulties associated with fitting parameters in a differential equation, and, to avoid unnecessary complications, we confine our attention to models of the form (1.1.1) with observations taken directly on the state variables.

The problem of finding p to minimize (1.1.3) can be nasty. Bard[6,p231] gives a concise description of some of the difficulties that can occur. Generally, the problem is difficult because of the vast range of solutions that (1.1.1) can have as a function of p. This can result in all sorts of local minima for (1.1.3). Stability problems with the differential equation can also arise. In some cases the solution to the initial value problem (1.1.1) is a discontinuous function of p and this can create difficulties. Also, many dynamic models are attempts to describe phenomena operating on different time scales and thus stiff initial value problems can be expected to arise in practice.

Parameter fitting in the predator-prey dynamic model

$$\begin{aligned} y_1' &= p_1 y_1 - p_2 y_1 y_2 / (1+p_5 y_1) - p_6 y_1^2 \\ y_2' &= -p_3 y_2 + p_4 y_1 y_2 / (1+p_5 y_1) \end{aligned} \tag{1.1.7}$$

is a typical example of the type of problem considered in this thesis. For more details concerning this dynamic system, we refer the reader to Bazykin[7] and to Chapter 4 of this thesis. As shown in Chapter 4 (which is devoted exclusively to this model) a poor initial approximation to the optimal parameters

can often lead to a local minimum in parameter space at which the solution to the above initial value problem is qualitatively quite different from the observations. Several methods are presented in Chapter 3 which are designed to overcome poor initial parameter estimates. Special methods are also developed in Chapter 3 for the important (and often difficult) case when observations are not available on all state variables. We also consider methods designed to overcome instabilities in the initial value problem by the use of continuation parameters and break points. In Chapter 4 we present extensive experiments with many of the techniques developed in Chapter 3 applied to the above dynamic model. In Chapter 5 we present a promising technique for improving poor parameter estimates using a sequential reestimation approach. Experiments with this technique applied to problems involving the above dynamic model are also given in Chapter 5.

Experience indicates that the successful resolution of a "real world" parameter fitting problem involving a dynamic model usually requires many optimization runs. Strategies such as freezing or rescaling parameters are also often useful. Frequently the model evolves as attempts are made to fit it to the data. It is thus desirable to rapidly acquire experience with a given model. We conclude that an interactive approach can be valuable for resolving, in a reasonable time, a parameter fitting problem involving a dynamic model. This is inherently expensive; however, as computer technology advances, the cost

factor becomes less important. It is our view that the first point to consider when designing a good interactive package is the set of numerical algorithms to be employed. The user-program interface should then be constructed to make optimal use of these algorithms. Of course the process works the other way too: interactive algorithms should be designed with a user-program interface in mind. One goal of this thesis is to develop dynamic model parameter fitting algorithms that exploit user interaction. Another goal is to organize a selection of these algorithms into an interactive package so that the various approaches developed complement one another. The end result of this work is the interactive package PARFIT documented in Appendix A.

1.2 MOTIVATION

Dynamic models occur extensively in practice. To be meaningful, such models must be related to physical observations and this often involves adjusting some parameters in the models. In this section we briefly describe some areas where parameter fitting in dynamic models is important.

Chemistry is one field where parameter fitting in dynamic models occurs that is frequently mentioned in the literature (see for example, Rosenbrock and Storey[61,p.189,p.204], Bard[6,p.222], van Domselaar and Hemker[71], and Bellman et al[9]). Typically a dynamic model is set up to describe a chemical reaction. The unknown parameters are reaction rates and the state variables represent concentrations of various

reactants. Since reaction rates can vary greatly, stiff dynamic models are important in chemistry. An example of such a model is considered in Section 3.2. The study of parameters in dynamic models with a large number of state variables is also of interest to chemists (see for example Farrow and Edelson[22]).

Many dynamic models occur in the fields of medicine and biology. Some of these models involve organic chemistry, while others are more directly related to biological processes. There are models describing enzyme activity in the blood (van Domselaar[70], van Domselaar and Hemker[71]). Models describing blood cell population dynamics are currently of interest (Mackey[42]). Parameter estimation in a model involving the electrical activity of the heart has been studied by Bellman et al[11]. Biological processes frequently operate on different time scales and thus stiff dynamic models are important here.

At present, there is a strong interest in dynamic models describing ecological processes. Several such models are considered in this thesis. Often ecological data is sparse and has a large random error and this makes parameter fitting difficult. Models describing predator-prey interactions occur extensively in ecology. In Chapters 4 and 6, we give several examples with such models using both simulated observations and observations obtained from physical experiments. For more background concerning parameter fitting in dynamic models in ecology, see for example Swartz and Bremermann[66], Vandermeer[69], Martin et al[44], Parker[53], Long[40].

1.3 MORE BACKGROUND

Following the observations of Bard[6,p220], we elect to use methods of the form

$$p^{(\ell+1)} = p^{(\ell)} - \rho_\ell R_\ell \nabla F(p^{(\ell)}) \qquad (1.3.1)$$

to minimize F(p) in (1.1.3) where ∇F represents the gradient of F, ρ_ℓ is a scalar, and R_ℓ is a matrix (usually positive definite). The Gauss-Newton (see for example [50,p.267]) and Levenberg-Marquardt[39],[43] methods are particular examples of such techniques. To apply these methods ∇F must be found. There are basically two ways this can be done. One way is to calculate ∇F through finite differences. Since the determination of F(p) involves the integration of an initial value problem, we expect the use of finite differences to approximate ∇F to be an expensive undertaking. More inportant, however, is the fact that with a dynamic model, the accurate calculation of ∇F by finite differences can be a tricky task. In particular let $\partial F(p)/\partial p_i$ be approximated by the difference

$$\frac{F(p+\Delta p_i\, e_i) - F(p)}{\Delta p_i} \qquad (1.3.2)$$

where e_i is the unit vector with a 1 in position i. As pointed out by Bard[6,p226], several factors affect how well this difference approximates $\partial F/\partial p_i$. For a good approximation, Δp_i must be small; however, if it is too small, rounding-error dominates and a poor approximation to the derivative is obtained. More important, however, is the fact that F(p) is

obtained by integrating an initial value problem and thus its accuracy depends on the discretization used and on the order of the method used to numerically solve the differential equation. Thus for example if $F(p+\Delta p_j e_j)$ and $F(p)$ are determined independently (each with its own discrete set of time values), they must be found with sufficient accuracy so that the difference approximation to $\partial F/\partial p_j$ is valid. This is an expensive undertaking and is not the proper way to proceed. It is more productive to think of F in terms of the discrete method used to find it. Thus at a given point p in parameter space, $\nabla F(p)$ is approximated by integrating (1.1.1) m+1 times using the same discrete set of time values. This avoids for example the possibility of getting discontinuities in F(p) due to varying sets of discrete steps at neighboring points in parameter space. Of course such a discontinuity would play havoc with the finite difference approximation to the gradient. Thus to calculate F(p) and approximate $\nabla F(p)$ requires m+1 integrations of (1.1.1), only one of which involves error control. However, (1.1.1) is often a nonlinear initial value problem.

The second alternative for calculating ∇F, which we shall use, employs the sensitivity equations. These are a set of linear initial value problems coupled in only one direction to the given initial value problem (1.1.1). In our notation they are

$$\begin{aligned} y'_{p_j} &= g_y y_{p_j} + g_{p_j} \\ y_{p_j}(0) &= \partial y(0,p)/\partial p_j \end{aligned} \qquad (1.3.3)$$

for $j=1,\ldots,m$ where the subscript p_j denotes partial differentiation with respect to p_j, and where g_y is the Jacobian matrix $(\partial g/\partial y)$. These equations may be obtained by differentiating (1.1.1) with respect to p. To use the sensitivity equations, g_y and g_p must be found; however, for a wide selection of important dynamic models this is not too difficult a task. For more details on sensitivity equations see Tomović[67], Tomović and Vukobratović[68]. In Chapter 2, further details are given on the integration of the sensitivity equations. We observe that for models of the form (1.1.1), the solution to the sensitivity equations immediately gives $\nabla F(p)$. Following the notation used in (1.1.2), denote by J the kr x m Jacobian matrix of f with respect to p. The elements of J are given by

$$J_{r(\ell-1)+s,\,j}(p)=w_{r(\ell-1)+s}\frac{\partial y_{s(s)}}{\partial p_j}(t_\ell) \qquad (1.3.4)$$

$$s=1,\ldots,r;\ l=1,\ldots,k;\ j=1,\ldots,m,$$

and the gradient of F(p) is

$$\nabla F(p)=2J(p)^T f(p). \qquad (1.3.5)$$

We note that in the process of finding ∇F, we have found part of the Hessian matrix of F(p). (The Hessian matrix is

$$2\left(J^TJ+\sum_{q=1}^{kr} f_q G_q\right)$$

where G_q is the matrix of second partials of f_q with respect to p.)

Beale[8] distinguishes three basic problems in nonlinear parameter estimation. First there is the problem of determining the optimal parameter vector p. This is primarily a matter of numerical analysis and it is the problem we concentrate on. Second there is the problem of defining a theoretically satisfactory confidence region or an approximate confidence region for p. Third, Beale identifies the problem of describing this confidence region so that it can be easily interpreted. Beale deals extensively in [8] with the second problem.

Although we are primarily concerned with the first problem in this thesis, some attention to the second and third problems is mandatory since a measure of the reliability of parameters greatly enhances their value to the model builder. Also, as seen in Chapter 4, a study of some of the statistical aspects of the problem can be useful for detecting linear relationships among parameters in a model. Following Bard[6,p.187], we make the

DEFINITION 1.3.1:

The γ-joint confidence region is a bounded closed subset S(W) in parameter space depending on the data sample W such that

$$\Pr[p^* \in S(W)] = \gamma \qquad (1.3.6)$$

for all possible data samples W where p^* is the exact (and unattainable) value for the parameter vector, and Pr denotes probability.

Specifically, we use for S(W) an m dimensional ellipsoid

centred at the estimate $\bar{p}$ of p^*. Following Bard[6,pl89], we approximate $f(p)$ defined in (1.1.2) by a linear function in a neighborhood of p:

$$f(p) \simeq f(\bar{p}) + J(\bar{p})(p-\bar{p}). \tag{1.3.7}$$

We assume the errors in the observations are normally distributed with zero mean. Form the theory of multiple linear regression with $\underline{V}$ representing the kr x kr covariance matrix for the observations, the variable

$$\mathcal{J} = (p-\bar{p})^T (J^T(\bar{p}) \underline{V}^{-1} J(\bar{p}))(p-\bar{p}) \tag{1.3.8}$$

has a χ^2 distribution with m degrees of freedom. (We are considering the general objective function $f^T \underline{V}^{-1} f$.) Unless stated otherwise, J is evaluated at $\bar{p}$ in the following discussion. Also,

$$\mathcal{J} = f(\bar{p})^T \underline{V}^{-1} f(\bar{p}) \tag{1.3.9}$$

has a χ^2 distribution independent of $\mathcal{J}$ with kr-m degrees of freedom. Thus

$$\frac{(kr-m)\mathcal{J}}{m\mathcal{J}}$$

has an $\underline{F}_{m,kr-m}$ distribution (m degrees of freedom in the numerator, kr-m degrees of freedom in the denominator). In particular when the covariance matrix $\underline{V}$ is of the form $\sigma^2 I$, (that is, when all observations are independent) then

$$\frac{(kr-m)\delta}{m\delta} = \frac{(kr-m)(p-\bar{p})^T J^T J(p-\bar{p})}{mf(\bar{p})^T f(\bar{p})}$$

$$= \frac{(kr-m)(p-\bar{p})^T J^T J(p-\bar{p})}{mF(\bar{p})} \qquad (1.3.10)$$

Thus the γ-joint confidence region is

$$\left\{p: \frac{(kr-m)(p-\bar{p})^T J^T J(p-\bar{p})}{mF(\bar{p})} \leq F_{m,kr-m}(\gamma)\right\} \qquad (1.3.11)$$

which is an m dimensional ellipsoid in parameter space. Following van Domselaar and Hemker[71], we can use the singular value decomposition of $J(\bar{p})$ to extract further information on this ellipsoid. Let

$$J(p)=Q(p)\Sigma(p)R^T(p) \qquad (1.3.12)$$

where Q and R are orthogonal matrices of size kr x m and m x m and Σ is the mxm diagonal matrix $[\mathrm{diag}(s_1,\ldots,s_m)]$ of singular values arranged in descending order of magnitude. Let

$$\frac{mF_{m,kr-m}(\gamma)F(\bar{p})}{kr-m} = \varepsilon \quad ,$$

$$\delta q=R^T(\bar{p})(p-\bar{p}) . \qquad (1.3.13)$$

Our ellipsoid may thus be written

$$\{\delta q : \delta q^T \Sigma^2 \delta q \leq \varepsilon\} . \qquad (1.3.14)$$

Thus the principal axes of the confidence region have lengths $\sqrt{\varepsilon}/s_j$, j=1,...,m. For our confidence intervals on p_j, j=1,...,m we take the projections of the above ellipsoid onto the

coordinate axes in p-space with the origin translated to the estimate $\bar{p}$. Thus the confidence interval for $\bar{p}_j$ is

$$[\bar{p}_j - \sqrt{\epsilon(J^T(\bar{p})J(\bar{p}))^{-1}_{jj}}\ ,\ \bar{p}_j + \sqrt{\epsilon(J^T(\bar{p})J(\bar{p}))^{-1}_{jj}}\ ]. \qquad (1.3.15)$$

This is the confidence interval we use in our program PARFIT described in Appendix A. To find $(J^TJ)^{-1}$ we use

$$(J^TJ)^{-1} = R\,\Sigma^{-2}R^T. \qquad (1.3.16)$$

The expected value of $\bar{p}-p^*$ is zero and in the special case considered above when $\underline{V}=\sigma^2 I$, the covariance matrix for $\bar{p}-p^*$ is

$$E((\bar{p}-p^*)(\bar{p}-p^*)^T) = \sigma^2(J^TJ)^{-1} \qquad (1.3.17)$$

where E(.) denotes expectation (see for example van Domselaar and Hemker[71], Bard[6,p.59]). This matrix can be found using (1.3.16). The matrix of correlation coefficients has elements

$$\rho_{ij} = \frac{(J^TJ)^{-1}_{ij}}{\beta_i\,\beta_j} \qquad (1.3.18)$$

where $\beta_i^2 = (J^TJ)^{-1}_{ii}$. Thus using the singular value decomposition,

$$\rho_{ij} = \cos(S_i\ ,\ S_j) \qquad (1.3.19)$$

where S_i, S_j are the i'th and j'th row vectors of $S=R\Sigma^{-1}$.

Finally we note that the above confidence intervals were derived under the assumption that f(p) could be well approximated by a linear function near $\bar{p}$. This is often not the case. Bard[6,p191] gives a simple empirical way of checking this. We want the linearity approximation to hold over the confidence region that has been found. That is F(p) should be

near

$$F(\bar{p}) + .5(p-\bar{p})(J^TJ)(p-\bar{p})$$

in the confidence region. This can easily be checked at the boundary of the confidence region. We also note that when $F(\bar{p})$ is large (and f is only moderately nonlinear), then J^TJ is a poor approximation to the Hessian matrix of $F(\bar{p})$ and the quadratic approximation to F(p) stated above cannot be very good. Thus the confidence region stated above loses validity as the residuals increase. This can also be seen by observing that ξ in (1.3.13) varies directly as $F(\bar{p})$, and thus as $F(\bar{p})$ increases, the confidence intervals can quickly become larger than the parameters themselves. For further comments on this case see Rosenbrock and Storey[61,p202].

We end this chapter with a brief discussion on conditioning. There is a close connection between nonlinear and linear least squares problems. For example, the Gauss-Newton iterative method for solving a nonlinear least squares problem can be viewed in terms of a sequence of linear least squares problems. That is given an estimate $p^{(g)}$ for the parameter vector which minimizes (1.1.3), we seek a new estimate $p^{(g+1)}$ for this minimizing parameter vector such that

$$||f(p^{(g)}) + J(p^{(g)})\,s^{(g+1)}||_2^2 \qquad (1.3.20)$$

is minimized where

$$s^{(g+1)} = p^{(g+1)} - p^{(g)} . \qquad (1.3.21)$$

(In practice a robust least squares optimization procedure must do more. For example, it should guarantee a reduction in the sum of the squares of the residuals after each iteration.) For simplicity of notation in the following discussion, we neglect superscripts and we let $-s^{(g+1)}=x$. Also, we assume J is of full rank. The condition number of the matrix J is defined to be $\chi(J)=||J||_2||J^\dagger||_2$ where $J^\dagger=(J^TJ)^{-1}J^T$ is the pseudo-inverse of J.

First we consider the effect of small errors in f. The vector $\bar{x}$ which minimizes

$$||f-Jx||_2^2 \tag{1.3.22}$$

is given by $\bar{x}=J^\dagger f$. When J is of full rank $\bar{x}$ is unique. Let $\hat{f}$ be an approximation to f and let f_p and $\hat{f}_p$ be projections of f and $\hat{f}$ onto the range space of J. Provided $f_p \neq 0$, it follows (see for example Stewart[65,p.221])

$$\frac{||J^\dagger f-J^\dagger \hat{f}||_2}{||J^\dagger f||_2} \leq \frac{\chi(J)||f_p-\hat{f}_p||_2}{||f_p||_2} \tag{1.3.23}$$

Thus when J is ill-conditioned ($\chi(J)$ is large) or when the projection of f onto the range space of J is small, relatively small errors in f can have a strong influence on the accuracy with which we can determine $\bar{x}$.

The effects of errors in J are much more complicated. The following theorem (see for example Stewart[65,p.223]) addresses this situation.

THEOREM 1.3.1:

Let J be of full rank and let f be defined as above. Let E

be a matrix of the same dimensions as J and let E_ρ, f_ρ and $\hat{E}_\rho$, $\hat{f}_\rho$ be projections of E and f onto the range space of J and the orthogonal complement of the range space of J respectively. (The projection of a matrix onto a subspace is defined to be the matrix whose i'th column is the projection of the i'th column of the given matrix onto the subspace.) If

$$||J^\dagger||_2 ||E_\rho||_2 \leq .5$$

then J+E is of full rank and

$$\frac{||\bar{x}-\hat{x}||_2}{||\bar{x}||_2} \leq 2\chi \frac{||E_\rho||_2}{||J||_2} + 4\chi^2 \frac{||\hat{E}_\rho||_2}{||J||_2} \frac{||\hat{f}_\rho||_2}{||f_\rho||_2} + 8\chi^3 \frac{||\hat{E}_\rho||_2^2}{||J||_2^2} \quad (1.3.24)$$

where $\hat{x}=(J+E)^\dagger f$ and $\bar{x}=J^\dagger f$.

As noted by Stewart[65,p.224], if f is almost in the range space of J then χ is the condition number of the least squares problem, while if $||\hat{f}_\rho||_2/||f_\rho||_2$ is large then χ^2 is the effective condition number of the least squares problem. Thus depending on the orientation of f with respect to the range space of J, the least squares problem can be extremely sensitive to the condition number of J.

The condition number of J can be influenced by certain transformations in parameter space. At times, such transformations can drastically reduce the condition number; however, on other occasions they can worsen the conditioning of a problem. For example consider the logarithmic transformation of p_j where we transform to $\hat{p}_j=\ln(p_j)$. We consider the effect of this transformation on the conditioning of the stiff problem

$$y_1' = -(1-y_2)y_1 + p_2 y_2$$
$$y_2' = p_1((1-y_2)y_1 - (p_2+p_3)y_2) \qquad (1.3.25)$$

with initial condition

$$y(0) = (1,0)^T$$

at the point

$$(1000,\ .99,\ .01)^T$$

in parameter space. Without any scaling, the condition number was approximately 59600. The matrix J was evaluated using the observation times given in Section 3.2 where this initial value problem is given further consideration. The trapezoidal integration scheme available in the package PARFIT described in the appendices was used to approximate J. For this problem, a logarithmic scaling of the parameters reduced the condition number to approximately 4.4. There are cases, however, where this scaling increases the condition number (this occurs for example, in certain cases with exponential fitting problems), and thus it must be used with caution.

CHAPTER 2

OPTIMIZATION AND INTEGRATION

2.1 NONLINEAR LEAST SQUARES

Our goal is to concentrate on the differential equation aspect of parameter fitting in dynamic models and not to compare the fine points of various optimization algorithms. However, the nonlinear least squares problem occurs repeatedly in this thesis and thus a brief discussion of this problem is in order. When choosing a method, it is important to keep in mind that the efficiency and reliability of an optimization program depend on both the particular algorithm and the details of its implementation.

We must choose between methods designed specifically for least squares problems, and more general methods designed for nonlinear optimization. For our particular problem the calculation of first derivatives is often expensive (through the sensitivity equations for example) and the calculation of second derivatives is even worse. Thus we elect to use only first derivative methods and we must settle for approximations to the Hessian matrix. Some comparisons between least squares methods and more general optimization methods applied to least squares problems (Bard[5], Bus et al[14]) indicate special least squares methods are to be preferred. We recognize, however, that there are cases when a more general method can be superior (see Ramsin and Wedin[57], McKeown[46]). The results of Bus et al favor the

Levenberg-Marquardt method ([39], [43]) over more general optimization methods and over the Gauss-Newton method, (we include under the name Gauss-Newton, modified Gauss-Newton algorithms employing step length adjustment) while Bard finds his implementation of the Levenberg-Marquardt method to be as reliable but slightly less efficient than his implementation of the Gauss-Newton method. Bard, however, employs some special techniques in his implementation of the Gauss-Newton algorithm that are absent in the experiments of Bus et al. This further points out the sensitivity of test results to particular details of implementation. Of course all results are also problem dependent. The Levenberg-Marquardt method appears to be a good algorithm for the nonlinear least squares problem and we employ it extensively in this thesis. Specifically we iterate according to

$$p^{(\ell+1)} = p^{(\ell)} - (J^T(p^{(\ell)})J(p^{(\ell)}) + \lambda I)^{-1} J^T(p^{(\ell)}) f(p^{(\ell)}) \tag{2.1.1}$$

where J, f, p were defined in Section 1.1 and λ is a positive parameter chosen so that the sum of the squares of the residuals is reduced by the above iteration.

In the case when J^TJ is positive definite (that is, when J is of full rank), we have the following important facts about this technique (see Marquardt[43]).

(1) Let $\delta^{(\ell+1)} = p^{(\ell+1)} - p^{(\ell)}$. That is

$$(J^TJ + \lambda I)\delta = -J^Tf \tag{2.1.2}$$

where we have dropped the superscripts. It follows that

$$\delta_0 = -(J^T J + \lambda I)^{-1} J^T f \tag{2.1.3}$$

minimizes

$$||J\delta + f||_2^2 \tag{2.1.4}$$

over

$$\{\delta : ||\delta||_2 = ||\delta_0||_2\}. \tag{2.1.5}$$

(2) For δ satisifying (2.1.2), $||\delta(\lambda)||_2^2$ is a continuous monotone decreasing function of λ and

$$\lim_{\lambda \to \infty} ||\delta(\lambda)||_2^2 = 0. \tag{2.1.6}$$

(3) The quantity

$$\gamma = \cos^{-1}(\delta \cdot \delta_g) \tag{2.1.7}$$

where $\delta_g = -J^T f$, is a continuous monotone decreasing function of λ and

$$\lim_{\lambda \to \infty} \gamma = 0. \tag{2.1.8}$$

Thus as $\lambda \to \infty$, the descent direction δ given by (2.1.2) approaches the steepest descent direction, given by $-J^T f$, and its magnitude approaches zero. This can create problems if this algorithm is not carefully implemented. That is, we do not want λ to become so large that we are taking very small steps in a direction that is essentially the steepest descent direction.

To overcome this possible drawback, a check is made to see if δ is within 45° of the steepest descent direction. If it is, λ is not increased but instead a search in the direction of the current δ is carried out to obtain a better parameter vector. Marquardt[43] outlines the necessity of using this strategy.

The following theorem (see Dennis[19]) gives conditions for the local convergence of the Levenberg-Marquardt method.

THEOREM 2.1.1:

let $\bar{p}$ be a local minimum of $F(p)$ and let μ be the smallest eigenvalue of $J(\bar{p})^T J(\bar{p})$. Let γ be a scalar such that for all p in a neighborhood of $\bar{p}$,

$$||(J(p)-J(\bar{p}))^T f(\bar{p})||_2 \leq \gamma ||p-\bar{p}||_2 \qquad (2.1.9)$$

If $\gamma < \mu$ then for any bounded sequence $\{\lambda^{(s)}\}$ of real numbers, there exists a neighborhood of $\bar{p}$ such that if $p^{(0)}$ is in this neighborhood, the Levenberg-Marquardt iterations defined by $p^{(0)}$ and $\{\lambda^{(s)}\}$ converge to $\bar{p}$.

For further reading concerning the convergence of the Levenberg-Marquardt algorithm see Osborne[51],[52].

There is a lot of choice available in a particular implementation of the Levenberg-Marquardt method. Much of this centres around the strategy for adjusting λ. For a summary of some of the strategies we refer the reader to Van Loan[72].

Marquardt[43] suggests that we rescale parameter space at each iteration in such a way that $J^T J$ in the scaled parameters has diagonal elements equal to one. The rationale for this

scaling is that the Levenberg-Marquardt method is biasing the descent direction towards the steepest descent direction and the steepest descent direction is scale dependent. Our implementation of the Levenberg-Marquardt technique has this scaling available as an option. Experimental results indicate that at times this scaling can be quite effective. A listing of our implementation is given in Appendix B.

Next, we mention a couple of recent developments in nonlinear least squares problems which, when they become more fully understood, may be very useful for our problem where function and gradient evaluations are expensive.

Steen and Byrne[64] propose an interesting nonlinear least squares algorithm which adjusts the descent direction between the steepest descent and Gauss-Newton directions in a complicated, but (experimentally) apparantly more efficient way than that of the Levenberg-Marquardt method. Also, their algorithm does not suffer from the stepsize going to zero as the steepest descent direction is approached. This algorithm is of particular interest to us because it requires a substantially fewer number of function evaluations than does the Levenberg-Marquardt method on a fairly wide range of test problems considered in [64]. On most of the test problems considered, this method is superior to the similar SPIRAL method proposed by Jones[34].

The second recent development of particular interest involves work on the large residual least squares problem. In

this case J^TJ is no longer a good approximation to the Hessian matrix, and the Gauss-Newton and Levenberg-Marquardt methods are slowed down. This situation can arise when we have a lot of noise in the observations, when we have a poor model, or when our least squares algorithm is converging to a local minimum with a large residual. (As the experiments of Chapter 4 indicate, numerous local minima can occur when we try to estimate parameters in dynamic models.) Recently, there has been some interesting work done on special algorithms which approximate the second derivatives in the Hessian of a least squares problem by techniques modelled after the quasi-Newton methods. (See for example Dennis[18,p.171,177], Dennis et al[20].) When the reliability of such methods increases, their application to parameter fitting problems in dynamic models seems worthwhile. If a reduction is achieved in the number of numerical integrations required, the extra work invested to approximate the Hessian matrices should be well worth it. For a good discussion of recent work in nonlinear least squares, we refer the reader to Dennis[19].

2.2 INTEGRATION OF MODEL AND SENSITIVITY EQUATIONS

In the course of fitting parameters in a dynamic model, several <u>different</u> initial value problems must be solved. That is, every time the parameter vector changes, a new initial value problem must be solved. Moreover, the solutions to these problems can vary drastically. Consider, for example, the algae growth model of Section 6.3 where small changes in the parameter

values can produce huge changes in the solution to the initial value problem. To handle such problems, a parameter fitting algorithm requires a good general purpose numerical package for solving initial value problems. The ability to handle parameter estimation problems involving stiff dynamic models is also important in practice. For several examples with such problems we refer the reader to van Domselaar and Hemker[71]. In this thesis, we employ the automatic integration package developed by Gear[25], [26], [27]. In addition to the standard predictor corrector methods, this package has available a set of stiffly stable multistep methods of varying order. In our package, PARFIT, the user can easily switch between a regular multistep method and a stiffly stable method.

When fitting parameters in a dynamic model, the need for a general integration program is clear; however, often a much simpler integration procedure is adequate. Of course to mimimize the error introduced into the parameter estimates by the discretization of the initial value problem, an integration scheme with error control should be used. In the interests of economy, it is advisable to start with a simple integration scheme not employing stepsize control if this is possible. In particular we make use of the trapezoidal method without error control. In PARFIT, the discrete times used by the trapezoidal method are the "sample times" which by default are the observation times; however, the sample times can be interactively modified. Use of the trapezoidal method allows us

to handle stiff problems. Finally we note that other integration schemes can easily be added to PARFIT.

In addition to integrating the given initial value problem, the sensitivity equations defined in Section 1.3 must also frequently be integrated. The way the sensitivity equations (which are coupled in only one direction to the given initial value problem) are integrated can be crucial to the success of a parameter fitting algorithm. For example, as observed by Bard[6,p.231], the integration of the sensitivity equations at a point p in parameter space should not have any influence on the discrete steps used in the integration of (1.1.1) at p. That is, F(p) should be independent of whether or not gradient information is extracted at p. The sensitivity equations in theory provide a means of determining how the continuous solution y(t) to the given initial value problem varies as a function of the parameter vector p. However, in practice the given initial value problem can only be solved approximately according to some discrete model (for example the trapezoidal method). For the purposes of numerical optimization, it is the solution to this discrete analog of the continuous problem that is being fit to the observations, and when finding the gradients of the objective function we really want to know how the solution to the discrete approximation to the initial value problem varies as a function of p at the observation times. Thus for the purposes of numerical optimization, the way the sensitivity equations are integrated should be related to the

solution method of the original initial value problem. This is especially important if a coarse approximation to the given initial value problem is used.

There is an analog to the above considerations in optimal control problems where the discrete approximation to the adjoint equations should be tailored to the particular discretization of the state equations. This can be a tedious undertaking in control problems as demonstrated by Kelly and Denham[35]. However, the main difficulty with control problems arises because the state equations are integrated forward in time and the adjoint equations are integrated backward in time. Fortunately for our purposes since we are integrating in only one direction in time, the relation of the discrete sensitivity equations to the discrete state equations is much simpler. In particular all we must do is to ensure that the same method with the same discrete set of time values is used on the sensitivity equations as is used on the state equations. To be more specific, the diagram in Figure 2.2.1 must commute where S_c represents the operator which produces sensitivity equations from an initial value problem, D_i represents a discretization operator, and S_d represents the operator which produces discrete sensitivity equations from discrete state equations. In our numerical integration procedures we take advantage of the one way coupling between the sensitivity equations and the state equations. Specifically, we integrate the state equations from time t to time t+h (under error control if applicable). Then

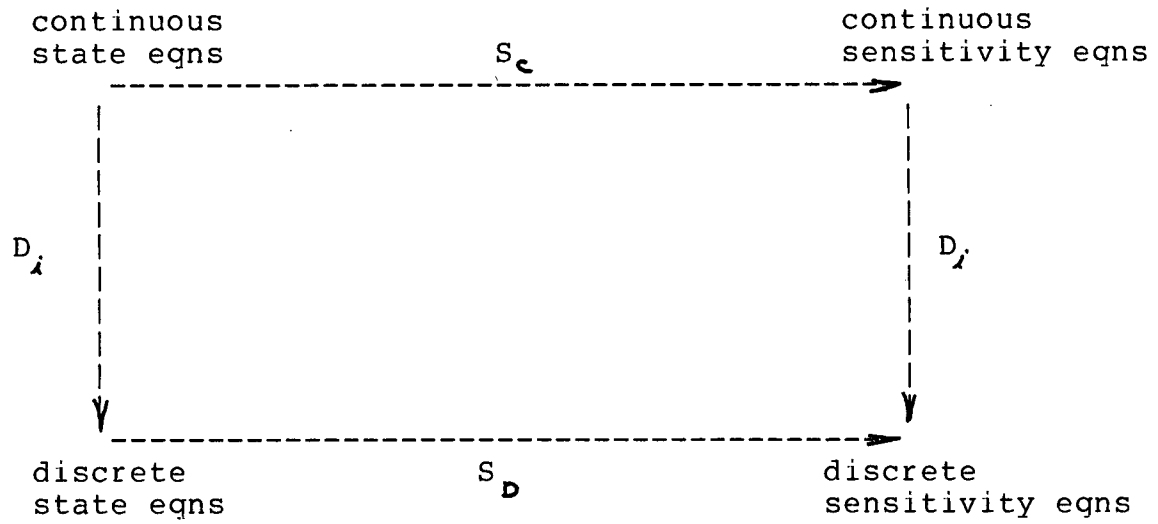

Figure 2.2.1
Discretization of sensitivity equations

assuming there are m parameters and n state equations, we integrate each of the m sensitivity initial value problems, each having n equations, from time t to time t+h using exactly the same discretization that was used to integrate the state equations. This is consistent with the comments by Bard[6,p.231] mentioned earlier in this section. We note that the sensitivity equations are linear initial value problems. When Gear's program is used, the above technique involves extracting information on the current stepsize and order from the integration package, and then integrating the sensitivity equations over the same step with the same order method. Gear's program employs the Nordsieck[49] formulation of a multistep method where approximations to higher derivatives are stored

instead of previous values. This makes changing the stepsize very easy. Our integration of the sensitivity equations employs exactly the same technique in phase with Gear's package. For more details see Appendix B. No error control is used when integrating the sensitivity equations. As observed by van Domselaar and Hemker[71], an integration scheme has the same stability properties on the sensitivity systems as it has on the system of state equations.

2.3 WHEN TO INTEGRATE THE SENSITIVITY EQUATIONS

We must decide when to integrate the sensitivity equations. Two strategies are considered. These extra linear equations may be integrated every time (1.1.1) is integrated (strategy one), or they may be integrated only when the Jacobian matrix and the gradient are required (strategy two). We expect the choice of strategy to depend in part on the implementation details of the optimization algorithm employed. For example, if several objective function evaluations are required at each iteration, then the first strategy would clearly be inefficient. On the other hand, if only one objective function evaluation is required per iteration (this is generally not the case, although at times several iterations follow this pattern), then strategy one would hold the advantage. To further complicate matters, the integration of the sensitivity equations is generally easier than the integration of (1.1.1) which is usually nonlinear and at times even stiff. Also, the step size adjustments are made when integrating (1.1.1) and not when integrating the

sensitivity equations.

To aid in a more detailed analysis of the two alternatives, we make the following definitions.

(a) Let W denote the basic unit of work for a particular problem defined as the work required to integrate (1.1.1).

(b) Let w denote the fraction of the work W required to integrate one sensitivity equation.

(c) Let m be the number of sensitivity equation systems. This is just the number of parameters.

(d) Let I be the number of iterations in the optimization run.

(e) Let J be the number of objective function evaluations where no gradient is required. These are the evaluations in the searches during the iterations.

Under the first strategy, the number of times the initial value problem (1.1.1) is solved is

$$I+J-(I-1)=J+1 \tag{2.3.1}$$

and this is also the number of times the sensitivity equations are solved. Therefore the total function evaluation work done in solving for the optimal parameters is

$$W_1=(J+1)W+m(J+1)wW. \tag{2.3.2}$$

Under the second strategy, the number of times the initial value problem is solved is I+J, and the number of times the sensitivity equations are solved is I. Therefore the total function evaluation work done in solving for the optimal parameters is

$$W_2 = (I+J)W + mIwW. \tag{3.3.3}$$

Thus

$$\begin{aligned} W_1/W_2 &= (J+1)(mw+1)/(I+J+mIw) \\ &= (J/I+1/I)(mw+1)/(1+J/I+mw). \end{aligned} \tag{2.3.4}$$

We neglect 1/I compared to J/I. This is equivalent to ignoring the starting function evaluation in the first strategy and thus in most cases, this approximation is not too significant. Thus

$$W_1/W_2 \simeq J/I(mw+1)/(J/I+(mw+1)) \tag{2.3.5}$$

Table 2.3.1 contains some values for W_1/W_2 as a function of J/I and mw. Entries less than 1 correspond to cases where it is

J/I \ mw	.5	1	2	4	8
1	0.6	0.667	0.75	0.833	0.9
1.2	0.667	0.75	0.857	0.968	1.06
1.5	0.75	0.857	1.0	1.15	1.29
2	0.857	1.0	1.2	1.43	1.64

Table 2.3.1
Work ratios W_1/W_2

more efficient to integrate the sensitivity equations each time the initial value problem is integrated.

To carry this analysis further, we need some typical values for w. Consider Euler's method with no stepsize adjustment. Assume there are n state variables and let the work required to evaluate one component of a vector function or one element of a Jacobian matrix be W_c. The initial value problem (1.1.1) is

$$y'=g(t,y,p)$$

and the sensitivity equations are of the form

$$y'_{p_j} = g_y(t,y,p)y_{p_j} + g_{p_j} \quad ,j=1,\ldots,m.$$

We do not make any special allowances for parameters that occur only in the initial conditions (thus making $g_{p_j}=0$) in the following analysis. Also, additions and multiplications associated with taking an Euler step are neglected. The work required to advance one time step in the solution of (1.1.1) is nW_c, and the work required to advance one step in the solution of one sensitivity equation system is $(n^2/m+n)W_c$. Thus

$$mw=(n^2+mn)/n=n+m. \tag{2.3.6}$$

In the case of Euler's method without error control, it seems advisable to solve the sensitivity equations only when required. The same conclusion applies to explicit multistep methods without error control since to advance one step in time requires only one evaluation of the vector function on the right hand side of (1.1.1) (see for example Gear[25,p.104]).

The situation is, however, different with implicit multistep methods. Here a generally nonlinear system of algebraic equations must be solved for each advance of one time step when integrating (1.1.1). If a predictor-corrector method is used to solve the these nonlinear systems of equations then typically two or three corrections are required at each time step (Gear[25,p.114]). Assume the predictor-corrector method

ends with an evaluation of y' at the new time value. To be concrete, assume four evaluations of g are required per time step. To solve the implicit equations associated with the integration of the sensitivity equations, no extra function evaluations beyond those required for an explicit multistep method are needed. However, m linear systems of equations each involving the same nxn matrix must be solved. This takes on the order of $n^3/3+mn^2$ multiplications (see for example Stewart[65,p.136]). The number of multiplications required to evaluate a component of g can vary greatly between problems. To describe this variation, let W_c require l multiplications. We will vary l in the following analysis. The work required to advance one time step , without error control, in the solution of (1.1.1) is

$$4nW_c=4nl \text{ multiplications} \tag{2.3.7}$$

and the number of multiplications required to advance one step in the solution of a sensitivity equation is

$$(n^2/m+n)l+n^3/(3m)+2n^2. \tag{2.3.8}$$

The multiplications required to form $g_y y_p$ have been included in the above estimate. Thus

$$mw=((n+m)l+n^2/3+2nm)/(4l). \tag{2.3.9}$$

Some values for mw and for W_1/W_2 are given in Table 2.3.2. The ordered pair (m,n) is given below the mw values. From Table 2.3.2, we see there are cases when it is advantageous to

	l=5				l=25			
mw / J/I	1.92 (3,2)	3.92 (3,5)	3.27 (6,2)	6.17 (6,5)	1.38 (3,2)	2.38 (3,5)	2.25 (6,2)	3.43 (6,5)
1.0	0.745	0.831	0.810	0.878	0.704	0.772	0.765	0.816
1.2	0.850	0.965	0.937	1.03	0.798	0.886	0.877	0.944
1.5	0.991	1.15	1.11	1.24	0.921	1.04	1.03	1.12
2.0	1.19	1.42	1.36	1.56	1.09	1.26	1.24	1.38

Table 2.3.2
Some work ratios for a predictor-corrector method

integrate the sensitivity equations every time the nonlinear initial value problem is integrated. This strategy would be even more desirable if an error control were used on the nonlinear initial value problem.

The case when the initial value problem is stiff requires special attention. In this case it is desirable for stability reasons to use an implicit integration scheme; however, a predictor-corrector method can require very small step sizes to converge and thus a Newton-like method is indicated (Gear[25,p.216]). This requires information on g_y; however, the exact g_y is not required and the usual strategy is to update g_y only when necessary (see Gear[25,p.217]). To be concrete, we assume α evaluations of g_y are required per time step in the solution of the given initial value problem (usually $\alpha < 1.$). There is a further complication in finding work estimates in the stiff case. If the sensitivity equations are to be integrated then we must find g_y and the factors of the associated linear

system matrix at each time step. Thus this information is freely available when we are integrating the given dynamic system. In the following estimates we assume no advantage is taken of this free information. We have

$$mw=((n+m)1+n^2/3+2nm)/(4(n+1)+\alpha(nl+n^2/3)). \qquad (2.3.9)$$

In Tables 2.3.3 and 2.3.4, results analogous to those in Table 2.3.2 are given for the cases $\alpha=.3$ and $\alpha=.6$ respectively.

We conclude that for implicit methods, it can be advantageous to follow strategy one, especially when the problem is stiff. Of course the optimal strategy is strongly dependent on the optimization method used and for strategy one to be best, exact searches at each iteration should not be made. Finally we note that from the results on ten test problems considered in [5], $J/I\simeq 1.2$ for the Levenberg-Marquardt method.

	l=5				l=25			
mw / J/I	1.22 (3,2)	1.57 (3,5)	2.08 (6,2)	2.47 (6,5)	1.12 (3,2)	1.49 (3,5)	1.83 (6,2)	2.15 (6,5)
1.0	0.690	0.720	0.755	0.776	0.680	0.713	0.739	0.759
1.2	0.779	0.818	0.864	0.891	0.766	0.810	0.842	0.869
1.5	0.895	0.947	1.01	1.05	0.879	0.936	0.980	1.02
2.0	1.05	1.12	1.21	1.27	1.03	1.11	1.17	1.22

Table 2.3.3
Work ratios for a stiff method (α=.3)

	l=5				l=25			
mw / J/I	1.10 (3,2)	1.31 (3,5)	1.88 (6,2)	2.06 (6,5)	0.997 (3,2)	1.19 (3,5)	1.62 (6,2)	1.72 (6,5)
1.0	0.678	0.697	0.742	0.753	0.666	0.687	0.724	0.731
1.2	0.764	0.789	0.847	0.862	0.750	0.775	0.823	0.832
1.5	0.875	0.909	0.986	1.01	0.857	0.891	0.954	0.966
2.0	1.02	1.07	1.18	1.21	0.999	1.05	1.13	1.15

Table 2.3.4
Work ratios for a stiff method (α=.6)

2.4 AN EXAMPLE WITH KNOWN DIAGONAL COVARIANCE MATRIX

Consider the initial value problem

$$
\begin{aligned}
y_1' &= -p_1 y_1 y_2 + p_2 y_3 \\
y_2' &= -p_1 y_1 y_2 + p_2 y_3 - p_4 y_2 y_3 + p_5 y_5 - p_6 y_2 y_4 \\
y_3' &= p_1 y_1 y_2 - p_2 y_3 - p_3 y_3 - p_4 y_2 y_3 \\
y_4' &= p_3 y_3 + p_5 y_5 - p_6 y_2 y_4 \\
y_5' &= p_4 y_2 y_3 - p_5 y_5 + p_6 y_2 y_4
\end{aligned}
\tag{2.4.1}
$$

with the initial condition

$$y(0)=(1,\ 1,\ 0,\ 0,\ 0)^T. \qquad (2.4.2)$$

This problem is considered by Bard[5]. Observations used by Bard on all five state variables are given in Table 2.4.1. The

Time	Observations				
	y_1	y_2	$10y_3$	$1000y_4$	$1000y_5$
12.5	.945757	.961201	.494861	.154976	.111485
25	.926486	.928762	.690492	.314501	.236263
37.5	.917668	.915966	.751806	.709300	.311747
50	.928987	.917542	.771559	1.19224	.333096
62.5	.927782	.920075	.780903	1.68815	.340324
75	.925304	.912330	.790539	2.19539	.356787
87.5	.925083	.917684	.783933	2.74211	.358283
100	.917277	.907529	.779259	3.20025	.361969

Table 2.4.1
Observations

problem we consider, designated problem 3d1 by Bard[5], assumes a known diagonal covariance matrix for the error in the observations on the state variables of

$$V=\mathrm{diag}(25E\text{-}6,\ 25E\text{-}6,\ 25E\text{-}8,\ 25E\text{-}10,\ 25E\text{-}12). \qquad (2.4.3)$$

The objective function is

$$.5\mathrm{Tr}(V^{-1}M(p)) \qquad (2.4.4)$$

where M(p) is defined in Section 1.1. This is just a maximum likelihood estimate with the observation times known exactly. Since V is a diagonal matrix, this problem can be handled by our weighted least squares formulation. For convenience we ignore the .5 and minimize $\mathrm{Tr}(V^{-1}M(p))$. Now

V =diag(4E4, 4E4, 4E6, 4E8, 4E10)

and thus we weight the residuals associated with state variables 1, 2, 3, 4, and 5 by 200, 200, 2000, 20000, and 200000 respectively. Bard's starting approximation for p was

$$(.01, .01, .001, .001, .02, .001)^T.$$

Our implementation of the Levenberg-Marquardt method using Gear's predictor-corrector implementation to integrate the initial value problem gave the results in Table 2.4.2. Our

Component of p	Our estimate	Estimate in [5]
1	.6358233E-2	.6358106E-2
2	.6774440E-1	.6774396E-1
3	.5920433E-4	.5916273E-4
4	.4943161E-3	.4943798E-3
5	.1018610	.1018756
6	.4204069E-3	.4202537E-3

Table 2.4.2
Optimization results

minimum for the expression in (2.4.4) was 21.38429 and Bard's minimum was 21.37944. Considering the complexity of the programs and the fact that we used Hermite interpolation to get integration results at the observation times, these results seem to be in good agreement. We note that no constraints were required to get the above optimum. Bard uses penalty functions to impose the constraint $p_j>0$, j=1,...,6 on this problem. They appear unnecessary for us; however, Bard[5,p185] does require

the imposition of constraints to successfully resolve some parameter fitting problems involving the above dynamic system.

CHAPTER 3

SPECIAL METHODS FOR THE INTERACTIVE APPROACH

3.1 THE INTERACTIVE APPROACH

Interactive techniques provide a powerful tool for nonlinear parameter estimation in general and they are especially valuable for difficult problems such as those involving dynamic models. Indeed, even the resolution of simple nonlinear parameter estimation problems often requires several runs to adjust such things as termination criteria, and starting approximations to the parameters. With dynamic models, the problems of instabilities, overstabilities, and at times numerous local minima in the optimization problem can make parameter estimation a tedious task. An interactive approach using algorithms designed with user intervention in mind can reduce many of the difficulties associated with parameter estimation in dynamic models. However, for the optimum use of any parameter estimation package there is no substitute for a good understanding of the model under consideration. The design of a good interactive package is an involved task. For a detailed introduction to interactive applications in numerical analysis see Smith[63]. The work by Aaro[1],[2] on a software system for interactive computing seems to hold promise for producing good transportable interactive packages with good user interfaces. We list below some of the major considerations involved in producing a good interactive parameter estimation

package.

(1) Special algorithms that exploit user judgement and intervention should play a large role in the interactive package. The development of such algorithms for the fitting of parameters in dynamic models is one goal of this thesis.

(2) An extensive set of commands should be available; however, a typical user should not be forced to learn a detailed command language in order to use the program. One way to attain this goal is for the interactive program to display lists of options (such as output options for example) and to prompt the user for the necessary details to complete a command. There is, however, a tradeoff here and for certain highly repetitive commands (such as those involved with stepping through a nonlinear optimization interactively) prompting should be kept to a minimum. Finally, the program should be relatively "user proof". That is, internal checking should be done so that regardless of what the user enters, the program should not end with a terminal error. Making a package user proof is largely a matter of detailed programming, and in the interests of efficient program development, it should be left until near the end. Another goal of this thesis is to develop a set of simple commands that are useful for interactive parameter fitting in dynamic models.

(3) Careful consideration must be given to what information the interactive program displays. Graphical information seems

to be the most useful. In the case of parameter fitting in dynamic models, integration results, observations, and in the two state variable case, phase plane plots are obvious candidates for graphical display.

(4) A decision must be made on how the user should describe the parameter fitting problem to the interactive package. There are basically two choices. First, the dynamic model can be defined through a user written subroutine, which is separately compiled and then loaded with the interactive package. This subroutine can also contain other necessary analytical information (such as Jacobian matrices) required by the integration and parameter fitting algorithms. Alternatively, the model can be entered and modified interactively (and of course saved on file for later use so that it need not be re-entered each time an interactive session begins). This is a more versatile approach, but it requires extensive programming. Its main advantage is that the user can interactively modify as well as fit a model. Starting with a simple model and gradually working up to a more complex model is one way of getting starting approximations to parameters (see Bard[6,p.123]). However, the consequences of adding a new term to a dynamic model can be dramatic and interactive modification of dynamic models demands a lot from the user. In the interests of efficiency, interactive model entry and modification require that any partial derivatives needed by the numerical

algorithms should be found symbolically. This is feasible, but it adds to the complexity of the overall program. The second option described above is a good long range goal; however, it is essentially a matter of programming and not numerical analysis and it should wait until the numerical aspects of the interactive package have been settled upon. This second option can be added later to a working package using the first option.

(5) A good interactive package should be well structured so that new commands can easily be added, and so that existing commands can be easily modified and extended. Also attention should be paid to the desirability of eventually producing a transportable program. Transportability is especially sensitive to the way the package uses graphics software. Thus it is desirable to isolate the interface between the interactive program and particular graphics procedures. Of course, the interactive program should only use generally available graphics operations.

It is our view that the first goal (to develop good interactive algorithms for the parameter fitting problem) is the most important in that decisions made here influence the details concerning the way the other goals are attained, and even more importantly, the algorithms employed play a major role in determining how effective the overall package is. Of course good algorithms can be degraded if the user-machine interface is neglected.

An efficient way to proceed seems to be to develop an experimental interactive package concentrating on the first goal, but also paying strong attention to the second third and fifth goals. This is the strategy we have employed in developing our interactive package PARFIT described in detail in Appendix A.

In this chapter, we consider several special techniques for fitting parameters in differential equations. Our goal is the development of techniques which lend themselves well to an interactive approach, and which are less sensitive to the initial parameter guess than the direct approach using the sensitivity equations. However, there is a tradeoff and coarse but well behaved methods should only be expected to give approximate values to the optimal parameters and should not for example be expected to distinguish between neighboring local minima in the full least squares problem. The first special method we consider is the derivative fitting approach. This is one of the most straightforward of the coarser methods. For experimental results with another implementation of this technique, we refer the reader to Swartz and Bremermann[66].

3.2 DERIVATIVE FITTING (DFIT)

Assume observations are given directly on a set of state variables in the dynamic model under consideration. Thus, theoretically, the desired derivatives of these state variables can be approximated as follows: (For the moment, we assume observations are available on all n components of the state

vector $y(t)$.) Each component $y_i(t)$ of $y(t)$ is approximated by a function $s_i(t)$ fitting the data at the observation points which is at least continuously differentiable. This can ideally be done interactively. The problem of finding p to minimize $F(p)$ in (1.3.1) can now be approximated by the problem of finding p to minimize

$$||s'(t)-g(t,s(t),p)||_2^2 \tag{3.2.1}$$

where $s(t)=(s_1(t),\ldots,s_n(t))^T$.

Since $s(t)$ generally approximates noisy data, a careful determination of p to minimize the above expression in the L_2 sense cannot be justified. Thus for computational purposes, we minimize the semi-norm on $C'[t_0,t_k]$ defined by

$$\left(\sum_{l=1}^{k} ||s'(t_l)-g(t_l,s(t_l),p)||_2^2\right)^{1/2} \tag{3.2.2}$$

where $\{t_l : 1\leq l\leq k\}$ is the set of observation times introduced in Section 1.1. Our parameter fitting problem has thus become a problem in nonlinear functional approximation which is much cheaper and usually much easier than the original problem. Indeed, if g is linear in p, all we have is an ordinary linear least squares problem.

The above observations make the derivative fitting approach very attractive; however, as noted by Bard[6p.128], it has some severe flaws. These occur in part because it requires approximations to $y(t)$ and $y'(t)$ and good approximations to these functions are often hard to obtain. This is the case for

example when the observations have large separations in time or when the data is noisy. Bard further notes that an error analysis for the parameter estimates is difficult with this approach. This latter drawback is not too important from our point of view since we propose this method only as an intermediate technique and for the final determination of parameters and confidence intervals, we recommend the use of the sensitivity equations. A more severe drawback of this method is that with a poor approximation to y or y', parameters may be produced at which the solution to the initial value problem deviates greatly from the observations or even blows up. This can be remedied at times by the technique of the next section where there is no need for an approximation to y'(t). The errors in the approximations to y and y' are not the only factors affecting the reliability of this technique. Other characteristics of the least squares problem of minimizing (3.2.2) must also be considered. In particular the results concerning conditioning mentioned at the end of Chapter 1 are important. For example, in view of (1.3.23), it is possible for relatively small errors in our approximation to y' to have a large influence on the parameters estimated by the DFIT method. This is especially important to keep in mind since the differentiation of data tends to be error prone. The integral fitting method of the next section avoids this dependence on approximated derivatives.

To implement the derivative fitting approach, a technique

for using the observations to approximate y(t) and y'(t) is required. If there are fairly large errors in the data, it is not reasonable to use a differencing technique to approximate y', and some smoothing method is called for. In our package PARFIT, we use least squares piecewise cubic spline approximations and least squares piecewise cubic Hermite approximations to the data. Approximations are made individually on each observed state variable and the positioning of the joints for each piecewise polynomial is left to the user. This should ideally be done interactively.

Cubic spline approximations are adequate for many cases; however, they run into difficulties when sharp bends occur in the function being approximated. This causes problems, for example, when the dynamic model under consideration is stiff. Piecewise cubic Hermite approximations which are C^1 as opposed to the C^2 cubic spline approximations do not have as great a problem with sharp bends and are thus more suited for handling the sharp turns that occur in stiff problems.

A couple of limitations with the derivative fitting technique immediately come to mind. In the problem formulated in Section 1.1, it was possible to have parameters arising only through the initial conditions. A derivative fitting algorithm cannot give us any information on these parameters. Another difficulty arises with derivative fitting when observations are not available on all the state variables, but there is a way around this problem in some of the cases where no parameters

occur exclusively in the subset of state equations corresponding to unobserved state variables. Specifically, we do a derivative fit on the subset of state equations corresponding to observed state variables. At each least squares function evaluation at a point in parameter space during the iterative solution of this reduced derivative fitting problem, we integrate the subset of state equations corresponding to unobserved state variables. Thus, up to date information is always available on the unobserved state variables in the reduced derivative fitting problem. However, this method has severe limitations. The obvious limitation is that it only applies to a restricted set of problems. Another limitation involves stability problems which can arise when a subset of a system of differential equations is integrated. Nevertheless, this technique has experimentally proven successful in some cases and for this reason we mention it here. Next we give an example of the kind of stability problem that can occur.

Consider the problem

$$y'=Gy \tag{3.2.6}$$

where y is of length 2 and

$$G=\begin{bmatrix} -30 & -22 \\ 28 & 20 \end{bmatrix} \tag{3.2.7}$$

The eigenvalues of G are -8 and -2 and the solution to (3.2.6) decays exponentially; however, if we fix $y_1=s_1(t)$ and integrate

$$y_2' = 28 s_1(t) + 20 y_2, \tag{3.2.8}$$

depending on $s_1(t)$ of course, the solution can grow exponentially. The stability problems that can arise with the above modified derivative fitting technique also have an impact on the design of stable iterated integral and derivative fitting algorithms in Section 3.4. To illustrate the derivative fitting technique, consider the following problem involving a set of coupled chemical reactions (see van Domselaar and Hemker[71]). The state equations are

$$\begin{aligned} y_1' &= -(1-y_2) y_1 + p_2 y_2 \\ y_2' &= p_1 ((1-y_2) y_1 - (p_2+p_3) y_2) \end{aligned} \tag{3.2.9}$$

and the initial condition is

$$y(0) = (1,\ 0)^T. \tag{3.2.10}$$

This represents a model of a chemical reaction and all parameters should remain positive. Observations were generated by integrating the above initial value problem at

$$(1000,\ 0.99,\ 0.01)^T.$$

All simulated observations in this thesis were found by integration under error control. In the interests of economy, all parameter fitting integrations were done without stepsize adjustment unless otherwise indicated. For this stiff problem, we used observations on both state variables at the same observation times that were used in [71]. These observations (to four figures) are listed in Table 3.2.1. The observations

Time	y_1	y_2	Time	y_1	y_2
0.0002	0.9998	0.1648	0.06	0.9991	0.4998
0.0004	0.9997	0.2753	0.08	0.9989	0.4997
0.0006	0.9996	0.3493	0.1	0.9989	0.4997
0.0008	0.9996	0.3990	1.0	0.9945	0.4986
0.001	0.9996	0.4322	2.0	0.9895	0.4974
0.0012	0.9995	0.4545	5.0	0.9747	0.4936
0.0014	0.9995	0.4695	10.0	0.9502	0.4872
0.0016	0.9995	0.4795	15.0	0.9260	0.4808
0.0018	0.9995	0.4862	20.0	0.9021	0.4743
0.002	0.9995	0.4907	25.0	0.8786	0.4677
0.02	0.9993	0.4998	30.0	0.8553	0.4610
0.04	0.9993	0.4998			

Table 3.2.1
Observations for stiff problem

on y_1 were approximated with a least squares piecewise cubic Hermite polynomial with one joint at t=10, and the observations on y_2 were approximated with a least squares piecewise cubic Hermite polynomial with joints at t=.0007, .0014, .0016, 15. Our starting guess at the parameters was

$$p^{(0)} = (50,\ 5,\ .5)^T.$$

The DFIT method found the point

$$(969.6,\ 1.113,\ -.1080)^T$$

in parameter space. Unfortunately, p_3 is negative. This suggests we try a square root transformation of p_3 and estimate $\hat{p}_3 = \sqrt{p_3}$. With this transformation, the DFIT method found the point (unscaled)

$$(969.6,\ 1.005,\ 9.33E\text{-}7)^T$$

in parameter space. We note that the DFIT method was quite capable of finding a good approximation to p_1. This should be compared with the results in the next section where the IFIT method was applied to this problem.

The direct method using the sensitivity equations also succeeded when it was started from $p^{(0)}$ given above; however, it was slow to begin modifying p_1 upward from 50.

Before we give our next set of test problems, some conventions must be established concerning the presentation of graphical information. In particular, we use the following conventions:

Observation points on y_1: +

Observation points on y_2: ×

Integration results on y_1: ———

Integration results on y_2: — — — — — — —

Smoothing of data for y_1: — - — - — - —

Smoothing of data for y_2: — - - — - - — - - — - - —

Guessed behavior for y_1: ——— — ——— — ———

Guessed behavior for y_2: — — — — — — — —

Phase plane trajectories are solid lines.

The above conventions are sufficient for most of our plots. Other conventions are introduced as they become necessary. We use the method developed by McConalogue[45] to produce smooth curves for our plots.

In this section and Section 3.3, we make some comparisons between derivative fitting and integral fitting on a problem

involving a change in equilibrium. The effect of noisy data on these two parameter estimation techniques is investigated for the test problem,

$$y_1' = p_1 y_1^3 + p_2 y_1^2 + p_3 y_1 - y_2$$
$$y_2' = p_4 y_1 \qquad (3.2.11)$$

with the initial condition

$$y(0) = (1.5,\ 1.0)^T. \qquad (3.2.12)$$

Our first set of observations was obtained by integrating the above problem at the parameter vector

$$(-.1,\ -1,\ 2.4,\ .1)^T.$$

These generated observations, at times .5(.5)20, are shown in Figure 3.2.1. For clarity we present graphical results only on y_1. State variable y_2 does not go through any rapid jumps. Observations are used on both state variables. A piecewise cubic Hermite approximation to the observations on y_1 using joints at t=2, 4, 6, 7, 8, 9, 10, 12, 14, 18 is also shown in Figure 3.2.1. The joints for the corresponding approximation to the observations on y_2 were at t=3, 8, 15. Using these approximations to the observations, the DFIT method produced the parameters

$$(-.07740,\ -.7995,\ 1.725,\ .09948)^T.$$

No starting approximation to the parameters was required since the DFIT method involved solving a linear least squares problem

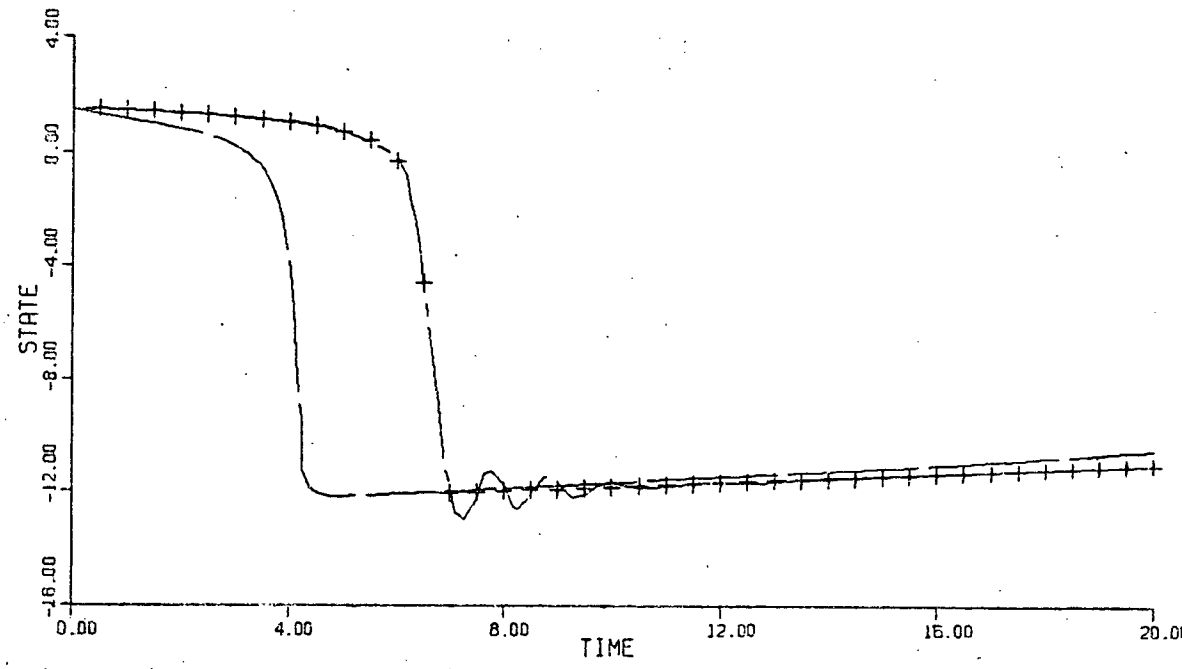

Figure 3.2.1
Equilibrium change--no error in observations

in this case. Integration results for y_1 at the above parameters are also shown in Figure 3.2.1. All integrations of (3.2.11) were done using stepsize adjustment. These results should be compared with those in Figure 3.3.1 where results obtained with the IFIT method applied to this problem are shown.

Next a normally distributed random error with mean 0 and standard deviation $\sigma = 1$ was introduced into the above observations. The resulting observations on y_1 and their piecewise cubic Hermite approximation using joints at t=6,7,9,14 are shown in Figure 3.2.2. Joints at t=6,12 were used for the smoothing of the observations on y_2. Using this smoothing, the DFIT method produced the parameters

$$(-.03942, -.4047, 1.168, .1007)^T.$$

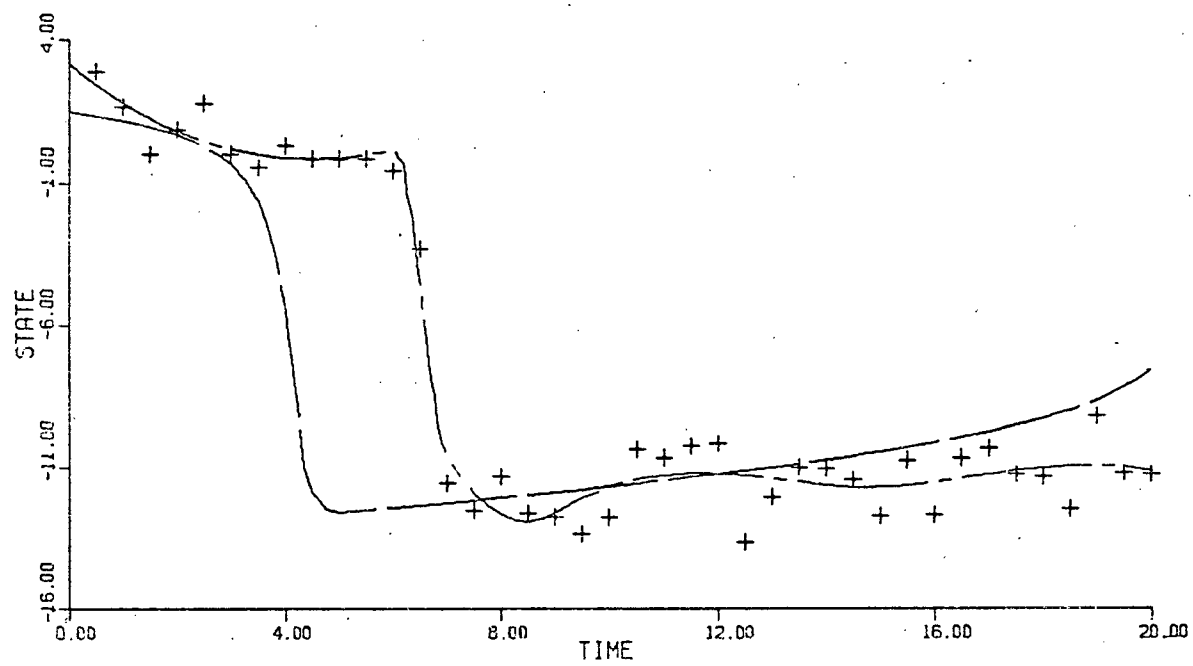

Figure 3.2.2
Equilibrium change--error in observations(σ=1)

Integration results for y_1 at these parameters are also shown in Figure 3.2.2. These results should be compared with those in Figure 3.3.2 where results with the IFIT method are presented for this problem.

Our next experiment involved an increase in the noise; however, the change in equilibrium is still visually discernable. A normally distributed random error with mean 0 and standard deviation σ=2 was introduced into the observations. The resulting observations along with a piecewise cubic Hermite smoothing function are shown in Figure 3.2.3. The joints for the smoothing functions were the same as for the case σ=1. With this smoothing , the DFIT method produced the parameters

$$(-.02113,\ -.2394,\ .6449,\ .1023)^T.$$

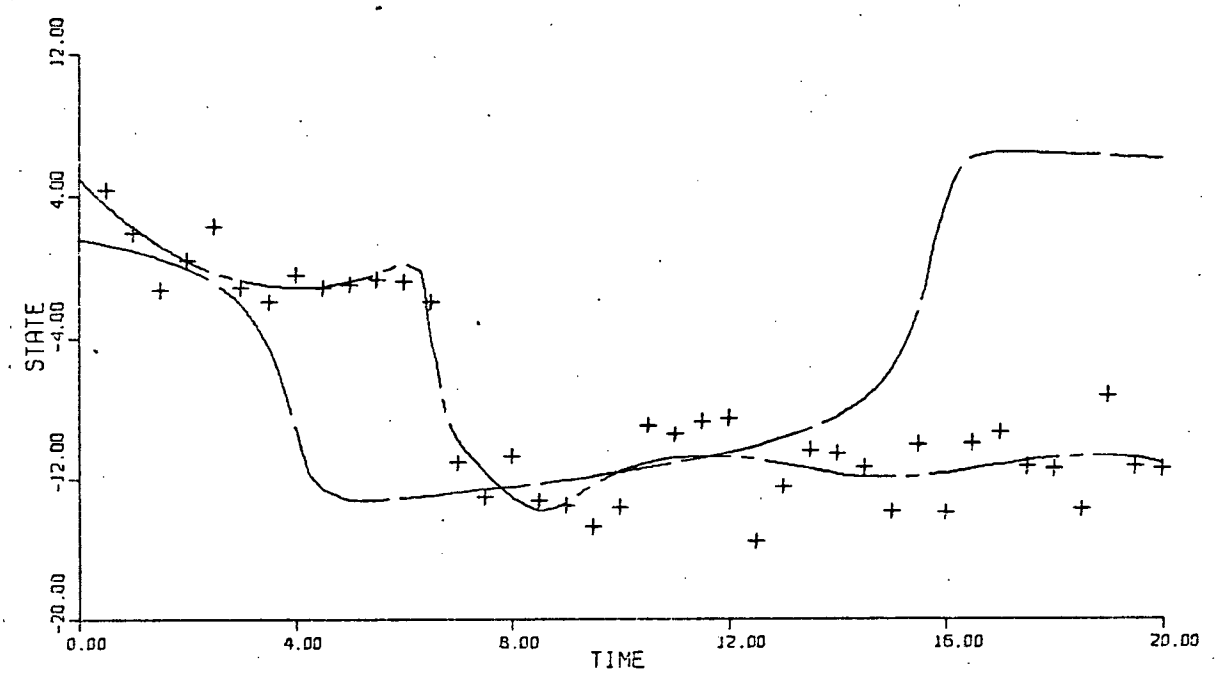

Figure 3.2.3
Equilibrium change--error in observations (σ=2)

Integration results for y_1 at these parameters are shown in Figure 3.2.3. By comparison with Figure 3.3.3, we see that the IFIT method produced much better results in this case.

3.3 INTEGRAL FITTING (IFIT)

The derivative fitting approach works well in many cases; however, it has the drawback that it requires the numerical differentiation of (at times) noisy data. Thus it is reasonable to try to fit integrals instead of derivatives. For some additional background to integral fitting we refer the reader to Bard[6,p.219]. As in the previous section, we assume an approximation to the desired solution to our initial value problem is available in the n-vector $s(t)=(s_1(t),\ldots,s_n(t))^T$ which is obtained by some smoothing technique. The integral

fitting problem is to find the vector p to minimize

$$f^{T}(p)f(p)$$

where

$$f_{n(\ell-1)+i} = (y_{0,i}(p) + \int_{t_0}^{t_\ell} g_i(t,s(t),p)dt) - s_i(t_\ell) \qquad (3.3.1)$$

where $\ell=1,\ldots,k$; $i=1,\ldots,n$; $y_{0,i}(p)$ is the i'th component of the vector of starting values for the initial value problem $y'=g(t,y,p)$, and $g_i(t,s,p)$ is the i'th component of $g(t,s,p)$. The Jacobian matrices for the linear least square problems that arise when p is found iteratively have elements

$$J_{qj} = \frac{\partial f_q}{\partial p_j} \quad q=1,\ldots,kn;\ j=1,\ldots,m \qquad (3.3.2)$$

which are found by evaluating the integrals

$$\frac{\partial y_{0,i}}{\partial p_j}(p) + \int_{t_0}^{t_\ell} \frac{\partial g_i}{\partial p_j}(t,s,p)dt \qquad (3.3.3)$$

for $\ell=1,\ldots,k$; $i=1,\ldots,n$; $j=1,\ldots,m$.

Note that we are not solving our initial value problem with these integrals. We are just integrating functions of time since s(t) is known. Thus in terms of the number of evaluations of the function g, this method is equivalent to the derivative fitting algorithm when a simple integration method such as the trapezoidal method using the times $t_0,t_1,\ldots,t_k$ is employed. Furthermore, this method has the same degree of linearity as does the derivative fitting method. In particular the above least squares problem for p is linear in p when g and $y_0(p)$ are

linear in p. Also, the integral fitting method can provide information on parameters which occur only in the initial conditions, and this can be very useful. As demonstrated in Chapter 4, access to the initial conditions can also be very important when no parameters occur in the initial conditions.

One might expect the IFIT problem to be better conditioned than the DFIT problem; however, it is possible for the integral fitting problem to be singular even when the derivative fitting problem is well conditioned, but this does not appear to be a serious drawback in practice. To see how this singularity can arise, consider the case when there are 2 parameters, 1 state variable, and three observation times, t_1, t_2, and t_3 with equal spacing h. Let the initial condition (at t_1) be independent of the parameters and let the derivative fitting Jacobian be

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ -1 & 0 \end{bmatrix}$$

Using the trapezoidal integration method, the integral fitting Jacobian is the rank one 2x2 matrix

$$\begin{bmatrix} h/2 & h/2 \\ h/2 & h/2 \end{bmatrix}$$

formed by multiplying the derivative fitting Jacobian by

$$\begin{bmatrix} h/2 & h/2 & 0 \\ h/2 & h & h/2 \end{bmatrix}$$

We note again that this was a specially contrived situation and such a difficulty does not appear to arise in practice.

Our first experiment with the IFIT method is on the stiff problem (3.2.9). Using the same smoothing and $p^{(0)}$ as were used in the previous section, the IFIT method found the point

$$(607.3,\ .9901,\ .009281)^T$$

in parameter space. The parameter p_1 was not as well approximated as it was with the DFIT method; however, no parameters have gone negative and good approximations have been obtained for p_2 and p_3.

Next we give some experiments with the initial value problem (3.2.11) involving a change in equilibrium. Using the observations and smoothing of the previous section for the case $\sigma=0$ (no error), the IFIT method gave the parameters

$$(-.1006,\ -.9913,\ 2.569,\ .1004)^T.$$

Integration results for y_1 at this parameter vector are shown in Figure 3.3.1. Using the observations and smoothing of the previous section for the case $\sigma=1$, the IFIT method gave the parameter vector

$$(-.00354,\ -.5883,\ 2.223,\ .09735)^T.$$

Integration results for y_1 at the above parameters are shown in Figure 3.3.2. These results should be compared with those in Figure 3.2.2. Finally using the observations and smoothing functions of the previous section for the case $\sigma=2$, the IFIT method gave the parameters

$$(-.03277,\ -.2655,\ 1.813,\ .09433)^T.$$

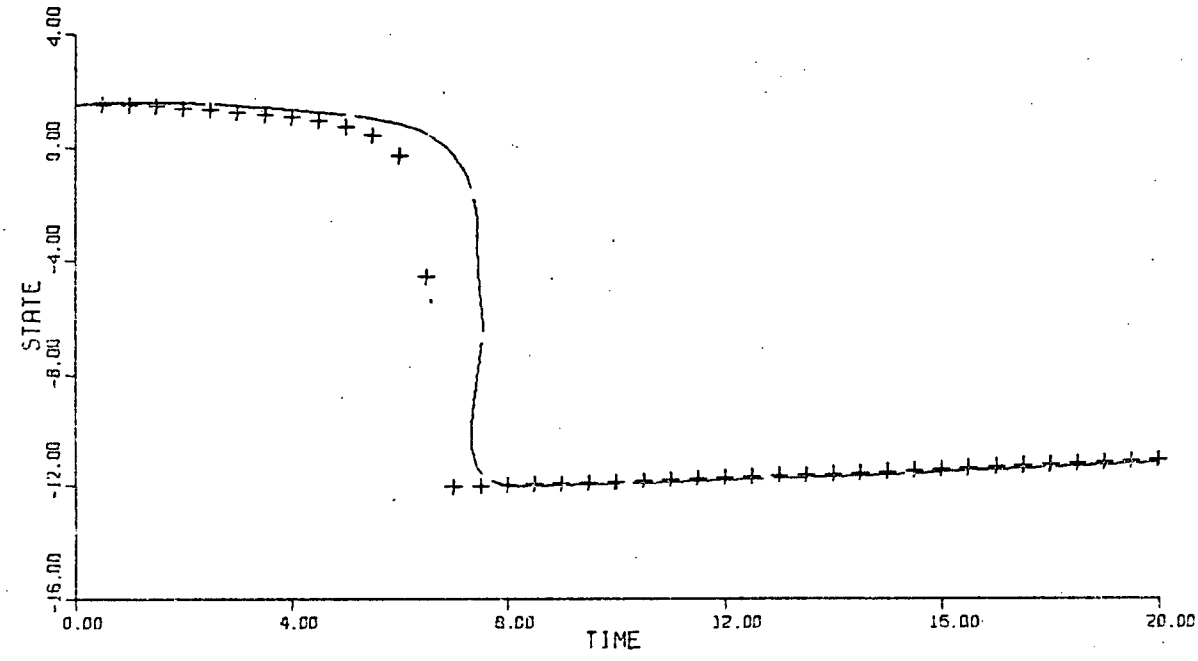

Figure 3.3.1
Equilibrium change--no error in observations

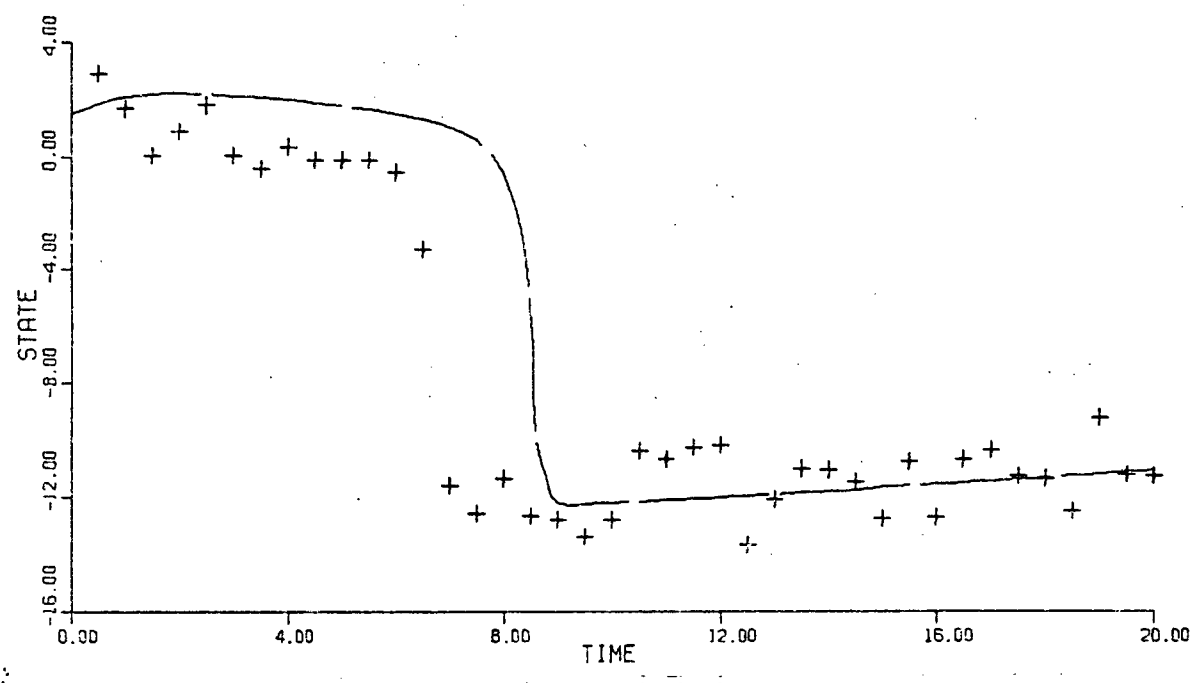

Figure 3.3.2
Equilibrium change--error (σ=1)

integration results for y_1 at the above parameters are shown in Figure 3.3.3. We observe that there was a substantial

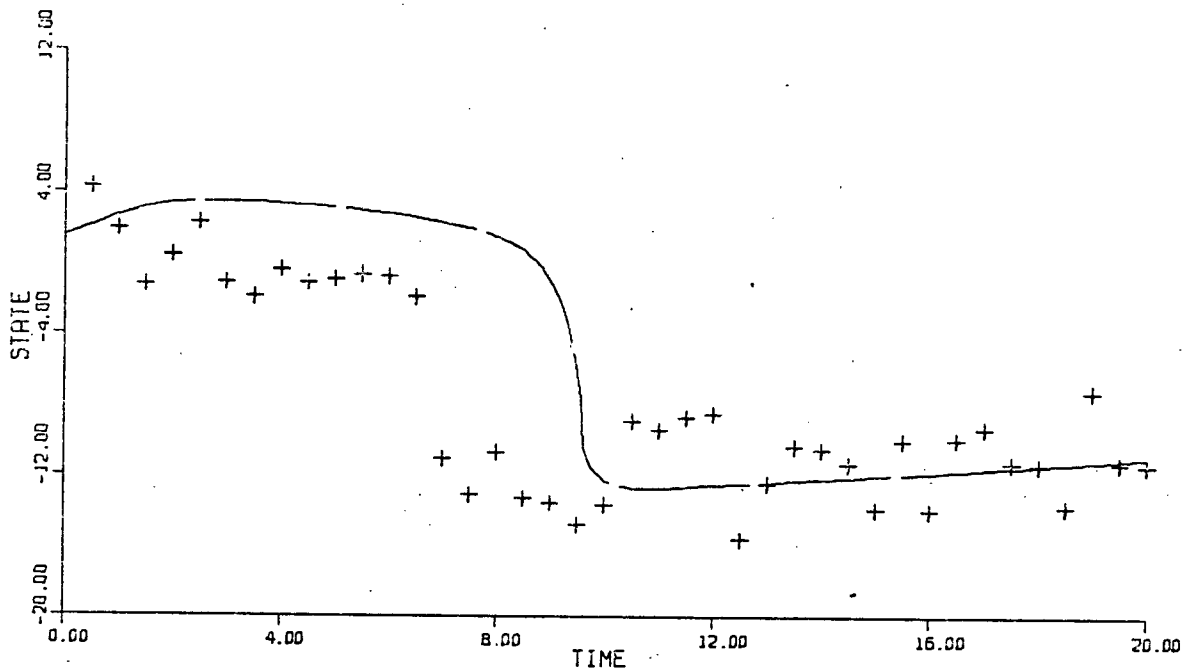

Figure 3.3.3
Equilibrium change--error (σ=2)

improvement over the corresponding results for the DFIT method shown in Figure 3.2.3.

3.4 ITERATED INTEGRAL AND DERIVATIVE FITTING

In this section, the important special case when observations are not available on all state variables is considered. This situation was mentioned briefly in Section 3.2; however, the technique presented there was highly restrictive in the class of problems it could handle and it was prone to instabilities.

At times the model builder knows approximately how unobserved state variables should behave to be consistent with

the observed state variables. Thus an intelligent guess can be made and fictitious observations on the unobserved state variables can be created. With these created observations along with the physical observations, the DFIT and IFIT methods can be applied. Our interactive package PARFIT contains facilities for setting up guessed observations on unobserved state variables in the important special case when only two state variables are present. PARFIT could easily be modified to handle cases where more than two state variables are present; however, as the number of unobserved state variables increases, the practicality of this method diminishes. The use of created observations can be a great help in determining the proper parameters, but, as one would expect, the success of this method depends on how well the proper behavior of the unobserved state variables can be anticipated. This is again a good place for an interactive approach. The model builder can interactively modify the guessed observations, apply the DFIT or IFIT methods, observe the integration results at the parameters obtained (if integration is possible), and then readjust the created observations. There are also automatic ways of improving the guessed observations and we concentrate on such methods for the remainder of this section.

One stable way to iteratively improve guessed observations and reestimate parameters is with a nonlinear block Gauss-Seidel technique. (For a discussion of nonlinear Gauss-Seidel techniques, see for example Ortega and Rheinboldt[50,p.224].)

For simplicity of notation, we look at the two state variable case with observations available on only one state variable. The extension to n state variables with observations available on r state variables (r<n) is immediate. Note, however, that generally the parameter estimation problem becomes more difficult as the number of unobserved state variables increases. Indeed, removing observations on a state variable can change a well defined parameter estimation problem into a singular problem. An example of this is given in the next chapter. Without loss of generality for our two state variable discussion, we assume that observations are missing on the second state variable. For the discrete set of time values t_ℓ, $\ell=1,\ldots,N$ let $s^\ell=s(t_\ell)$ (where we are using a superscript to avoid confusion with the case when s(t) is a vector) be the smoothed approximation to the observations on state variable y_1 at time t_ℓ, and let $s^{\ell\prime}=s'(t_\ell)$ be the corresponding approximation to the derivative of the observations at time t_ℓ. Let c_ℓ be the approximation to the unobserved state variable y_2 at time t_ℓ. Usually, the points t_ℓ, $\ell=1,\ldots,N$ are the observation times defined in Section 1.1. We seek p and

$$c=(c_0,\ldots,c_N)^T$$

to minimize

$$F=\sum_{\ell=0}^{N}(g_1(t_\ell,(s^\ell,c_\ell)^T,p)-s^{\ell\prime})^2+\sum_{\ell=1}^{N}(d_\ell(c))^2 \qquad (3.4.1)$$

where $d_\ell(c)$ represents a discretization of the second state

equation. That is we want the c_ℓ to approximately satisfy the second state equation at the minimum of the above expression. For example if the trapezoidal method is used to discretize the second state equation then

$$d_\ell(c) = .5\left(g_2(t_{\ell-1}, (s^{\ell-1}, c_{\ell-1})^T, p) + g_2(t_\ell, (s^\ell, c_\ell)^T, p)\right) - \frac{c_\ell - c_{\ell-1}}{t_\ell - t_{\ell-1}}. \tag{3.4.2}$$

Define

$$\nabla_p F = \left(\frac{\partial F}{\partial p_1}, \ldots, \frac{\partial F}{\partial p_m}\right)^T$$
$$\nabla_c F = \left(\frac{\partial F}{\partial c_0}, \ldots, \frac{\partial F}{\partial c_N}\right)^T. \tag{3.4.3}$$

Thus the gradient of F is

$$\nabla F = (\nabla_p F^T, \nabla_c F^T)^T.$$

A necessary condition for

$$(p^T, c^T)^T = (\bar{p}^T, \bar{c}^T)^T$$

to minimize F is that

$$\nabla F((\bar{p}^T, \bar{c}^T)^T) = 0. \tag{3.4.4}$$

One way to solve (3.4.4) is to use the nonlinear block Gauss-Seidel method starting with an initial guess $(p^{(0)T}, c^{(0)T})^T$ for $(\bar{p}^T, \bar{c}^T)^T$. There are of course other ways to solve for $(\bar{p}^T, \bar{c}^T)^T$. For example, we could minimize (3.4.1) by a standard nonlinear least squares technique such as the Levenberg-Marquardt method. To be efficient, however, such

methods should take advantage of the particular sparsity structure of the Jacobian matrix associated with (3.4.1). The use of a block Gauss-Seidel technique on the other hand requires only a standard nonlinear least squares technique such as the Levenberg-Marquardt procedure, along with a least squares technique designed for cases where the approximations to the Hessian matrices are banded matrices. A description of our algorithm for minimizing (3.4.1) and an outline of a local convergence proof for this method follows. Assume a starting approximation $(p^{(0)T}, c^{(0)T})^T$ to $(\bar{p}^T, \bar{c}^T)^T$ is given. First hold c fixed at $c^{(0)}$ and determine $p^{(1)}$ to minimize (3.4.1). This is just a standard nonlinear least squares problem and we solve it using the Levenberg-Marquardt method. Next with p fixed at $p^{(1)}$ determine $c^{(1)}$ to minimize (3.4.1). This is a sparse nonlinear least squares problem and we solve it by the Gauss-Newton method using the normal equations and a standard library procedure for solving banded positive definite linear systems by a Cholesky decomposition. Strategy (a) in Bard[5,p175] was employed for step length adjustment in our sparse Gauss-Newton procedure. In practice fairly rapid convergence was obtained with this part of the algorithm; however, we expect this to vary depending on the nonlinearity in y_2 of the given initial value problem. The overall algorithm proceeds by successively reestimating $p^{(i)}$ and $c^{(i)}$ where on each estimation, the most recent information on p and c is employed. Usually only a few full iterations were required for the algorithm to settle down to a value for

$(\bar{p}^T, \bar{c}^T)^T$.

The above algorithm is just a special case with $\omega=1$ of a block nonlinear successive overrelaxation process (see for example Ortega and Rheinboldt[50,p.325, p.332]). Thus a local convergence proof is standard and we confine ourselves to just a brief outline of convergence.

Let $x=(p^T,c^T)^T$, $\bar{x}=(\bar{p}^T,\bar{c}^T)^T$ and assume $\bar{x}$ is a local minimum of F. That is

$$\nabla F(\bar{x})=0.$$

Assume further that F is twice continuously differentiable and that the Hessian matrix H of F is positive definite in the open neighborhood S_o of $\bar{x}$. Split the Hessian into

$$H(x)=D(x)-L(x)-L^T(x)$$

where D(x) is a block diagonal matrix, and L(x) is a block lower triangular matrix and where the entries in L corresponding to the blocks in D are zero. Since $H(\bar{x})$ is symmetric and positive definite $D(\bar{x})$ is is symmetric and positive definite. Also $D-\omega L$ is nonsingular for any ω and in particular for $0\leq\omega\leq 2$. Thus (see Varga[73,p.77]), for $0<\omega<2$

$$\rho((D(\bar{x})-\omega L(\bar{x}))^{-1}((1-\omega)D(\bar{x})+\omega L^T(\bar{x})))<1$$

where $\rho(.)$ denotes spectral radius.

It follows (Ortega and Rheinboldt[50,p.326]) that there exists an open ball S centred at $\bar{x}$ and contained in S_o such that there is a unique sequence $\{x^{(i)}\}$ in S satisifying our

nonlinear block Gauss-Seidel algorithm (ω=1) and

$$\lim_{i\to\infty}(x^{(i)})=\bar{x}.$$

There are other ways to estimate p in conjunction with iterations on guessed observations. For example instead of estimating p to minimize (3.4.1) for a fixed c, we could estimate p using the DFIT or IFIT methods. Use of the DFIT method to estimate p is equivalent to minimizing (3.4.1) with

$$d_\ell(c)=g_2(t_\ell,(s^\ell,c_\ell)^T,p)-c'_\ell$$

where c'_ℓ approximates the time derivative of c at t_ℓ. The detailed form of d_ℓ depends on how c'_ℓ is approximated.

Alternatively, we could replace the DFIT portion of the above iterative algorithm by an application of the IFIT method. This has been found to work well. We comment that the resulting algorithm is not equivalent to minimizing $F((p^T,c^T)^T)$ in (3.4.1) with the first summation replaced by its integral fitting counterpart:

$$\sum_{l=0}^{N}\left(y_l(0)+\int_{t_0}^{t_\ell} g_1(t,(s(t),c(t))^T,p)\,dt-s(t_\ell)\right)^2.$$

If the above term is put in place of the first sum in (3.4.1) then the least squares problem for updating c loses the sparsity structure that was present with derivative fitting. Thus we do not consider full iterated integral fitting methods, but only iterated integral fitting methods where p is updated using the

IFIT method and c is updated using a sparse technique.

There is another way c can be updated besides using the function (3.4.1) and experimentally this method has proven effective; however, it suffers from potential instabilities and must be used with caution. This method simply involves the generation of c at the new parameter vector p by integrating the second state equation holding y_1 fixed at s(t). (We are still assuming y_2 is unobserved.) As indicated in the derivative fitting section of this chapter, this subsystem can be very unstable at a particular point p in parameter space even when the system y'=g(t,y,p) has no stability problems. An example where this method works very well is given in the next chapter. Finally we observe that no differential equation stability problems occur when we update c to minimize (3.4.1).

A couple of examples using integral fitting and our sparse Gauss-Newton method to improve guessed observations follow. Consider the Lotka-Volterra predator-prey model[41].

$$y_1' = p_1 y_1 - p_2 y_1 y_2$$
$$y_2' = -p_3 y_2 + p_4 y_1 y_2$$

Here y_1 represents the prey and y_2 represents the predator and p_1, p_2, p_3, p_4 are all positive. This model is a special case of the model considered by Bazykin[7] which we look at in Chapter 4. For our first example, we generated observations by integrating the above system starting at $y_1 = 12$, $y_2 = 2$ and using

$$p = (.15, .03, .8, .1)^T.$$

The observation times were 1(1)20 and observations were made available only on y_1. The generated observations on y_1 and integration results for y_2 at the above simulation parameters are shown in Figure 3.4.1. The function s(t), defined by the

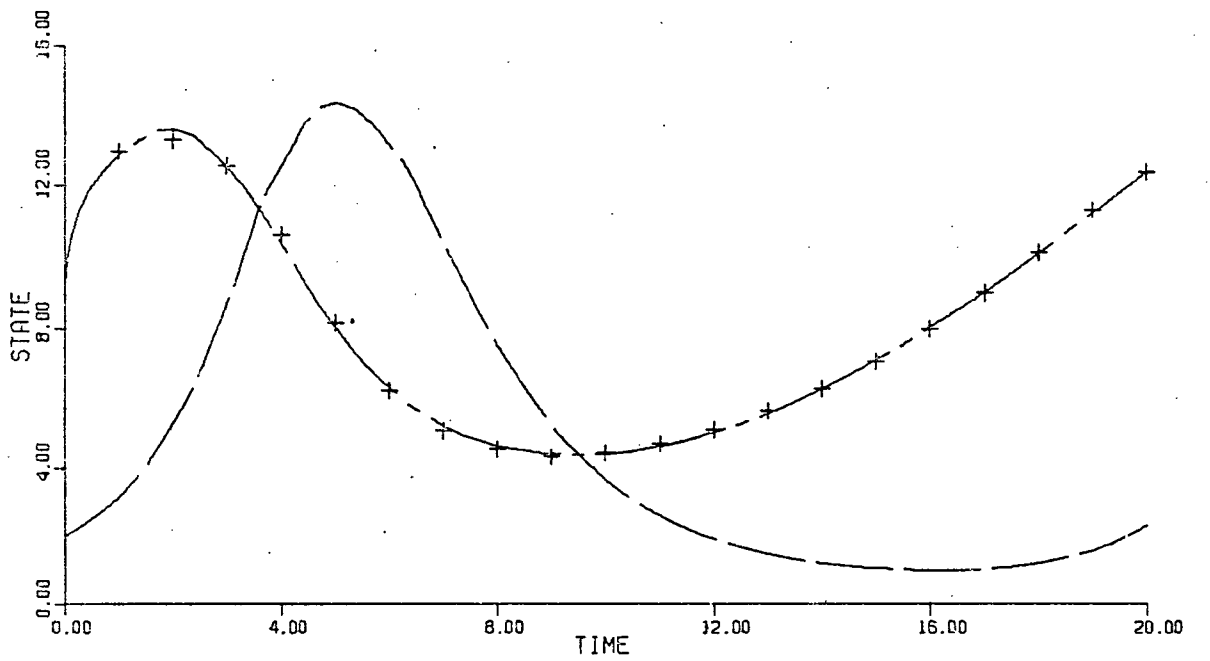

Figure 3.4.1
Simulation results and spline approximation

least squares cubic spline approximation to the observations using joints at t=2, 3, 5, 9, 16, is also shown in Figure 3.4.1. The initial guess at the observations is shown in Figure 3.4.2. Successive improvements in the guessed observations are also shown in Figure 3.4.2. Integration results at successive parameter estimates are shown for y_1 in Figure 3.4.3. Observations on y_1 are also shown in Figure 3.4.3. Observe that the integral fitting problem is linear and thus an initial guess at the parameter values was not required. The least squares

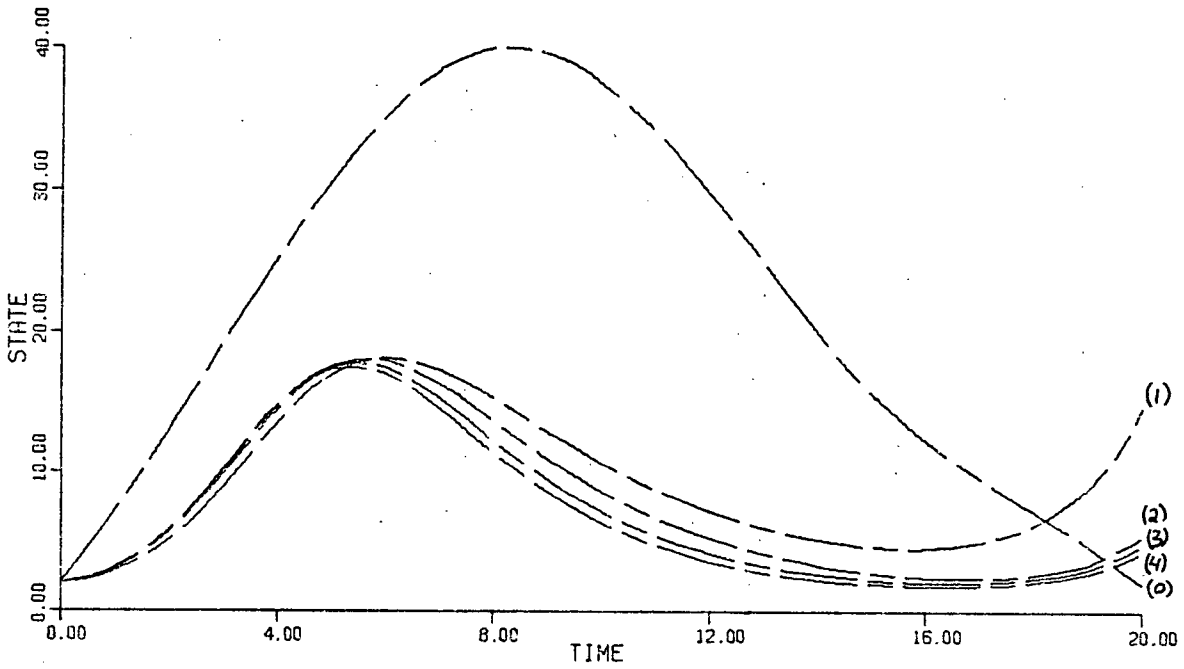

Figure 3.4.2
Iterations on the guessed observations

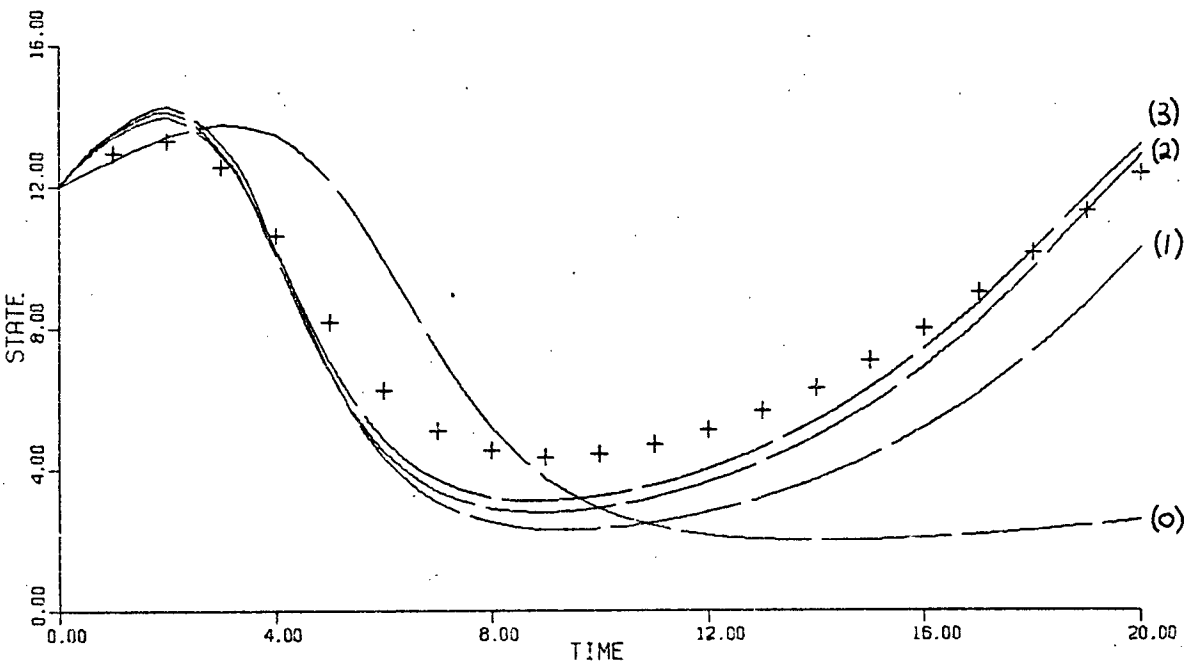

Figure 3.4.3
Observations and successive integrations of y_1

problem for updating the guessed observations is also linear in this example. Convergence was not too sensitive to the height of the initial approximation to y_2, but it was sensitive to the position of the peak. This, however, is quite easily adjusted interactively. Thus an iterated approach to this problem reduces a potentially nasty nonlinear problem (especially if there are no good guesses at the values of $p_1,\ldots,p_4$) to a simple interactive procedure of adjusting one quantity (the position of the peak) over a well defined interval. Of course some intuitive idea about the "proper" behavior of y_2 is required; however, we would expect this information to often be more readily available than a good approximation to the optimal parameter vector.

Our second example is the same as the above example except for the new observation times .5(.5)12.5. Also our initial guess at the behavior of y_2 was much less informed than it was for the previous example. The joints for the smoothing spline were at t=1.25, 2.5, 5, 9. Figure 3.4.4 shows the successive improvements in the guessed observations and Figure 3.4.5 compares integration results with observations for y_1 at the successive parameter estimates. We observe that the iterations on the guessed observations did not attain the maximum that y_2 does in Figure 3.4.1; however, this does not seem critical in view of the results in Figure 3.4.5.

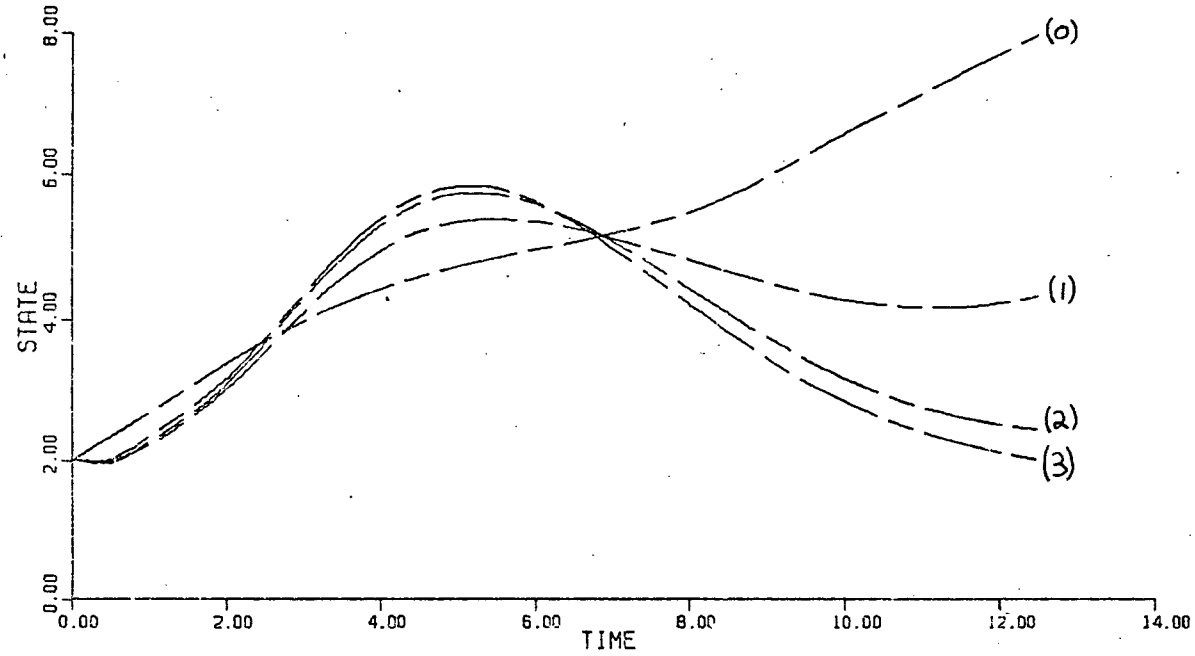

Figure 3.4.4
Iterations on guessed observations

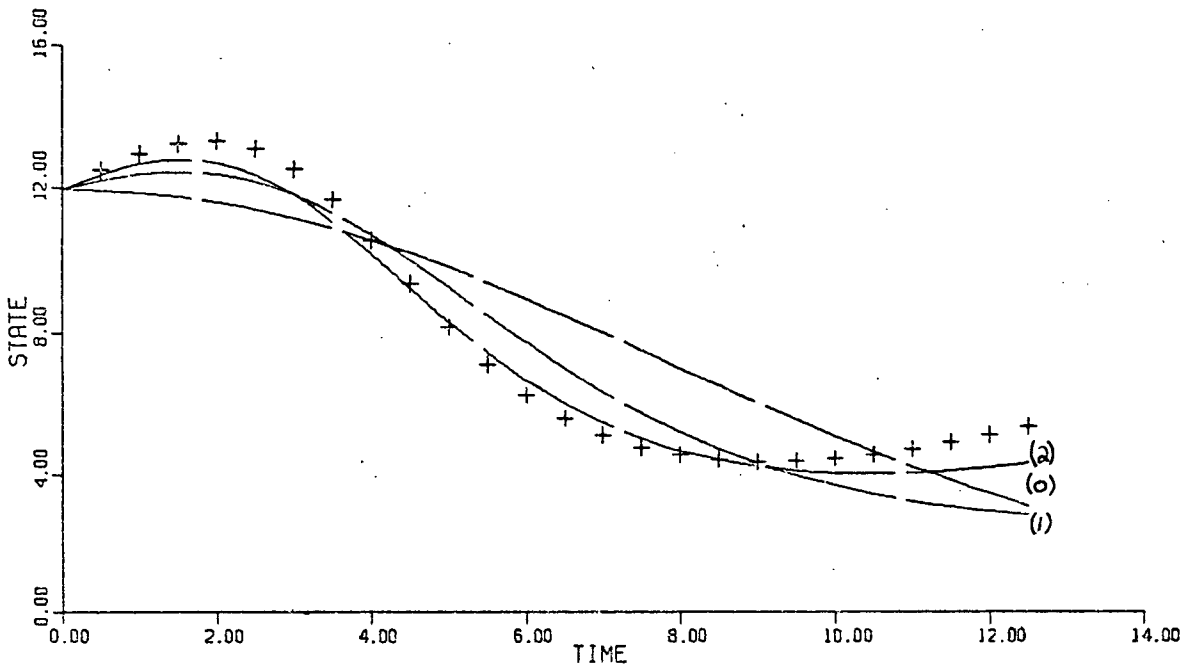

Figure 3.4.5
Integrations at successive parameter estimates

3.5 CONTINUATION AND QUASI MULTIPLE SHOOTING

The methods presented in this section are designed to bridge the gap between the coarse but well behaved techniques of the previous sections and the full nonlinear least squares problem. These coarse methods are good to start out with if a good initial approximation to the optimal parameter vector is unavailable. However, the parameters produced by these methods can be inadequate. For example the solution to the given initial value problem may blow up at the parameters found by a coarse method. The problem of instabilities at the starting parameter values is a common difficulty encountered when fitting parameters in initial value problems. Various strategies have been suggested (see for example Bard[6,p.233]); however, there does not appear to be any preferred technique. The methods suggested in this section lend themselves well to an interactive attack on the problem.

First we consider a continuation method between the integral fitting (IFIT) technique and the full least squares problem. Assume, for now, that observations are available on all state variables. These observations provide us with approximations $s_i(t)$, $i=1,\ldots,n$ to the desired behavior of the state variables $y_i(t)$, $i=1,\ldots,n$. Consider the problem of finding p to fit the solution of

$$\begin{aligned} u' &= g(t,(1-\gamma)s(t)+\gamma u,p) \\ u(t_o) &= y_o(p) \end{aligned} \qquad (3.5.1)$$

to s(t) in the least squares sense at the observation times where $0 \leq \gamma \leq 1$. When $\gamma = 0$, this is just the integral fitting technique and when $\gamma = 1$ it is the full least squares problem (on the smoothed data). The eigenvalues of g_u determine the stability of the above initial value problem. Now

$$g_u = \gamma g_y$$

for $y = (1-\gamma)s + \gamma u$. Thus for small γ, it should be possible to integrate (3.5.1) even when it is unstable for $\gamma = 1$. Put another way, as γ increases, more and more of the "differential equation nature" of the problem is taken into account.

There are two basic ways a continuation problem can be approached. One way is to treat it as an initial value problem in the continuation parameter. This is the Davidenko approach (for a good summary of Davidenko's work see Rall[56]). The second way is to treat it as a sequence of nonlinear problems each associated with a larger value of γ. This is the Lahaye approach (see Rheinboldt[58],[59], Rall[56], Ficken[23]).

For our purposes the Davidenko approach appears to involve an excessive amount of calculation. The Lahaye approach on the other hand lends itself well to an interactive attack on the problem. That is we start with $\gamma = 0$ (IFIT) and then successively fit the solution to (3.5.1) to s(t) with progressively larger values for γ. If γ is increased too much and the integration of (3.5.1) becomes impossible, then γ can be interactively reduced until (3.5.1) can be integrated. Our limited experience

seems to indicate that this approach by itself is not very effective (for a full evaluation more experiments on a wide selection of problems are required); however, combined with the use of break points as described later in this section, the continuation approach seems to be a viable way to escape from an unstable region in parameter space. One of the main drawbacks with such techniques is of course the expense involved. This however is becoming less important with the increasing availability of powerful computers.

Experimental results indicate that the use of break points in a quasi-multiple shooting technique along with a continuation method can be an effective combination for handling the stability problem. Specifically consider break points at

$$T_1 < T_2 < \ldots < T_g$$

corresponding to observation times

$$t_{i_1}, \ldots, t_{i_g}.$$

Further let denote a continuation parameter vector for the break points. The vector α is of length n where n is the number of state variables and $0 \leq \alpha_i \leq 1$, i=1,...,n. If break points alone are used (that is, $\gamma = 1$) then observations need not be available on all state variables. The components of α corresponding to unobserved state variables should be equal to 1. Let $u^-(T_i)$ denote the solution to (3.5.1) arrived at by integrating up to break point T_i (which for i>1 involves passing through i-1 previous break points). In the process of integrating (3.5.1),

we reset u at the break points according to

$$u^{+}(T_i)=Au^{-}(T_i)+(I-A)s(T_i)$$

where $A=[diag(\alpha_1,\ldots,\alpha_n)]$, and $u^{+}(T_i)$ acts as the new initial condition at T_i. Of course the appropriate modification to the solution of the sensitivity equations must also be made at the break points. For purposes of least squares approximation, the value $u^{-}(T_i)$ is used at T_i. Thus we can also weight the residuals at the break points with weights $w_1,w_2,\ldots,w_\beta$. To summarize we have at our command

(a) a continuation parameter γ

(b) a set of break points $T_1,\ldots,T_\beta$

(c) a continuation parameter vector α for the break points

(d) a set of weights $w_1,\ldots,w_\beta$ for the break points.

This gives the user a powerful set of options to play with and for their optimal use, an interactive approach is indicated. Extensive experiments over a wide range of problems are required before a proper evaluation can be made of the interactive facilities suggested above. For the purposes of this thesis; however, we limit ourselves to a few examples in this section which indicate the potential power of the above facilities in an interactive environment. For details on another approach employing break points to aid in the fitting of parameters in dynamic models, we refer the reader to van Domselaar and Hemker[71]. For more background on shooting methods in general see for example Roberts and Shipman[60].

Consider again the Lotka-Volterra predator-prey model described in the previous section. We will try to fit this model to the data shown in Figure 3.5.1. This data is contrived

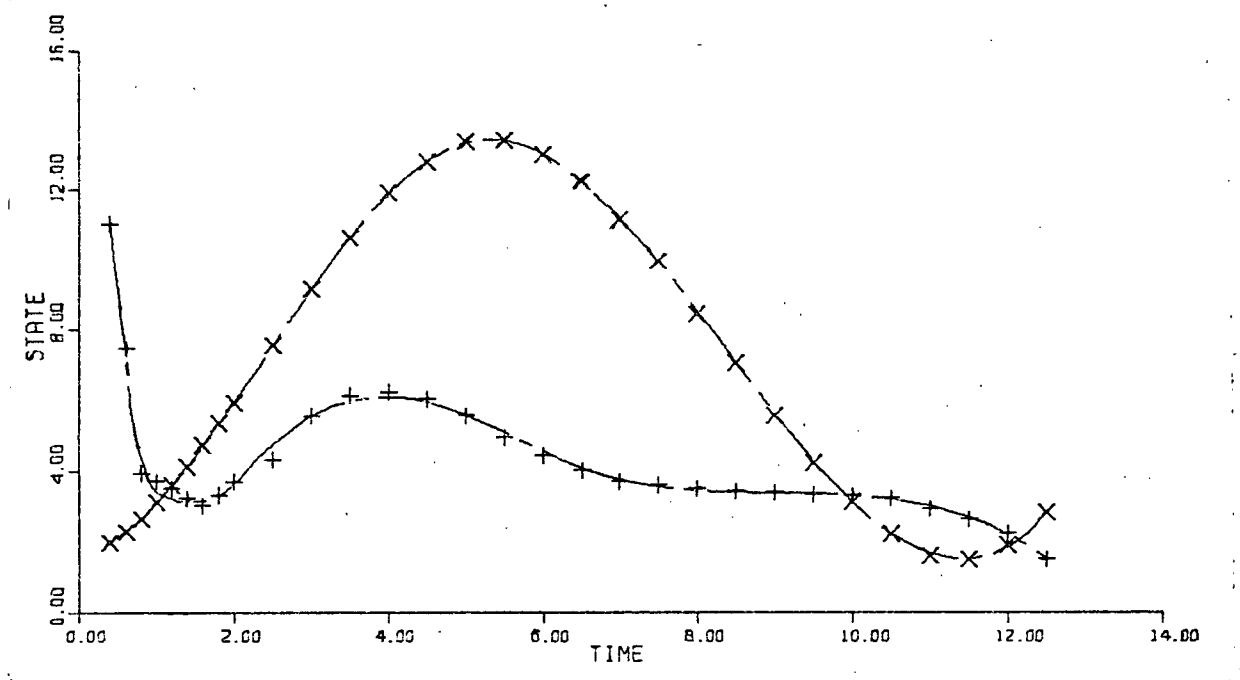

Figure 3.5.1
Data and smoothing for continuation tests

and there is no reason to expect a good fit with this model. Piecewise cubic Hermite least squares approximations to the data were used to define s(t). The joints for $s_1(t)$ were at t=1, 3, 6 and the joint for $s_2(t)$ was at t=5. The components of s(t) are shown in Figure 3.5.1. The observations were generated so that the IFIT method produced parameters where the solution to the Lotka-Volterra equations with initial conditions at $y=(12, 2)^T$ blew up. It is worthy of note that a fair amount of experimentation was required before such observations could be contrived. That is for this problem, the IFIT method did not

seem likely to produce parameters corresponding to an instability. The IFIT technique (γ=0) produced the parameters

$$(-.7273, -.06719, 1.525, .3295)^T$$

at which the given initial value problem was unstable. Since the observations have no physical meaning, there is no reason to require the paramerters to be positive. We are just looking for a set of parameters to minimize a sum of squares.

Using break points at

.6, 1.6, 4.5, 9.5

and using γ=.2, $\underline{\alpha}$=(0,0) and with weights of 1 at the break points, the following parameters were found:

$$(.07608, .02078, 1.032, .2150)^T.$$

The initial value problem was stable here and integration results at these parameters along with the observations are shown in Figure 3.5.2. The Levenberg-Marquardt technique had no problem converging from the above parameters to the optimum

$$(-.6820, -.05733, .8184, .1941)^T.$$

Integration results at the above paramters are shown in Figure 3.5.3. The sum of the squares of the residuals was approximately 196.

We comment that it is not necessarily advantageous to proceed with the continuation process once stability has been attained. Parameters giving rise to an instability may be produced and even with the use of break points, the careful

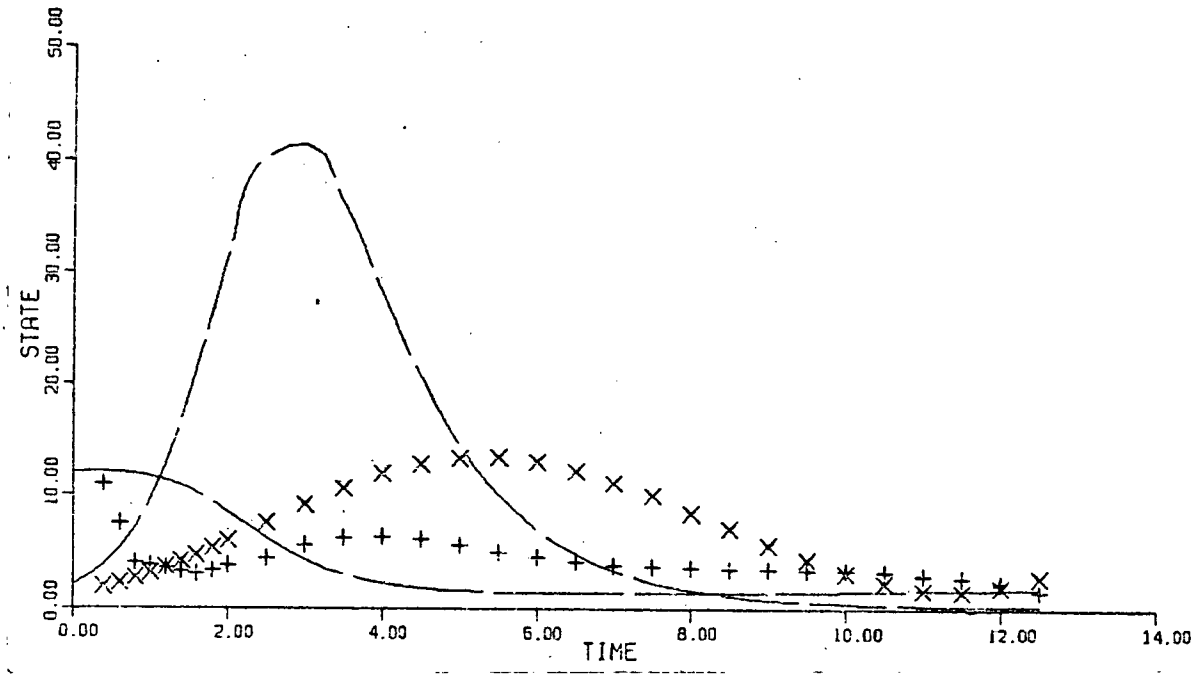

Figure 3.5.2
Results using break points with $\gamma = .2$

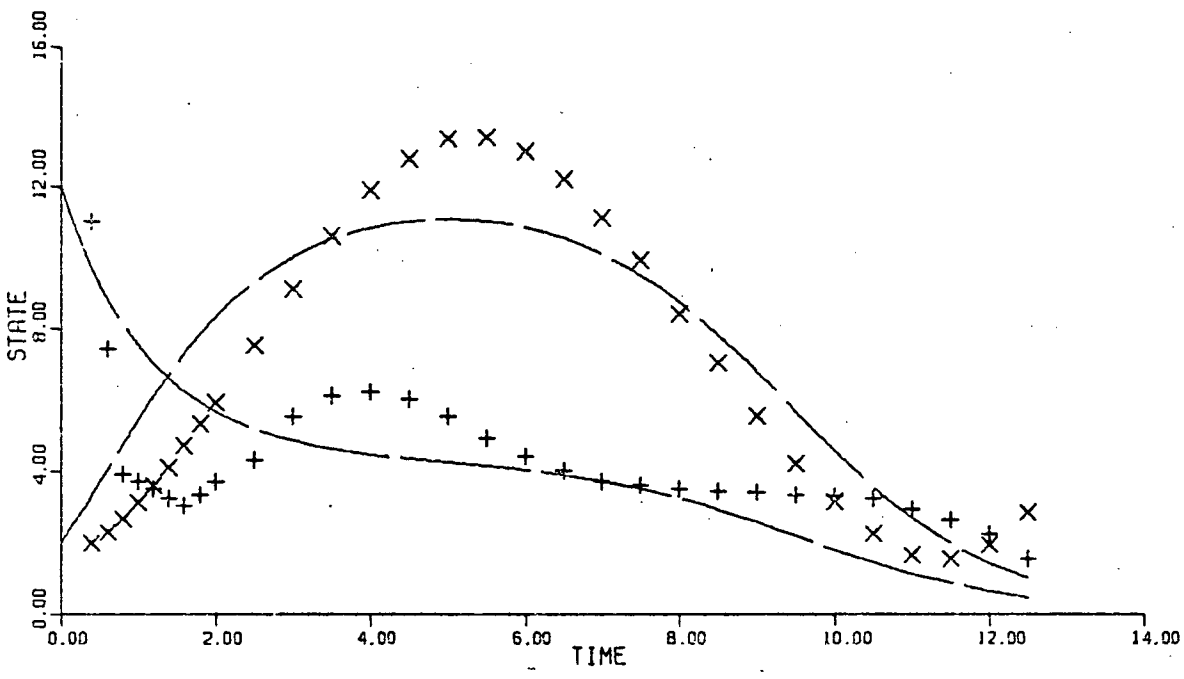

Figure 3.5.3
Results at optimal parameters

increasing of continuation parameters can prove to be a tedious and expensive undertaking. Again interactive monitoring and control of the overall process is desirable and if the process appears to get stuck a good strategy is to return to a smaller continuation parameter and modify the break points.

The use of the continuation parameter γ without break points did not appear productive on this problem. For example consider the sequence of γ's in Table 3.5.1. The process worked fairly well until γ got near .75. It then became difficult to

γ	Continuation fit result p_1	p_2	p_3	p_4	Stability
0	-.7273	-.06719	1.525	.3295	u
.3	-.5877	-.04820	1.524	.3341	u
.6		u n s t a b l e			
.5		u n s t a b l e			
.4	-.5766	-.04640	1.397	.3092	u
.5	-.5622	-.04465	1.242	.2783	u
.6		u n s t a b l e			
.55	-.5697	-.04491	1.184	.2667	u
.6	-.5759	-.04541	1.130	.2559	u
.7		u n s t a b l e			
.65		u n s t a b l e			
.63	-.5723	-.04492	1.081	.2463	u
.65	-.5704	-.04462	1.057	.2402	u
.68	-.5741	-.04497	1.021	.2343	u
.7	-.5809	-.04576	1.010	.2320	u
.73		u n s t a b l e			
.72	-.5843	-.04610	.9881	.2279	u
.74	-.5936	-.04724	.9853	.2270	u
.76	-.9258	-.08865	1.043	.2637	u
	very difficult to continue				

Table 3.5.1
A continuation experiment

increase γ and difficult to do the optimizations.

The use of break points alone (with γ=1) was effective on

this problem; however, instabilities arose very easily if too few break points were used. Here again we have a situation where the interactive approach can provide a powerful tool. Starting at the parameters determined by the IFIT algorithm, with $\gamma=1$, $\alpha=(0,0)$ an instability arose in the integration of (3.5.1) when break points were used at

1.8, 4, 7.5.

With break points at

.6, 1, 1.8, 4, 4.5, 6, 7.5, 10

the integration of (3.5.1) blew up just after t=5. With break points at all the observation times, (3.5.1) could be integrated and the optimal parameters obtained were

$(-.4166, -.02940, .6279, .1280)^T$.

The Volterra equations were stable at these parameters and starting from this point in parameter space, the Levenberg-Marquardt method produced the optimum illustrated in Figure 3.5.3.

We conclude that for this problem the use of break points alone is effective, but tricky, while the use of break points along with a continuation parameter in the differential equation can be very effective.

The techniques of this section demand a degree of judgement from the user; however, they provide a framework in which to tackle the instability problem and in this sense they are to be

preferred over blind probing in parameter space.

3.6 IMPLEMENTATION OF AN INTERACTIVE PACKAGE

The development of a good interactive package is an evolutionary process. Our package, PARFIT, represents the first stage in such a process. We outline below some of the specifications that should be kept in mind as the package evolves. We also describe with the aid of flow charts how the various facilities of PARFIT can be used to complement one another. In view of the increasing power of mini-computers and the inherent expense of using a program such as PARFIT, the feasibility of implementing a version of PARFIT on a mini-computer appears to be a worthy topic of study. The work of Aaro[1], [2] should provide a very valuable tool for developing such an implementation.

One goal when developing an interactive package for parameter fitting in differential equations should be to make special parameter fitting algorithms such as those of this chapter easily accessible and complementary. The package should also allow for the testing and addition of new algorithms. It is advantageous to be able to easily switch between techniques in an interactive manner since the success of a particular technique is often closely related to the problem on which it is employed. To facilitate the easy switching from one algorithm to another the interactive package must be carefully designed. For example piecewise polynomial smoothing of data is used by several algorithms. Thus there should be a single data

structure for smoothing functions and it should be accessible (and perhaps modifiable) by all the procedures using smoothed data. This is just common sense; the general philosophy should be to define data structures at the most general level practicable. The various algorithms then function in an environment established by these data structures. It is useful for this environment to contain much more than just the set of data structures used by the numerical procedures, and it is convenient to think of it as a data structure itself containing information on how the package is to communicate with the user, on various controlling parameters such as the integration error criteria and integration method, and on many other aspects of the operation of the package. We list below some of the items which can be thought of as being part of the environment.

(1) Communication mode:

This governs the amount of program guidance and descriptive information provided during an interactive session. The way information is entered (for example using a keyboard, or a light pen) is also indicated here.

(2) Echo flags:

These indicate what information produced during an interactive session is to be retained for later hard copy output.

(3) Graphics display control:

This indicates what information is to be displayed on the graphics device. For example the most recently generated data

(such as smoothing or integration results) may be displayed as the default. The user should be able to modify the display to include for example only selected state variables. The default display should be intelligent enough to display appropriate plots during an interactive session. For example when smoothing is being done, the data and approximating function for the given state variable should be displayed. When iterated integral fitting is being done, the display of successive iterations on the guessed observations would be valuable. It should of course be possible to override the graphics display control and request that specific data be plotted.

(4) Numerical data structures:

These contain, among other things, smoothing information and integration results.

(5) Numerical control parameters:

These include integration error criteria and stepsize constraints as well as parameters used in the optimization procedures.

(6) Problem and algorithm selection data structures:

These indicate what integration and optimization procedures are being used. The use of various scalings (such as the square root or logarithmic scalings available in PARFIT) is indicated here and an indication of any frozen parameters is given here. In more sophisticated packages, constraints on parameters can be indicated here and a particular choice from a selection of various objective functions to optimize (from statistical

considerations) can be indicated here.

(7) Notations accumulated during an interactive session:

As a user experiments with a particular problem, he should have the facilities to make notes. Thus a list of notes containing for example "promising parameter values" can be created. The environment thus becomes tailored to a particular problem. Of course it should be possible to save the notes from run to run.

There are two fundamental modes in which an interactive program can operate. One mode employs extensive control by the interactive program and the other relies on the user to initiate the appropriate sequence of actions to solve a problem. A good approach when developing an interactive package of the first type is to first build a package of the second type. Strategies for the solution to the problems under consideration are then developed through extensive use of this package and these strategies can eventually be incorporated into a package employing extensive program control over the solution strategy. PARFIT is a package designed for user initiated and conducted strategies. Thus its effective use requires a detailed knowledge of all its facilities and how these can be used to complement one another in an effective manner. In this section we give a description by way of flow charts of how the various approaches to the parameter estimation problem can be used together in an integrated package. The strategies outlined in these flow charts form the skeleton for a version of PARFIT that

would attempt to guide the user to the various facilities required for the solution to a problem. The implementation of such a program is a nontrivial endeavor, and to justify it, there should be a good demand for such a package. Indeed, the sophistication required of the model builder to develop good dynamic models argues against the need for detailed computer control of the interactive process. When the user is required to pick an appropriate strategy without program guidance, the strategies outlined in the following flow charts should be kept in mind.

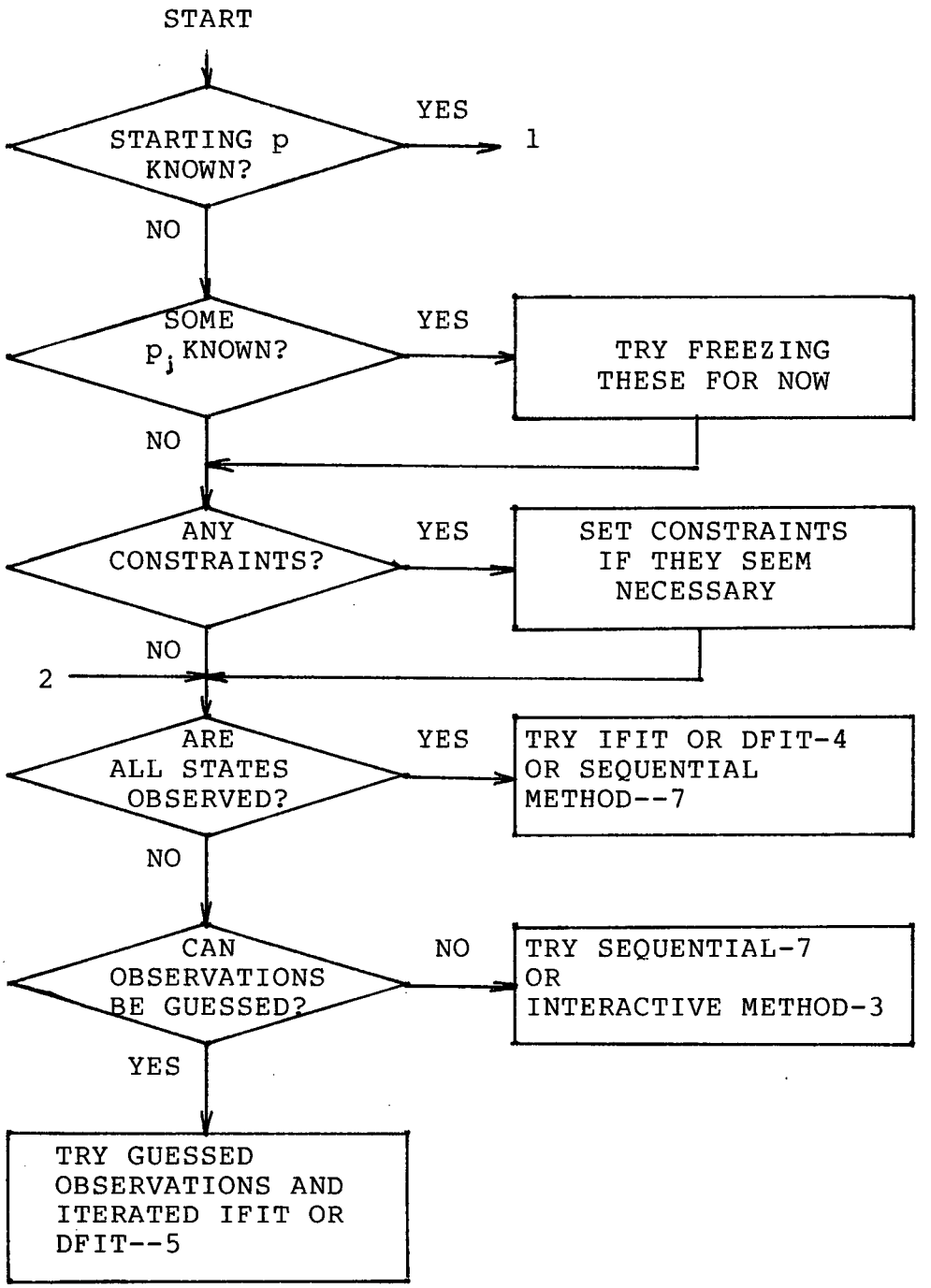

Figure 3.6.1
Overall strategy

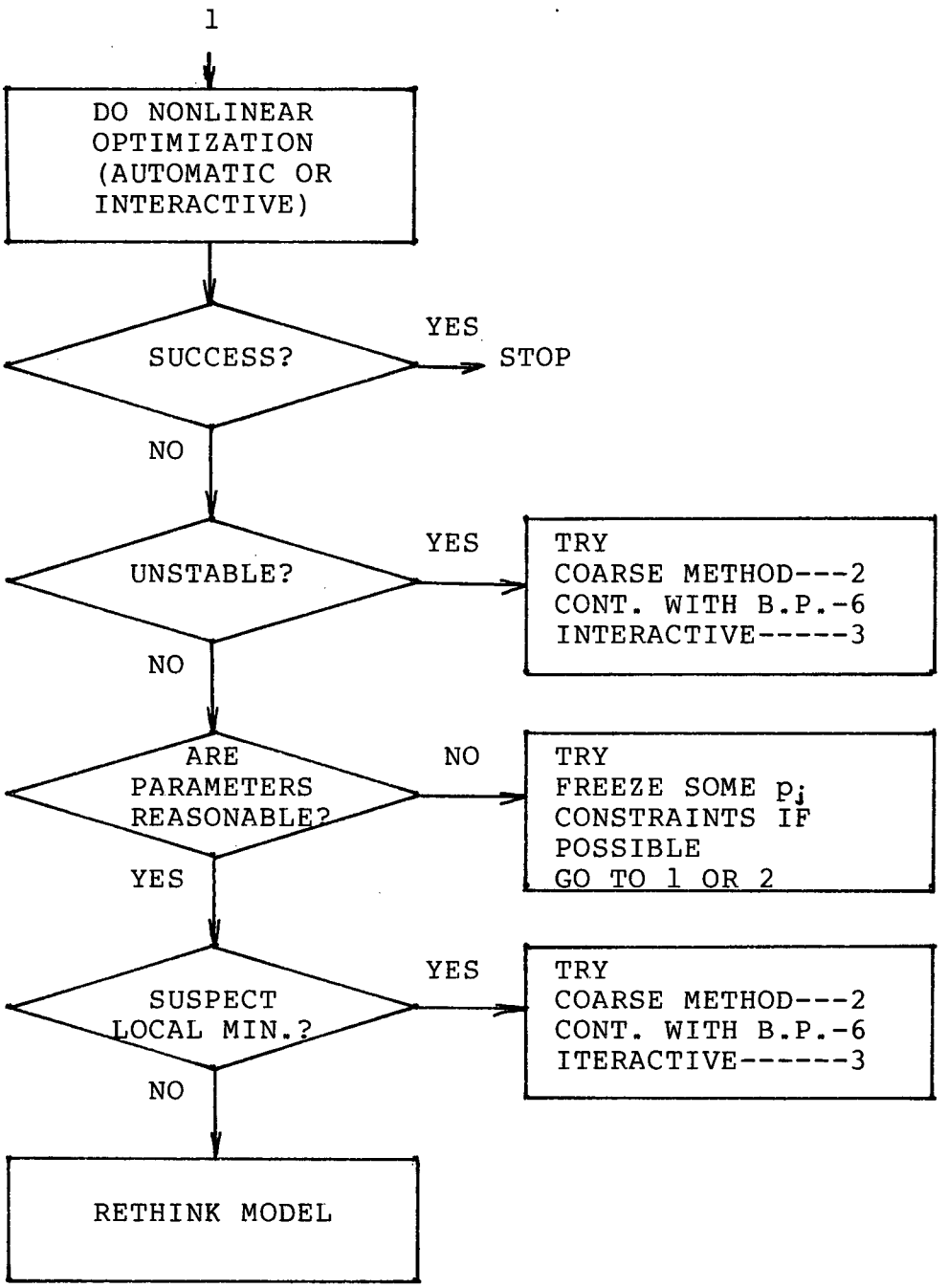

Figure 3.6.2
Refined parameter fitting

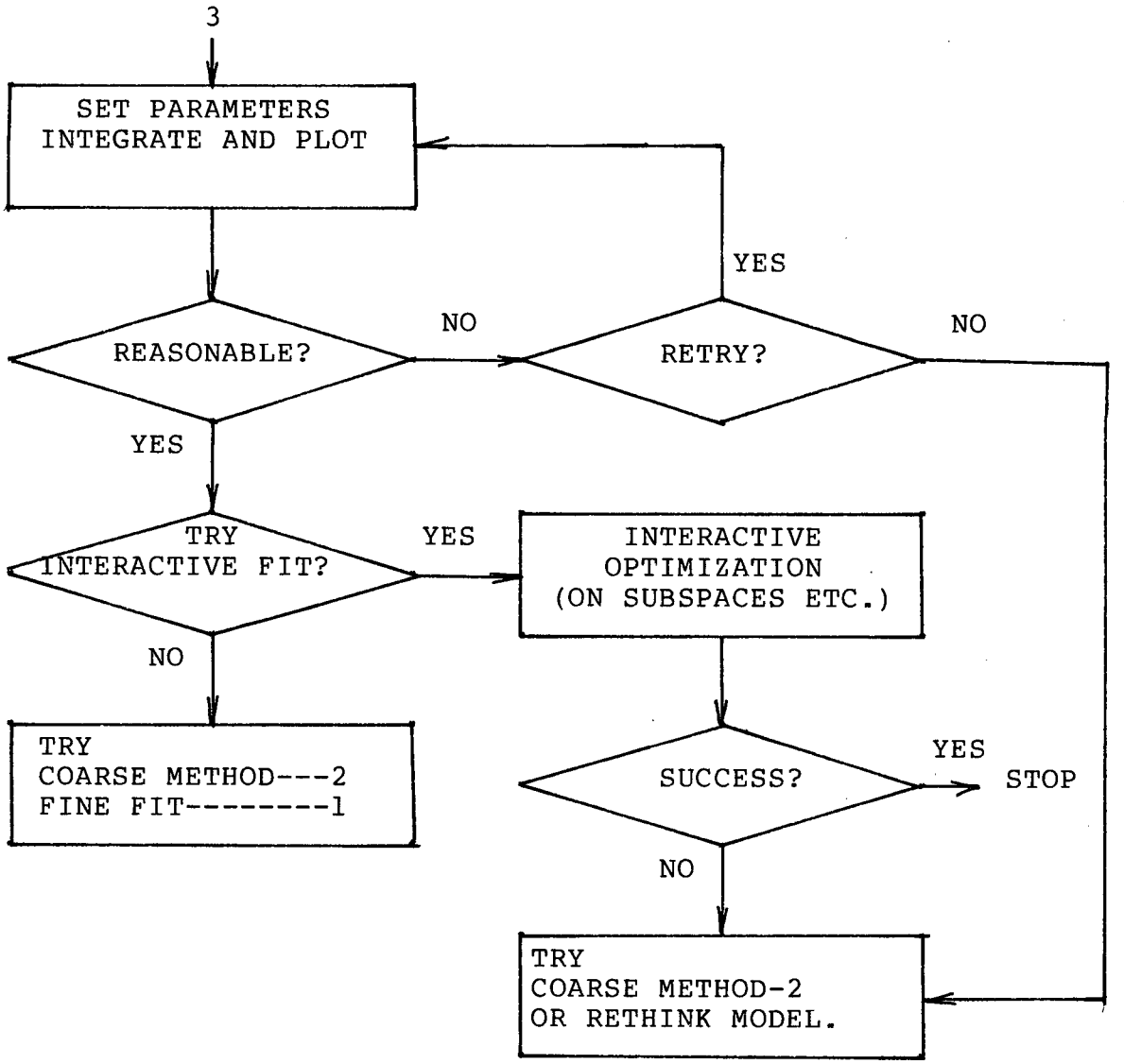

Figure 3.6.3
Interactive optimization

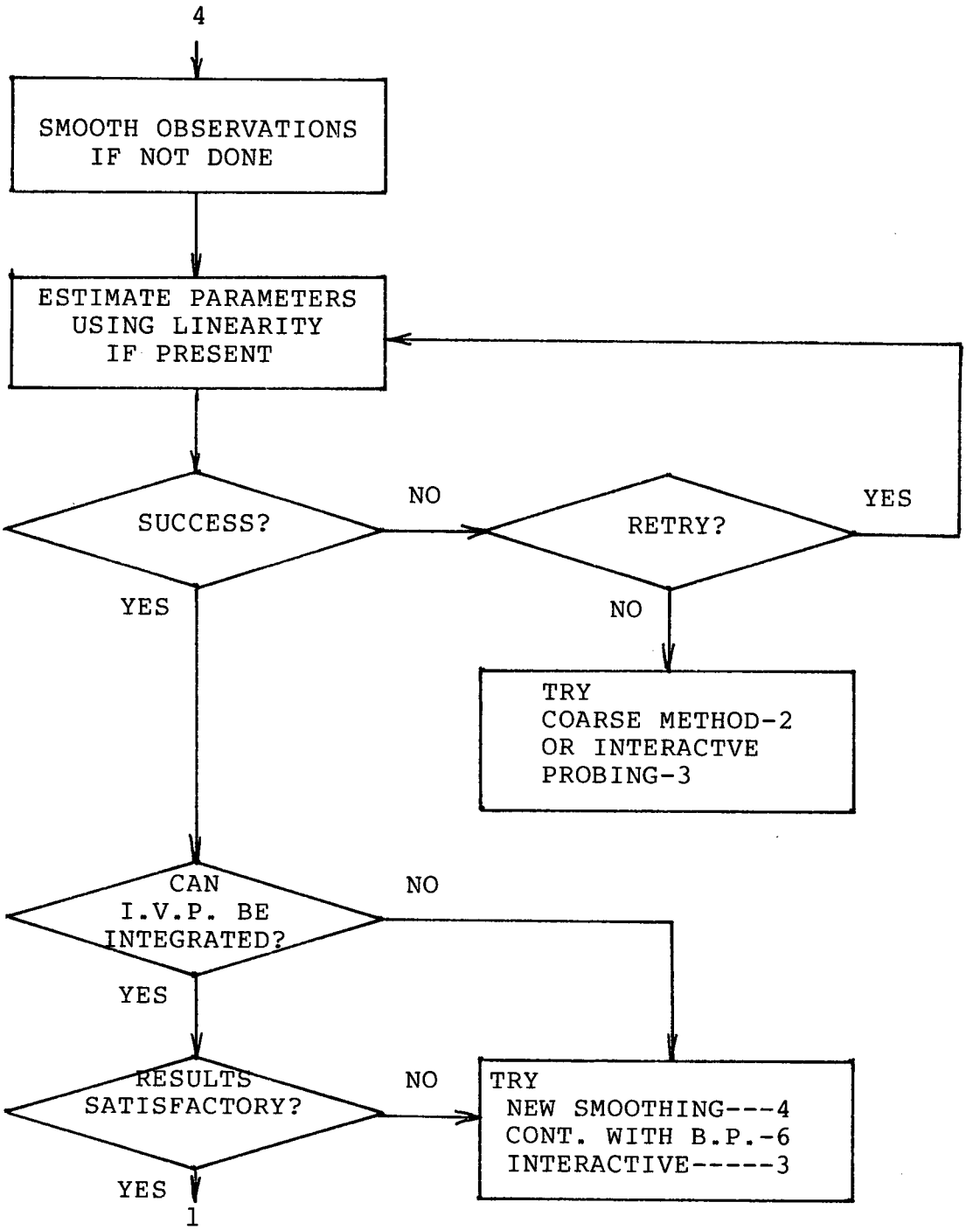

Figure 3.6.4
Derivative and integral fitting

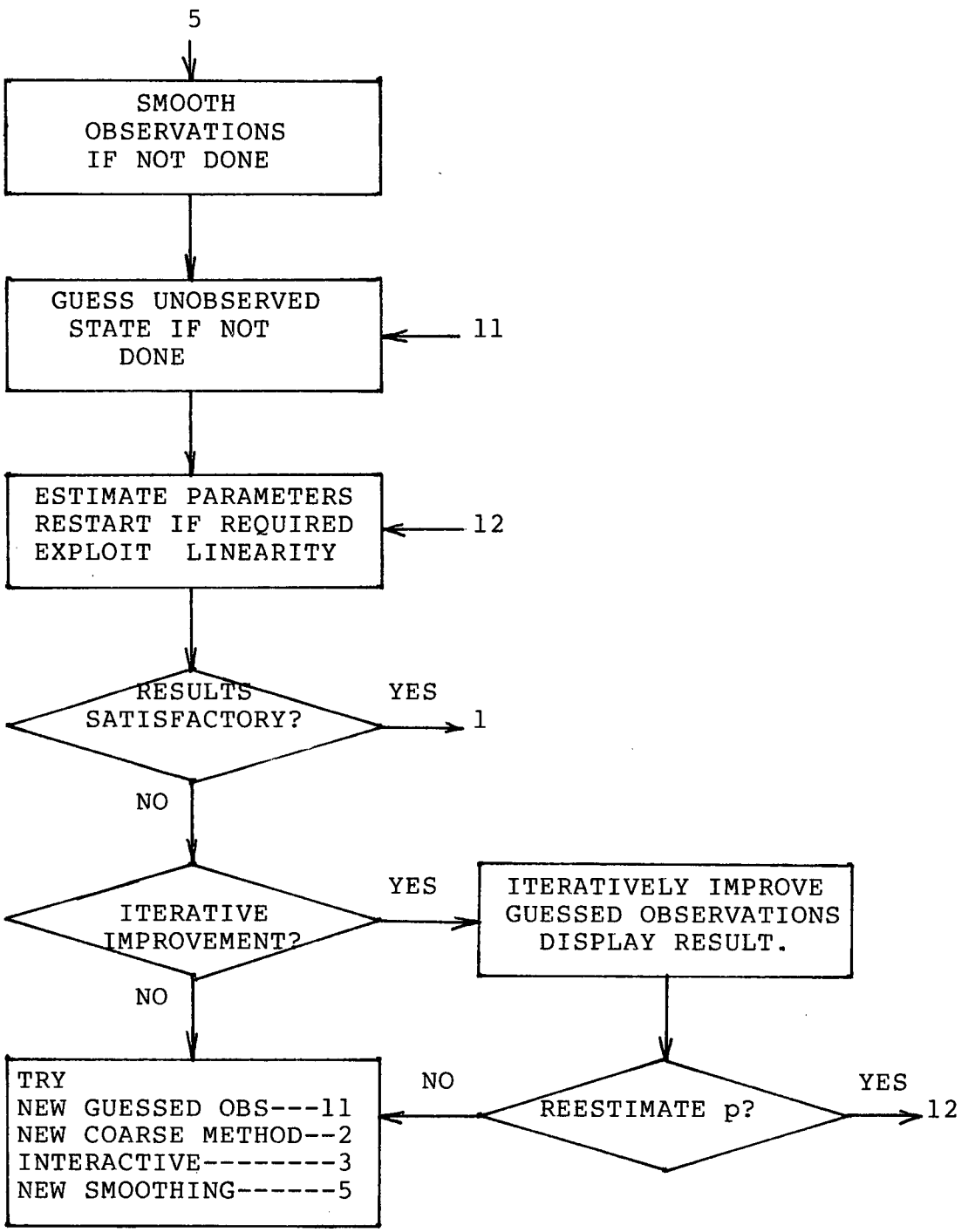

Figure 3.6.5
Guessed observations and iterative improvement

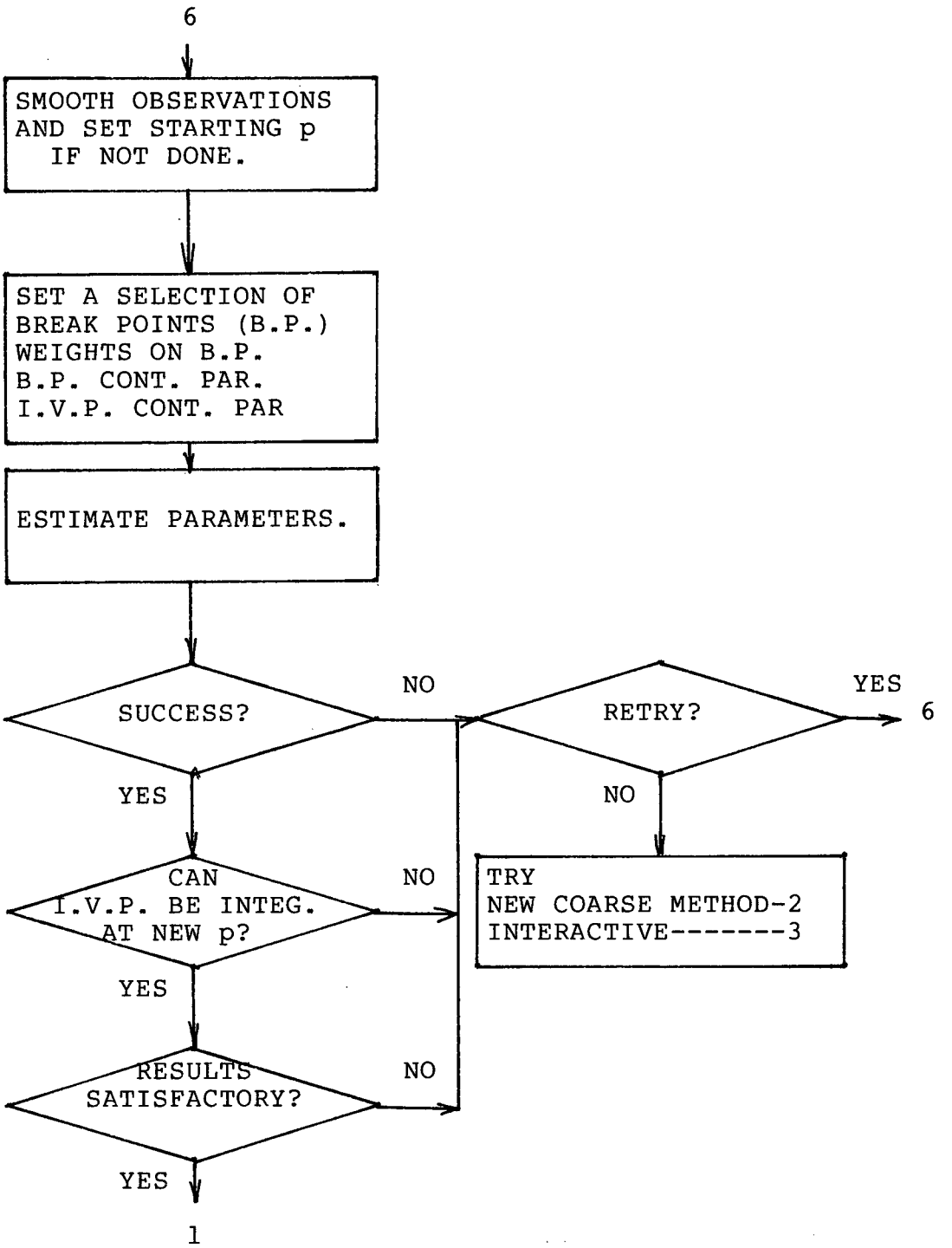

Figure 3.6.6
Continuation and quasi multiple shooting

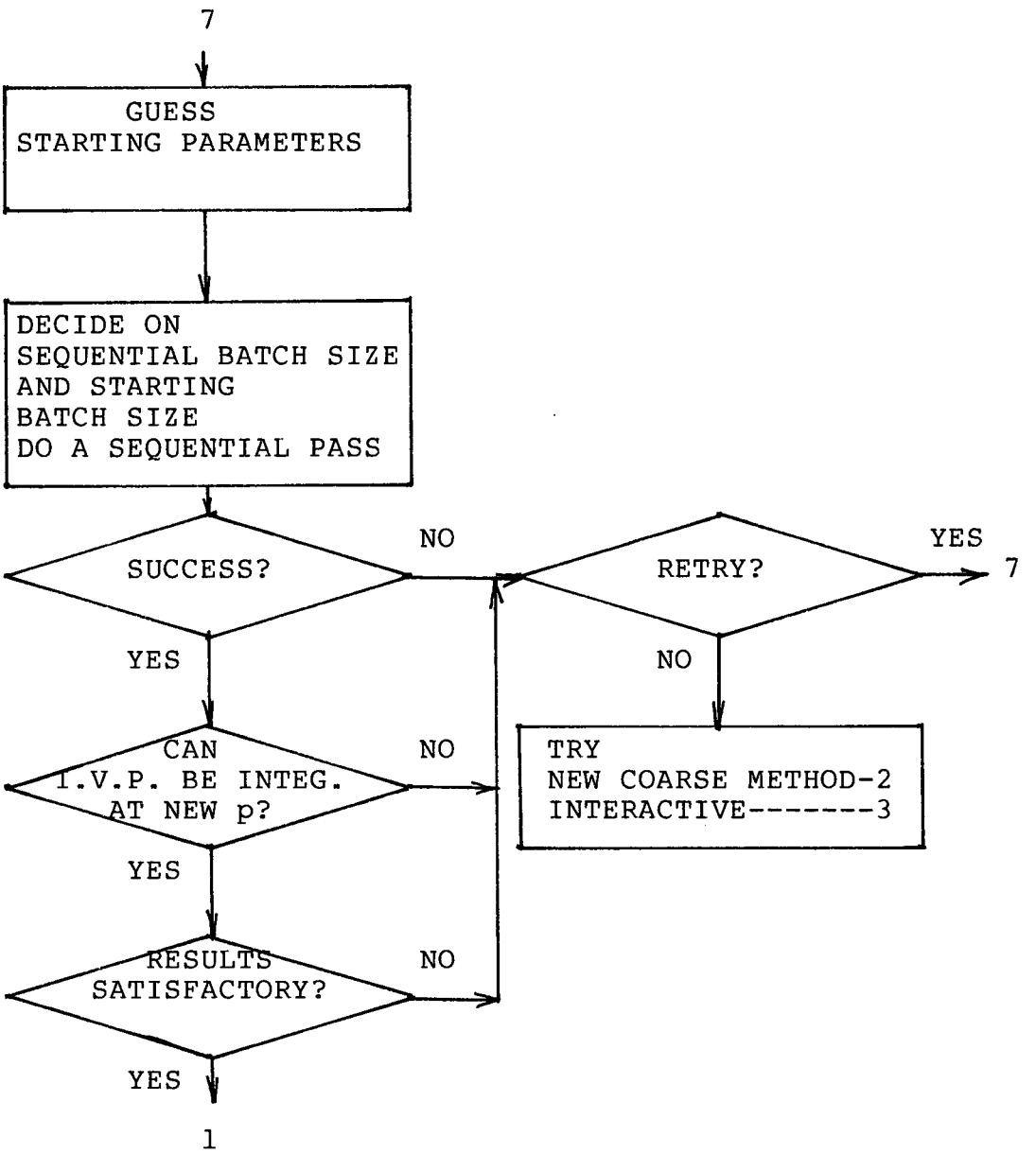

Figure 3.6.7
Sequential reestimation (not implemented in PARFIT)

CHAPTER 4

PARAMETER FITTING IN A PREDATOR-PREY DYNAMIC MODEL

4.1 INTRODUCTION

We consider in detail the problem of fitting parameters in a predator-prey dynamic model studied by Bazykin[7]. This model has several aspects which are attractive to ecologists modelling population dynamics. Depending on the parameter values, there are several possible phase plane configurations for the model, some of which contain limit cycles. Certain limits to population growth are also inherent in the model and this is physically appealing. Unfortunately, the varied behavior of the model which makes it rich from an ecological point of view , complicates the parameter fitting problem. For example, with a poor guess at the optimal parameters, we may find ourselves with a phase plane configuration quite different from the one indicated by the observations. The ability of some of the methods of Chapter 3 to handle this problem is explored in this chapter. Only one parameter in the model considered by Bazykin occurs nonlinearly in the differential equations, and thus we expect some of the more global methods of Chapter 3 to be well suited to this model.

4.2 A PREDATOR-PREY DYNAMIC MODEL

The dynamic model considered in [7] has the form:

$$y_1' = p_1 y_1 - p_2 y_1 y_2/(1+p_5 y_1) - p_6 y_1^2$$
$$y_2' = -p_3 y_2 + p_4 y_1 y_2/(1+p_5 y_1) - p_7 y_2^2 \qquad (4.2.1)$$

We confine ourselves to a very brief interpretation of this model. For more details see [7]. State variable y_1 represents the prey population, and state variable y_2 represents the predator population. The above equations evolved from the Lotka-Volterra model introduced in Section 3.4. Equations (4.2.1) reduce to the Lotka-Volterra model equations when $p_5 = p_6 = p_7 = 0$. The term $1/(1+p_5 y_1)$ in the above equations represents the satiation of the predators. That is, as the number of prey increases, the ability of the predators to consume prey is limited, and when the number of prey is large, the growth rate of the predators becomes independent of the prey population. The terms $p_6 y_1^2$ and $p_7 y_2^2$ represent competition among the prey and among the predators respectively. As the prey population y_1 increases, it becomes limited by such things as territory. Similar limitations apply to the predator population.

We consider one of the special cases studied by Bazykin. In this case, $p_7 = 0$ and there are two equilibrium points A and B in the phase plane (see Bazykin[7]). Point A is at

$$(p_3/(p_4 - p_5 p_3),\ (p_4/p_2)(p_1(p_4 - p_5 p_3) - p_6 p_3)/(p_4 - p_5 p_3)^2)$$

and point B is at

$$(p_1/p_6,\ 0).$$

Since all the parameters are positive, for A to have physical meaning, we require $p_6 < p_1 (p_4 - p_5 p_3)/p_3$. Bazykin further shows that if point A is stable then $p_6 > p_2 p_5 a_2/(1+p_5 a_1)^2$ where A has coordinates (a_1, a_2), and if A has biological significance and is unstable, it is necessary that a limit cycle occur. We consider three situations:

(1) A has biological significance and is stable;

(2) A has biological significance and is unstable; and

(3) A has no biological significance.

Case (1) arises for example at the parameter values

$$(1, .1, 3, 1, .1, .15)^T.$$

In this case A is at (4.29, 5.10) and B is at (6.67, 0). Case (2) arises at the point

$$(.5, .1, 5, 1, .15, .01)^T$$

in parameter space. In this case A is at (20.0, 12.0) and is unstable and B is at (50.0, 0). Case (3) arises at the point

$$(1, .1, 3, 1, .2, .15)^T$$

in parameter space. In this case A has no biological significance and B is at (6.67, 0). Phase plane plots corresponding to the above three parameter vectors are shown in Figures 4.2.1, 4.2.2, and 4.2.3. We use integration results at the above three points in parameter space to simulate observations for our test problems.

In particular, we make use of the following three test

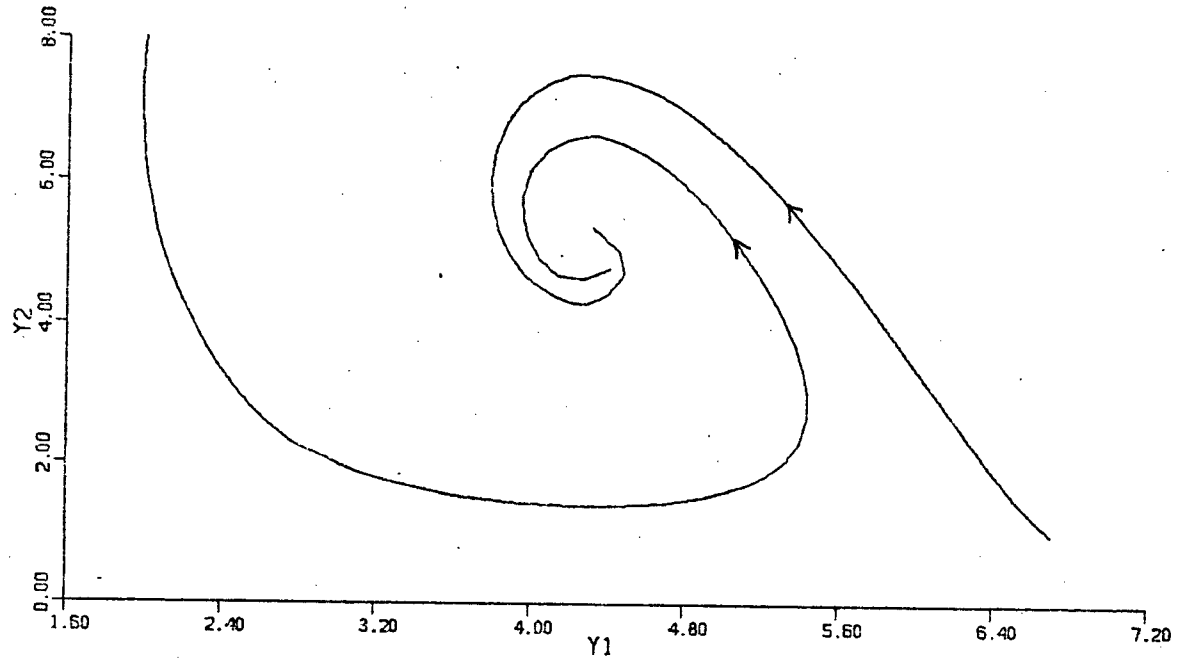

Figure 4.2.1
Phase plot for case (1)

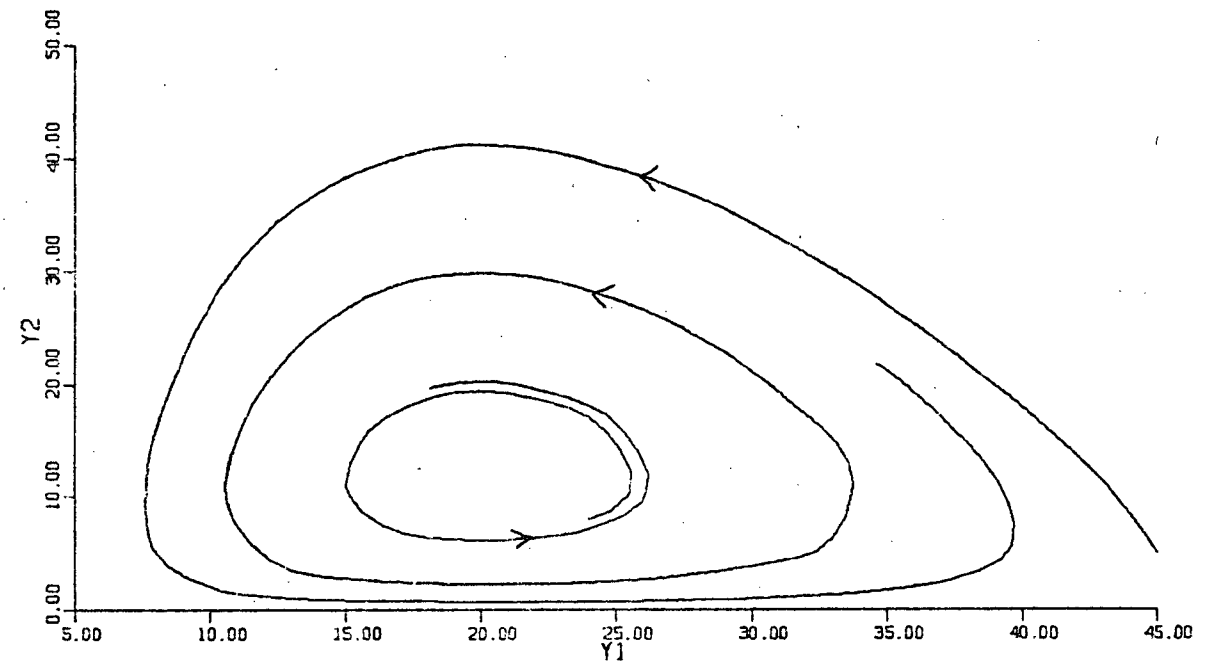

Figure 4.2.2
Phase plot for case (2)

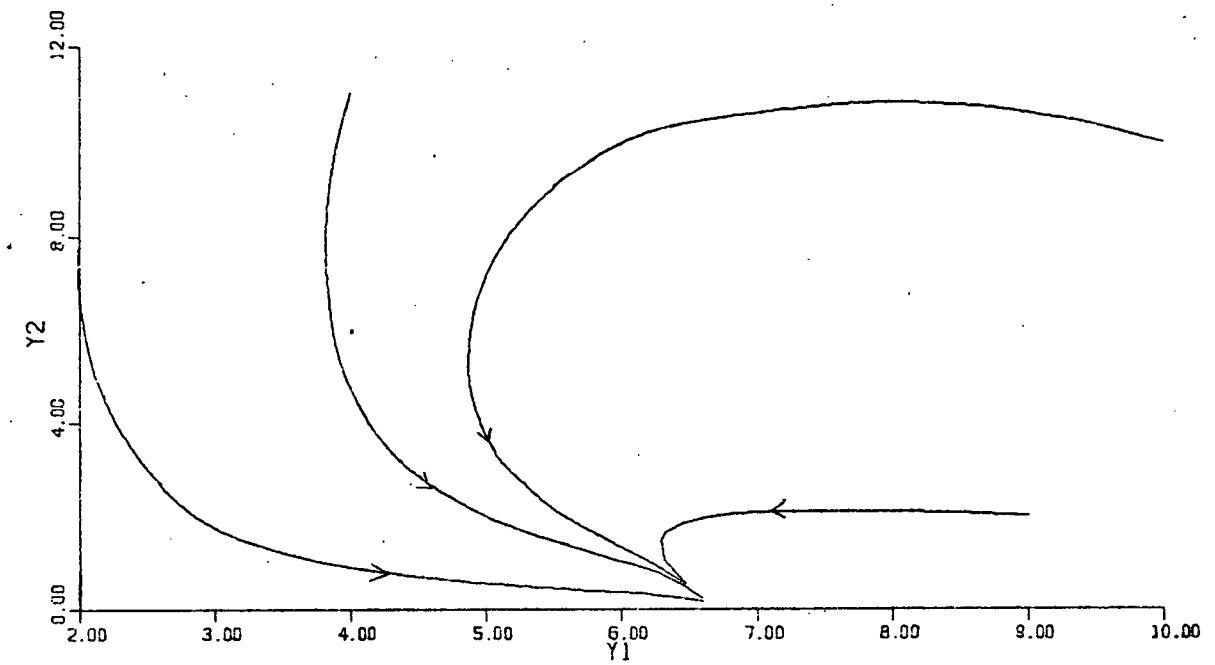

Figure 4.2.3
Phase plot for case (3)

problems:

PROBLEM 4.2.1:

Observations are generated using the parameters of Figure 4.2.1 and the initial condition $y(0)=(2, 8)^T$.

PROBLEM 4.2.2:

Observations are generated using the parameters of Figure 4.2.2 and the initial condition $y(0)=(24, 8)^T$.

PROBLEM 4.2.3:

Observations are generated using the parameters of Figure 4.2.3 and the initial condition $y(0)=(10, 10)^T$.

In all of the above problems simulation runs were made using Gear's program [27]. No random error was introduced into the generated observations, and observations were generated for both

state variables at times .5(.5)12.5 in all cases. The observations for Problems 4.2.1, 4.2.2, and 4.2.3 are displayed in Figures 4.2.4, 4.2.5, and 4.2.6 respectively. We also show

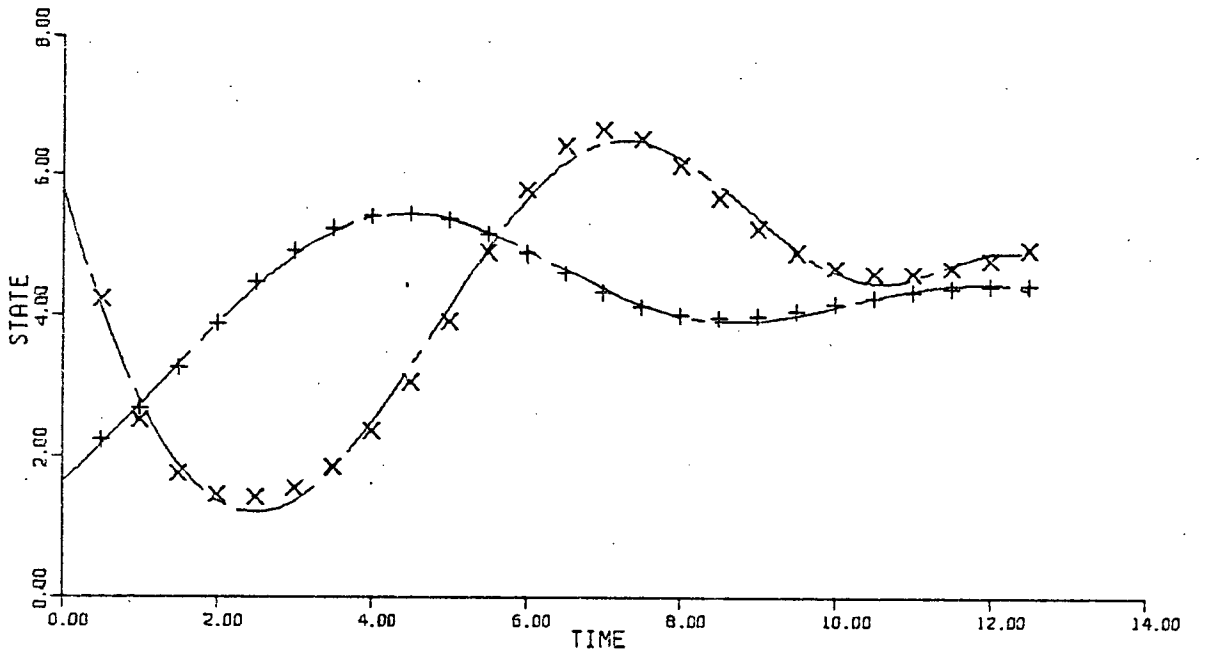

Figure 4.2.4
Observations for Problem 4.2.1

the cubic spline least squares approximations to the observations that are used throughout this chapter. In Figure 4.2.4, the joints for the spline approximating the observations on y_1 are at t=4,8. The joints for the cubic spline approximating the y_2 observations are at t=7,10.5. In Figure 4.2.5, the joints for the spline approximating the y_1 observations are at t=6.5,11.25, and the joints for the spline approximating the y_2 observations are at t=2,3.75,8.75. In Figure 4.2.6, the joints for the spline approximating the observations on y_1 are at t=2.5,7.5, and the joint for the

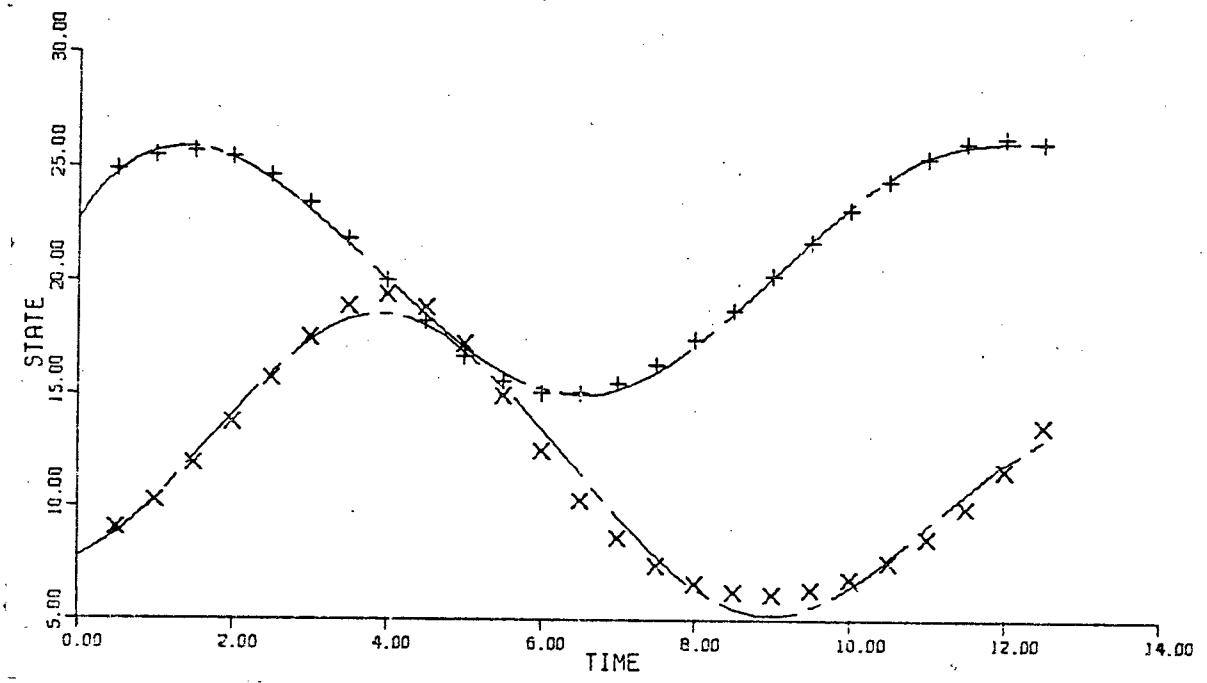

Figure 4.2.5
Observations for Problem 4.2.2

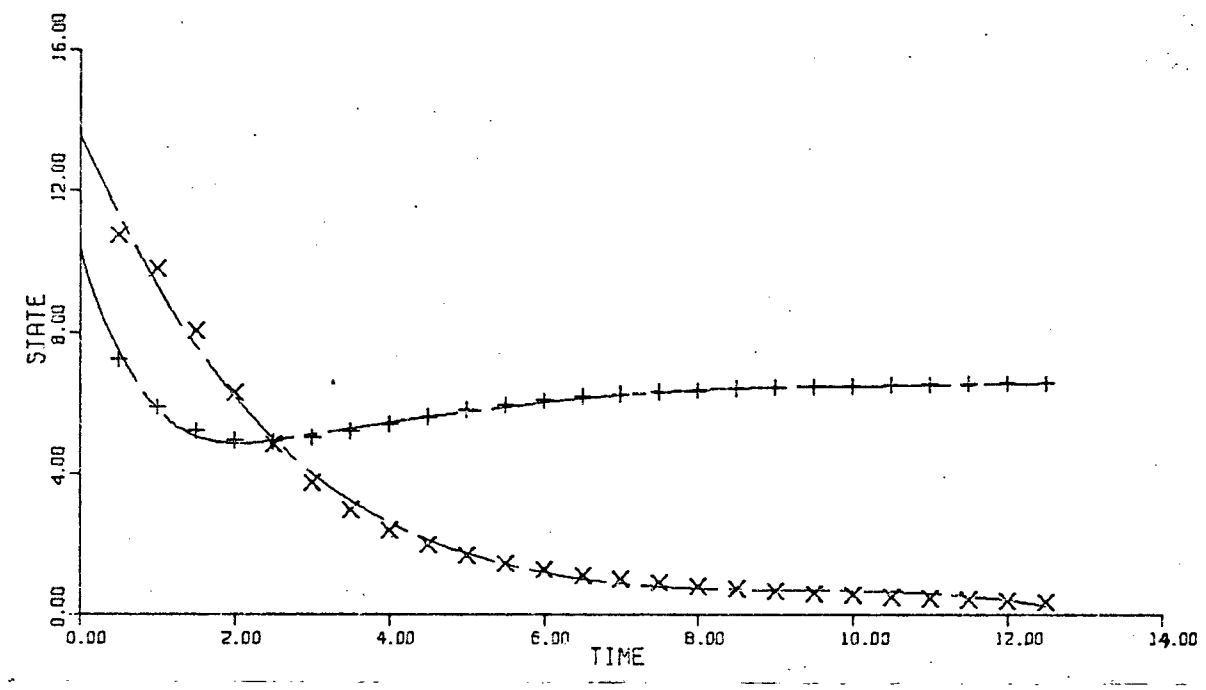

Figure 4.2.6
Observations for Problem 4.2.3

spline approximating the observations on y_2 is at t=5.

4.3 IMPROVING STARTING PARAMETERS

In Table 4.3.1 we present results using the Levenberg-Marquardt method and employing the sensitivity equations (for convenience we call this the FIT technique) along with results using the derivative fitting (DFIT) method followed by the FIT method to refine the parameter values, and results using the integral fitting (IFIT) method followed by the FIT method to refine parameter values. The derivative and integral fitting methods are used as initial techniques to improve our approximations to the optimal parameter values. From Table

Problem	Starting p	Results with indicated method		
		( )-a figure, C-convergence, L-local minimum		
		FIT	DFIT+FIT	IFIT+FIT
4.2.1	(4.2.2)	L(4.3.1)	C	C
	(4.2.3)	C	C	C
4.2.2	(4.2.1)	C	C	C
	(4.2.3)	L(4.3.2)	C	C
4.2.3	(4.2.1)	C	L(4.3.3)	C
	(4.2.2)	C	L(4.3.3)	C

Table 4.3.1
FIT compared with DFIT+FIT and IFIT+FIT

4.3.1, we see that both the DFIT and IFIT methods work well for improving the approximations to the optimal parameters. This is not too unexpected since the differential equations are linear in all but one parameter. We observe that a direct method using

the sensitivity equations can lead to difficulties. This reflects the increased nonlinearity that arises when we employ direct integration of the initial value problem. The situation that arose when the Levenberg-Marquardt technique was applied to Problem 4.2.1 starting with the parameters of Figure 4.2.2 is typical of the sort of thing that can happen. In this case a local minimum at

$$(3.211,\ -.4441,\ -3.405,\ 3.159,\ -1.182,\ .6314)^T$$

was found. Integration results at these parameters are shown in Figure 4.3.1. At this local minimum the peaks and troughs in

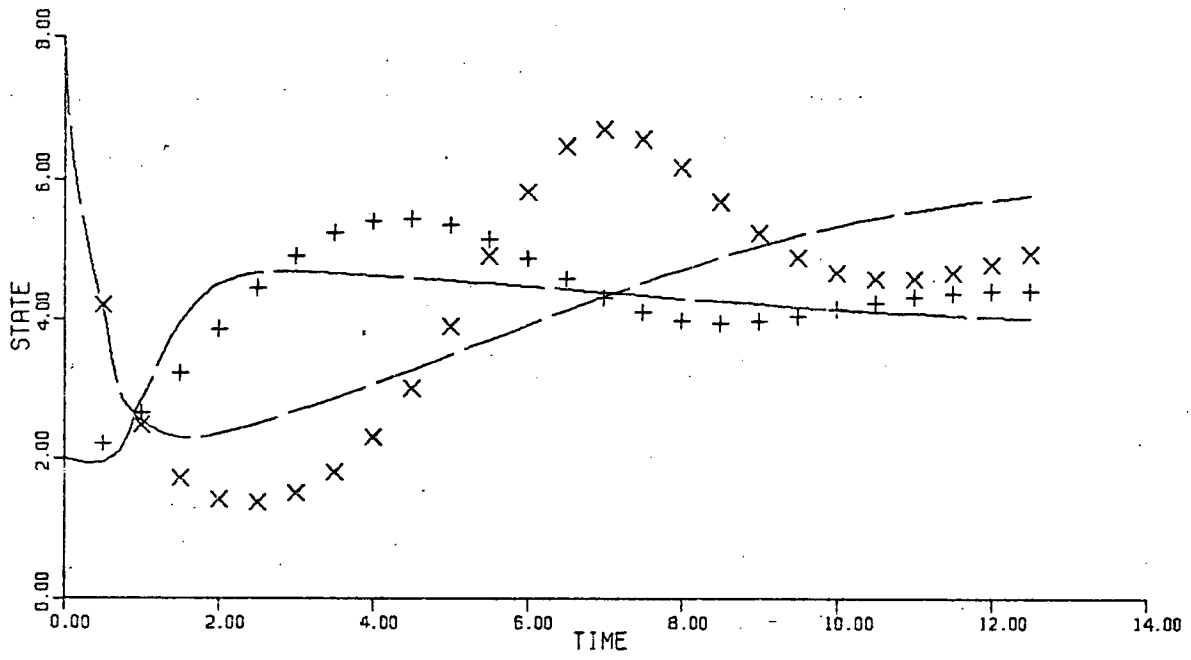

Figure 4.3.1
A local minimum for Problem 4.2.1

the observations are being balanced against one another.

The FIT approach worked for Problem 4.2.2 starting at the parameters of Figure 4.2.1, but difficulties were encountered.

Specifically, the parameter λ in the Levenberg-Marquardt procedure had to be adjusted to avoid certain points in parameter space where the integration blows up. The use of constraints might also be useful here; however, it is noteworthy that no such difficulties arose with the DFIT+FIT and IFIT+FIT methods on this problem.

The FIT approach to Problem 4.2.2 starting at the parameters of Figure 4.2.3 produced a local minimum at

$$(3.087,\ .6111,\ 3.086,\ .7745,\ .2007,\ .08077)^T.$$

Integration results at these parameters are shown in Figure 4.3.2. No difficulties arose in this case with the DFIT+FIT and

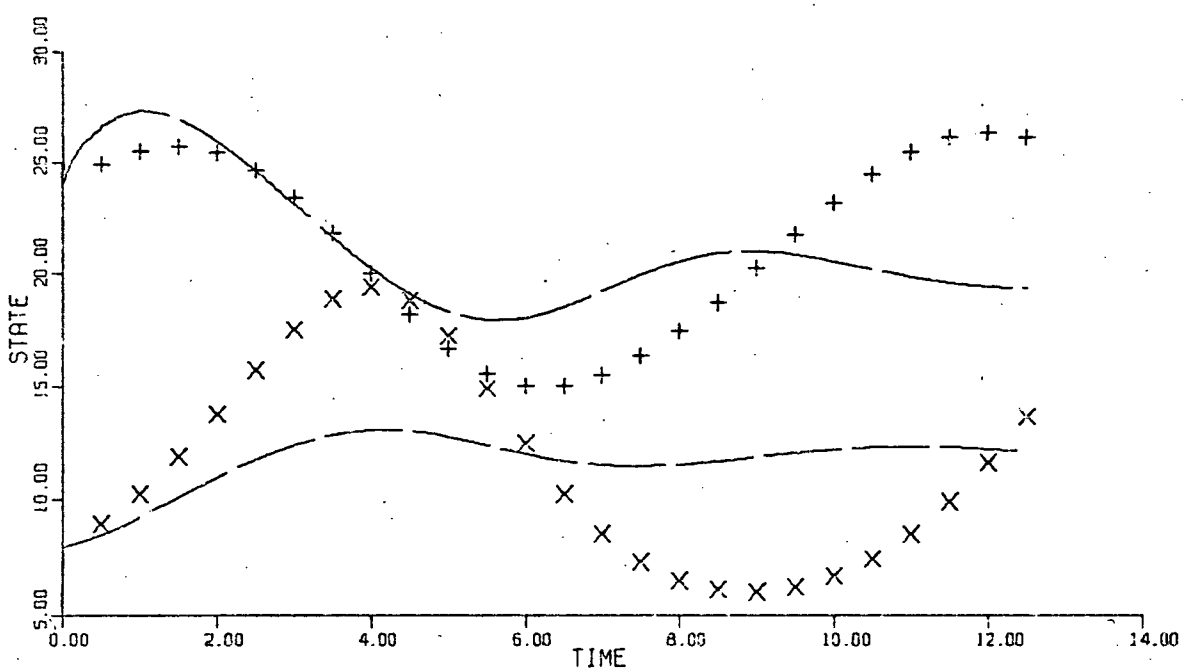

Figure 4.3.2
A local minimum for problem 4.2.2

IFIT+FIT methods.

Problem 4.2.3 requires further discussion. At first

glance, it appears to be the simplest of the three problems since the observations seem to exhibit no special features. However, there is a local minimum in parameter space corresponding to a solution to the differential equation which comes very close to the given observations. This appears to result from the small peak in y_2 near t=0.5, and from the contrived nature of the problem with no random error in the observations. The observations start at t=0.5 and if we do not look at the initial conditions, the small peak near t=0.5 is invisible. Thus the DFIT method found a point in parameter space where this peak was absent. Consequently the DFIT+FIT combination found a local minimum at the parameters

$$(1.028,\ .2255,\ 1.290,\ .7186,\ .5493,\ .1517)^T.$$

The sum of the squares of the residuals at this point in parameter space was approximately 10. Integration results at the above parameters are shown in Figure 4.3.3. If there were some random error in the observations, the solution shown in Figure 4.3.3 might appear quite adequate. However, in this contrived case there exists a more optimal solution with the objective function equal to zero. Our implementation of the IFIT method which has access to the initial values had no difficulty with this problem. The FIT approach to this problem managed to extract the global optimum, but not without some difficulties. Starting with the parameters corresponding to Figure 4.2.1, no problems arose; however, starting with the

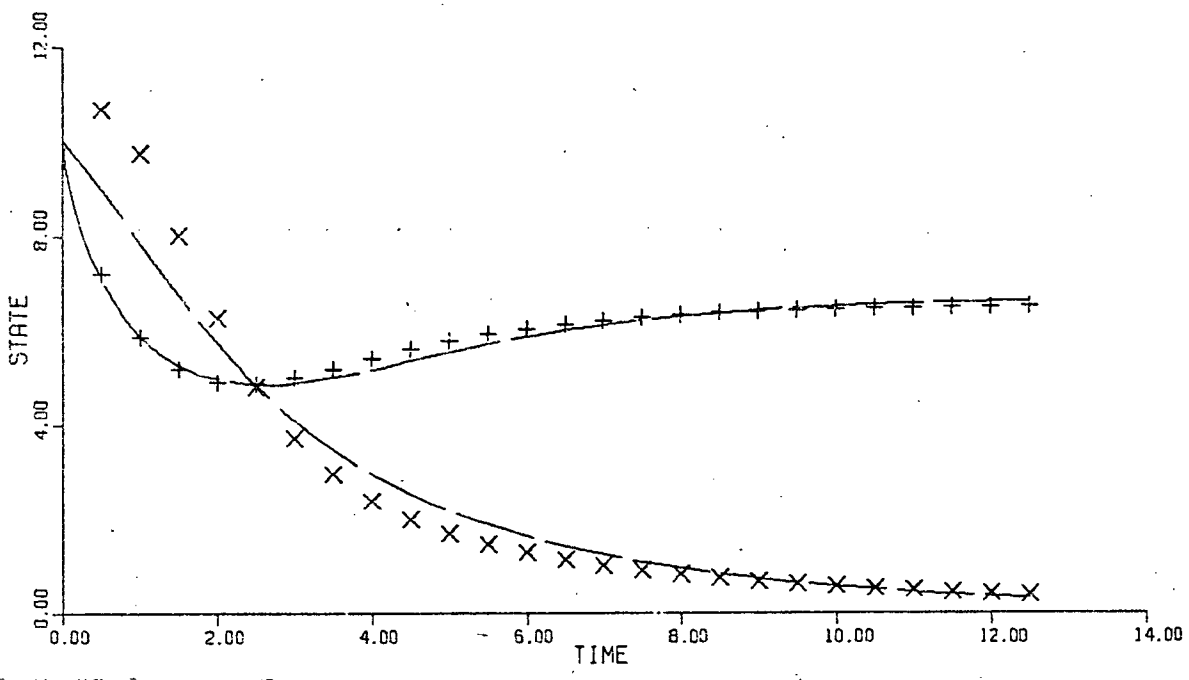

Figure 4.3.3
A local minimum for Problem 4.2.3

parameters of Figure 4.2.2, difficulties were encountered and λ in the Levenberg-Marquardt procedure had to be increased to avoid certain regions in parameter space.

4.4 GUESSED OBSERVATIONS AND ITERATED METHODS

In this section we present some experiments with the techniques introduced in Section 3.4. These techniques are designed for the important case when observations are not available on all state variables. First we experiment with the use of guessed observations for unobserved state variables. To facilitate the guessing of state variable behavior, an interpolating cubic spline is passed through a set of guessed observation points. This eliminates the need to enter long lists of guessed observations. Experiments are conducted on

PROBLEM 4.4.1:

This problem is the same as Problem 4.2.1 except observations are only available on state variable y_1 and $y_2(0)=8$.

Initially we looked at Problem 4.4.1 with $y_2(0)$ an unknown parameter. However, this problem was singular due to a linear realtionship between p_2 and the initial condition on y_2. The columns corresponding to these two parameters in the least squares Jacobian matrices are multiples of one another and the correlation coefficient between these two parameters is 1. This occurs because the state equation involving y_2' is homogeneous in y_2. Thus y_2 can be replaced by cy_2 and the second state equation is not altered, except for the initial condition on y_2 which is divided by c. However, in the first state equation, p_2 is replaced by cp_2. Thus a relation exists between p_2 and the initial condition on y_2.

To apply the derivative fitting method to Problem 4.4.1, we approximate the observations on y_1 with the least squares cubic spline used for experiments with Problem 4.2.1 presented in Table 4.3.1. In Figure 4.4.1, some guesses at possible observations on y_2 are shown. Referring to this figure, curve (a) is an interpolating cubic spline for the points

(0, 8), (2.5, 1.5), (7, 6.5), (10, 4.5), (12.5, 5),

curve (b) is an interpolating cubic spline for the points

(0, 8), (2.5, 3), (8.75, 5), (12.5, 4).

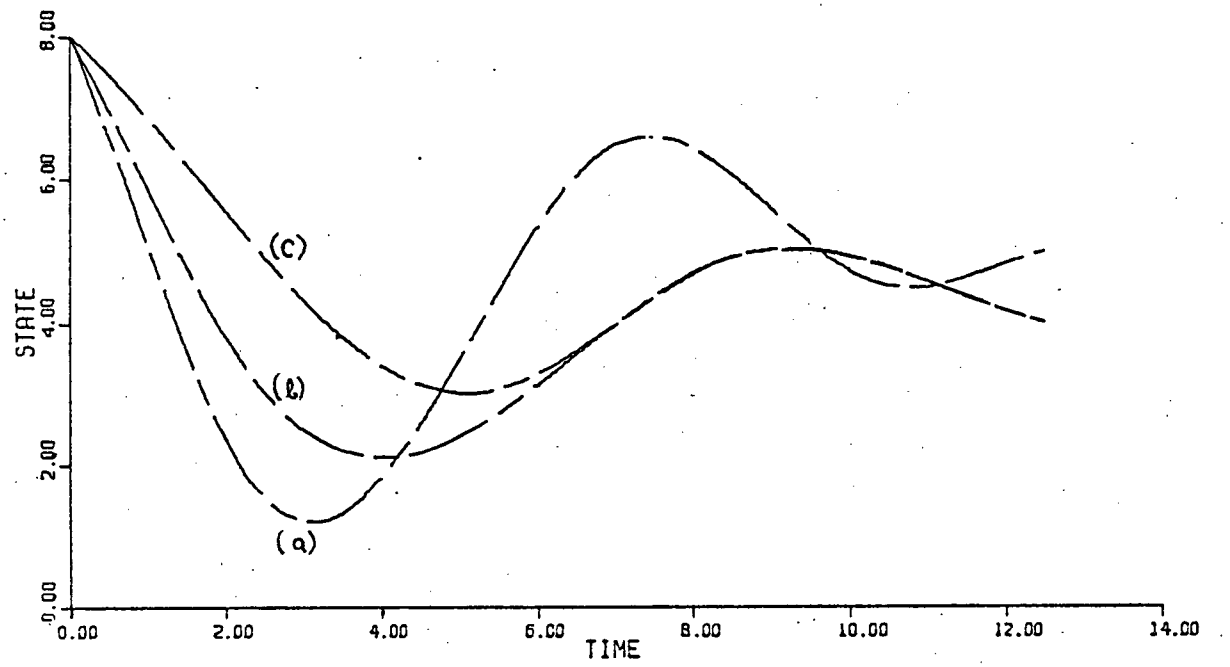

Figure 4.4.1
Guessed observations for problem 4.4.1

and curve (c) is an interpolating cubic spline for the points

(0, 8), (5, 3), (8.75, 5), (12.5, 4),

(that is, one point was moved). The guessed observations shown in curve (a) are fairly close to the observations on y_2 in Problem 4.2.1, while the guessed observations in curve (c) are substantially different from those of Problem 4.2.1. The guessed observations in curve (b) are intermediate to those in curves (a) and (c). End conditions for the interpolations are described under the CREOBS command in Appendix A.

Figure 4.4.2 shows the integration results at the parameter vector

$$(.2354, -.1977, 4.983, 1.049, .01350, .05878)^T$$

obtained with the FIT approach (with error controlled integrations) to problem 4.4.1 using the starting parameters of

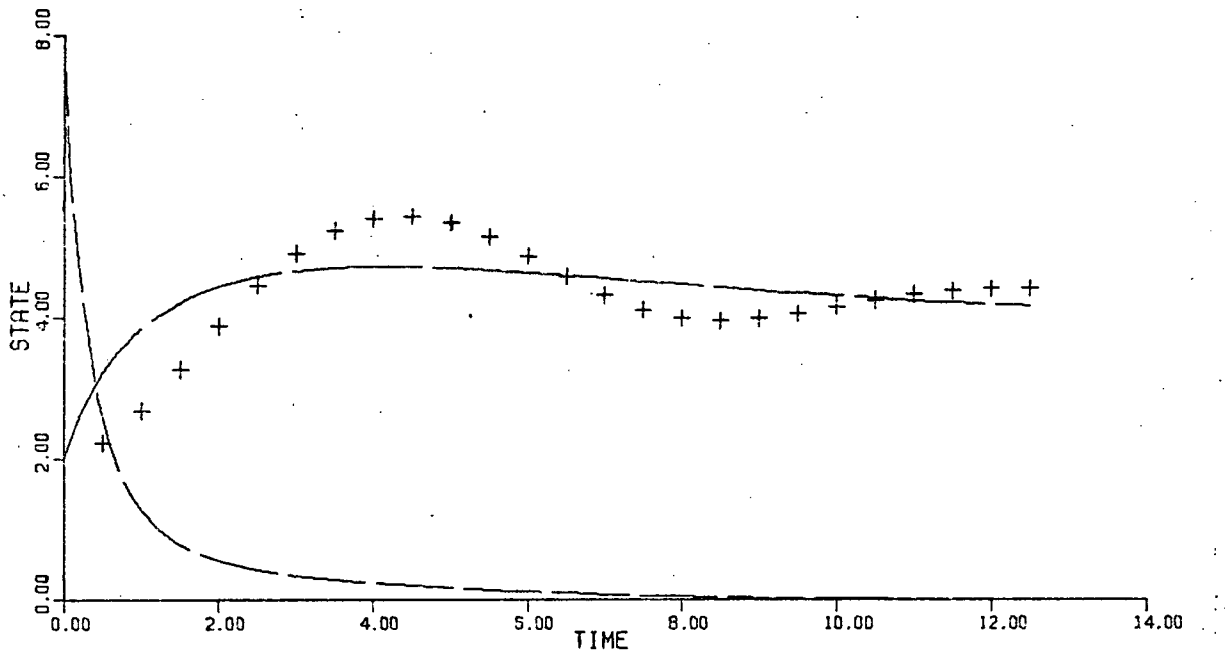

Figure 4.4.2
FIT on Problem 4.4.1

Figure 4.2.2. A local minimum has been obtained.

Using points on curve (a) in Figure 4.4.1 for guessed observations, the DFIT+FIT, and IFIT+FIT combinations both produced the correct parameters. The starting parameter values were those corresponding to Figure 4.2.2. Using guessed observations from curve (b) in Figure 4.4.1, the DFIT+FIT combination found a local minimum at

(.2507, -.3758, 1.676, 2.729, 1.698, .07026).

Integration results at these parameters are shown in Figure 4.4.3. The IFIT+FIT combination with these guessed observations produced the correct minimum. When guessed observations from

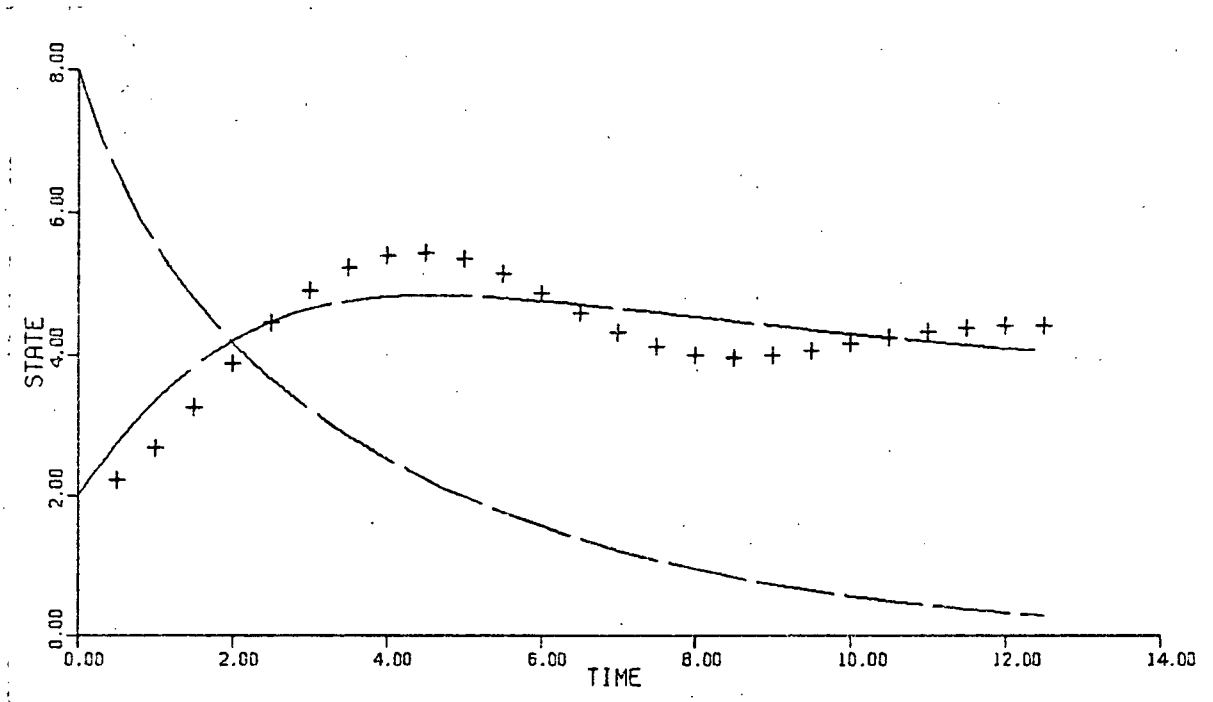

Figure 4.4.3
DFIT+FIT on Problem 4.4.1 using guess (b)

curve (c) in Figure 4.4.1 were used, both the DFIT and IFIT methods produced parameters from which the FIT method was unsuccessful due to instabilities. The change of the position of one data point from t=2.5 to t=5 in the guessed observations of curves (b) and (c) of Figure 4.4.1, meant the difference between disaster and the aquisition of a minimum. The above experiments indicate that some interactive experimentation with guessed observations and with particular methods can be profitable.

Next some experiments using the iterative techniques of Section 3.4 for improving guessed observations are presented. These techniques can reduce the dependence on a good set of guessed observations.

Three methods are considered. First we consider iterated derivative fitting where the function (3.4.1) with $d_\ell(c)$ defined by (3.4.2) is minimized. Second, we consider iterated integral fitting where our parameter estimates are updated by the integral fitting technique and the guessed observations are updated to minimize the function (3.4.1). Finally, we consider an iterative scheme where the parameters are updated using the integral fitting technique, and the guessed observations are updated by integrating a subsystem of the given system of differential equations. Experiments are conducted on Problem 4.4.1. The case when there is a random error in the observations is considered in the next section. In all experiments, the initial condition for the unobserved state variable remains fixed at 8. We observe that the least squares problem for updating the guessed observations with p held fixed is linear in this example. Starting with the guessed observations of curve (b) in Figure 4.4.1 and the parameters corresponding to Figure 4.2.2, all three methods converged to a parameter vector from which the FIT method converged to the desired solution. In all cases, the observations on y_1 were approximated with the spline shown in Figure 4.2.4. The iterated derivative and iterated integral fitting methods employing (3.4.1) produced similar results and we present graphical results only for the iterated derivative fitting case. In Figure 4.4.4, the observations on y_1 and integration results for y_1 at successive parameter estimates obtained with the

iterated derivative fitting method are shown. In Figure 4.4.5,

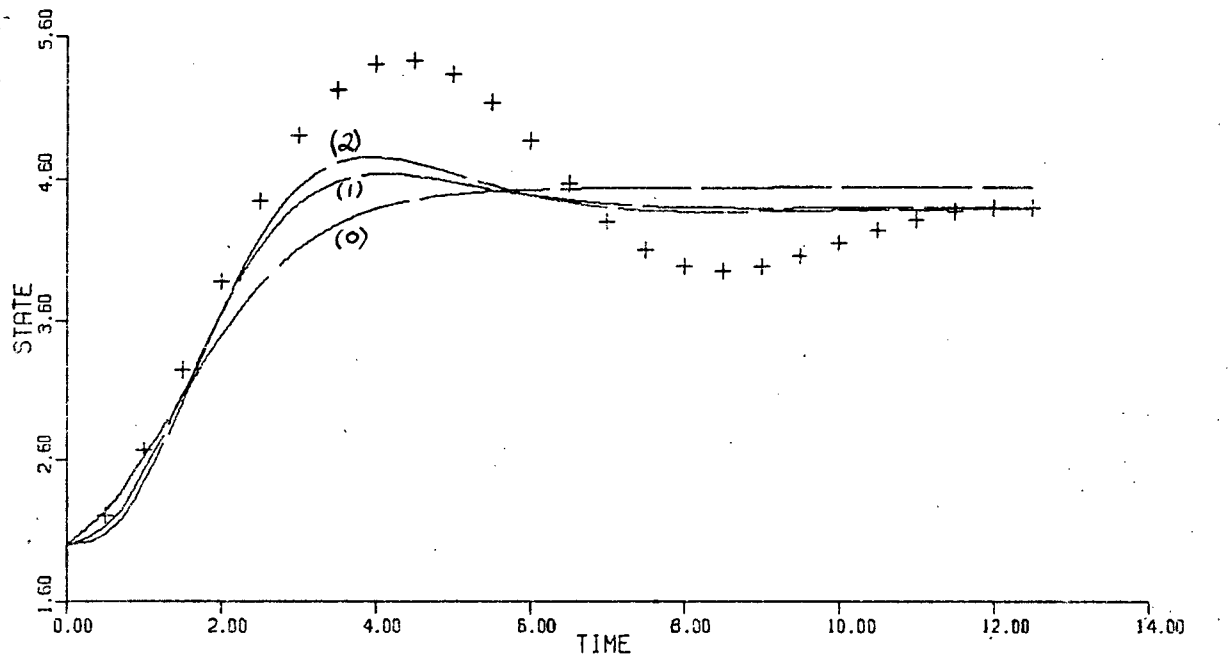

Figure 4.4.4
Iterated DFIT results

the iterations on the guessed observations corresponding to the results presented in Figure 4.4.4 are shown. We note that the derivative fitting method using guessed observations from curve (b) in Figure 4.4.1, did not produce parameters from which the FIT method could find the global minimum, but the iterated derivative fitting method did produce parameters from which the FIT method was successful. In this case the iterations were crucial to obtaining the desired solution.

Next we consider iteratively improving the guessed observations by integrating the unobserved state variable with the observed state variable held fixed at the smoothed approximation to the observations. In Figures 4.4.6 and 4.4.7

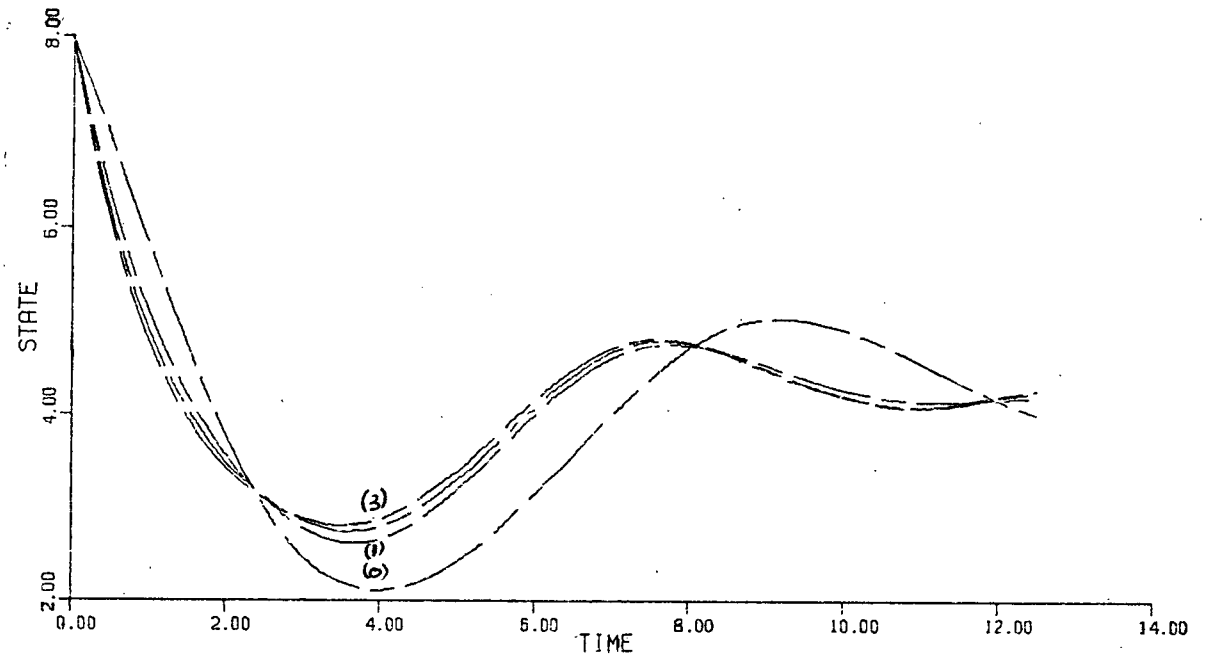

Figure 4.4.5
Iterated DFIT guessed observation iterations

we give results with this technique applied to Problem 4.4.1 starting with the guessed observations of curve (b) in Figure 4.4.1. Figure 4.4.6 shows the observations on y_1 and integration results on y_1 at successive parameter estimates. With this technique, the difference between iteration (0) and iteration (1) was substantial, while iterations (1) and (2) were essentially identical. The iterations shown in Figure 4.4.7 correspond well to the simulation results in Figure 4.2.4. The FIT approach had no trouble converging to the global minimum starting at the parameters provided by this iterative process.

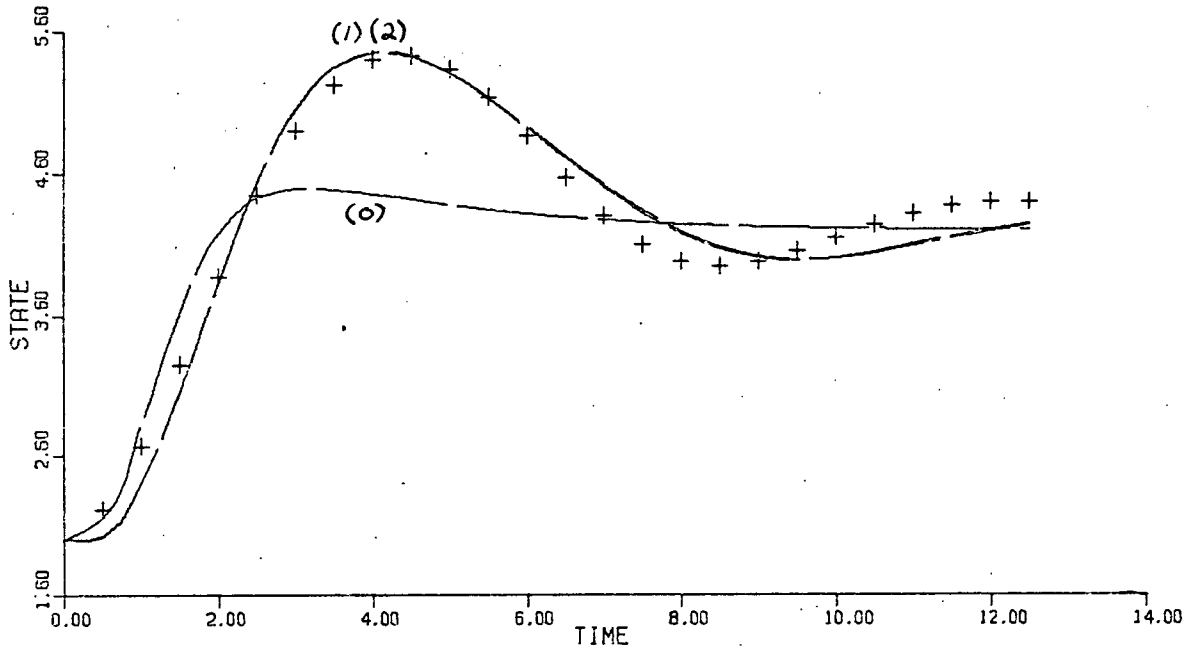

Figure 4.4.6
Iterated IFIT results

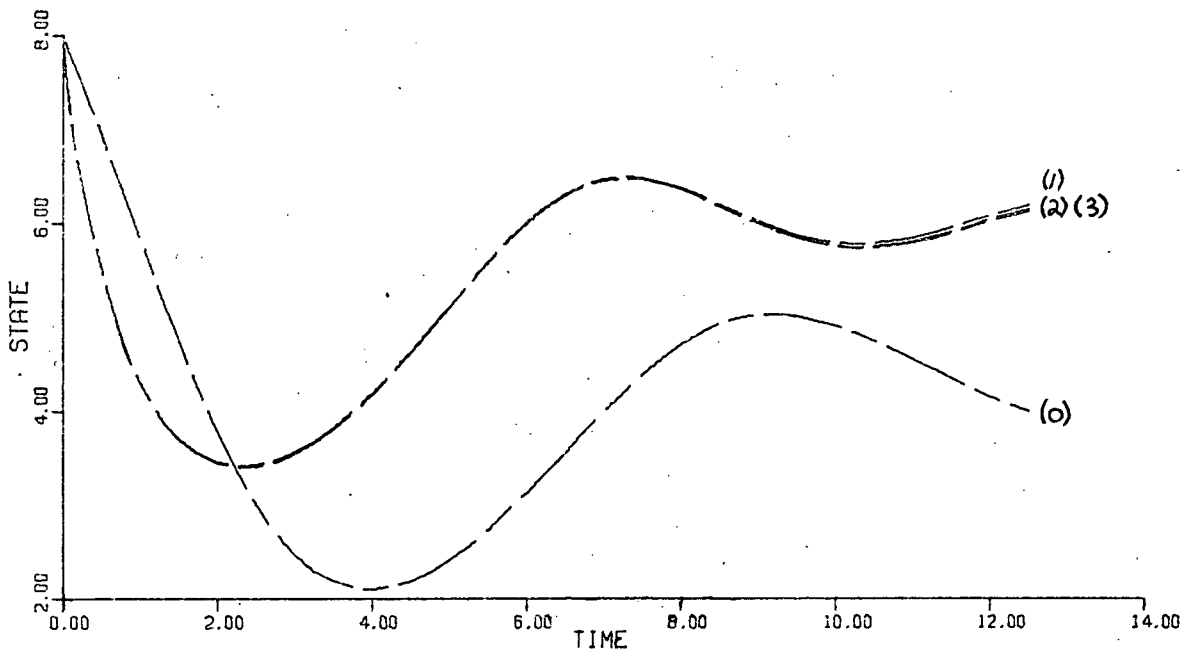

Figure 4.4.7
Iterated IFIT guessed observations

4.5 THE PRESENCE OF NOISE

In this section we consider a selection of the previous problems with a random error introduced into the observations. The problems considered are:

PROBLEM 4.5.1

This problem is the same as Problem 4.2.1 except a normally distributed random error with zero mean has been introduced into the observations. The standard deviation for the error in y_1 is .5 and the standard deviation for the error in y_2 is 1.

PROBLEM 4.5.2

This problem is the same as Problem 4.2.2 except a normally distributed random error with zero mean and standard deviation 2 has been introduced into the observations.

PROBLEM 4.5.3

This problem is the same as Problem 4.5.1 except the observations on y_2 have been removed. The initial condition for y_2 is fixed at 8.

We smooth the observations for Problems 4.5.1, 4.5.2, and 4.5.3 using least squares piecewise cubic splines with the same joints as were used for the smoothings in Problems 4.2.1, 4.2.2, and 4.4.1 respectively.

First we consider Problem 4.5.1. Starting at the parameters correspomding to Figure 4.2.2, a direct approach (FIT method) found the point

$$(.3668, -.08827, 4.874, 1.237, -.4058, .07037)^T$$

in parameter space. Integration results at this point were qualitatively quite different from the observations. The sum of the squares of the residuals at the above point in parameter space was approximately 515. The IFIT+FIT and DFIT+FIT methods both found a minimum at

$$(.8865, .04991, 2.983, .6827, -.009550, .1483)^T.$$

In both cases, the starting parameters were those corresponding to Figure 4.2.2. The sum of the squares of the residuals at the above point in parameter space was approximately 15.5. The FIT method also found the above minimum when it was started from the parameters corresponding to Figure 4.2.3. Unfortunately, p_5 is negative. This suggests we try a square root scaling of p_5 to constrain it to be positive. With this scaling and starting at the parameters of Figure 4.2.2, the IFIT+FIT combination produced the parameters (unscaled)

$$(.8962, .05327, 3.076, .7405, .001897, .1498)^T.$$

The sum of the squares of the residuals at the above parameters was approximately 15.5. Integration results at the above parameters are shown in Figure 4.5.1. Finally, with this scaling, the FIT method also produced the above parameters when started from the parameters of Figure 4.2.2.

Next we consider Problem 4.5.2. Starting at the parameters corresponding to Figure 4.2.3, the FIT method drew us to the point

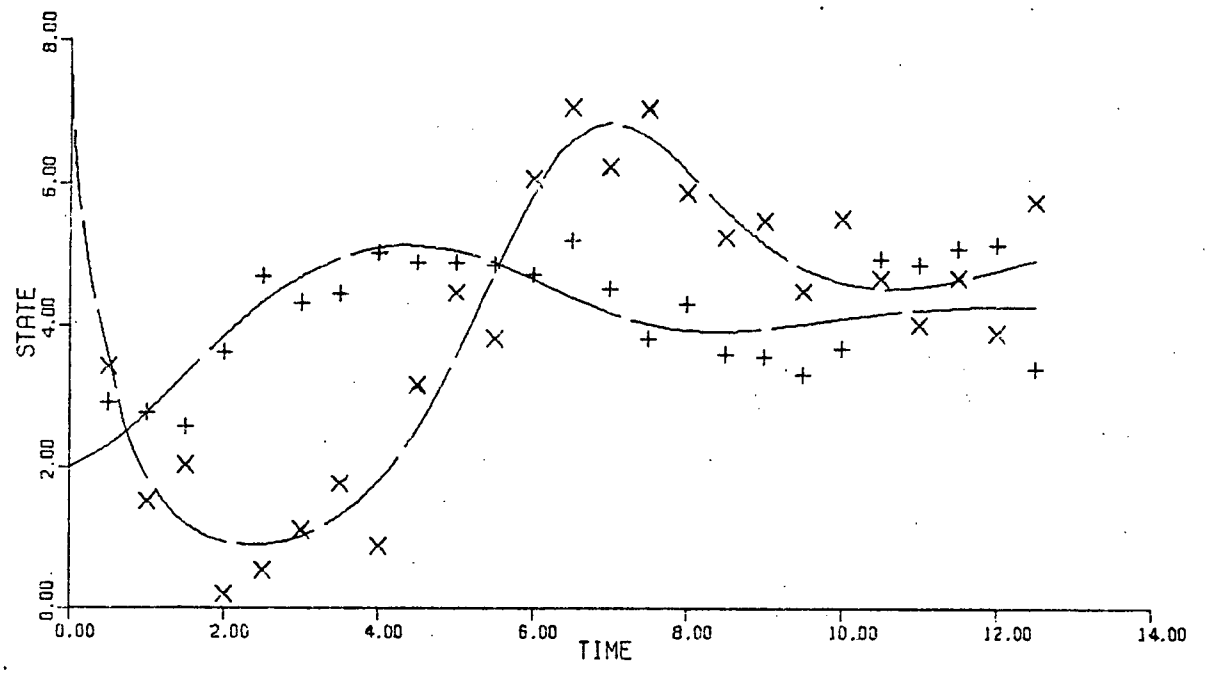

Figure 4.5.1
Results for Problem 4.5.1

$$(5.544,\ .7024,\ 2.647,\ -0.6093,\ -.07095,\ 1.021)^T$$

in parameter space at which the integration results and the observations were qualitatively quite different. (The integration results on y_1 contained a rapid and damped oscillation.) The sum of the squares of the residuals at the above parameters was approximately 1000.

Both the DFIT+FIT, and the IFIT+FIT methods found an optimum at

$$(.4018,\ .03851,\ 1.254,\ .07606,\ .01059,\ .001893)^T$$

The sum of the squares of the residuals at the above parameters was approximately 147. Integration results at the above point

in parameter space are shown in Figure 4.5.2. The FIT method,

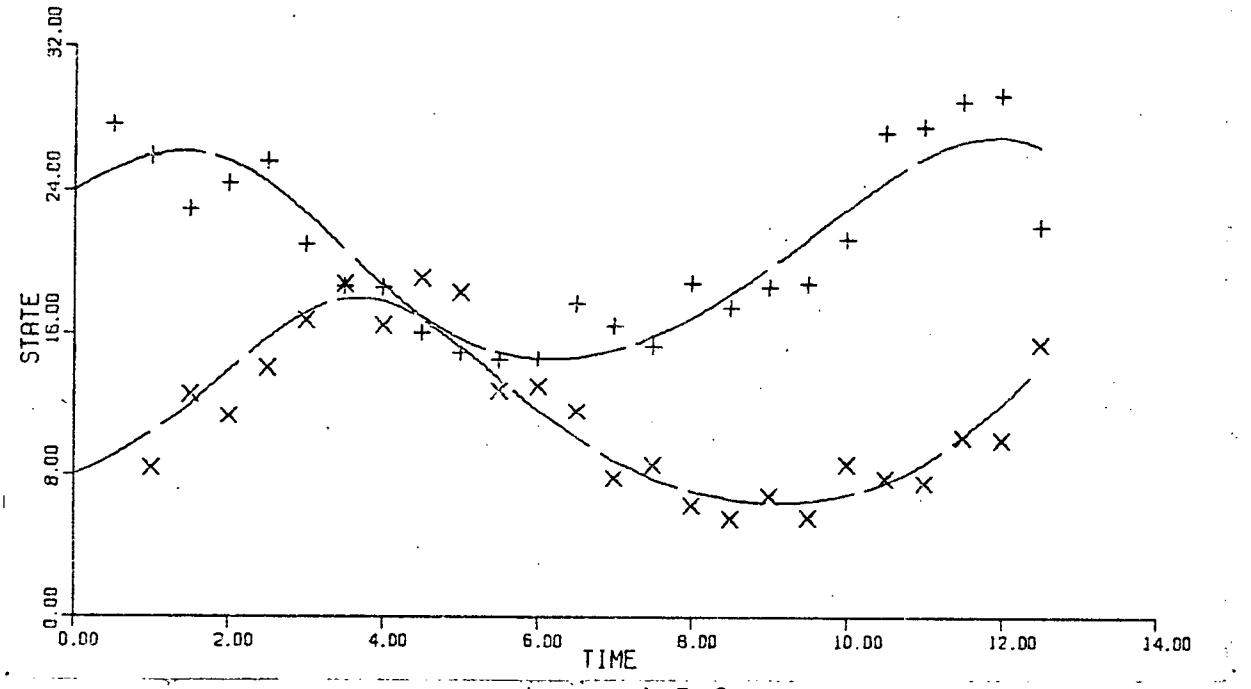

Figure 4.5.2
Results for Problem 4.5.2

using a square root scaling of p_4 and p_5 and starting at the parameters of Figure 4.2.3, also found the above point in parameter space.

Finally, we consider Problem 4.5.3. The random error in the observations combined with the missing observations on y_2 make this a rather nasty problem. The noise is fairly large in this example, but this much noise is not uncommon in problems involving population counts. Starting with the parameters corresponding to Figure 4.2.2, the FIT approach found an optimum at

$$(.5945,\ .0005587,\ 5.760,\ .7368,\ -.1121,\ .1073)^T$$

The sum of the squares of the residuals was approximately 5.4 at

the above parameters. Graphically, integration results at the above parameters look very good. However, p_5 is negative, and the integration results for y_2 at the above parameters are very different from the simulation results shown in Figure 4.2.4. (the integration results for y_2 have a spike which extends to around $y_2=400$ at $t\simeq 7.5$.) Using the guessed observations from curve (b) in Figure 4.4.1, and starting at the parameters corresponding to Figure 4.2.2, three iterations of the iterated derivative fitting method produced the parameters

$$(1.522,\ .3921,\ 3.186,\ 1.709,\ .3031,\ .1829)^T.$$

The observations and integration results at the above parameters are presented in Figure 4.5.3.

Using the same starting conditions as for the previous experiment, three iterations of the iterated integral fitting method (using subsystem integrations) gave the parameters

$$(1.402,\ .2126,\ 3.584,\ 2.497,\ .4548,\ .2249)^T.$$

Integration results at the above parameters are presented in Figure 4.5.4.

Using the same starting conditions as in the previous experiment, three iterations of the iterated integral fitting method (using (3.4.1) to update the guessed observations) produced the parameters

$$(1.388,\ .2153,\ 4.652,\ 2.493,\ .2981,\ .2275)^T.$$

Integration results at the above parameters are presented in

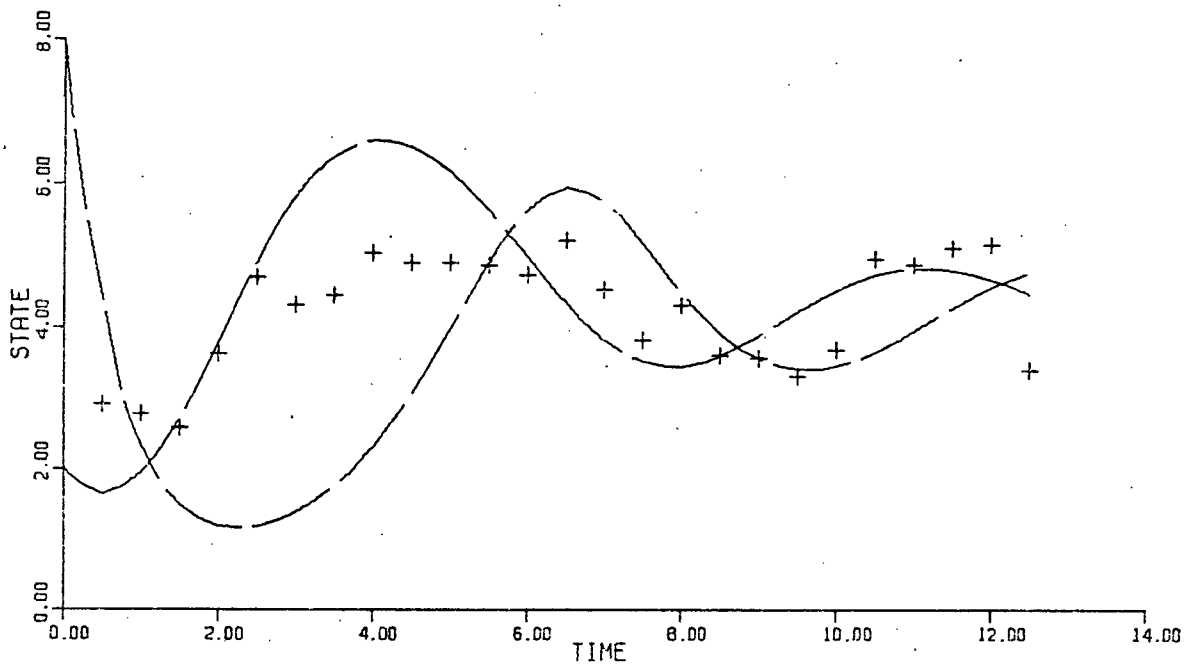

Figure 4.5.3
DFIT results

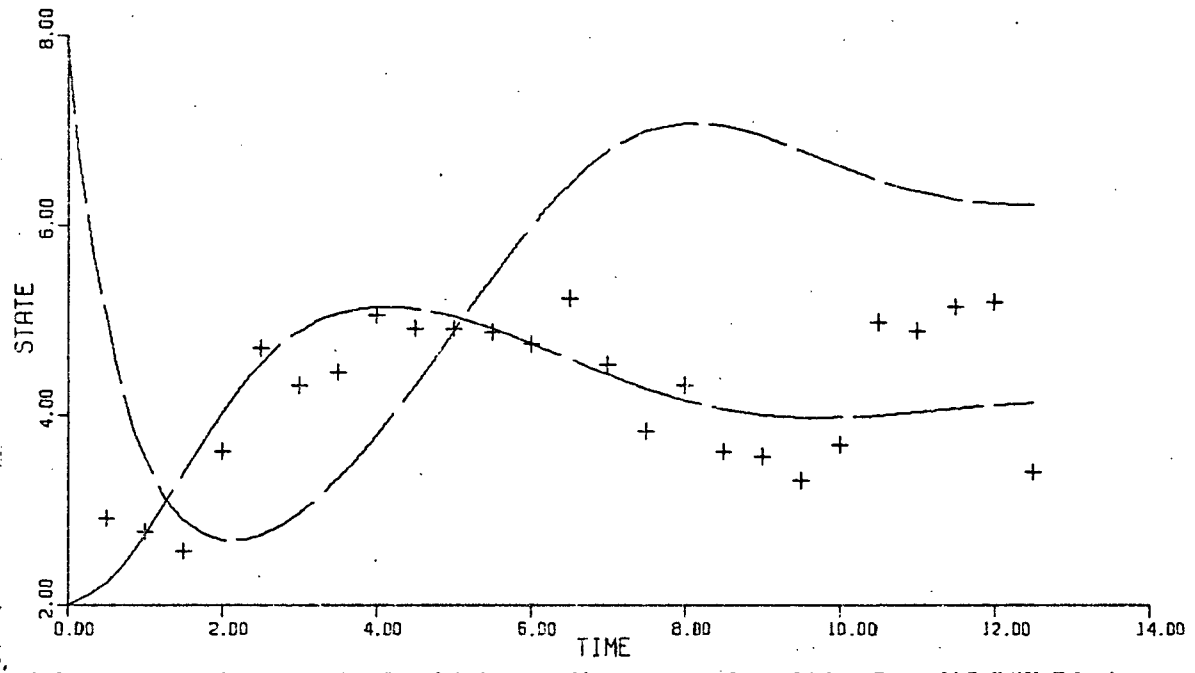

Figure 4.5.4
IFIT results (using subsystem integrations)

Figure 4.5.5.

From each of the above parameter vectors, the FIT method found the same minimum that was found when the FIT method alone was used. Since results with the iterated methods appear fairly

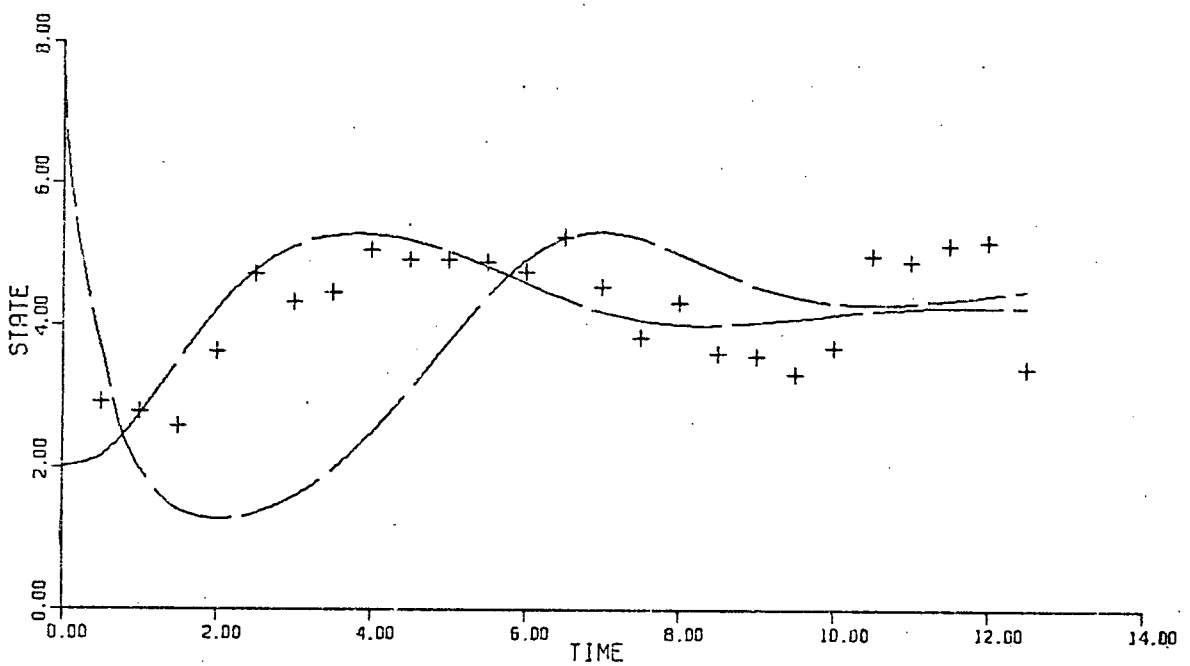

Figure 4.5.5
IFIT results (using (3.4.1))

good graphically, and the parameters generated by the iterated methods are positive, it appears worthwhile to try a square root scaling on p_5 starting from the results of an iterated method. Starting from the results of the iterated integral fitting method (using (3.4.1)) and using this scaling, we found an optimum at (unscaled)

$$(.6859,\ .03515,\ 6.092,\ 1.481,\ .2708E\text{-}5,\ .1213)^T.$$

The sum of the squares of the residuals at the above point in parameter space was approximately 5.8. Integration results at

the above parameters are shown in Figure 4.5.6.

If we return to the direct FIT approach starting with the

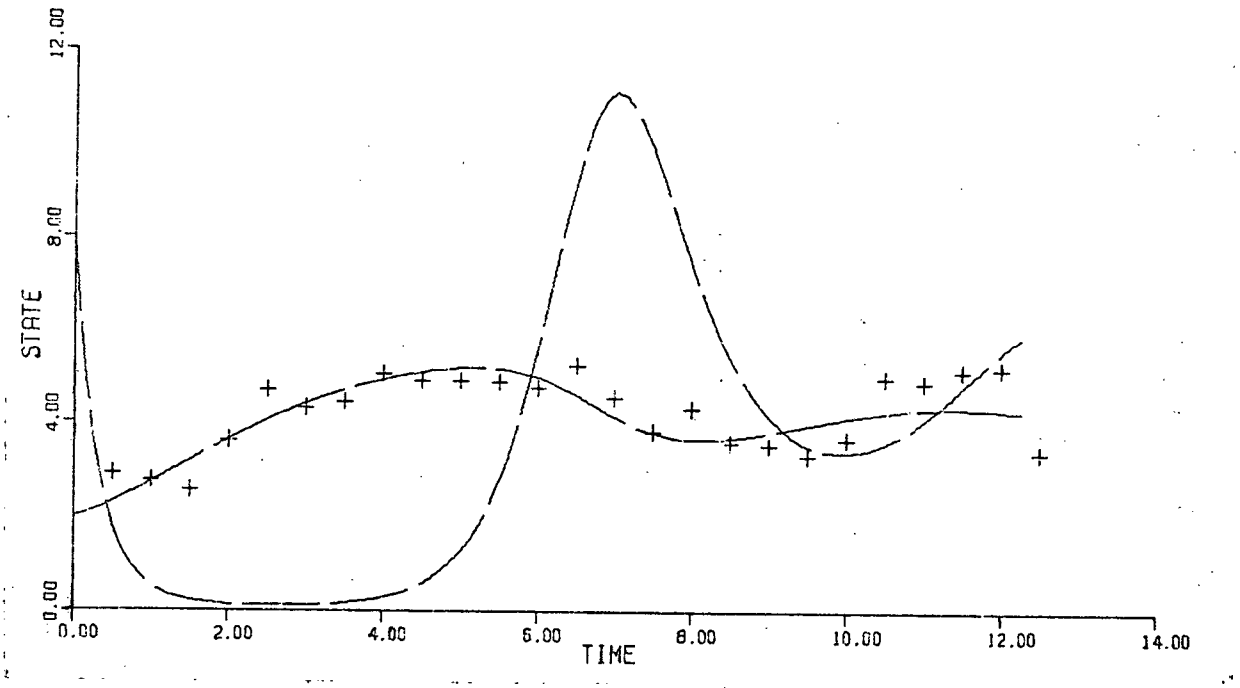

Figure 4.5.6
FIT results (p_5 scaled) starting from IFIT results

parameters corresponding to Figure 4.2.2 and with p_5 scaled with a square root transformation, problems occur. The parameter p_2 becomes negative. The scaling of both p_2 and p_5 did not prove very profitable either. With this scaling and starting at the parameters corresponding to Figure 4.2.2, we were drawn to a point in parameter space where all the parameters were positive, but where the integration results on y_2 rapidly went to zero, and the results on y_1 went to an equilibrium. Thus the iterated approach was extremely valuable for getting a solution to this problem. The use of guessed observations may be thought of as a means of guiding us to a preferred solution.

CHAPTER 5

SEQUENTIAL TECHNIQUES

5.1 INTRODUCTION

The direct approach, using the sensitivity equations, for fitting parameters in dynamic models involves the choice of an initial approximation to the optimal parameter vector, integrating at this point in parameter space, and then, with the aid of information from the solution to the sensitivity equations, finding a more optimal set of parameters. Often the initial integration deviates greatly from the observations, and it may even blow up. In these cases, the first few data points contain valuable information that can be used to improve the parameter estimates. It is intuitively appealing to use this information to improve some parameter values before we commit ourselves to a full integration over the whole time interval under consideration. To carry this idea a little further, an algorithm where we sequentially update parameter estimates each time taking into account a few more data points seems worthy of consideration. In a sense, such an algorithm is using the observations to guide us along the correct path in state space.

Sequential reestimation has received a great deal of attention. Much of this attention has been from a statistical point of view. For a concise introduction, we refer the reader to Young[76]. Problems involving the analysis of a large quantity of continuously arriving data and requiring a "real

time" solution have been one of the main motivations for the development of sequential estimation techniques. A typical example of such a situation occurs in the estimation of missile trajectories from, say, radar data. Frequently, sequential techniques are used to estimate the state of a dynamic system; however, they can also be used to estimate parameters in a dynamic system. For a good discussion of sequential estimation techniques applied to dynamic systems, we refer the reader to Gelb[28].

As mentioned above, our motivation for considering sequential techniques does not come from the need to rapidly process a large quantity of data, but instead from the need to overcome poor initial parameter estimates. Our goal is to use the observations in a manner that addresses itself to the basically sequential nature of an initial value problem.

5.2 A SEQUENTIAL ALGORITHM

There are many ways to approach the development of a sequential algorithm. At one extreme there is the approach of solving a sequence of parameter fitting problems, each using progressively more data points. Since we are dealing with a dynamic model, such an approach would be expensive. At the other extreme, we have the stochastic approximation techniques where parameters are updated by less refined but very fast reestimation algorithms ([6,p.251], [3], [33]). The method explored in this chapter falls between these two extremes.

We assume the data points are processed in batches ending

at the observation times

$$t_{k_0}, \ldots, t_{k_q}. \tag{5.2.1}$$

Batch s is the set of observations taken at the observation times

$$t_{k_{s-1}+1}, t_{k_{s-1}+2}, \ldots, t_{k_s} \tag{5.2.3}$$

where $t_{k_{-1}+1}$ is the first observation time. Define $f_{(0)}(p)$ according to (1.1.2) for the observation times

$$t_1, \ldots, t_{k_0} \tag{5.2.3}$$

and let

$$F_{(0)}(p) = f_{(0)}^{T}(p) f_{(0)}(p). \tag{5.2.4}$$

To start the sequential process we estimate p as well as possible to minimize (5.2.4). Note that k must be large enough that this least squares problem makes sense. That is, we do not want fewer data points than parameters. We expect this problem to often be singular since it is unreasonable to expect that an estimate of all the parameters can be obtained with just a few data points.

Denote by $p_{(0)}$ the optimal parameter vector obtained by minimizing (5.2.4), and denote the Jacobian matrix for $f_{(0)}$, defined by (1.3.4), by $J_{(0)}(p)$. Next we include the batch of data points at the observation times

$$t_{k_0+1}, \ldots, t_{k_1}.$$

Define $f_{(1)}(p)$ according to (1.1.2) for the above observation

times. There is some ambiguity to be resolved concerning $f_{(1)}(p)$. The elements in $f_{(1)}(p)$ depend on the solution $y(t)$ to (1.1.1) at the above observation times. We would, however, like to integrate from time t_{k_0} and not from time t_0 in order to define $f_{(1)}(p)$. Thus initial conditions are required at time t_{k_0} for this integration. Consistent with the linearization employed below, we define the initial condition on y_i at time t_{k_0} by

$$y_{0,i}(t_{k_0}) = y_i(p_{(0)}) + (p-p_{(0)})^T \left(\frac{\partial y_i}{\partial p_1}, \ldots, \frac{\partial y_i}{\partial p_m}\right) \qquad (5.2.5)$$

for $i=1,\ldots,n$. Thus the initial conditions for the integration between times t_{k_0} and t_{k_1} are functions of the parameters, and this must be considered when the sensitivity equations are integrated.

Define the Jacobian matrix $J_{(1)}(p)$ corresponding to $f_{(1)}(p)$ according to (1.3.4). This matrix is found by integrating the sensitivity equations from time t_{k_0} to time t_{k_1}. Let

$$(\Delta p)_{(0)} = p - p_{(0)} \qquad (5.2.6)$$

and define $f^{(1)}(p)$ by

$$f^{(1)}(p) = \left(\left(f_{(0)}(p_{(0)}) + J_{(0)}(p_{(0)}) \right) (\Delta p)_{(0)} \right)^T, f_{(1)}^T(p) \Big)^T. \qquad (5.2.7)$$

Our new parameter estimate, which we denote by $p_{(1)}$, minimizes

$$f^{(1)T}(p) f^{(1)}(p). \qquad (5.2.8)$$

The Jacobian matrices for $f^{(1)}(p)$ are

$$J^{(1)}(p) = \begin{bmatrix} J_{(0)}(p_{(0)}) \\ J_{(1)}(p) \end{bmatrix}. \qquad (5.2.9)$$

We have just approximated $f_{(0)}(p)$ by the first term in its Taylor expansion at the point $p_{(0)}$. The success of a sequential technique is closely related to the size of the region in which this approximation is accurate. We continue in the above manner reestimating p for each new batch of data points. Thus when batch s is under consideration, we are finding the estimate $p_{(s)}$ using

$$f^{(s)}(p) = \begin{bmatrix} f_{(0)}(p_{(0)}) & +J_{(0)}(p_{(0)})(\Delta p)_{(0)} \\ & \vdots \\ f_{(s-1)}(p_{(s-1)}) & +J_{(s-1)}(p_{(s-1)})(\Delta p)_{(s-1)} \\ & f_{(s)}(p) \end{bmatrix} \qquad (5.2.10)$$

and

$$J^{(s)}(p) = \begin{bmatrix} J_{(0)}(p_{(0)}) \\ \vdots \\ J_{(s-1)}(p_{(s-1)}) \\ J_{(s)}(p) \end{bmatrix} \qquad (5.2.11)$$

The new parameter estimate, $p_{(s)}$, minimizes

$$f^{(s)T}(p) f^{(s)}(p). \qquad (5.2.12)$$

Note that only $J_{(s)}(p)$ is changing in $J^{(s)}(p)$ during the determination of $p_{(s)}$. This fact can be used to advantage when implementing a sequential algorithm. For example, if we are using the singular value decomposition of $J^{(s)}$ then information

obtained when producing the decomposition of $J^{(\alpha-1)}(p_{(\alpha-1)})$ can be used to efficiently obtain the decomposition of $J^{(\alpha)}(p)$.

5.3 EXPERIMENTAL RESULTS

We experiment with Problems 4.2.1, 4.2.2, 4.5.1, and 4.5.2. The last two of these test problems have noise in the observations. The starting parameters for our experiments with Problems 4.2.1 and 4.5.1 are those corresponding to Figure 4.2.2, and the starting parameters for our experiments with Problems 4.2.2 and 4.5.2 are those corresponding to Figure 4.2.3. We comment that with these parameters a direct approach (FIT) was unsuccessful on all four problems. Our experiments with Problems 4.2.1 and 4.2.2 start with an initial batch of observations corresponding to the first five observation times. We then proceed through the remaining observation times in increments of five observation times. For Problems 4.5.1 and 4.5.2 the use of five observation times proved to be insufficient to get started (the initial parameter estimates were too inaccurate). For these two problems we start with ten observation times and then proceed in increments of five observation times. An interactive approach could be valuable here.

For Problem 4.2.1, the parameter estimates after the first and last batches were respectively

$$(.9743,\ .1281,\ 3.873,\ 2.101,\ .3153,\ .1473)^T$$

and

$$(1.014,\ .1178,\ 3.548,\ 1.406,\ .1632,\ .1539)^T.$$

For Problem 4.2.2, the parameter estimates after the first and last batches were respectively

$$(.5550,\ .09277,\ 3.256,\ .6304,\ .1390,\ .01231)^T$$

and

$$(.4687,\ .06548,\ 3.3837,\ .4226,\ .07886,\ .008364)^T.$$

For Problem 4.5.1, the parameter estimates after the first and last batches were respectively

$$(.6689,\ .02493,\ 4.085,\ 1.309,\ .06232,\ .1154)^T$$

and

$$(.7690,\ .04540,\ 3.238,\ .8006,\ .01421,\ .1286)^T.$$

For Problem 4.5.2, the parameter estimates after the first and last batches were respectively

$$(7.227,\ 1.303,\ 3.337,\ 1.333,\ .3334,\ .2266)^T$$

and

$$(.6987,\ .2829,\ 6.917,\ 2.703,\ .3389,\ .01377)^T.$$

A direct approach (FIT method) converged to the desired solution for Problems 4.2.1, 4.2.2, and 4.5.1 starting at the first and last estimates given above. A square root transformation of p_5 was required in the case of Problem 4.5.1 to prevent p_5 from becoming negative. The direct approach on

Problem 4.5.2, starting from the parameters obtained using only the first batch of observations, found a local minimum at

$$(3.624,\ .8877,\ 4.064,\ 1.277,\ .2642,\ .1002)^T$$

where the sum to the squares of the residuals was approximately 840. However, convergence of the direct method to the desired solution was obtained when we started from the final result of the sequential pass on Problem 4.5.2. Thus the sequential updating was essential in this case. We summarize the above results in Table 5.3.1. We conclude that it can be advantageous

Problem	FIT	FIT (first batch)	FIT (last batch)
4.4.1	L	C	C
4.4.2	L	C	C
4.5.1	L	C	C
4.5.2	L	L	C

(L-local minimum, C-desired minimum)

Table 5.3.1
Results with sequential approach

to consider the observations sequentially to obtain an improved approximation to the optimal parameters before we commit ourselves to a full optimization attempt over the whole time interval. Indeed, in three of the above four cases a sufficient improvement to allow the FIT method to converge to the desired solution was obtained using only the first few observation points.

A great deal of work remains to be done to fully evaluate

the use of sequential methods for improving starting parameters; however, we have attained our limited goal of demonstrating the feasibility of using a sequential strategy.

CHAPTER 6

REAL WORLD PROBLEMS

6.1 INTRODUCTION

All the parameter fitting problems in this chapter involve physical observations as opposed to observations generated by a simulation. Such problems are a good deal more difficult than those using generated data. This difficulty occurs partly because the dynamic model under consideration often cannot, for any parameter values, give an adequate description of the process being modelled. Also, experience indicates that the least squares surface for parameter fitting in dynamic models is often plagued with numerous local minima. Starting with one of the more global methods of Chapter 3, it is fairly easy to find one of these local minima. (See for example the experiments with Bazykin's model in Chapter 4.) The problem for the model builder is to decide if there is a more optimal set of parameters somewhere else in parameter space or if the qualitative difference between the model and the data at the current minimum is just the result of a poor or incomplete model. This is usually a difficult decision to make. Situations such as this arise frequently in nonlinear problems and a standard strategy is to start optimizing from different points in parameter space. If the same minimum is determined starting from several different points, then we can be more confident that the minimum is a global minimum. An interactive

approach is ideal for experimentally checking on a minimum; however, a good understanding of the model and the physical meaning of the parameters is also very valuable. This latter understanding can be augmented through an interactive analysis of the effects of various parameter changes.

An interactive parameter fitting package can be very useful in the design of models as well as in the fitting of individual models. For example, the model builder may start with a simple but incomplete model and find in the course of fitting it that it cannot account for some of the qualitative behavior of the observations. This would be indicated for example if the best fitting parameters produced a model which smoothed out a crucial peak in the data. With luck, in the course of fitting this model, some of its deficiencies may be determined and some insight into improvements may be gained. For this type of application, it would be helpful to allow the dynamic redefinition of the model.

We stress that the above process is very tenuous and puts a large emphasis on the intuition and judgement of the model builder. It is in such situations, however, where an interactive approach can be extremely advantageous.

6.2 A DYNAMIC MODEL FOR AGGRESSIVE AND DOCILE MICE

The model considered in this section proposes a population consisting of two interacting types of mice to account for observations on the total mouse population. For an introduction to this problem, we refer the reader to Myers and Krebs[48] and

Krebs et al[37]. Let u be the population density of docile mice, and v the population density of aggressive mice. Let p_3 be the basic birth rate, p_4 the basic death rate, and let p_5 describe the sensitivity of docile mice to crowding. Let α be the proportion of aggressive mice in the offspring. Further assume the aggressive mice reproduce poorly (almost sterile) and that this can be described by multiplying the basic birth rate by u/(u+v). Under the effects of crowding, the docile mice are assumed to either emigrate or die, and this is described by the term $-p_5 u(u+v)$ in the equation for u'. The dynamic model is thus

$$\begin{aligned} u' &= p_3(1-\alpha)u^2/(u+v) - p_4 u - p_5 u(u+v) \\ v' &= p_3 \alpha u^2/(u+v) - p_4 v. \end{aligned} \tag{6.2.1}$$

Questionable assumptions such as those given above are typical of dynamic models in ecology, and with such assumptions we should not be too disappointed if the model cannot describe the observations very well. Mouse population measurements are available only on the total population u+v. A more general formulation of the parameter fitting problem in dynamic models (see Bard[6,p.221]) could handle this directly; however, the problem can be transformed to conform to our formulation. In so doing, we arrive at a problem with observations on only one state variable and some of the techniques of Chapter 3 can be used to produce starting approximations to the parameters. If we define $y_1 = u$, $y_2 = u+v$, and $p_6 = p_3(1-\alpha)$ then the above dynamic

model may be written as

$$y_1' = p_6\, y_1^2/y_2 - p_4\, y_1 - p_5\, y_1\, y_2$$
$$y_2' = p_3\, y_1^2/y_2 - p_4\, y_2 - p_5\, y_1\, y_2 . \qquad (6.2.2)$$

For our initial conditions we take

$$y_1(0) = p_1/(1+\exp(-p_2))$$
$$y_2(0) = p_1, \qquad (6.2.3)$$

where we have ensured that $y_1(0) < y_2(0)$. Observe that all parameters in this model, with the possible exception of p_2, should be positive.

The 44 observations on y_2 are shown graphically in Figure 6.2.1. We comment that it is with reference to the scale of

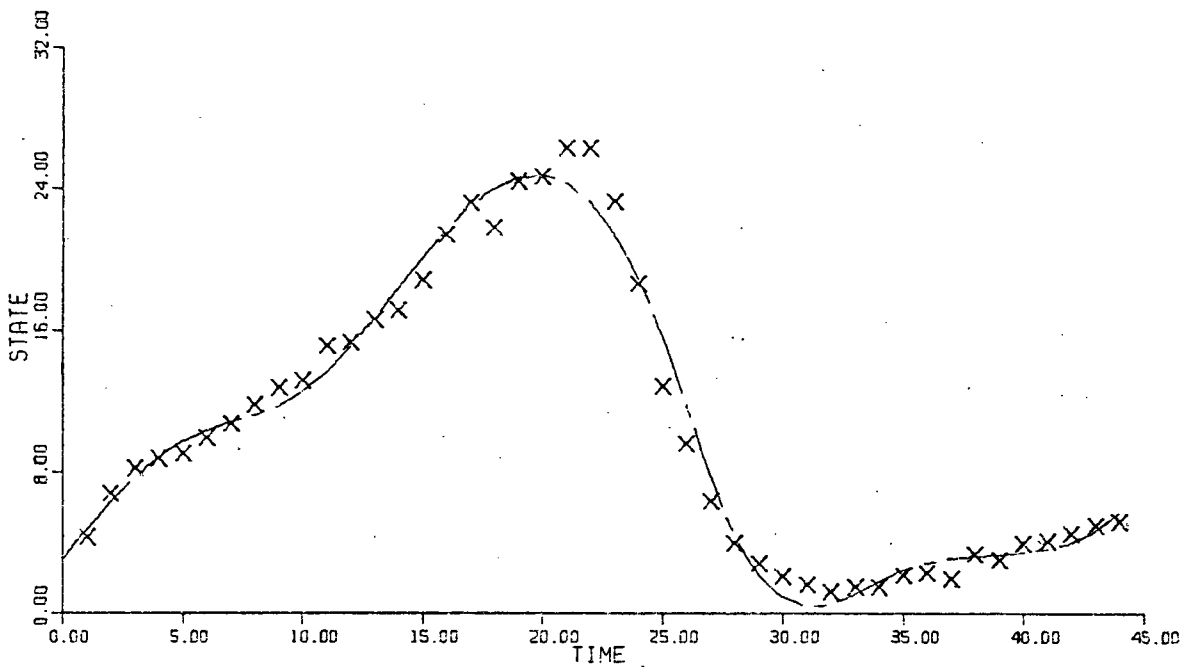

Figure 6.2.1
Observations and spline approximation

Figure 6.2.1 that the parameter estimates should be interpreted.

(The basic unit of time is two weeks and the basic population density unit is ten animals per acre.) A cubic spline approximation to the observations using joints at t=5,10,20,25,28,35 is also shown in Figure 6.2.1.

This problem is relatively difficult. In the following discussion, we outline a particular sequence of experiments which leads to a model which fits the observations quite well numerically. Such experiments, of necessity, involve a good deal of trial and error, and thus an interactive approach is ideal. In what follows, we try to give an indication of this interactive process.

Since observations are available on only one state variable, the iterative methods of Section 3.4 may be useful for getting initial approximations to the parameters. Furthermore, we observe that with the exception of p_2, all the parameters occur linearly in the residual functions of the IFIT and DFIT algorithms when guessed observations are used on y_1. However, to apply one of the iterative algorithms of Section 3.4, a starting guess at the behavior of the unobserved state variable is required. It must be less than y_2, and we expect it to mimic in some sense the behavior of y_2. A reasonable guess is curve (0) in Figure 6.2.2. Note that one of the most prominent features of y_1 with respect to y_2 is the position of the proposed maximum of y_1. As indicated by the experiments in Chapter 4, the position of such a maximum can be critical. An interactive approach can be very valuable here. To get starting

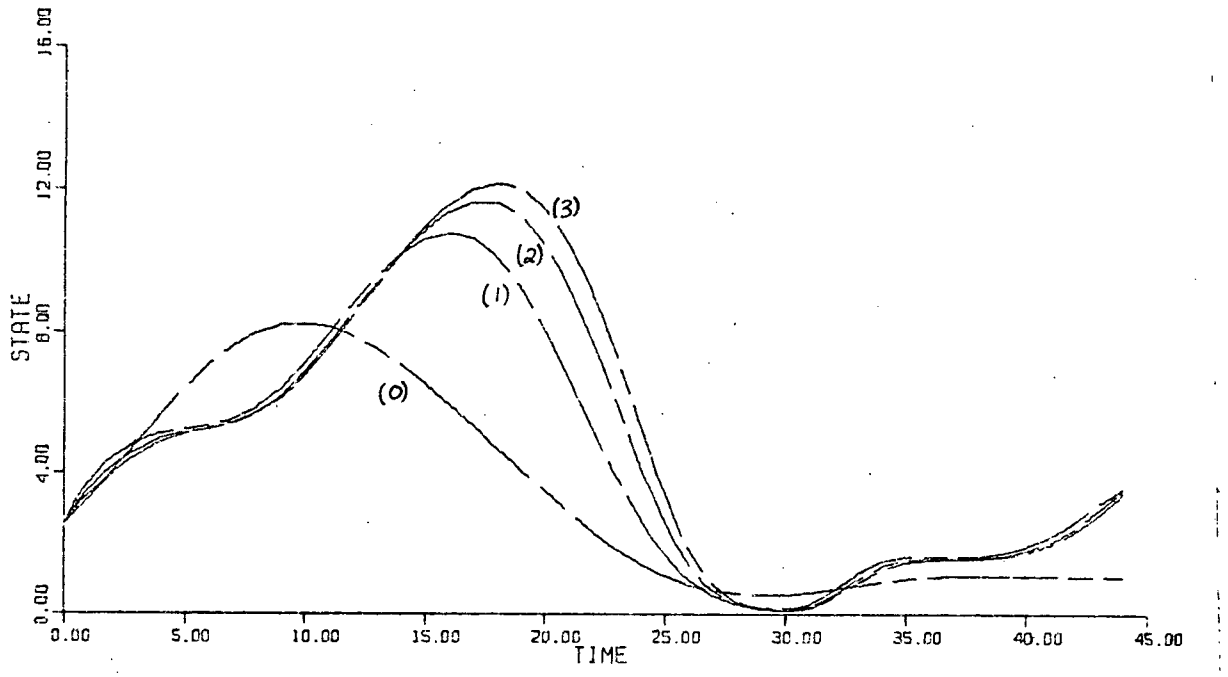

Figure 6.2.2
Iterations on guessed observations

approximations to the parameters, we used the iterated integral fitting method (using (3.4.1)) taking advantage of the linearity of the parameters in (6.2.2). We froze p_1 and p_2 at 2.9 and 2 respectively and iterated on the remaining four parameters. Figure 6.2.2 shows successive improvements in the guessed observations on y_1. In Figure 6.2.3, the solution $y_2(t)$ obtained from integrating (6.2.2) at the successive approximations to p are shown. The observations on y_2 are also shown in Figure 6.2.3. The trapezoidal discretization was used throughout this section. Curve (i) in Figure 6.2.3 corresponds to curve (i) in Figure 6.2.2. The parameters corresponding to curve (2) in Figure 6.2.3 were

$$(2.9,\ 2.0,\ 1.234,\ .2445,\ .003339,\ .6114)^T.$$

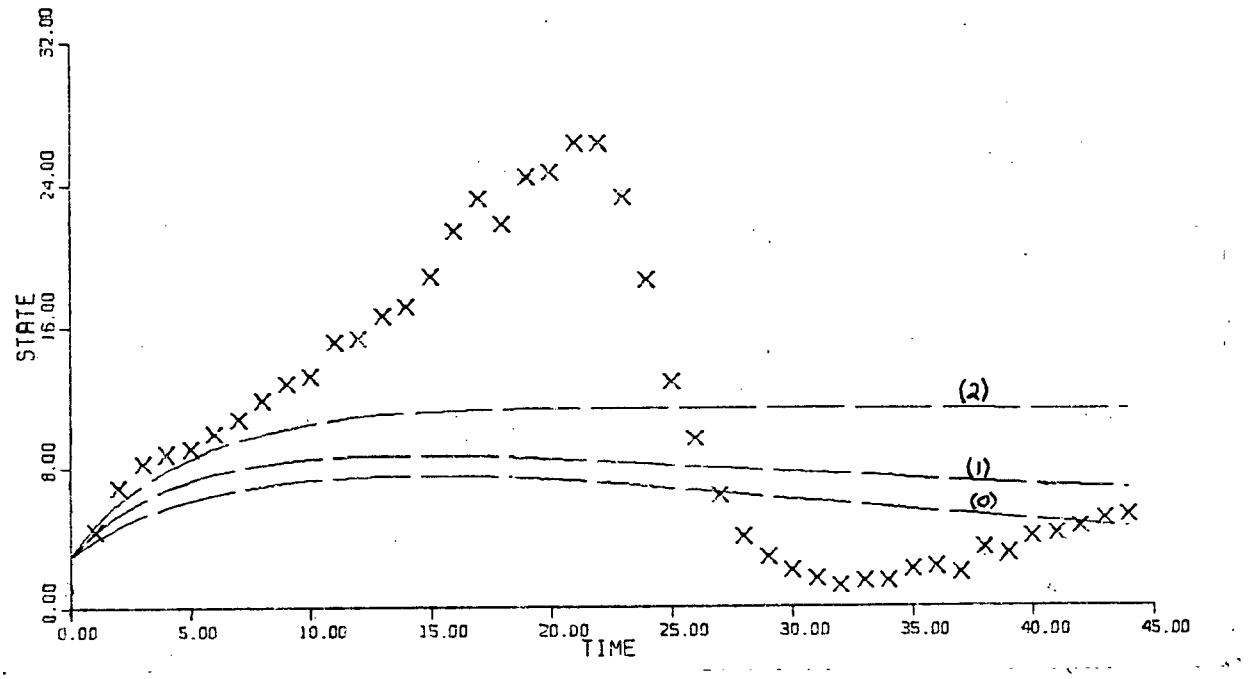

Figure 6.2.3
Integrations at successive parameter approximations

Graphically, there appears to be substantial room for improvement in the parameter values; however, the iterated integral fitting method has efficiently provided starting approximations to the last four parameters.

Now that we have starting approximations to the parameters at which the initial value problem can be integrated to produce reasonable results, it is worthwhile to try to refine the parameters values using a direct approach employing the sensitivity equations. The Levenberg-Marquardt method gave the parameters

$$(2.828,\ 1.963,\ 1.142,\ .06483,\ .008582,\ .5279)^T$$

Integeration results corresponding to this parameter vector are shown in Figure 6.2.4. The sum of the squares of the residuals

at the above point in parameter space was approximately 1350. Clearly, an improvement has been made. However, the peak in the

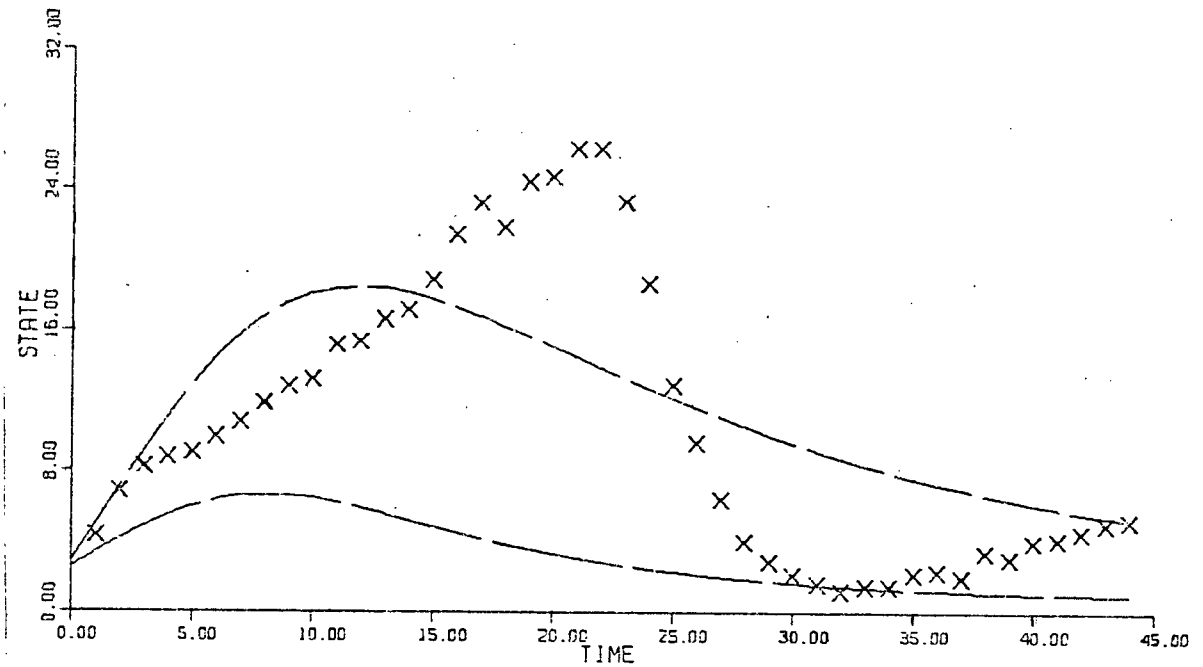

Figure 6.2.4
Optimum starting from iterated IFIT results

observations is not being approximated well. Considering the way the model was derived, there is no reason to expect that the peak in the observations can be well approximated. If we want to try to fit the peak in the data, we can experiment at other points in parameter space and try other algorithms such as those presented in Section 3.5. An interactive package can be very valuable for such probing and experimenting. Extensive experiments and the use of special methods such as the quasi-multiple shooting technique of Chapter 3 did not produce parameter values which described the peak well. A logarithmic transformation of p_5 which works well on the following two

models did not work here either. It is impossible to say no such parameters exist; however, in view of the better fit obtained later in this section when more flexibility was put into the model, we are led to the conclusion that the model cannot adequately describe the observations. Such a conclusion is a matter of judgement and all a good numerical package can do is to provide information to make that judgement more informed.

It is fairly easy using an interactive approach to produce parameters that give integration results which peak in the same vicinity as do the observations. For example, the following parameters were obtained interactively:

$$(1.4,\ 1.24,\ .8272718,\ .07,\ .006,\ .5037043)^T.$$

The interactive procedure employed involved freezing parameters, optimizing on subspaces, and the experimental resetting of parameters. Integration results at these parameters are shown in Figure 6.2.5. The difficulty seems to be that the sudden drop in the observations cannot be imitated. Thus when we tried a full least squares optimization starting at the above parameters, we obtained a possibly local minimum at

$$(1.292,\ 1.171,\ .9662,\ .06040,\ .01371,\ .6242)^T.$$

Integration results at the above parameters are shown in Figure 6.2.6. The sum of the squares of the residuals was approximately 1150 at the above parameters. Observe that the results are slightly different than those of Figure 6.2.4, but the peak in the data is still not being imitated. Indeed, even

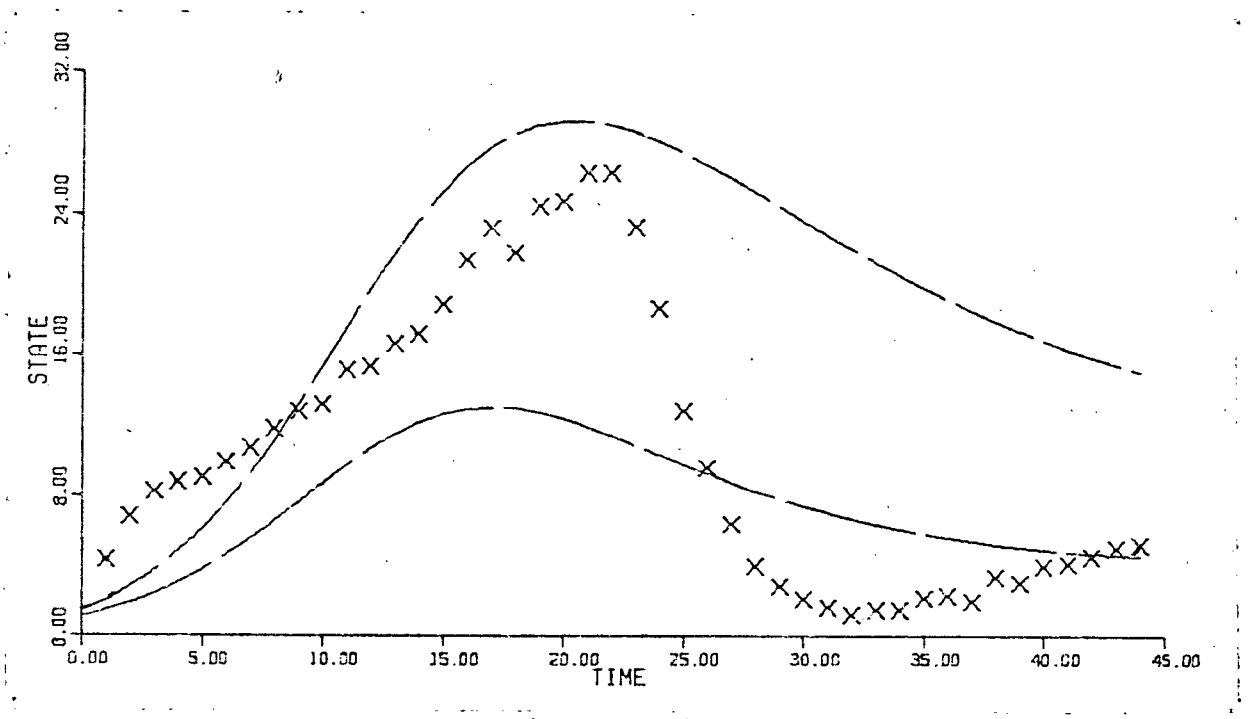

Figure 6.2.5
Integration at parameters found interactively

the interactive optimization on subspaces continually drew the integration results down to balance off the peak and the trough in the observations. Of course we could weight some of the data points in the least squares problem to emphasize the peak, but this is artificial and experiments indicate that while a good approximation up to the peak can easily be obtained, the rapid drop in the observations cannot be imitated.

Next we experimented with a model with more flexibility in the form of another parameter. Specifically, we let the death rates for the docile and aggressive mice be different. Denote by p_4 the death rate for docile mice and let p_7 denote the death rate for aggressive mice. Our dynamic model now is

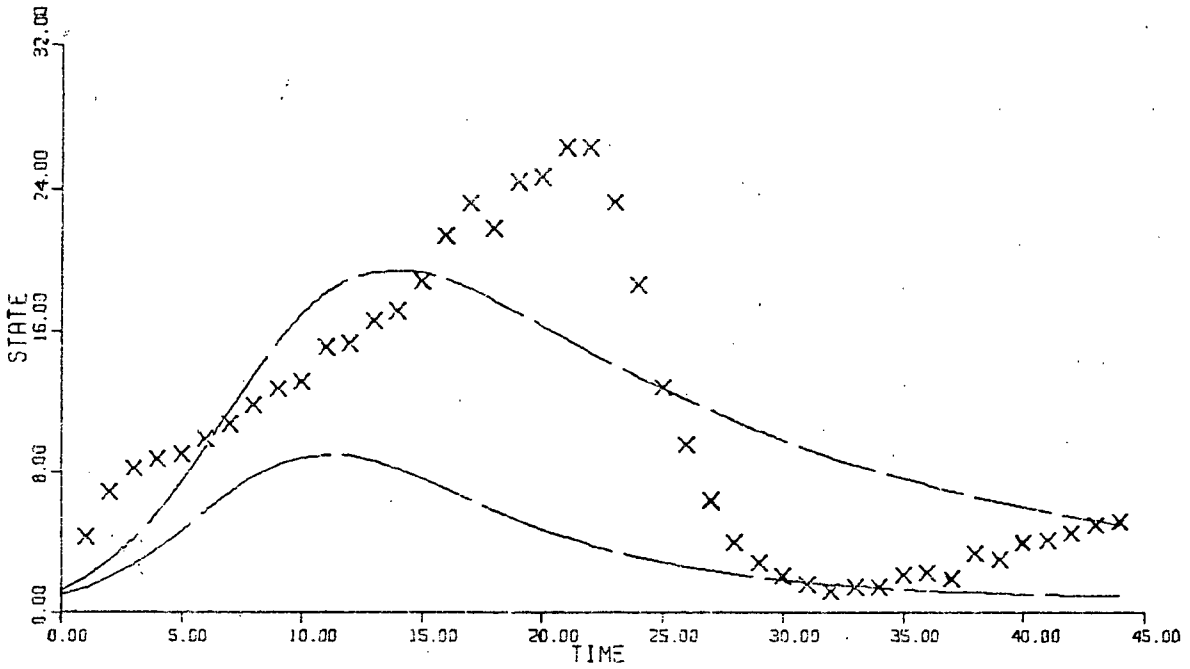

Figure 6.2.6
Optimum starting from parameters found interactively

$$y_1' = p_6 y_1^2/y_2 - p_4 y_1 - p_5 y_1 y_2$$
$$y_2' = p_3 y_1^2/y_2 - p_4 y_1 - p_7 (y_2 - y_1) - p_5 y_1 y_2 \qquad (6.2.4)$$

Using this model and starting at the parameters obtained interactively with $p_7 = p_4$, a local minimum to the full unweighted least squares problem was found at

$$(1.402,\ 1.241,\ .8365,\ .1875,\ .01174,\ .6748,\ .005763)^T.$$

The sum of the squares of the residuals at the above parameters was appproximately 960. Although p_4 and p_7 differ, integration results at the above parameters do not differ substantially from those shown in Figure 6.2.6.

This is in reality a local minimum. Later in this section, we return to this model and achieve a much smaller but still perhaps local minimum. The specific strategy used to obtain this new minimum grew out of experiments with the following model.

Experimentally, the following model fits the observations well. It is the same as the previous model except for a modification in the growth term for the aggressive mice. As in Equation (6.2.1), let u be the population density of docile mice, and let v be the population density of aggressive mice. Our proposed dynamic model is

$$\begin{aligned} u' &= p_6 u^2/(u+v) - p_4 u - p_5 u(u+v) \\ v' &= p_3 uv/(u+v) - p_7 v \end{aligned} \tag{6.2.5}$$

which with $y_1 = u$, and $y_2 = u+v$ becomes

$$\begin{aligned} y_1' &= p_6 y_1^2/y_2 - p_4 y_1 - p_5 y_1 y_2 \\ y_2' &= (p_6 - p_3) y_1^2/y_2 + (p_3 + p_7 - p_4) y_1 - p_7 y_2 - p_5 y_1 y_2 . \end{aligned} \tag{6.2.6}$$

The initial conditions are given in (6.2.2). Our starting approximation to the optimal parameters was

$$(2.9,\ 2.5,\ .8,\ .07,\ .006,\ 1,\ .5)^T,$$

and the optimal parameters obtained were

$$(2.958,\ 2.630,\ .6503,\ .8892,\ -.007497,\ 1.108,\ .3108)^T.$$

The sum of the squares of the residuals at the above parameters was approximately 250. Graphically, the integration results at the above parameters are almost the same as those in Figure

6.2.7 except y_2 gets close to zero around t=36. Unfortunately, p_5 is negative. We note that p_5 is much smaller than the other six parameters and this suggests we try the logarithmic transformation $\hat{p}_5 = \ln(p_5)$ and estimate $\hat{p}_5$ instead of p_5. It is important to be selective when choosing a logarithmic transformation. For example, the rescaling of p_3 to p_7 by logarithmic transformations did not lead to the success described below when p_5 alone was rescaled. Thus again we have a place for interactive experimentation. As mentioned at the end of Chapter 1, the use of logarithmic transformations also affects the conditioning of the problem. For example the condition number of the Jacobian matrix in the least squares problem at the above parameters is 1.1E5. If p_5 is scaled as suggested then the condition number becomes 1.3E4 and if $p_3, \ldots, p_7$ are all scaled by logarithmic transformations, the condition number becomes 2.6E4. With the transformed p_5, we started the Levenberg-Marquardt procedure at the last set of parameters preceding the first occurance of a negative value for p_5 in the previous run, namely

$$(2.88,\ 2.57,\ .728,\ .437,\ -5.17,\ .687,\ .465)^T$$

where we have renamed $\hat{p}_5$ to be p_5. The optimal parameter values found were

$$(5.335,\ 3.011,\ .08038,\ 3.966,\ -6.895,\ 4.266,\ -.01545)^T$$

where now p_7 has taken on a negative value. (The transformed values of scaled parameters are given in this discussion.) The

sum of the squares of the residuals was approximately 56 at the above parameters. Graphically, the integration results at the above parameters are very similar to those in Figure 6.2.7 except y_2 curves up slightly for t>30. The best parameters before p_7 went negative were

$$(4.362, 2.989, .1307, 2.644, -6.232, 2.909, .01303)^T.$$

It appears that a logarithmic transformation of p_7 may be profitable. With this transformation and starting at the above point in parameter space, we found the minimum

$$(5.403, 2.691, .1042, 3.507, -6.807, 3.848, -4.688)^T.$$

The sum of the squares of the residuals at these parameters was approximately 62. Integration results at the above parameter values are shown in Figure 6.2.7.

From a numerical point of view, the fit in Figure 6.2.7 appears excellent; however, from a biological point of view it has a flaw. The first state variable gets caught at zero and y_2 goes to an equilibrium. Biologically, we would like the solution to the initial value problem to oscillate in time. Numerically, it appears that more data points should be available if we want to look for oscillatory behavior.

The remarkable improvement achieved when a logarithmic transformation was used on p_5 suggests that we go back to the first two models and try this strategy. Starting with the parameters obtained interactively, this strategy did not change the results with the first model; however, starting at the

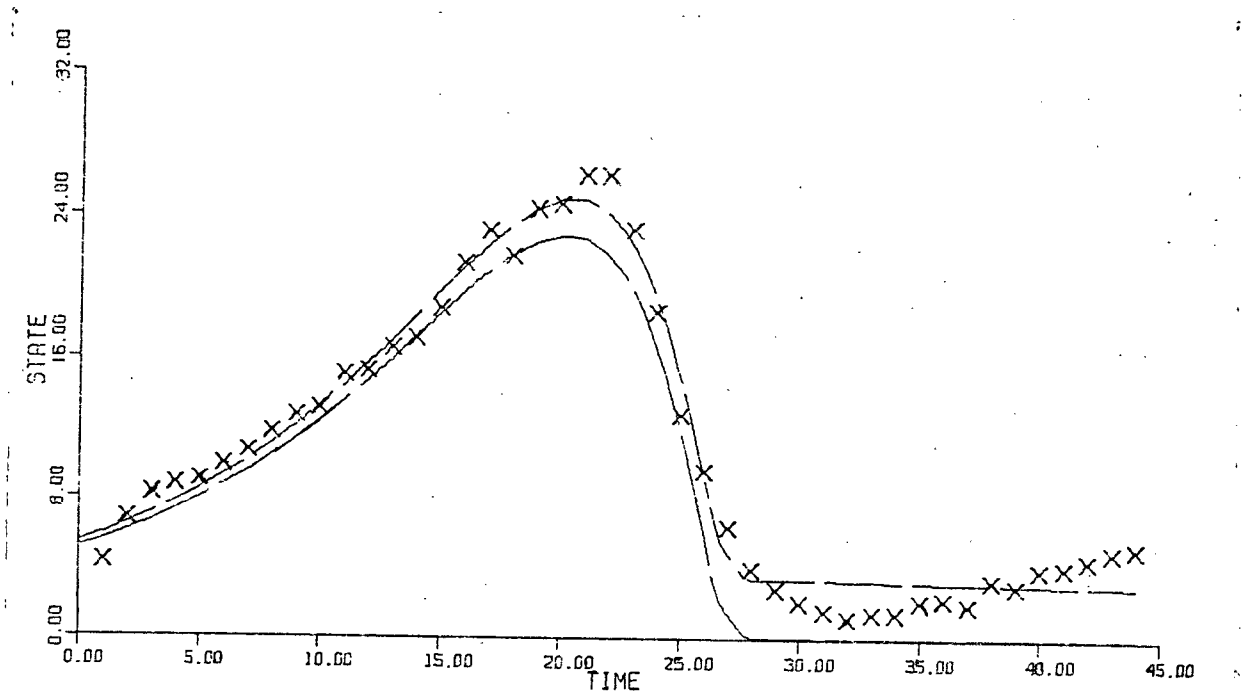

Figure 6.2.7
Optimum with logarithmic scaling on p_5 and p_7

parameters obtanied interactively and with $p_7 = p_4$, this strategy worked very well on the second model where two death rates were used. As with the third model, p_7 became negative and small and a logaritmic transformation of p_7 was also used. The final parameters obtained were

$$(5.416,\ 2.350,\ 3.832,\ 3.394,\ -7.037,\ 3.822,\ -5.062)^T,$$

and the sum of the squares of the residuals and integration results at the above parameters were essentially the same as those of Figure 6.2.7. It is interesting to note that for the parameter vector

$$(4.205,\ 2.586,\ 2.905,\ 2.562,\ -6.636,\ 2.896,\ -5.298)^T,$$

the sum of the squares of the residuals was approximately 54000.

This sort of behavior is typical of initial value problems and is one reason why so many difficulties are encountered when fitting parameters in dynamic models. Finally, we comment that our investigation has been confined to numerical parameter fitting and no attempt has been made to interpret the parameter values in a physical sense. Before any parameters can be accepted, they must of course be physically reasonable.

6.3 A MODEL INVOLVING A CHANGE IN EQUILIBRIUM

The model considered in this section represents an early attempt to describe data collected from Lake Placid, British Columbia [75]. The observations are on phytoplankton in the lake. Throughout much of the summer, their total mass remains fairly constant at a relatively low level. Then, within the space of a few days, it jumps to a much higher level. The next few observations contain a lot of noise, but it appears that the level of phytoplankton remains high until near the end of the year when it drops back to a low level. The observations on the phytoplankton are shown in Figure 6.3.1. The units for the time axis are in days and time t=0 corresponds to the beginning of May. One unit along the state variable axis in Figure 6.3.1 corresponds to 1/75 milligram of phytoplankton per litre of water. It is postulated that the sudden jump in the observations can be described by a dynamic model which loses a lower equilibrium and moves to a higher equilibrium. In particular we consider the dynamic model (proposed by Dr. C.J. Walters, Institute of Animal Resource Ecology, University of

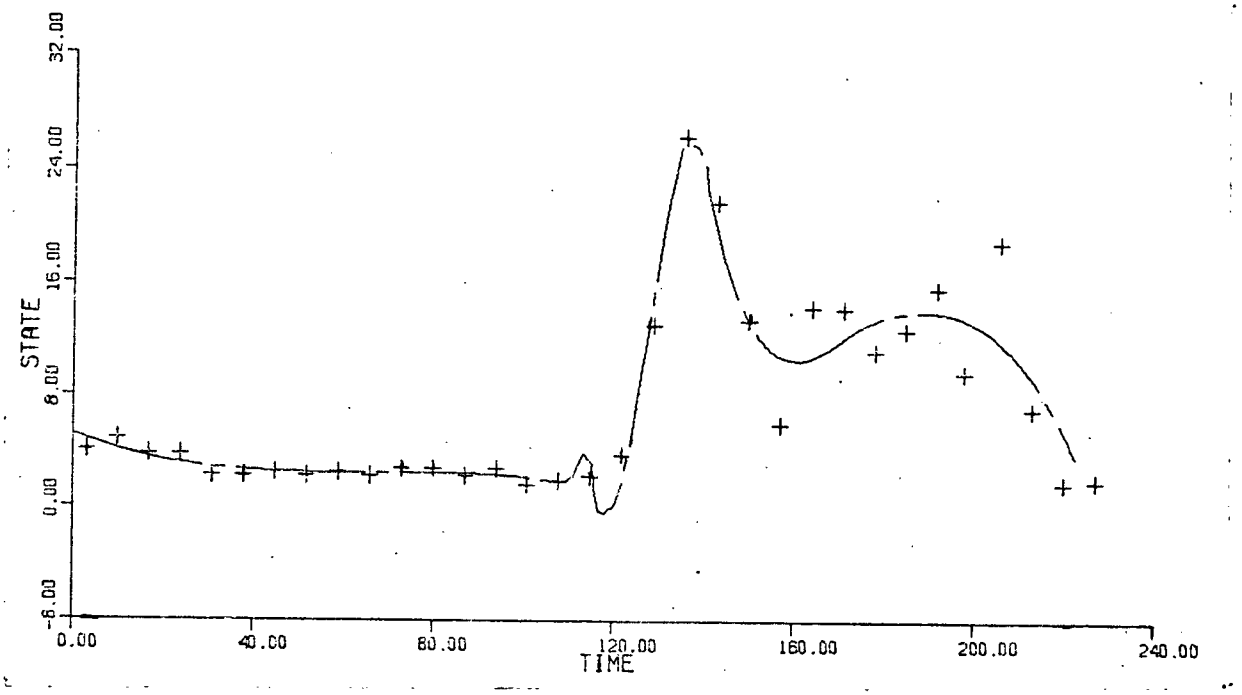

Figure 6.3.1
Phytoplankton observations and smoothing function

British Columbia)

$$y' = p_1 r(t) y(1-y/25) - p_3 z(t) y^2/(p_2^2+y^2)$$
$$y(0) = 3.2. \qquad (6.3.1)$$

In this model, y represents the density of phytoplankton, and z(t) represents the density of the predator zooplankton. The function r(t) represents the effect of sunlight on the growth rate of the phytoplankton. The term $r(t)p_1(1-y/25)$ describes the growth of the phytoplankton, and the term $-z(t)p_3 y^2/(p_2^2+y^2)$ describes the feeding effect of the zooplankton.

To determine the equilibria of the above dynamic model, we set y'=0, and solve for y. Thus for $y \neq 0$, the equilibria occur at the roots of

$$p_1 r(t)(1-y/25)(p_2^2+y^2)-p_3 z(t)y=0 \qquad (6.3.2)$$

Depending on p_1, p_2, p_3, $r(t)$, and $z(t)$, the above algebraic equation in y can have one, two, or three real roots. Thus it is theoretically possible to gain or lose an equilibrium when (6.3.1) is integrated.

For $r(t)$ we take the function

$$r(t)=\exp(-((t-110)/55)^2). \qquad (6.3.3)$$

The function $z(t)$ was obtained from physical observations. To produce a continuous approximation to $z(t)$, we used a least squares piecewise cubic Hermite approximation to the observations. The joints used in this approximation were at $t=60$, 120, 180. The data points and the continuous approximation to $z(t)$ are shown in Figure 6.3.2. The observations have been scaled so that the maximum ordinate is 1.

We take

$$p^{(0)}=(1,\ 1,\ .5)^T$$

for our initial parameter estimate. The success in Section 3.3 of the integral fitting technique for this type of problem suggests we start with this method. First we smoothed the observations with a least squares piecewise cubic Hermite polynomial with joints at $t=110$, 115, 140, 180. This smoothing function is shown along with the observations in Figure 6.3.1. With this smoothing, the IFIT method produced the parameters

$$(.1433,\ 5.509,\ .7300)^T.$$

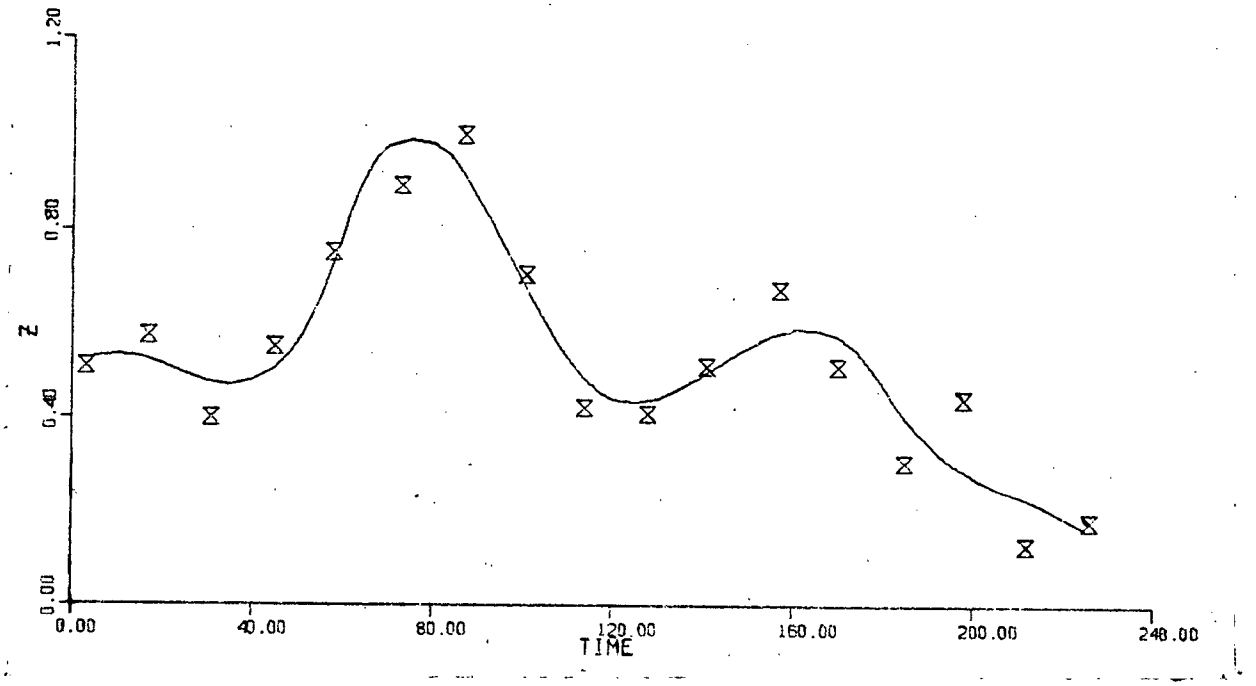

Figure 6.3.2
Zooplankton densities

Integration results at the above parameter vector are shown in Figure 6.3.3. Next we refined these parameters using the Levenberg-Marquardt technique and the sensitivity equations. This approach found the optimum

$$(.1562,\ 3.336,\ .8963)^T$$

where the sum of the squares of the residuals was approximately 556. Integration results at the above parameters are shown in Figure 6.3.4. A direct approach using the sensitivity equations and starting at $p^{(0)}$ defined above also produced this result.

In Table 6.3.1, we list some roots of (6.3.2) as a function of time at the above parameters. The results in this table indicate that at the start of the time interval under consideration there is one relatively low equilibrium. Later on

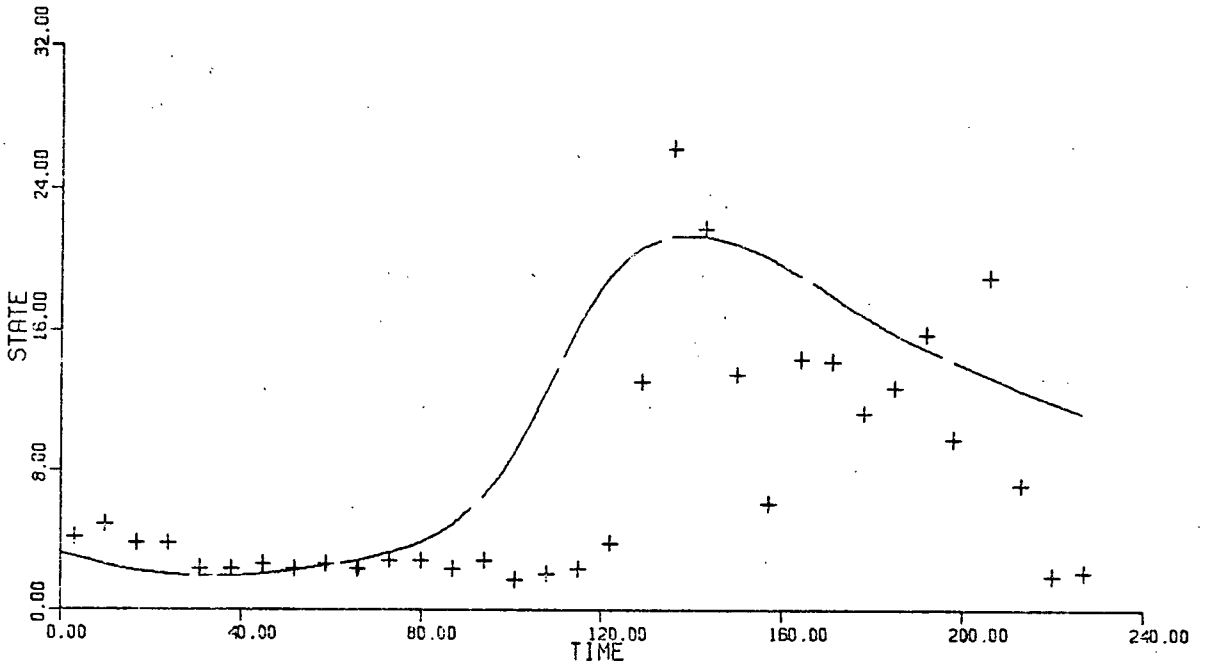

Figure 6.3.3
Integration at IFIT results

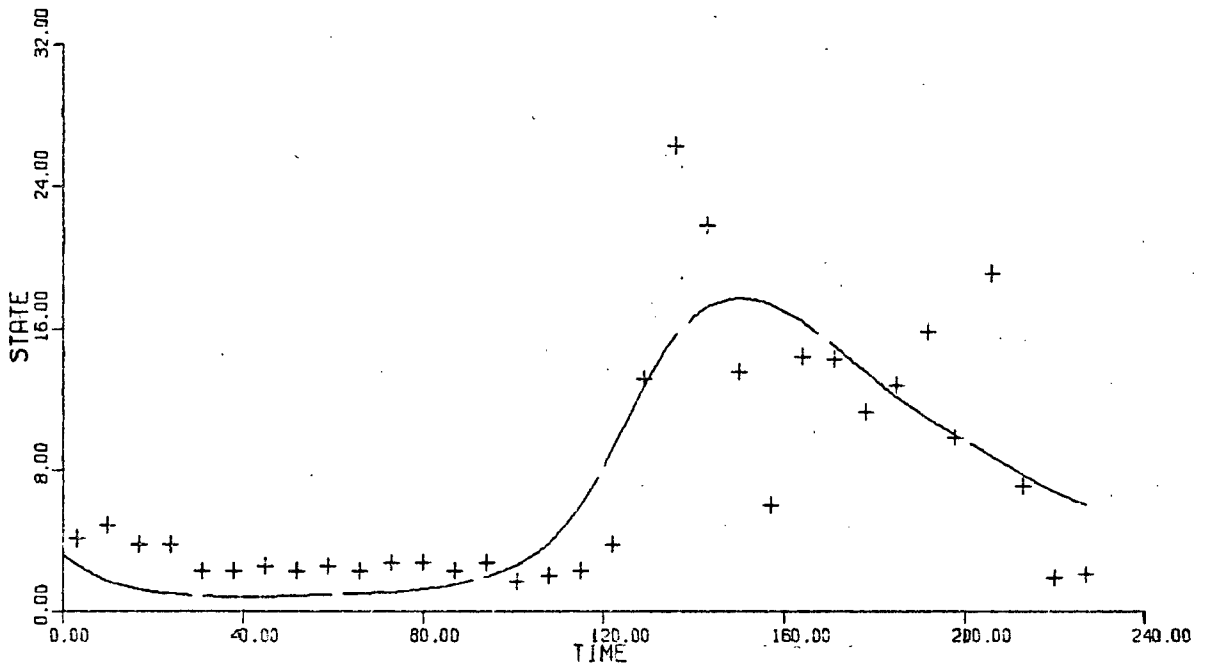

Figure 6.3.4
Integration results at IFIT+FIT results

Time	Number of real roots	Real roots		
1	1	0.07464		
84	1	2.093		
85	3	2.216	9.331	13.45
89	3	3.130	5.395	16.47
90	1	16.95		
151	1	17.14		
152	3	3.260	5.142	16.60
156	3	2.134	10.83	12.03
157	1	1.994		
226	1	0.1436		

Table 6.3.1
Roots of (6.3.2) corresponding to Figure 6.3.4

we have three equilibria, and still later we have only one relatively high equilibrium. Still later in the time interval, we again acquire three equlilibria. At the end of the time interval, we are back to only one lower equilibrium. However, no rapid jumps to new equilibria are evident in Figure 6.3.4. In the results which follow, a more pronounced jump to a higher equilibrium was obtained.

The above parameter vector corresponding to Figure 6.3.4 is not the only point in parameter space where the solution to (6.3.1) fits the observations well. Starting at $p^{(o)}$ defined above, and following an interactive strategy of freezing parameters, optimizing on subspaces, and resetting parameters, we arrived at the point

$$(.9900,\ .7284,\ 2.796)^T$$

in parameter space. The sum of the squares of the residuals at the above local minimum was approximately 662. Integration

results at the above parameters are shown in Figure 6.3.5.

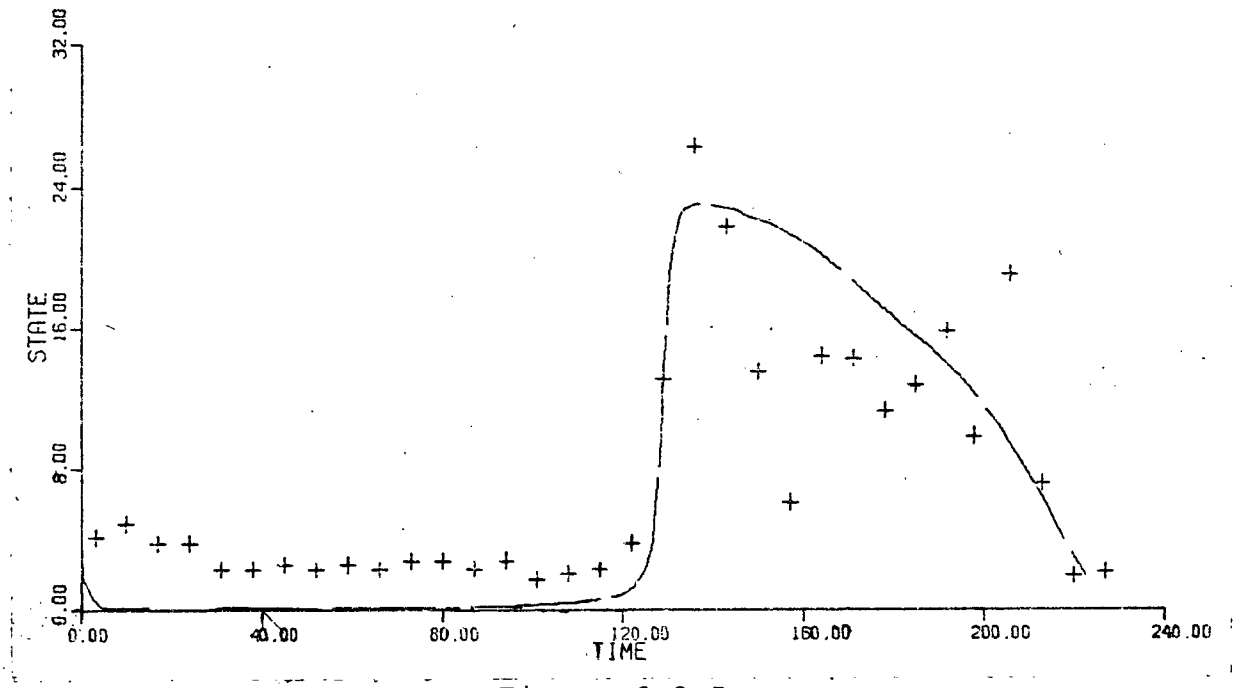

Figure 6.3.5
Integration at interactively obtained optimum

(Integrations under error control were employed near the above point in parameter space.) This point was difficult to find and it is unlikely that an automatic approach would have much luck in finding it. The difficulty seems to stem from the nature of the least squares surface near the above point in parameter space. For example if we change p_2 to .7 then the integration results become essentially zero for the full time interval under consideration, and the sum of the squares of the residuals becomes 3085. Integration results corresponding to this sum of squares are shown in Figure 6.3.6. Starting from the parameters of Figure 6.3.6, the FIT approach again produced the optimum illustrated in Figure 6.3.4.

In Table 6.3.2, we list some of the roots of (6.3.2) as a function of time at the parameters of Figure 6.3.5. At the

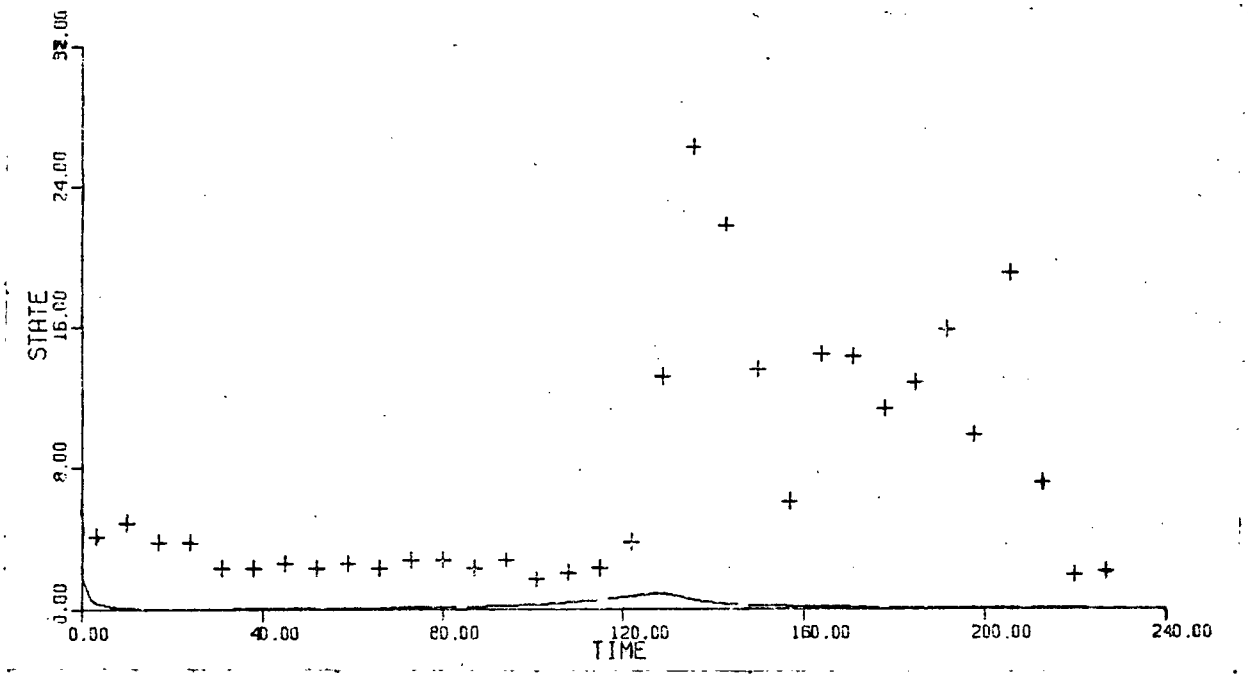

Figure 6.3.6
Integration near interactively obtained optimum

Time	Number of real roots	Real roots		
1	1	0.007244		
42	1	0.08443		
43	3	0.08730	10.66	14.25
112	3	0.6511	0.8676	23.48
113	1	23.52		
129	1	23.51		
130	3	0.6316	0.8947	23.47
176	3	0.08548	12.34	12.57
177	1	0.08367		
226	1	0.01396		

Table 6.3.2
Roots of (6.3.2) corresponding to Figure 6.3.5

start of the time interval, we have only a relatively low equilibrium. Around t=43, we acquire three equilibria, and around t=113, the lower two of these three equilibria vanish and we are left with only a relatively high equilibrium. Later, we again acquire three equilibria, and around t=177, the higher two of these three vanish and we are left with a relatively low equilibrium which remains until the end of the time interval. The disappearance of the lower two equilibria around t=113 corresponds to the rapid increase in the solution to (6.3.1) shown in Figure 6.3.5.

The large error in the observations makes it difficult to choose between the two solutions obtained for this problem; however, qualitatively the latter solution is more pleasing. In conclusion, the interactive approach has provided us with a solution which qualitatively behaves in the desired fashion. It is now the task of the model builder to interpret and perhaps build on these results. When interpreting the parameters corresponding to to Figure 6.3.5, the model builder must of course take into account the drastic change possible in the sum of the squares of the residuals due to a relatively small change in parameter space.

6.4 A REINDEER POPULATION GROWTH MODEL

The model presented below represents an attempt to describe the reindeer population in Alaska from the year 1891 to the year 1963 [2]. The Bazykin predator-prey model discussed in Chapter 4 is employed. The reindeer correspond to the predators and the

forage corresponds to the prey. Let y_1 represent forage in units of 100 tons per square mile, and let y_2 represent the reindeer population density in units of animals per square mile. We consider the dynamic model (proposed by Dr. C.J. Walters, Institute of Animal Resource Ecology, University of British Columbia)

$$\begin{aligned} y_1' &= p_1 y_1 (1-cy_1) - (r/100) y_1 y_2 / (p_2 + y_1) \\ y_2' &= p_3 r y_1 y_2 / (p_2 + y_1) - p_4 y_2 \end{aligned} \tag{6.4.1}$$

where p_1 represents the growth rate of the forage for small y_1 in the absence of grazing, 1/c represents the equilibrium for the forage in the absence of grazing, r represents the reindeer feeding rate, p_2 and p_3 are measures of the reindeer feeding efficiency, and p_4 represents the reindeer death rate. Of course all parameters should be positive. Our initial conditions are

$$y_1(0)=10,\ y_2(0)=.001$$

where t=0 corresponds to the year 1891. The initial condition on y_1 represents 1000 tons of forage per square mile. We take the constant c=.1. That is, the equilibrium for the forage in the absence of grazing is 1000 tons per square mile, the initial condition on y_1. The reindeer population density observations are derived from population counts over an area of approximately 20000 square miles. Thus the initial condition on y_2 represents very few (approximately 16) reindeer in this area. We fix r at 2 (in units of tons per year) for the reindeer in this area.

Our initial approximation to the parameters was

$$p^{(o)} = (.3,\ 1,\ .1,\ .06)^T.$$

In Figure 6.4.1, the observations on y_2 are shown along with integration results at the above parameter vector. There are no

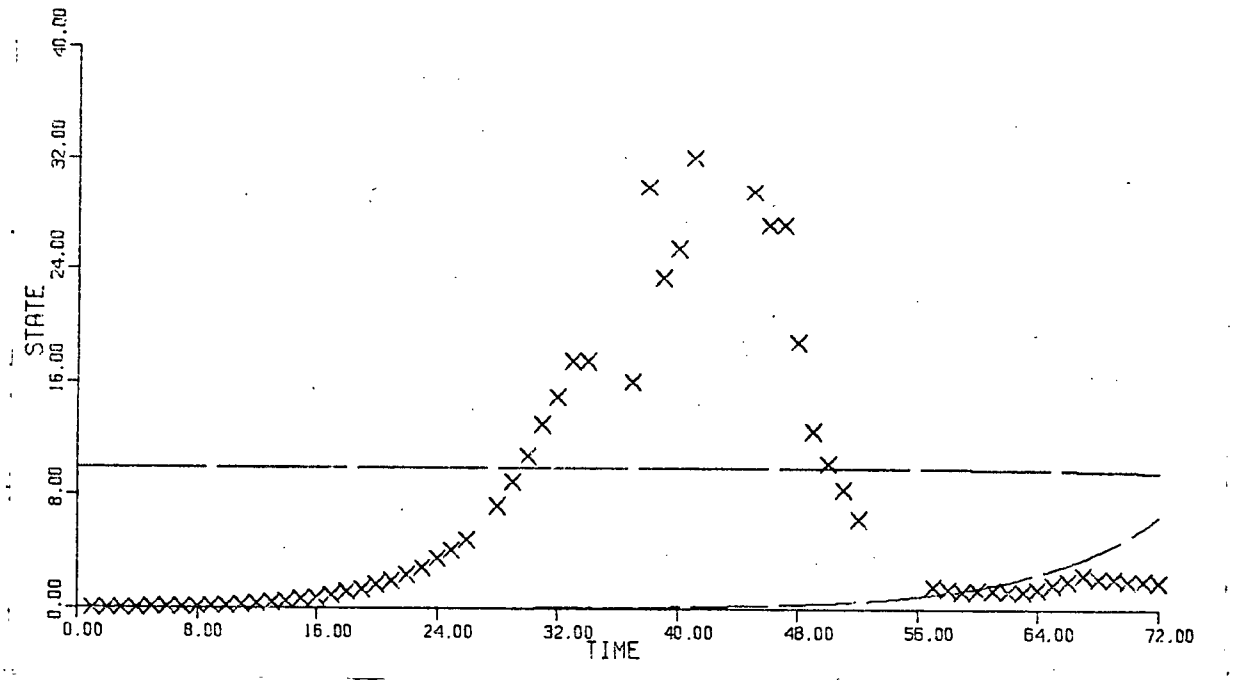

Figure 6.4.1
Observations and integration results at $p^{(o)}$

observations on the forage, and the initial condition on y_1 represents a rough guess.

In the following discussion, we demonstrate through a set of experiments, the power of an interactive approach on this fairly difficult problem.

EXPERIMENT 1

It seems worthwhile to begin with a direct attack on the problem. If this strategy succeeds, then we are finished, and

if it does not , we can try a more sophisticated strategy. Even if this direct attempt does not work, it may suggest further experiments. Starting at the above $p^{(0)}$, the FIT approach produced the optimal parameters

$$(-.2691,\ 24.82,\ .8093,\ .1763)^T.$$

The sum of the squares of the residuals at this mimimum was approximately 494. Integration results at these parameters are shown in Figure 6.4.2. The results look excellent graphically,

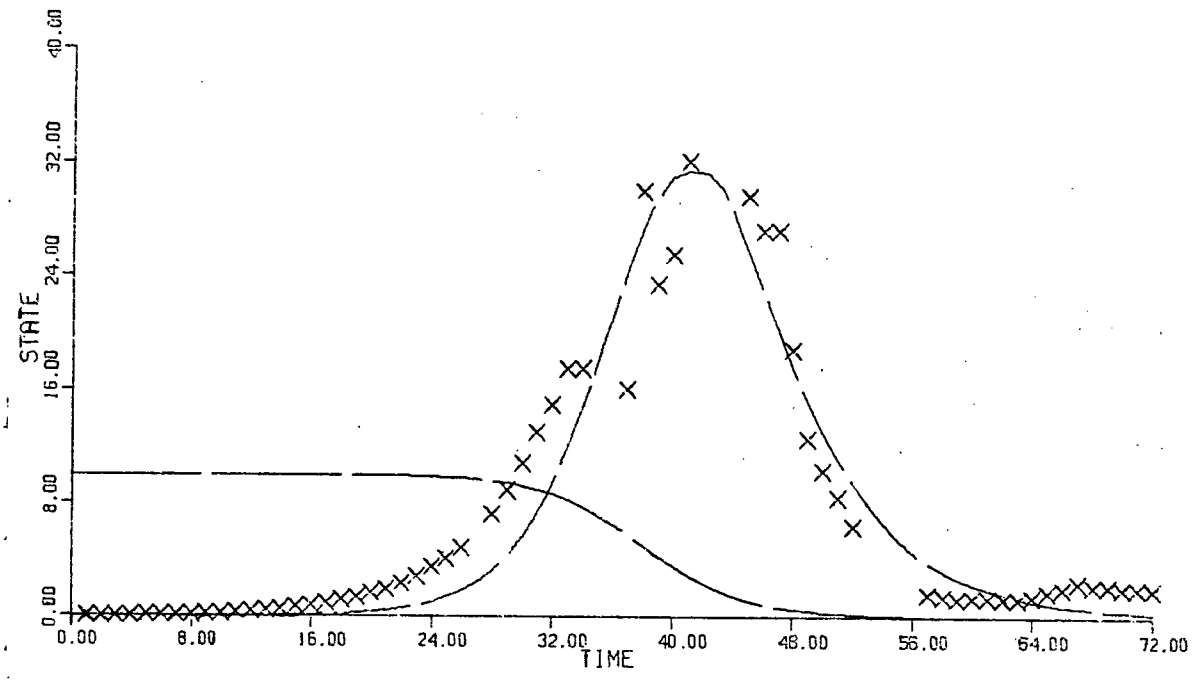

Figure 6.4.2
Integration results for Experiment 1

but p_1 is negative and p_2 is much too large. The 95% confidence intervals , as defined in Chapter 1, were

$$(\pm 4.041,\ \pm 3788,\ \pm 83.11,\ \pm .1385)^T.$$

The parameter p_2 has strayed into a region where the least

squares surface is very insensitive to it. The negative sign of p_1 suggests we constrain it to be positive; however, this strategy starting at $p^{(0)}$ did not prove effective. (The parameter p_1 became small, but the integration results remained essentially the same as those in Figure 6.4.1.) Freezing p_1 at .3 and starting at $p^{(0)}$ did not help either; however, the next experiment was successful.

EXPERIMENT 2

In this experiment, p_2 was frozen at 1 and the optimization was started at $p^{(0)}$ defined above. This strategy found an optimum at

$$(-.2667,\ 1.,\ .2250,\ .1366)^T.$$

The sum of the squares of the residuals at these parameters was approximately 757. Graphically, the solution appears to model the observations quite well; however, p_1 is negative. Our next experiment produced qualitatively promising behavior with all parameters positive.

EXPERIMENT 3

In this experiment, p_1 and p_2 were frozen at .3 and 1 respectively, and we let p_3 and p_4 start at the values obtained in the previous experiment. Optimizing on the resulting two dimensional subspace of parameter space produced the parameters

$$(.3,\ 1,\ .9704,\ 1.488)^T.$$

The sum of the squares of the residuals at the above parameters was approximately 20,000. Integration results at the above parameters are shown in Figure 6.4.3. In spite of the large sum

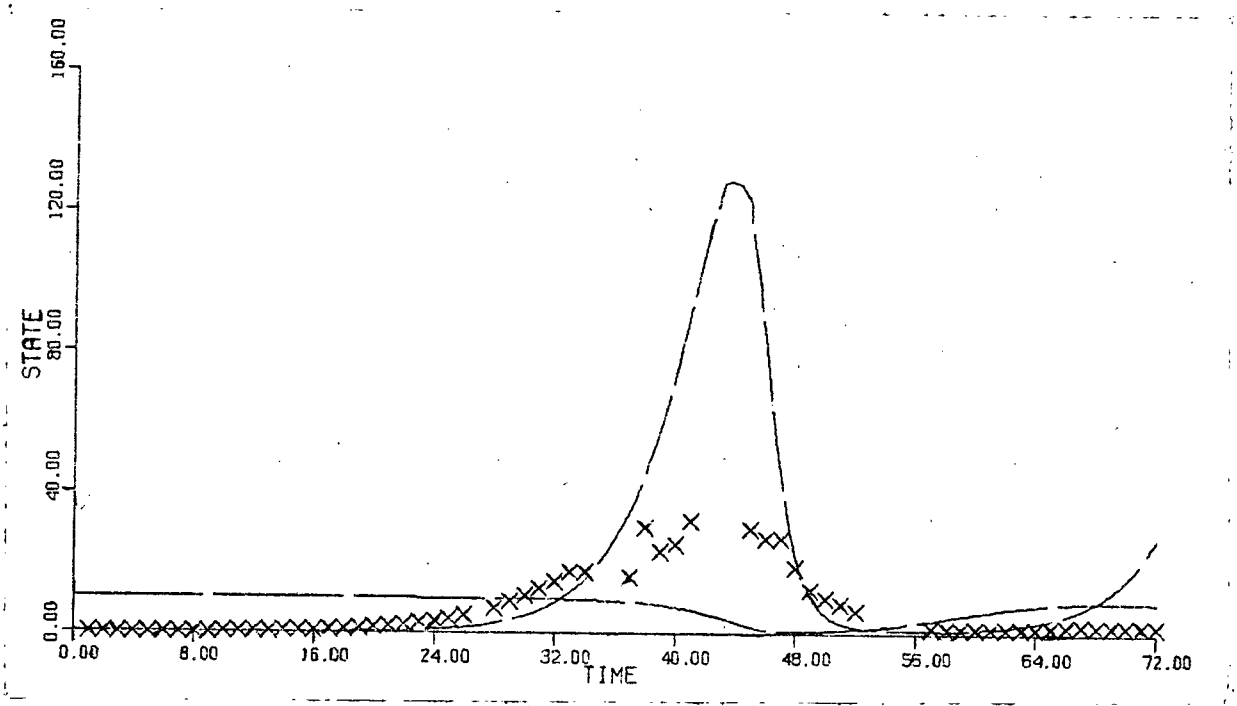

Figure 6.4.3
Integration at results of Experiment 3

of squares, the results look promising. The model equations suggest that by reducing p_3 and p_4, a much better result should be possible; however, interactive experiments along these lines produced instabilities very easily. We improve on the above results in the next experiment.

EXPERIMENT 4

Starting at the parameters obtained in the previous experiment, and not employing any freezing, we found an optimum at the point

$$(.04046,\ 14.61,\ 3.635,\ 2.669)^T$$

in parameter space. The sum of the squares of the residuals at this point was approximately 490. Integration results at these parameters are shown in Figure 6.4.4. The parameters are now

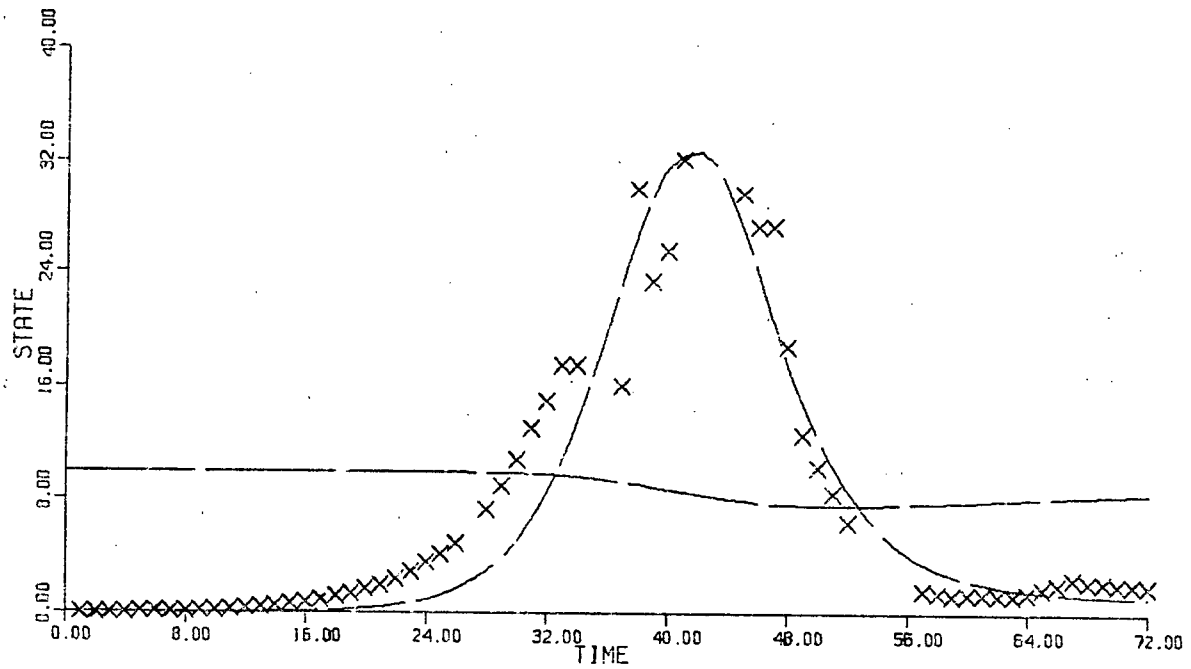

Figure 6.4.4
Integration results at optimum of Experiment 4

all positive and the integration results at these parameters fit the data quite well. However, p_2 is again much too large, as are p_3 and p_4, and y_1 does not appear to be as active as it should be. It appears that the absence of observations on y_2 leaves too much flexibility in the model.

EXPERIMENT 5

Our last experiment in this section is with guessed observations and iterated integral fitting methods. The

smoothing of the observations was accomplished with a least squares cubic spline using joints at t=5, 10, 21, 23, 42, 57, 59. The initial guessed observations for the forage were points on the cubic spline that interpolates

(0,10), (10,10), (20,9), (30,6), (50,3), (60,2), (72,1).

The end conditions for this interpolation are described under the CREOBS command in the PARFIT documentation in Appendix A. The iterated integral fitting approach (employing subsystem integrations) worked well; however, it drew us to the parameters found in Experiment 1. Freezing p_2 and using the above guessed observations produced a negative p_1 on the first iteration, and further iterations did not correct this situation. Next, p_1 and p_2 were frozen at .3 and 1 respectively. The iterated integral fitting approach (using (3.4.1)) produced the parameters corresponding to Figure 6.4.3.

While the methods employed in this experiment did not provide any new parameter estimates, they at least demonstrated the strong preference that exists in the model for the estimates obtained in the previous experiments.

The above experiments are a distillation of a few fairly short interactive sessions. To gain comparable results with a noninteractive approach would take a good deal longer and would demand a lot of patience and determination on the part of the user.

Much more could be done with this problem. For example, we

could search for other local minima. We could also try more parameters or different combinations of parameters. (For example, we could fix one of the above parameters and let r be a parameter.) Our main goal, however, is to gain experience with PARFIT on a variety of problems. Thus we turn to our next example, a problem involving an ocean plankton model.

6.5 AN OCEAN PLANKTON MODEL

Our final example in this chapter involves a model of the stages in the life cycle of certain ocean zooplankton. We consider a simplified model where only three stages in the life cycle are represented by state variables. Let y_1, y_2, and y_3 represent the population densities of these three stages in units of population per cubic metre of sea water. The adult population density is represented by y_3. Physically, there are death rates p_1, p_2, and p_3 associated with y_1, y_2, and y_3 respectively, and there is a transfer rate p_4 from y_1 to y_2, and a transfer rate p_5 from y_2 to y_3. We drive the system with a function $x(t)$ which represents the population density of the stage in the life cycle before that represented by y_1. Let p_6 represent the transfer rate from $x(t)$ to y_1. Our dynamic model is thus ([54])

$$\begin{aligned} y_1' &= -(p_1+p_4)y_1 + p_6 x(t) \\ y_2' &= p_4 y_1 - (p_2+p_5)y_2 \\ y_3' &= p_5 y_2 - p_3 y_3 \end{aligned} \tag{6.5.1}$$

We note that this model is linear. Observations on all state

variables and on x(t) were taken using large plastic bags (suspended from the ocean surface) which were designed to isolate samples of ocean water[16]. The observations are shown in Figures 6.5.1, 6.5.2, and 6.5.3. For these three graphs, time is measures in days, observations on y_3 are symbolized by ◇, and integration results for y_3 are represented by an unbroken line.

The function x(t) was approximated by

$$22.15+8.036t -.3099t^2 +.003156t^3 . \qquad (6.5.2)$$

The initial conditions were

$$y(1)=(.407, .271, .291)^T,$$

and our starting parameter vector was

$$p^{(o)} =(.03, .02, .01, .1, .2, .06)^T.$$

All parameters should remain between 0 and 1. A direct approach (FIT method) found the point

$$(.1256, -.1121, .1626, .06709, .2219, .07167)^T$$

in parameter space. The sum of the squares of the residuals at the above parameters was approximately 1690. Unfortunately p_2 is negative. We note that no parameters have exceeded 1 and this is encouraging. Next we constrained p_2 with a square root scaling. With this transformation, the FIT method found an optimum at (unscaled)

$$(.09615, 1.160E-6, .1087, .09013, .1523, .06958)^T$$

when it was started from $p^{(b)}$ defined above. The sum of the squares of the residuals at the above parameters was approximately 1710. All the parameters are now between 0 and 1. Integration results at the above parameters are shown along with the observations in Figures 6.5.1, 6.5.2, and 6.5.3.

Numerically this was a rather easy problem compared with the other three examples considered in this chapter. Visually, there appear to be peaks in the observations which could be approximated better, and this is a subject for further study.

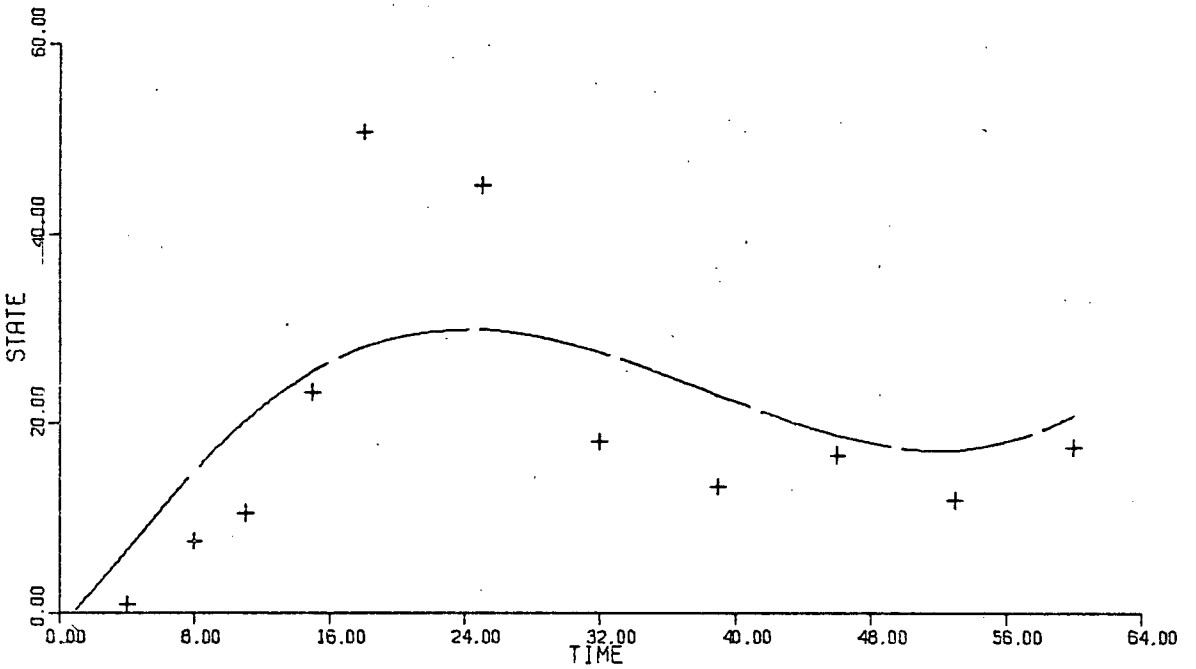

Figure 6.5.1
Observations and best fit for y_1

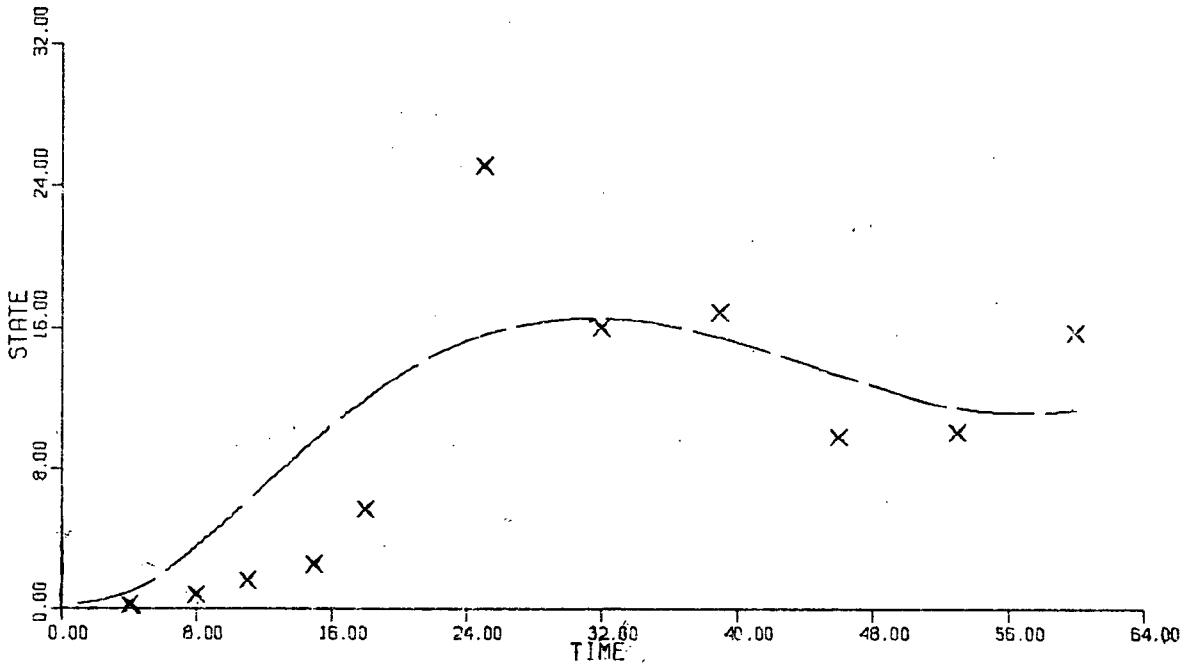

Figure 6.5.2
Observations and best fit for y_2

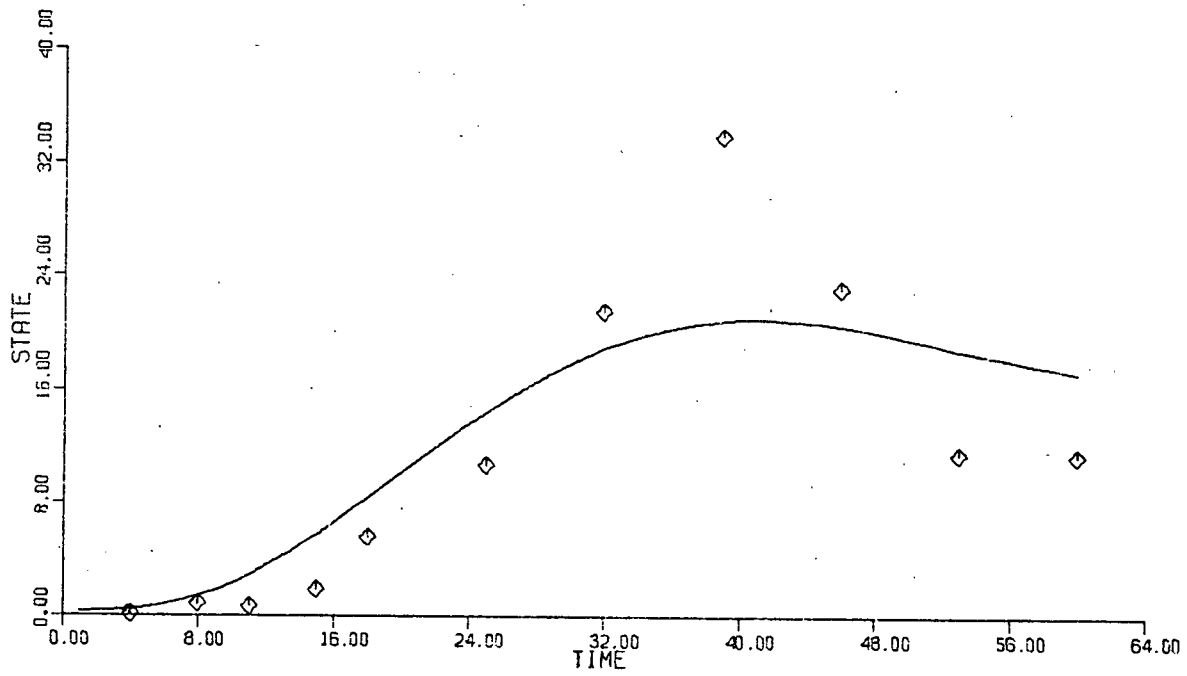

Figure 6.5.3
Observations and best fit for y_3

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

Our goal was to develop techniques designed to overcome poor initial approximations to the optimal parameters. The necessity for such techniques became evident with experiments (such as those of Section 4.2) using the direct approach employing the sensitivity equations. The basic conclusion from such experiments was that the direct approach often produced a highly nonlinear problem where the effect of small parameter changes could be dramatic. Thus we turned to such methods as derivative and integral fitting. As shown in Section 4.2, these approaches are very useful. It also became evident that to resolve a problem often required several runs employing different complementary procedures. Thus an interactive approach appeared to be a good way to proceed.

The case when observations are not available on all state variables arises often in practice, and experiments indicated that this situation can be very difficult if approached directly with poor initial parameter estimates. Thus we sought an extension of the derivative and integral fitting techniques that could handle this problem. As a result we developed the effective approach of guessing the desired behavior of the unobserved states and then iteratively improving this guess. As shown in Section 4.5, this approach can be very effective when

the direct approach presents all sorts of difficulties.

In keeping with our goal of developing tools for overcoming (at least in part) the difficulties associated with a direct approach, we investigated briefly in Chapter 5 the value of sequentially processing the observations. We conclude that this strategy can be superior to a direct attack on the problem.

The main trouble with a coarse method is that it can produce parameters from which the direct approach is still incapable of succeeding. For example the initial value problem may be unstable at the parameters produced by the coarse method. In Section 3.5, we proposed methods employing continuation parameters and break points to bridge the gap between the coarse approach and the full nonlinear problem. We conclude from the example of Section 3.5 that this approach can be effective on problems involving an instability. Our approach to the use of break points differs from that of van Domselaar and Hemker[71], and we are unaware of any work connected with parameter estimation in dynamic models that involves the use of break points along with continuation parameters.

We conclude from our experience on real world problems that with an interactive approach it is possible, in a relatively short time, to make substantial progress on these fairly difficult problems. However, to be effective such an interactive package must be well organized and have a fairly wide range of options available. The implementation of such a package is an evolutionary process. Our package PARFIT

represents the initial stage in this process.

7.2 SUGGESTIONS FOR FUTURE WORK

The iterated integral and derivative fitting methods developed in Chapter 3 appear to be worthy of further study. For example in the experiments presented in Section 4.4, the iterated integral fitting method using subsystem integrations worked much better than the iterated integral and derivative fitting methods employing (3.4.1). The reason behind this seems worthy of further study. Hopefully, such a study may lead to other effective ways of updating the guessed observations.

Other promising avenues of further research involve the use of continuation parameters and break points, and the use of sequential techniques for improving starting parameters. Finally there is the development of new and improved versions of a package such as PARFIT. The value of good software when approaching the sort of problems considered in this thesis cannot be over emphasized.

BIBLIOGRAPHY

[1] Aaro, I., Design and implementation of a tool for interactive communication between user and program, Dept. of Info. Proc., Roy. Inst. of Tech., Stockholm, Sweden, TRITA-NA-7705 (1977)

[2] Aaro, I., Design and implementation of a software system for interactive scientific computing, Dept. of Info. Proc., Roy. Inst. of Tech., Stockholm, Sweden, TRITA-NA-7707 (1977)

[3] Albert, A.E. and Gardner, L.A., Stochastic Approximation and Nonlinear Regression, M.I.T. Press, Cambridge, Mass. (1967).

[4] Avila, J., Continuation methods for nonlinear equations, SIAM J. Numer. Anal., 11 (1974) 102-122.

[5] Bard, Y., Comparison of gradient methods for the solution of nonlinear parameter estimation problems, SIAM J. Numer. Anal. 7 (1970) 157-186.

[6] Bard, Y., Nonlinear Parameter Estimation, Academic Press, New York (1974).

[7] Bazykin, A.D., Volterra's system and the Michaelis-Menton equation in Problems in Mathematical Genetics, Ed. by V.A. Ratner, U.S.S.R. Acad. Sci., Novosibrisk (1974) 103-142. (Available in English as Structural and dynamic stability of model predator-prey systems (1975), Institute of Resource Ecology, University of British Columbia, Vancouver, B.C., Report R-3-R.)

[8] Beale, E.M.L., Confidence regions in nonlinear estimation, J. Roy. Stat. Soc. 22 (1960) 41-88.

[9] Bellman, R., Jacquez, J., Kalba, R., and Schwimmer, S., Quasilinearization and the estimation of chemical rate constants from raw kinetic data, Math. Biosci. 1 (1967) 71-76.

[10] Bellman, R., Kagiwada, H., Kalaba, R., and Vasudevan, R., Quasilinearization and the estimation of differential operators from eigenvalues, Comm. ACM 11 (1968) 255-256.

[11] Bellman, R., Kashef, B., and Vasudevan, R., The inverse problem of estimating heart parameters from cardiograms, Math. Biosci. 19 (1974) 221-230.

[12] Berman, M., Weiss, M.F. and Shahn, E., Some formal approaches to the analysis of kinetic data in terms of linear compartmental systems, Biophys. J. 2 (1962) 289-316.

[13] Boggs, P.T. and Dennis, J.E. Jr., A stability analysis for perturbed nonlinear iterative methods, Math. of Comp. 30 (1976) 199-215.

[14] Bus, J.C.P., van Domselaar, B., and Kok, J., Nonlinear least squares estimation, Math. Cent. Amsterdam, NW 17/75 (1975).

[15] Byrne, G.D. and Hall, C.A. (Ed.) Numerical Solution of Systems of Nonlinear Algebraic Equations, Academic Press, New York (1973).

[16] CEPEX-Controlled Ecosystem Pollution Experiment, Patricia Bay, Sannich, British Columbia, Canada, unpublished working paper (1976).

[17] Curtis, A.R., and Edsberg, L., Some investigations into data requirements for rate constant estimation, A.E.R.E., Harwell, HL. 73/3744 (1973).

[18] Dennis, J.E. Jr., Some computational techniques for the nonlinear least squares problem, in Numerical Solution of Systems of Nonlinear Algebraic Equations Ed. by G.D. Byrne and C.A. Hall, Academic Press, New York (1973) 157-183.

[19] Dennis, J.E. Jr., Nonlinear least squares and equations, A.E.R.E. Harwell, Oxfordshire, CSS 32 (1976).

[20] Dennis, J.E. Jr., Gay, D.M., and Welsch, R.E., An adaptive nonlinear least squares algorithm, National Bureau of Economic Research, Cambridge, Mass. prelim. rep. (1977).

[21] Edsberg, L., KEMPEX-II, A program package for interactive simulation of some chemical reactors. Dept. of Info. Proc., Roy. Inst. of Tech., Stockholm, Sweden, TRITA-NA-7504 (1975).

[22] Farrow, L. and Edelson, D., The steady-state approximation: fact or fiction, Int. J. of Chem. Kin. VI (1974) 787-800.

[23] Ficken, F.A., The continuation methods for functional equations, Comm. Pure Appl. Math. 4 (1951) 435-456, Math. Rev. 13 (1952) 562-563.

[24] Fletcher, R., Generalized inverse methods for the least squares solution of systems of nonlinear equations, The Comput. J., 10 (1968) 392-399.

[25] Gear, C.W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1971).

[26] Gear, C.W., The automatic integration of ordinary differential equations, Comm. ACM 14 (1971) 176-179.

[27] Gear, C.W., DIFSUB for solution of ordinary differential equations, Algorithm 407, Comm. ACM 14 (1971) 185-190.

[28] Gelb, A. (Ed.), Applied Optimal Estimation, The M.I.T. Press, Cambridge, Mass. (1974).

[29] Gill, P.E., Golub, G.H., Murray, W. and Saunders, M.A., Methods for modifying matrix factorizations, Comp. Sci. Dept., Stanford, U., STAN-CS-72-322 (1972).

[30] Gill, P.E. and Murray, W., (Ed.) Numerical Methods for Constrained Optimization, Academic Press (1974).

[31] Goldstein, A.A. and Price, J.F., An effective algorithm for minimization, Numer. Math. 10 (1967) 184-189.

[32] Golub, G.H., and Reinsch, C., Singular value decomposition and least squares solutions - Handbook series in linear algebra, Numer. Math. 14 (1970) 403-420.

[33] Ho, Yu Chi, On the stochastic approximation method and optimal filtering theory, J. Math. Anal. and Appl. 6 (1962) 152-154.

[34] Jones, A., SPIRAL-a new algorithm for non-linear parameter estimation using least squares. The Comput. J. 13 (1970) 301-308.

[35] Kelley, H.J. and Denham, W.F., Modeling and adjoints for continuous systems, JOTA 3 (1969) 174-183.

[36] Kowalik, J. and Osborne, M.R., Methods for Unconstrained Optimization Problems, American Elsevier, New York (1968).

[37] Krebs, C.J., Gainer, M.S., Keller, B.L., Myers, J.H. and Tamarin, R.H., Population cycles in small rodents, Science 179 (1973) 35-41.

[38] Lawson, C.L. and Hanson, R.T., Solving Least Squares Problems, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1974).

[39] Levenberg, K., A method for the solution of certain non-linear problems in least squares, Quart. Appl. Math. 2 (1944) 164-168.

[40] Long, C.E., Model stability, resilience, and management of an aquatic community, Oecologia (Berl.) 17 (1974) 65-85.

[41] Lotka, A., Elements of Mathematical Biology, Dover, Pub. Inc., New York (1956), republication of Elements of Physical Biology, Wilkins Co. Inc. (1924).

[42] Mackey, M., Oscillation and chaos in physiological control systems, to appear in Science.

[43] Marquardt, D.W., An algorithm for least squares estimation of nonlinear parameters, SIAM J. 11 (1963) 431-441.

[44] Marten, G.G., Kleiber, P.M., and Reid, J.A.K., A computer program for fitting tracer kinetic and other differential equations to data, Ecology 56 (1975) 752-754.

[45] McConalogue, D.J., A quasi-intrinsic scheme for passing a smooth curve through a discrete set of points, The Comput. J. 13 (1970) 392-396.

[46] McKeown, J.J., Specialised versus general-purpose algorithms for minimising functions that are sums of squared terms, Math. Prog. 9 (1975) 57-68.

[47] Murray, W. (Ed.), Numerical Methods for Unconstrained Optimization, Academic Press, London (1972).

[48] Myers, J.H. and Krebs, C.J., Population cycles in rodents, Scientific American, June (1974) 38-46.

[49] Nordsieck, A., On the numerical integration of ordinary differential equations, Math. Comp. 16 (1962) 22-49.

[50] Ortega, J.M. and Rheinboldt, W.C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York (1970).

[51] Osborne, M.R., Some aspects of non-linear least squares calculations, in Numerical Methods for Non-linear Optimization, Ed. by F.A. Lootsma, Academic Press, London (1972) 171-189.

[52] Osborne, M.R., A class of methods for minimising a sum of squares, The Aust. Comp. J. 4 (1972) 164-169.

[53] Parker, R.A., The influence of environmental driving variables on the dynamics of an aquatic ecosystem model, Verh. Internat. Verein. Limnol. 19 (1975) 47-55.

[54] Parslow, J. and Sonntag, N., Parameter estimation of the natural population, Institute of Oceanography, University of British Columbia, Vancouver, B.C., (in preparation).

[55] Pearson, J.B., On nonlinear least squares filtering, Automatica, 4 (1967) 97-105.

[56] Rall, L.B., Davidenko's method for the solution of nonlinear operator equations, The Univ. of Wisconsin, Math. Res. Cent., MRC Tech. Summary Rep. 948 (1968).

[57] Ramsin, H. and Wedin, P.-Å., A comparison of some algorithms for the nonlinear least squares problem, BIT 17(1977) 72-90.

[58] Rheinboldt, W.C., On the solution of some nonlinear equations arising in the application of finite element methods, U. of Maryland Tech. Rep. TR-362 (1975).

[59] Rheinboldt, W.C., Numerical continuation methods for finite element applications, U. of Maryland Tech. Rep. TR-454 (1976), to appear in Formulation and Computational Algorithms in Finite Element Analysis, Proc. of U.S.-German Symp., MIT Press (1976)

[60] Roberts, S.M. and Shipman, J.S., Two Point Boundary Value Problems: Shooting Methods, American Elsevier, New York (1972).

[61] Rosenbrock, H.H. and Storey, C., Computaional Techniques for Chemical Engineers, Pergamon Press, Oxford (1966) 189-208.

[62] Skoog, R.O., Ecology of the Caribou (rangifer tarandus granti) in Alaska, Ph.D. Thesis, Univ. of California, Berkely (1968).

[63] Smith, L.B., The use of man-machine interaction in data fitting problems, Stanford U., Comp. Sci. Dept. Rep. CS 131 (1969).

[64] Steen, N.M. and Byrne, G.D., The problem of minimizing nonlinear functionals-I. Least squares, in Numerical Solution of Systems of Nonlinear Algebraic Equations Ed.

by G.D. Byrne and C.A. Hall, Academic Press, New York, (1973) 185-239.

[65] Stewart, G.W., Introduction to Matrix Computations, Academic Press, New York (1973).

[66] Swartz, J. and Bremermann, H., Discussion of parameter estimation in biological modelling: algorithms for estimation and evaluation of the estimates, J. Math. Bio. 1 (1975) 241-257.

[67] Tomović, R., Sensitivity Analysis of Dynamic Systems, McGraw-Hill, New York, Trans. by D. Tornquist. (1963).

[68] Tomović, R. and Vukobratović, M., General Sensitivity Theory, American Elsevier, Inc., New York (1972).

[69] Vandermeer, J.H., The competitive structure of communities: an experimental approach with protozoa, Ecology 50 (1969) 362-371.

[70] van Domselaar, B., A mathematical analysis of the heart-infarct (Dutch), Math. Cent., Amsterdam NN 4/74 (1974).

[71] van Domselaar, B. and Hemker, P.W., Nonlinear parameter estimation in initial value problems, Math. Cent., Amsterdam NW 18/75 (1975).

[72] Van Loan, C., Lectures in least squares, Dept of Comp. Sci., Cornell U., Ithaca, N.Y. TR 76-279 (1976).

[73] Varga, R.S., Matrix Iterative Analysis, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1962).

[74] Walters, C.J., and Hilborn, R., Adaptive control of fishing systems, J. Fish. Res. Board Can., 33 (1976) 145-159.

[75] Walters, C.J., Institute of Animal Resource Ecology, University of British Columbia, Vancouver, B.C., personal communication.

[76] Young, P., Recursive approaches to time series analysis, Inst. Math. Appl. 10 (1974) 209-224.

APPENDIX A

PARFIT DOCUMENTATION

Our interactive parameter fitting program, PARFIT, serves several purposes. It establishes an environment in which new algorithms can be tested. It is an experimental tool for investigating the utility of various commands for interactive parameter fitting in dynamic models. It is also a device for studying the organization of an interactive program of this nature. Finally, it is a practical tool for fitting parameters in dynamic models.

PARFIT is structured so that the various numerical algorithms can easily be extracted into individual procedures (see Chapter 3 for flow charts outlining the facilities of PARFIT). As mentioned in Chapter 3, the development of an interactive package such as PARFIT is an evolutionary process. The documentation in this appendix describes the first stage in this process.

PARFIT

INTERACTIVE PARAMETER FITTING IN DYNAMIC MODELS

CONTENTS

APPENDIX A

1. INTRODUCTION AND NOTATION

The program PARFIT is designed to allow a user to interactively fit parameters in an initial value problem when observations are available on the solution to the problem. Its use does not require the learning of any complex set of commands. The program is written in ALGOL W; however, a knowledge of FORTRAN is sufficient for its use. Specifically the program handles initial value problems of the form

$$y' = g(t,y,p)$$

$$y(t_o) = y_o(p)$$

where y is an n-vector of state variables, p is an m-vector of parameters, t is the independent variable which we will call time for convenience, and ' indicates differentiation with respect to time. Along with the initial value problem, we have a set of observations $v_1,\ldots,v_k$ taken at distinct times $t_1,\ldots,t_k$ respectively where t_ℓ may or may not equal t_o and where $t_\ell \geq t_o$, $\ell=1,2,\ldots,k$. Each v_ℓ is an r-vector where $r \leq n$. That is, not all components of y need be observed. Each v_ℓ, however, contains observations on the same components of y.

The central part of PARFIT is a command reader. This nucleus of the program ties together the various facilities such as integration, optimization and plotting. After PARFIT has executed a particular command, control returns to the command reader and PARFIT is ready for the next command. This continues until a QUIT command is issued. Once a command is issued, PARFIT usually elicits from the user all the information required to execute the command. This principle is, in the interests of efficiency, violated slightly in the interactive option of the FIT command.

It is useful to view the operation of PARFIT in terms of an environment and a set of commands which operate in and on this environment. The environment consists of such things as echo flags, parameter scaling and freezing indicators, sample times, output options, and algorithm selection indicators.

PARFIT is modular in nature and is designed to make the addition of new commands very easy. Existing commands are also easy to modify. For example at present the Levenberg-Marquardt technique is used to solve nonlinear least squares problems, but other optimization methods can easily be added. New integration schemes can also easily be added.

We summarize here some of the facilities of PARFIT. For more details consult the command descriptions. There are extensive output facilities for printing a variety of things such as Jacobian matrices, integration results, optimization details and smoothing results. Parameters can be frozen and scaled. There are facilities for determining the optimal parameters in a least squares sense by automatic and by interactive techniques. There are facilities for obtaining initial approximations to the optimal parameters using such techniques as derivative and integral fitting, guessed observations and iterative improvement of guessed observations for unobserved state variables, and methods employing continuation parameters and break points. A subsection of PARFIT can be used in batch mode. Certain statistical information on the optimal parameters can be calculated and various control parameters governing the way PARFIT runs can be reset by the user.

PARFIT functions with two sets of discrete time values. First we have the observation times where the observations $v_1,\ldots,v_k$ were taken and second we have the sample times. The sample times are the times at which plot points for the state variables and smoothed observations are taken and the times at which information on the continuous solution of the sensitivity equations (see below) can be extracted. For further details see the SAMPLE command and item 7 of the REPORT command.

Before continuing our description of PARFIT, we must establish more notation. Define the weighted residual vector f by

$$f_{r(l-1)+s} = w_{r(l-1)+s}\,(y_{i(s)}(t_l) - v_{ls})$$

for s=1,...,r, l=1,...k where r is the length of v_l, l=1....k, v_{ls} is component s of v_l and $y_{i(s)}(t_l)$ is the corresponding element of the vector $y(t_l)$. $w_{r(l-1)+s}$ is a weighting factor. We note that f has length kr. We seek p to minimize

$$F(p) = f^T(p)\,f(p).$$

The gradient of F(p) is

$$\nabla F(p) = 2J^T(p)\,f(p)$$

where J(p) is the kr x m matrix defined by

$$J_{ij} = \frac{\partial f_i}{\partial p_j} \quad i=1,\ldots,kr; \quad j=1,\ldots,m.$$

The elements in J are found by integrating the sensitivity

equations:

$$y'_{p_j} = g_y(t,y,p)\,y_{p_j} + g_{p_j}(t,y,p)$$

$$y_{p_j}(t_o,p) = (y_o)_{p_j}(p)$$

for j=1,...,m derived from the original initial value problem. In our notation y_{p_j} and g_{p_j} are n-vectors of partial derivatives of the components of y and g with respect to p_j, and g_y is the Jacobian matrix of g with respect to the vector y. Our integration program is specially designed to take advantage of the fact that our sensitivity equations are linear and coupled in only one direction to the original nonlinear initial value problem. All the step size and order adjustments are done on the nonlinear initial value problem and the linear problems are efficiently solved along with the nonlinear problem. This is the same strategy as that adopted by van Domselaar and Hemker[71].

The Levenberg-Marquardt technique for finding the optimal parameter vector p uses the iteration

$$p^{(q+1)} = p^{(q)} - (J^T(p^{(q)})J(p^{(q)}) + \lambda I)^{-1} J^T(p^{(q)})f(p^{(q)})$$

where $0 \leq \lambda < \infty$. As in [71] we employ the singular value decomposition in our implementation of this algorithm. This avoids the forming of J^TJ with its associated squaring of the condition number. A scaling option is also available in our implementation of the Levenberg-Marquardt method. Finally in our notation a steepest descent iteration to get the optimal p is

$$p^{(q+1)} = p^{(q)} - \delta J^T(p^{(q)})f(p^{(q)})$$

where δ is the step length referred to in the interactive option of the FIT command.

To run PARFIT the information detailed in Section 2 must first be supplied. The particular run command for PARFIT depends on what facilities of PARFIT the user desires. The basic run command is

```
$RUN PFIT:PAR1+PFIT:L1+CP+*PRPLOT 4=DP T=t
```

this provides all aspects of PARFIT except the SMOOTH, DFIT, IFIT, CONTIN and CREATE commands, and limits the integration method to the trapezoidal method. The file CP contains the object code for the procedure G_FUN described in the next section, and DP is the data file described in the next section.

If CP is the object code for the FORTRAN subroutine (named GF) version of G_FUN, then the MTS run command is

```
$RUN PFIT:PAR1+PFIT:L1+PFIT:LF+CP+*PRPLOT 4=DP T=t
```

To run PARFIT with Gear's program and the CREATE command available, but without the DFIT, IFIT, CONTIN and SMOOTH commands, the MTS run command is

```
$RUN PFIT:PAR1+PFIT:PAR2+PFIT:L12+CP+*PRPLOT 4=DP T=t
```

If CP is a compiled FORTRAN subroutine, then as in the previous command, PFIT:LF must be used. To run PARFIT with everything but Gear's program and the CREATE command, the MTS run command is

```
$RUN PFIT:PAR1+PFIT:PAR3+PFIT:L13+CP+*PRPLOT+*NUMLIB 4=DP T=t
```

(with the previous modification if CP comes from a FORTRAN program). Finally, to run the complete PARFIT, the run command is

```
$RUN PFIT:PAR1+PFIT:PAR2+PFIT:PAR3+CP+*PRPLOT+*NUMLIB 4=DP T=t
```

(with the appropriate change for a FORTRAN CP). It is suggested that a time limit be put on all runs to avoid the possibility of unnecessary expense. A good strategy is to start with the simple version of PARFIT and to use the more powerful facilities when they become necessary.

2. INFORMATION REQUIRED BY PARFIT

To run PARFIT, the user must supply a procedure which defines g(t,y,p) as well as the Jacobian functions g_y and g_p and which provides initialization information. Since PARFIT is written in ALGOL W, it is natural that this program should also be in ALGOL W, but as indicated in the run commands of the previous section, a FORTRAN subroutine can be used. If a FORTRAN subroutine is used, it should be named GF and all its real arguments should be double precision. If an ALGOL W procedure is used, it should be called G_FUN and it should have the following header: (The particular parameter names are of course not important and may be changed by the user.)

```
PROCEDURE G_FUN(LONG REAL VALUE T; LONG REAL ARRAY Y(*);
LONG REAL ARRAY P(*); INTEGER VALUE OPTION;
LONG REAL ARRAY G(*); LONG REAL ARRAY DGY(*,*);
LONG REAL ARRAY DGP(*,*); LONG REAL ARRAY ISEN(*,*));
```

where

T is the independent variable time,

Y is the vector of state variables y(t) of length n,

P is the vector of parameters of length m,

OPTION indicates which of various tasks G_FUN is to perform:

```
 1 to return g(t,y,p) and g_y(t,y,p)
-1 to return just g(t,y,p)
 2 to return g_p(t,y,p)
 3  to return initial y in Y and initial values for y_pj ,
 j=1,...,m in ISEN,
-3 to return just the initial y in Y.
```

G returns the n-vector g(t,y,p) when required,

DGY returns the n x n matrix $g_y(t,y,p)$ when required,

DGP returns the n x m matrix $g_p(t,y,p)$ when required,

ISEN returns the nxm matrix of initial values for the sensitivity equations.

An example of an ALGOL W procedure G_FUN for the problem

$$y_1' = -(1-y_2)y_1 + p_2 y_2$$

$$y_2' = p_1((1-y_2)y_1 - (p_2+p_3)y_2)$$

$$y_1(0)=1, \quad y_2(0)=0$$

is given in Figure A.1. (This dynamic model is considered by van Domselaar and Hemker[71].)

Besides providing the procedure G_FUN, the user must provide a data file containing the information outlined below. This data is read under free format with blanks acting as delimiters.

The first data card contains the model name associated with the particular set of differential equations. It consists of at most 30 characters with no embedded blanks. And it cannot be the word CREATE (see the CREATE command).

```
PROCEDURE G_FUN(LONG REAL VALUE T;
LONG REAL ARRAY Y(*);LONG REAL ARRAY P(*);
INTEGER VALUE OPTION;LONG REAL ARRAY G(*);
LONG REAL ARRAY G(*);LONG REAL ARRAY DGY(*,*);
LONG REAL ARRAY DGP(*,*);LONG REAL ARRAY ISEN(*,*));
   BEGIN
   COMMENT EXAMPLE OF A MODEL DEFINITION PROCEDURE;
   CASE ABS OPTION OF
      BEGIN
         BEGIN
         G(1):=-(1.L-Y(2))*Y(1)+P(2)*Y(2);
         G(2):=P(1)*((1.L-Y(2))*Y(1)-(P(2)+P(3))*Y(2));
         IF OPTION>0 THEN
            BEGIN
            DGY(1,1):=-(1.L-Y(2));
            DGY(1,2):=Y(1)+P(2);
            DGY(2,1):=P(1)*(1.L-Y(2));
            DGY(2,2):=-P(1)*(Y(1)+P(2)+P(3));
            END;
         END;
         BEGIN
         DGP(1,1):=0.L;
         DGP(1,2):=Y(2);
         DGP(1,3):=0.L;
         DGP(2,1):=(1.L-Y(2))*Y(1)-(P(2)+P(3))*Y(2);
         DGP(2,2):=-P(1)*Y(2);
         DGP(2,3):=-P(1)*Y(2);
         END;
         BEGIN
         Y(1):=1.L;
         Y(2):=0.L;
         IF OPTION>0 THEN
            BEGIN
            FOR I:=1 UNTIL 2 DO
            FOR J:=1 UNTIL 3 DO
            ISEN(I,J):=0.L;
            END;
         END
      END;
   END G_FUN.
```

Figure A.1
A typical model definition procedure

The second data card contains the initial time.

The next data card contains 5 integers separated by blanks. These integers are

(1) number of state variables
(2) number of components of p
(3) number of components of v
(4) number of observation times
(5) maximum number of sample times

The maximum number of sample times must be greater than the number of observation times by at least one when there are no observations at the initial time. When there are observations at the initial time, the maximum number of sample times can be greater than or equal to the number of observation times.

The next set of data cards contains the observations. Several observations can be put on each card. The observations are ordered first by state variable and then by time. For example if n=5 and state variables 2 and 5 are observed then all the observations on state variable 2 are entered and following this set of numbers, all the observations on state variable 5 are entered. It is suggested, but not mandatory that observations on a new state variable start on a new card.

The next set of data cards contains the observation times. These entries need not start on a new card, but for clarity it is suggested that they do.

The next set of data is a set of integers indicating which state variables have been observed.

Next, an initial guess at the optimal parameter values should be entered in the data file.

Finally, a set of weights corresponding to the observations may be entered by the user. This is optional and if no weights are entered, PARFIT by default sets all weights to one. If weights are to be entered, there should be an entry for each observation, these entries should be in the same order as the corresponding observations, and they must start on a new card.

The use of free format should make the data entry fairly

simple. For example real numbers are entered as -.1, -3.2, .005, 2, 5 etc. and integers are entered as 1, -5, 2 etc. Numbers in exponential notation follow FORTRAN conventions.

3. FILES USED BY PARFIT

PARFIT uses a number of MTS temporary files when it is running. Normally the user need not be concerned with these files. However, if PARFIT terminates with an error, the information in these files may be of value to the user. The temporary files used are:

-SC1 This file takes output from the integeration and optimization procedures. It is always emptied before it is reused.

-SC2 When required, this file accumulates the information in -SC1 for later output.

-GRAPH This file takes the output from the plotting procedures.

-GRAPHSTORE This file accumulates graphical data for later hard copy output.

-REPRT This file takes output from the report command. It also takes certain messages such as those indicating when a permanent copy of a graph has been requested.

-AECHO This file contains an echo of the sessions with the interactive option of the fit command.

-ECHO This file contains echo information as requested through the ECHO command. It never contains any of the information that is sent to -AECHO.

4. THE BATCH MODE OF PARFIT

PARFIT is designed as an interactive program. However, there are cases when, for reasons of economy say, a user would be advised to run sections of PARFIT in batch mode. For example, this might be the case in the final stages of parameter estimation when good approximations to the optimal parameter values are available and where these values are being further refined through the automatic option of the FIT command. The MTS RUN commands for a batch run are the same as for an interactive run. The PARFIT commands should follow directly after the RUN command. Alternatively, the user may specify SCARDS=filename in the RUN command, in which case the PARFIT commands are read form the file attached to SCARDS. When in batch mode, PARFIT prints the command designators on *SINK*. When in batch mode, -SC1 and -SC2 are not used. All output that goes to -SC1 in an interactive run goes directly to *SINK* when PARFIT is runnumg in BATCH mode.

5. COMMANDS IN PARFIT

The following are the commands currently available in PARFIT.

(1) ECHO	(2) OPTION	(3) FREEZE	(4) SCALE
(5) SAMPLE	(6) WEIGHT	(7) PLOT	(8) REPORT
(9) SET	(10) INTEG	(11) PROBE	(12) FIT
(13) STATS	(14) QUIT	(15) SUSP	(16) CREATE
(17) SMOOTH	(18) CREOBS	(19) LINEAR	(20) DFIT
(21) IFIT	(22) CONTIN		

With each of the above commands, we describe what action the command initaites and what interaction with PARFIT the user can expect after issuing the command.

(1) ECHO

The ECHO command allows the user to control the accumulation of a hard copy echo of an interactive session. When this command is issued, PARFIT requests an entry of 0 or 1. An entry of 0 turns off the echo and an entry of 1 turns the echo on. This command does not affect the echo associated with option 2 of the FIT command. At the start of an interactive session PARFIT turns the echo off.

(2) OPTION

This command allows the user to set certain control parameters in PARFIT that govern the way various procedures function. These control parameters are automatically given default values by PARFIT at the start of a run. After issuing the OPTION command, PARFIT returns with the following message indicating what control parameters the user can change

```
DESIGNATE CONTROL PARAMETER AND NEW VALUE (0 TO END)
OUTPUT-----------------------------------------1
INT. OUTPUT IN OPT. 0-YES,1-NO-----------------2
JACOBIAN 0-FULL 1-LEAST SQUARES----------------3
GEAR'S METHOD 0-ADAMS,1-STIFF------------------4
EPS FOR INTEGRATION----------------------------5
HMIN FOR INTEGRATION---------------------------6
HMAX FOR INTEGRATION---------------------------7
INTEGRATION PROCEDURE, 1-GEAR, 2-TRAPEZOIDAL-----8
```

The convention of using a 0 to end a sequence of input data of indeterminate length is used in several places in PARFIT.

Control parameter 1 governs the dumping of information during an integration. This option is useful during the debugging stages with a new model when the integration program encounters difficulties. This may happen for example if G_FUN is returning undefined values to the integration procedures. The default value of this control parameter is 0 in which case no output occurs. If this control parameter is set to the integer n, then after every n integration steps, the time and state variables are printed on -SC1.

Control parameter 2 indicates whether or not any output requested by a nonzero control parameter 1 is required when the integration procedure has been called by an optimization procedure. This control parameter is 0 if output is desired during an optimization and 1 otherwise. Its default value is 1. Control parameter 2 acts as a safety on control parameter 1 in that the user must specifically request integration output in an optimization run thus guarding against the chance of getting a large quantity of output by accident.

Control parameter 3 selects the information to be printed when output of the Jacobian matrix is requested. If this control parameter is 0 then the full Jacobian on all the state variables and at all the sample times is printed. If this

control parameter is 1 then only the entries of the full Jacobian that relate to observations on the state variables are printed. (That is the Jacobian J is printed.) The default value of control parameter 3 is 0. The full Jacobian is of value to the model builder because it can tell him (perhaps after further analysis) if and where he should take further observations to better determine his parameters. This is the case because the full Jacobian matrix is determined by solving the sensitivity equations and is thus independent of the observations.

Control parameter 4 indicates what option in Gear's integration procedure is to be employed. If this control parameter is 0 then integration of the initial value problem is done by Gear's implementation of an Adams' predictor corrector method[27]. If this control parameter is 1 then the integration is done by Gear's implementation of a multi-step method suitable for stiff problems[27]. The Adams' predictor-corrector method can be faster when stability is not a problem. The default value of control parameter 4 is 1.

Control parameter 5 contains the error criterion EPS to be used in gear's integration procedure. Its default value is .01. As this value is decreased, the user can expect his integrations to become more expensive. Among other things, the observation error should be considered when picking this control parameter. We comment that .01 is a very weak error criterion.

Control parameter 6 contains the minimum step size that integration procedures with stepsize control are allowed to use. Its default value is .00001.

Control parameter 7 contains the maximum step size that integration procedures with stepsize control are allowed to use. Its default value is 5.

Control parameter 8 indicates what integration procedure is to be employed. If it is 1, Gear's program is used. If it is 2, a trapezoidal method without error control is employed. The default value of control parameter 8 is 2.

(3) FREEZE

The FREEZE command allows the user to freeze selected parameters. These parameters then remain fixed until freezing is removed. There must be at least two active parameters for the optimization procedures to function properly. The freezing in the interactive option of the FIT command is at a higher level than the freezing indicated by the FREEZE command. When this command is issued, PARFIT asks the user to enter a list of subscripts of parameters to be frozen (0 to end). An entry of 0 removes all freezing.

(4) SCALE

The SCALE command allows the user to transform selected parameters. Currently logarithmic scaling and square root scaling are available. With logarithmic scaling for p_j, p_j is transformed according to $\hat{p}_j = \ln(p_j)$, and with square root scaling, p_j is transformed according to $\hat{p}_j = \sqrt{p_j}$. Thus with logarithmic scaling, p_j is repalced by $\exp(\hat{p}_j)$ in the model and with square root scaling p_j is repalced by $\hat{p}_j^2$. When this command is issued, PARFIT asks the user to enter pairs of integers indicating subscripts of parameters to be scaled and the scaling to be used. The integer 1 indicates logarithmic scaling and 2 indictes square root scaling. A 0 subscript terminates the entry of scaling instructions. An entry of 0 alone removes all scaling. When a parameter is scaled, or descaled, its current numerical value is automatically transformed.

(5) SAMPLE

This command allows the user to alter the set of sample times. As mentioned in the introduction, these are the times at which information is extracted from the continuous problem. The default sample times are the initial time and the observation times. When the user issues the SAMPLE command, a choice of three options is presented. The first option restores the sample times to their default values. The second option allows the user to specify a uniform mesh of sample times starting at the initial time by entering the number of sample times (not counting the initial time) and the sample time spacing. Of course an error results if more sample times than are allowed (as indicated in the data file) are requested. With this option, no connection to the observation times is maintained, and thus care must be used when this option is in effect. For example, the FIT command cannot be used when this option is in effect. The third option is for the interactive insertion of

sample times between existing sample times. The user is first asked if a listing of the existing sample times is desired. If it is desired, the user enters the subscript range (in the existing vector of sample times) where the listing is desired. To insert sample times between existing sample times, the user lists in sequence the upper index (in the vector of sample times) of the interval where the new times are to be inserted, and the number of points to insert. The points are inserted uniformly in the interval. The user can request insertion of times in several intervals. A 0 for an interval's upper index ends the input. This interactive insertion option can be of value when the sample times straddle a time interval where the differential equation solution warrants further investigation (for example, it might be taking a sudden jump.)

(6) WEIGHT

The WEIGHT command allows the user to interactively enter the weights for the observations. There are two options. First, the weights can all be set to their default value of one. Second, weights can be specified on selected observations on selected state variables which have been observed. Under the second option, the user is given the chance to take a permanent copy of the weights in the scratch file -WEIGHT.

(7) PLOT

The PLOT command allows the user to select various items to plot. When the command is issued, the following instructions appear

```
SEQUENCE OF ITEMS TO PLOT (END WITH 0)
STATE VARIABLES--------------------1
OBSERVATIONS-----------------------2
SMOOTHED OBSERVATIONS--------------3
GUESSED OBSERVATIONS---------------4
PHASE PLOT-------------------------5
```

Of course before item 1 can be selected, an integration must have been performed, and before item 3 can be selected, the SMOOTH command must have been used, and before item 4 can be selected, the CREOBS command must have been used. Item 5 applies only to 2 state variable problems. Before it can be used an integration must have been performed. After the desired items are selected, the user is given further choice. For example if items 1 and 2 are selected, the user is asked which state variables and which observations (i.e. on which state variables) he wants plotted. This flexibility allows the user

to isolate various aspects of the problem.

After the user has described the desired plot, a mini-print-plot appears at his terminal. The abscissae for the plotted points are the sample times.

After the plot is completed, the user is asked if a permanent record is required. If the answer is y for yes, a large scale version of the print plot is accumulated in the file -GRAPHSTORE for later output. A plot number (starting at 1) is associated with each graph accumulated and when a plot is accumulated, a message indication the current parameter values and the current plot number is written on the file -REPRT. The interface to the plotting programs is confined to one procedure in PARFIT and thus it is easy to modify the plotting facilities of PARFIT and the plotting hardware employed can easily be altered.

(8) REPORT

This command controls a set of output procedures with which PARFIT can display various information to the user. When this command is issued, PARFIT returns with the following message indicating what items the user can have printed.

```
ENTER LIST OF ITEMS TO PRINT
(END WITH 0)
GENERAL DATA -------1
OBS----------------2
PAR----------------3
CREATION-----------4
PTS AND STATE VARS--5
SMOOTHING DATA-----6
JACOBIAN-----------7
INTEG./OPT. DETAILS-8
STATISTICAL DATA----9
OPTION SETTINGS----10
WEIGHTS------------11
GUESSED OBS--------12
```

new items can easily be added to this list. The requested information is displayed at the terminal and also put in the file -REPRT so that a permanent record can be taken at the end

of a run with PARFIT.

When item 1 is selected, PARFIT displays the following basic information on the particular problem under consideration

(a) the number of parameters
(b) the number of state variables
(c) the number of state variables observed
(d) a list of state variables observed
(e) the number of observation times

Selection of item 2 causes a list of observations along with observation times to be printed.

Selection of item 3 causes the starting parameter values to be printed (that is the values read from the data file) along with the current parameter values and the freezing and scaling status of each parameter.

Report item 4 prints information for the special situation when test observations have been created by a simulation run. The parameter values for the simulation run are printed along with the standard deviation of the random error introduced into the generated observations.

Report item 5 prints the parameter values used in the last integration of the initial value problem along with the sample times and integration results at these sample times.

Report item 6 prints smoothing information generated by the SMOOTH command. First the type of smoothing used--either least squares cubic spline or least squares cubic Hermite is indicated. Next the joints used for the piecewise polynomial smoothing function are displayed and finally the smoothed observations and the smoothed derivatives at the sample times are listed.

Report item 7 prints the parameter values used in the last integration along with the jacobian matrix. Whether or not a full Jacobian matrix is printed depends on control parameter 3 which can be altered in the option conmmand.

Report item 8 prints optimization and or integration data. When an automatic optimization run is made, detailed information on the run (eg. sums of squares of residuals, parameter values etc.) is written on the scratch file -SC1. Depending on the settings of control parameters 1 and 2 (see the option command), integration information may be written on -SC1. When the user selects item 8 and when PARFIT is running interactively, the contents of -SC1 are displayed at the terminal. Since a lot of relatively useless output may be present in -SC1, the user is asked if the contents of -SC1 are to be accumulated for later output. If a later hard copy is desired, the contents of -SC1 are accumulated in -SC2. The file -SC1 is emptied before it is next required to accept output from PARFIT. An output reference number is attached to each use of -SC1. When an accumulation is made to -SC2, a message to this effect along with the current output reference number is written on -REPRT and the output reference number is incremented by 1. This allows a coordinated interpretation of the output from PARFIT.

Report item 9 prints statistical data resulting from the STATS command. The F distribution value used in determining confidence intervals on the parameters is displayed along with its corresponding percentile value and number of degrees of freedom in the numerator and number of degrees of freedom in the denominator. The sum of the squares of the residuals is printed and the parameter values along with their confidence intervals are printed. Finally, the correlation and covariance matrices are printed.

Report item 10 prints the current control parameter settings along with the default values for the control parameters.

Report item 11 prints the weights given to the observations.

Report item 12 prints the guessed observations and corresponding derivatives generated by the CREOBS command.

(9) SET

This command allows the user to redefine an element of the parameter vactor p by entering the integer subscript of the element and the new value of the element. This can be repeated for as many elements as desired. Entry of a 0 for a subscript terminates the command. For example to set the first and fourth

parameters to 3. and 6.2 respectively, the user would enter

1 3. 4 6.2 0

(The above set of numbers need not all be on one line.) Parameter subscripts refer to the full parameter vector and no allowance for frozen parameters is required. Scaling is ignored by the SET command.

(10) INTEG

This command requests that the initial value problem be numerically integrated using the current parameter values. The integration technique used is determined by control parameter's 4 and 8 which can be reset by the OPTION command. The integration results at the sample times are stored in an array for later use. For example these values may be plotted at some later time in an interactive session. When this command is issued, the user is asked if the Jacobian is desired, and if it is the sensitivity equations are integrated along with the given model equations and the full Jacobian is stored for later analysis. (for example by the PROBE command). If the integration procedure runs into difficulty, an error message is printed and the user is returned to the command section of PARFIT. A typical difficulty with the integration procedure is that it cannot meet the error criterion with the current minimum step size.

(11) PROBE

This command allows the user to investigate, among other things, the condition at the current point p in parameter space of the Jacobian matrix associated with the least squares problem. The user is asked if an integration is required to determine the Jacobian at the current parameter values. (it may not be if for example INTEG with a Jacobian option has just been executed.) A singular value decomposition is done on the Jacobian for the least squares problem. The user is given the option of taking a permanent record of the PROBE results in the file -REPRT. Furthermore, if control parameter 1 is set to request output in an integration, the user is given the chance to view the output and accumulate it in -SC2. Since the singular value decomposition is available when the PROBE command has been executed, the potential exists for adding a procedure here to further analyse the problem at the current point in parameter space.

(12) FIT

The FIT command puts the user in control of optimization procedures which apply directly to the least squares minimization problem mentioned in the introduction (as opposed to the DFIT command for example). Currently the user has two main choices with the FIT command. The first option uses the Levenberg-Marquardt algorithm. From a starting guess, this algorithm attempts to determine the optimum parameters automatically without user intervention. For effective use of this procedure, the initial guess at the optimal parameter vector should be fairly good. The user is asked to supply a starting value for the parameter λ used in the Levenberg-Marquardt algorithm as well as error tolerances for termination of the automatic optimization run. A negative value for λ tells PARFIT to pick its own starting value for λ. At times λ must be adjusted upward interactively to avoid points in parameter space where the differential equation cannot be integrated. The termination criterion take the form of a realtive (e_1) and an absolute (e_2) error tolerence. Termination of the optimization run occurs when either

$$F_q - F_{q+1} < e_1 F_{q+1} + e_2 \quad \text{or}$$

$$F_{q+1} < e_2$$

where F_q is the sum of the squares of the residuals on the q'th iteration. Of course computation costs increase as e_1 and e_2 are decreased and for fine tolerences a batch run of PARFIT is probably advisable. Choice of e_1 and e_2 should of course depend on the accuracy of the observations and on the error criterion chosen for the numerical integration technique.

Our implementation of the Levenberg-Marquardt algorithm has a provision for automatic scaling so that the diagonal elements of J^TJ are all 1. To request scaling, e_1 and e_2 should both be negative. PARFIT uses their absolute values for the termination criteria when scaling is requested. The use of scaling can dramatically speed up convergence.

The second option under the FIT command provides an interactive optimization approach where the user has extensive control over PARFIT through a set of optimization commands. Among other things, the user can reset parameters, freeze selected parameters, and plot graphs. Since it is anticipated that these commands will be used very frequently, the descriptive messages from PARFIT are kept to a minimum and the command designators are very short. A description of the

currently implemented optimization commands follows.

(a) T

This is the technique command where the user can choose either the Levenberg-Marquardt or the steepest descent optimization method. To request the steepest descent technique the user should enter

T SD

and then hit the return. One or more blanks must separate the T and the SD. To request the Levenberg-Marquardt technique, the user should enter

T MARQ

The default is the Levenberg-Marquardt technique.

(b) M

By entering

M r

where r is a real number, the current value of λ for the Levenberg-Marquardt technique or step length for the steepest descent technique is multiplied by r and this product replaces the current λ or step length. A new iteration of the current optimization technique is then attempted. PARFIT then reports on the success of this attempt and the user is asked to enter a new optimization command. The default value of λ is .01. There is no default value for the step length for the steepest descent method. The user should pick a starting value for the step length after observing the gradient.

(c) N

By entering

N r

where r is a real number, the current value of λ or step length (depending on the technique) is replaced by r.

(d) V

This is the view command. It does not take any arguments. It requests a display of the current parameters, and current gradient components. Frozen parameters are indicated--see command (e). The current technique is also displayed along with basic information associated with this technique. Finally, the current sum of squares is displayed.

(e) F

This is the freeze command. This command acts at a higher level than the main FREEZE command. It allows the user to selectively freeze various parameters at their current values and to continue the optimization on a subspace of parameter space. A 0 is used to terminate the list of parameters to be frozen. For example, to freeze the first and third parameters, the user would enter

F 1 3 0

Currently, no special programming is implemented to take advantage of the fewer sensitivity equations present when we are working on a subspace.

(f) DF

This is the defreeze command. It removes all or some of the parameters from the list of frozen parameters. This command does not influence freezing set by the FREEZE command. To remove all parameters from the list of frozen parameters enter

DF 0

To remove freezing on say the third and fourth parameters enter

DF 3 4 0

(g) SET

This is the SET command and it is identical, except for the printing of guiding instructions, to the main SET

command.

(h) PLOT

This command is identical to the main PLOT command. The state variables plotted are those from the last integration. Thus if a plot is requested after an unsuccessful Marquardt iteration attempt, the state variables plotted are those at the last set of trial parameters.

(i) Q

This is the quit command for the interactive optimization subsection. It returns control to the main command section of PARFIT.

A detailed record of all commands and all command results (including mini-print-plots) that occur in an interactive optimization session is kept in the file -AECHO. The user has the option of taking a hard copy of this file at the end of a run with PARFIT.

(13) STATS

This command requests PARFIT to produce certain statistical information on the parameters after they have been optimized by the FIT command. An assumption of linearity in the parameters near the optimum is made. The user is requested to enter the percentile for the confidence intervals. The program then finds the reqired value of the F distribution with the appropriate degrees of freedom. The confidence intervals on the parameters and the correlation and covariance matrices are calculated when the STATS command is issued.

(14) QUIT

This command terminates a run with PARFIT. Before execution is terminated, the user is given the chance to take a permanent copy of some or all of the information accumulated

during the interactive session. In batch mode a permanent copy of the accumulated plots is automatically taken.

(15) SUSP

This command suspends execution of PARFIT and returns the user to MTS. The MTS command $RESTART causes the execution of PARFIT to be resumed. One use of this might be to examine various scratch files PARFIT has created.

(16) CREATE

This command allows the user to make a simulation run with his model and to generate a data file from this simulation run. A primary purpose of this facility is in the debugging of new procedures in PARFIT. The user is asked for the parameter values for the simulation run, the observation times, the state variables observed, and the error tolerence, maximum step size, and minimum step size for Gear's integration program. (If the integration run does not succeed, its constraints can be reset and a new run can be made.) The user is further asked for the standard deviation of the random error in the generated observations and the vector of starting values for the parameters. Finally the user can have the generated data put in the file -DATA. When the create command is to be used, the data file should have a model name of CREATE in it and of course only the first three data cards described in section 2 are required when the CREATE command is to be used.

(17) SMOOTH

The SMOOTH command allows the user to fit a least squares piecewise polynomial to the observations on each state variable observed. This is normally used in preparation for the DFIT IFIT and CONTIN commands. The user has the choice of using either a piecewise cubic spline or a piecewise cubic Hermite polynomial--the latter should be used if the observations take any sudden violent jumps. In either case, PARFIT sequentially goes through the state variables on which observations have been taken. On each state variable, the user is asked to enter the number of joints and joint positions for the indicated set of observations. A maximum of 15 joints for each piecewise polynomial is allowed.

It should be very easy to make this curve fitting aspect of PARFIT more interactive, however for the present, the user must issue SMOOTH snd PLOT commands alternatively when doing

interactive curve fitting.

(18) CREOBS

This command allows the user to guess observations on the unobserved state variable in the two state variable case. This is in preparation for options 2 and 3 of the DFIT command and options 2, 3, and 4 of the IFIT command. The user is asked to enter the number of abscissae and the abscissae for an interpolating cubic spline to approximate the guessed observations. The first and last sample times must begin and end the list of abscissae. Next the user is asked to enter the corresponding ordinates. At each end, the interpolating cubic spline matches the slope of the line joining the two points of interpolation closest to the given end.

(19) LINEAR

The LINEAR command allows the user to specify that the DFIT and IFIT least squares problems are linear and thus starting parameters are not required since no iterations are required to obtain the optimal parameters. Currently this facility is implemented in options 2 and 3 of the DFIT command and in options 2 3 and 4 of the IFIT command.

(20) DFIT

This command has three options. The first option uses the results of the SMOOTH command to estimate optimal parameters by a rather coarse, but at times inexpensive technique. Furthermore, this procedure can be useful when our initial parameter values correspond to an unstable initial value problem. There are however some restrictions on the class of problems the DFIT command can handle. It cannot handle problems where some parameters occur only in the initial conditions, and for this version of the DFIT command, all parameters must occur in the subset of the differential equations defining the initial value problem that correspond to state variables on which observations have been taken. This program works by applying an automatic Levenberg-Marquardt procedure to the nonlinear curve fitting problem

$$s'=g(t,s,p)$$

where $s_i(t)$, the i'th component of the vector s(t), is a piecewise polynomial approximation to the observations on y_i. The nonlinear curve fitting is done in a least squares sense at the sample times. No weighting is employed at present. If not

all the state variables are observed, then at each iteration of the Levenberg-Marquardt procedure, a subset of the set of equations in our initial value problem is integrated (using s(t), the current parameter vector and the integration technique indicated by control parameter's 2 and 8). Thus this technique can become expensive too. Also instabilities at the starting parameter values may arize in the subsystem initial value problem. Furthermore, especially if the observations are far apart and have large errors, the parameters determined by this technique are not very reliable. However, they can serve as starting values for the FIT command. As in the automatic option of FIT command, the user is asked to enter a starting λ along with a relative and an absloute error tolerence for the Levenberg-Marquardt procedure.

The second option under the DFIT command currently applies only to the important special case when only two state variables are present and observations are available on only one of them. This option assumes the behavior of the unobserved state variable has been approximated using the CREOBS command. The observations on the other state variable must be smoothed with the SMOOTH command prior to using this option. This option of the DFIT command then fits derivatives using the smoothed and guessed observations. Only the nonfrozen parameters enter into the optimization. If a parameter occurs only in an initial condition, it must be frozen prior to the use of this command. If the least squares problem is linear, (as indicated by the LINEAR command), then a linear least squares technique using the singular value decomposition is employed. If the problem is nonlinear, then the Levenberg-Marquardt algorithm is employed and a starting lambda and relative and absolute error tolerances must be supplied.

The third option under the DFIT command provides a means of iteratively improving the guessed observations with the aid of a sparse Gauss-Newton procedure. This option currently applies only to the two state variable case. Starting with guessed observations on the unobserved state variable, PARFIT attempts to find an optimal parameter vector p and corresponding guessed observations $c=(c_0,\ldots,c_N)^T$, where c_ℓ is the guessed observation at time t_ℓ, to minimize

$$\sum_{\ell=0}^{N} (g_1(t_\ell,(s^\ell,c_\ell)^T,p)-s^{\ell\prime})^2 + \sum_{\ell=1}^{N} (d_\ell(c))^2$$

where for notational convenience we have assumed y_2 is unobserved, where $s^\ell=s(t_\ell)$ (a superscript is used to avoid confusion with our notation when s(t) is a vector), and where $d_\ell(c)$ represents a discretization of the state equation corresponding to the unobserved state variable. Currently

PARFIT uses the trapezoidal discretization for $d_\ell(c)$. That is

$$d_\ell(c)=.5(g_2(t_{\ell-1},(s^{\ell-1},c_{\ell-1})^T,p)+g_2(t_\ell,(s^\ell,c_\ell)^T,p))$$

$$-\frac{c_\ell-c_{\ell-1}}{t_\ell-t_{\ell-1}}.$$

PARFIT employs a nonlinear block Gauss-Seidel technique to find the optimal p and c. That is PARFIT starts by fixing c and finding the optimal parameter vector to minimize the above sum of squares. Then with p fixed at this optimum, an optimal c is determined to munimize the above sum of squares. This latter optimization is accomplished through a Gauss-Newton procedure which takes advantage of the particular sparsity structure of the problem. The user is requested to enter a tolerance e to define the stopping criterion for the iterative determination of c. The iteration terminates when two successive iterates $c^{(\ell)}$ and $c^{(\ell+1)}$ satisify

$$|c_\ell^{(\ell+1)}-c_\ell^{(\ell)}|<e(|c_\ell^{(\ell+1)}|+.001)$$

for $l=0,\ldots,N$. There is an option to fix the initial conditions of the guessed observations. PARFIT has facilities for handling the case when the estimation of p for a fixed c is a linear problem. When this problem is not linear, the user is requested to enter a starting lambda and relative and absolute error tolerances for the determination of the optimal p by the Levenberg-Marguardt technique. After an optimal c is found, the user is given the chance to further refine this vector by decreasing the tolerance e. Finally, after an iteration (the determination of a new p and a new c) the user is given the option of doing another iteration or terminating the iterative process. If the process is terminated, it may be restarted by issuing the same DFIT command that initiated the iterative process.

(21) IFIT

This command is similar to the DFIT command except here integrals are used instead of derivatives. Currently there are four options available under this command. The first option requires observations on all state variables and it determines parameters which minimize

$$f^T(p)f(p)$$

where

$$f_{n(\ell-1)+i} = y_{o,i}(p) + \int_{t_o}^{t_\ell} g_i(t,s(t),p)\,dt - s_i(t)$$

where $\ell=1,\ldots,k$; $i=1,\ldots,n$; $y_{o,i}$ is the i'th component of the initial condition vector and g_i is the i'th component of $g(t,s,p)$, and $s(t)$ is the smoothing function determined by the SMOOTH command. Currently there is no provision for the linear case and the Levenberg-Marquardt method is used to obtain the optimal parameters.

The second option under the IFIT command corresponds to the second option of the DFIT command. This option fits integrals to the smoothed observations and to the guessed observations. There are special facilities for handling the linear case. Currently this option of the IFIT command applies only to the two state variable case.

The third option under the IFIT command attempts to iteratively improve the guessed observations by an experimentally effective, but, occasionally, unstable technique. The iteration proceeds by first applying the method used in option two of the IFIT command to estimate the parameters, and then, holding the observed state variable equal to $s(t)$, a new set of guessed observations is generated by integrating the state equation corresponding to the unobserved state variable. Currently this option applies only to the two state variable case. As with the third option of the DFIT command, after an iteration is complete, the user is given the chance to terminate the iterative process.

The fourth option under the IFIT command is very similar to the third option under the DFIT command. The only difference is that the parameter vector is updated using the procedure employed in the second option of the IFIT command. Provisions are available for linear parameter estimation problems.

(22) CONTIN

The CONTIN command provides the user with a technique employing continuation methods and break points which is designed to bridge the gap between the coarse integral fitting method and the full least squares problem. A direct continuation method is available where the user can fit the solution of

$$u' = g(t,((1-\gamma)s(t)+\gamma u,p)$$

$$u(t_o) = y_o(p)$$

to s(t) in the least squares sense at the observation times where $0 \leq \gamma \leq 1$ and where s(t) represents a smoothing of the observations. The user can experiment with various values for the continuation parameter γ. When $\gamma=0$, we have the first option of the IFIT command, and when $\gamma=1$, we have the full least squares problem (on the smoothed observations). Observations must be available on all state variables to use this strategy in the CONTIN command. Another facility available under this command involves the use of break points. The user can specify break points at times

$$T_1 < T_2 < \ldots < T_g$$

corresponding to observation times

$$t_{i_1}, t_{i_2}, \ldots, t_{i_g}$$

The user can also specify a continuation parameter vector α for the break points. As we integrate the initial value problem through the break point at time T_i, u is reset according to

$$u(T_i) = Au^-(T_i) + (I-A)s(T_i)$$

where $A=\text{diag}(\alpha_1, \ldots, \alpha_m)$, and $u^-(T_i)$ is the result obtained by integrating up to time T_i. The user also has the chance to weight the break points with weights $w_1, \ldots, w_g$. Finally, when break points alone are employed ($\gamma=1$), observations need not be available on all state variables. The components of α corresponding to unobserved state variables should be set equal to 1. This command has no implementation restrictions on the number of state variables.

APPENDIX B

SELECTED PROGRAM LISTINGS

In this appendix we give selected listings from the code that defines PARFIT. PARFIT is coded in ALGOL W, and thus the listings given below should be fairly easy to read. In the interests of brevity and clarity, we have replaced all input/output statements with descriptive pseudo-statements indicated by %INPUT and %OUTPUT. In these statements, actual variables in the ALGOL W code are enclosed in brackets. PARFIT performs a major portion of its input and output using FORTRAN subroutines. This was done mainly to take advantage of the good "user proof" set of FORTRAN callable free format input procedures available at the University of British Columbia. External procedures are indicated by %EXTERNAL in the following listings. The last few pages of listings are devoted to the declaration statements for the external procedures not previously listed. Brief descriptions of the functions of these procedures are also included. Occasionally, PARFIT performs operations on MTS (Michigan Terminal System) files. These operations are indicated with %FILE in the following listings.

```
%%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

            Main driving program for PARFIT

%%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

BEGIN
COMMENT MAIN  DRIVING PROGRAM FOR PARFIT
START BY SETTING UP SIZE INFORMATION;
INTEGER N_STATE,N_PAR,N_STATE_OBS,N_OBS,MAX_PTS;
STRING(31) MODEL;
STRING(1) ANS;
LOGICAL BATCH;
LONG REAL INITIAL_TIME;
COMMENT %EXTERNAL CMD_AL, CHECK_BATCH;
COMMENT %EXTERNAL GEAR, TRAP, SVD_AL,
MARQUARDT, PARFIT;
COMMENT EXECUTION BEGINS HERE
##########################################
##########################################;
COMMENT ASSIGN UNIT NUMBERS FOR OUTPUT;
COMMENT EMPTY TEMPORARY FILES;
COMMENT DETERMINE IF IN BATCH MODE;
CHECK_BATCH(BATCH);
COMMENT IF IN BATCH OPTIMIZATION/INTEGRATION
INFORMATION IS WRITTEN OUT DIRECTLY
BY ASSIGNING SAME UNIT NUMBER FOR -SC1 AND OUTPUT TO USER;
COMMENT %INPUT FROM DATA FILE
(MODEL),(INITIAL_TIME),(N_STATE),(N_PAR),(N_STATE_OBS),
(N_OBS),(MAX_PTS);
PARFIT(N_STATE,N_PAR,N_STATE_OBS,N_OBS,MAX_PTS,MODEL,
INITIAL_TIME,BATCH,CHECK_BATCH,GEAR,TRAP,SVD_AL,
MARQUARDT,CMD_AL);
IF BATCH=TRUE THEN
COMMENT %FILE TAKE COPY OF -GRAPHSTORE;
ELSE
  BEGIN
  COMMENT %OUTPUT TO USER
  IS A LISTING OF FULL ECHO DESIRED? Y OR N;
  COMMENT %INPUT (ANS);
  IF ANS="Y" THEN
  COMMENT %FILE TAKE COPY OF -ECHO;
  COMMENT %OUTPUT TO USER
  IS A COPY OF FULL PLOTS AND REPORTS DESIRED? Y OR N;
  COMMENT %INPUT (ANS);
  IF ANS="Y" THEN
  COMMENT %FILE TAKE COPY OF -REPRT, -GRAPHSTORE;
  COMMENT %OUTPUT TO USER
  IS A COPY OF INTEG./OPT. AND INTERACTIVE FIT
  DESIRED Y OR N;
  COMMENT %INPUT (ANS);
```

```
    IF ANS="Y" THEN
    COMMENT %FILE TAKE COPY OF -SC2, -AECHO;
    END;
  END.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                Major procedure defining PARFIT

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE PARFIT(INTEGER VALUE N_STATE,FN_PAR,
N_STATE_OBS,N_OBS,MAX_PTS;
STRING(31) VALUE MODEL;
LONG REAL VALUE INITIAL_TIME;
LOGICAL VALUE BATCH;
PROCEDURE
 CHECK_BATCH, GEAR, TRAP, SVD_AL, MARQUARDT, CMD_AL);
  BEGIN
  COMMENT MAJOR  PROCEDURE IN PARFIT
  ;
  PROCEDURE EG_FUN(LONG REAL VALUE T;
  LONG REAL ARRAY Y(*);
  LONG REAL ARRAY P(*);
  INTEGER VALUE OPTION;
  LONG REAL ARRAY G(*);
  LONG REAL ARRAY DGY(*,*);
  LONG REAL ARRAY DGP(*,*);
  LONG REAL ARRAY ISEN(*,*));
    BEGIN
    COMMENT INTERFACE TO USER DEFINED G_FUN TO ALLOW
    USE OF    ENVIRONMENT    FOR
    SCALING AND FREEZING OF PARAMETERS;
    COMMENT %EXTERNAL G_FUN;
    INTEGER KK;
    LONG REAL ARRAY PS(1::FN_PAR);
    LONG REAL ARRAY TDGP,TISEN(1::N_STATE,1::FN_PAR);
    IF PFRZ=TRUE THEN
      BEGIN
      COMMENT COPY NONFROZEN PARAMETERS AND
      FROZEN PARAMETERS TO PS;
      KK:=0;
      FOR I:=1 UNTIL FN_PAR DO
      IF FRZ(I)=0 THEN
        BEGIN
        KK:=KK+1;
        PS(I):=P(KK);
        END
      ELSE
      PS(I):=FPAR(I);
```

```
  END
ELSE
FOR I:=1 UNTIL FN_PAR DO PS(I):=P(I);
IF PSCL=TRUE THEN
COMMENT SCALING PRESENT;
FOR I:=1 UNTIL FN_PAR DO
IF SCL(I)=1 THEN
PS(I):=LONGEXP(PS(I))
ELSE
IF SCL(I)=2 THEN
PS(I):=PS(I)**2;
G_FUN(T,Y,PS,OPTION,G,DGY,TDGP,TISEN);
CASE ABS OPTION OF
  BEGIN
    BEGIN
    END;
    BEGIN
    IF PSCL=TRUE OR PFRZ=TRUE THEN
      BEGIN
      COMMENT HANDLE FREEZING AND SCALING;
      KK:=0;
      FOR I:=1 UNTIL FN_PAR DO
      IF FRZ(I)=0 THEN
        BEGIN
        KK:=KK+1;
        IF SCL(I)=1 THEN
          BEGIN
          FOR J:=1 UNTIL N_STATE DO
          DGP(J,KK):=TDGP(J,I)*PS(I);
          END
        ELSE
        IF SCL(I)=2 THEN
          BEGIN
          FOR J:=1 UNTIL N_STATE DO
          DGP(J,KK):=TDGP(J,I)*2.L*P(KK)
          END
        ELSE
        FOR J:=1 UNTIL N_STATE DO
        DGP(J,KK):=TDGP(J,I);
        END;
      END
    ELSE
    FOR I:=1 UNTIL N_STATE DO
    FOR J:=1 UNTIL FN_PAR DO
    DGP(I,J):=TDGP(I,J);
    END;
    BEGIN
    IF OPTION>0 THEN
    IF PSCL=TRUE OR PFRZ=TRUE THEN
      BEGIN
      COMMENT HANDLE FREEZING AND SCALING;
```

```
        KK:=0;
        FOR I:=1 UNTIL FN_PAR DO
        IF FRZ(I)=0 THEN
          BEGIN
          KK:=KK+1;
          IF SCL(I)=1 THEN
            BEGIN
            FOR J:=1 UNTIL N_STATE DO
            ISEN(J,KK):=TISEN(J,I)*PS(I);
            END
          ELSE
          IF SCL(I)=2 THEN
            BEGIN
            FOR J:=1 UNTIL N_STATE DO
            ISEN(J,KK):=TISEN(J,I)*2.L*P(KK);
            END
          ELSE
          FOR J:=1 UNTIL N_STATE DO
          ISEN(J,KK):=TISEN(J,I);
          END;
        END
      ELSE
      FOR I:=1 UNTIL N_STATE DO
      FOR J:=1 UNTIL FN_PAR DO
      ISEN(I,J):=TISEN(I,J);
      END
    END;
  END EG_FUN;
COMMENT
;
PROCEDURE EXTRACT_JACOBIAN(LONG REAL ARRAY JAC(*,*));
  BEGIN
  COMMENT FROM A COMPLETE JACOBIAN AT ALL POINTS HIT AND ON
  ALL STATE VARIABLES, EXTRACT INTO JAC THE JACOBIAN
  DIRECTLY ASSOCIATED WITH THE OBSERVATIONS TAKEN,
  AND INCORPORATING THE WEIGHTS ON THE OBSERVATIONS;
  INTEGER M;
  COMMENT SET UP WEIGHTING SCALING OF LEAST SQUARES JAC;
  FOR J:=1 UNTIL N_STATE_OBS DO
  FOR I:=1 UNTIL N_OBS DO
  WT_VEC((I-1)*N_STATE_OBS+J):=WT(I,J);
  FOR K:=1 UNTIL N_PAR DO
    BEGIN
    M:=0;
    FOR I:=1 UNTIL N_PTS_HIT DO
    IF OBS_STATUS(I)~=0 THEN
    FOR J:=1 UNTIL N_STATE_OBS DO
      BEGIN
      M:=M+1;
      JAC(M,K):=JACOBIAN((I-1)*N_STATE+STATES_OBS(J),K)
      *WT_VEC(M);
```

```
      END;
    END;
  END EXTRACT_JACOBIAN;
COMMENT
;
PROCEDURE STANDARD_HIT( LONG REAL VALUE INITIAL_TIME;
INTEGER ARRAY OBS_STATUS(*);
LONG REAL ARRAY OBS_PLACE(*);
INTEGER VALUE N_OBS;
LONG REAL ARRAY PTS_TO_HIT(*);
INTEGER RESULT N_PTS_HIT);
  BEGIN
  COMMENT SET SAMPLE TIMES AS INITIAL TIME AND
  OBSERVATION TIMES;
  I:=1;
  PTS_TO_HIT(1):=INITIAL_TIME;
  OBS_STATUS(1):=0;
  IF ABS(OBS_PLACE(1)-INITIAL_TIME)<1.'-5 THEN
    BEGIN
    OBS_STATUS(1):=1;
    I:=0;
    END;
  FOR J:=2-I UNTIL N_OBS DO
    BEGIN
    PTS_TO_HIT(J+I):=OBS_PLACE(J);
    OBS_STATUS(J+I):=J;
    END;
  N_PTS_HIT:=N_OBS+I;
  END STANDARD_HIT;
COMMENT
;
PROCEDURE COPY_TRANS(LONG REAL ARRAY A(*,*);
LONG REAL ARRAY B(*,*);
LONG REAL ARRAY JA(*,*);
LONG REAL ARRAY JB(*,*);
INTEGER VALUE M,N,JM,JN,AUX_INT);
  BEGIN
  COMMENT TO REDUCE PAGING, INTEGRATION
  PROCEDURES INSERT DATA BY
  COLUMNS IN NXM ARRAY B,THEN B(TRANSPOSE) IS COPIED TO A
  TO ACCESS A BY COLUMNS FOR PLOTTING ETC.;
  FOR J:=1 UNTIL N DO
  FOR I:=1 UNTIL M DO
  A(I,J):=B(J,I);
  IF AUX_INT=1 THEN
  FOR J:=1 UNTIL JN DO
  FOR I:=1 UNTIL JM DO
  JA(I,J):=JB(J,I);
  END COPY_TRANS;
COMMENT
;
```

```
COMMENT %EXTERNAL SPLINT_AL, SPLN_AL, DSPLN_AL,
HERMIT_AL, HERM, DHERM, ECHO1;
PROCEDURE READ_CMD_DATA(INTEGER ARRAY CMD_DATA(*));
  BEGIN
  COMMENT READ A STRING OF INTEGERS SEPARATED
  BY BLANKS UNTIL A
  ZERO IS ENCOUNTERED, STORE THE INTEGERS INCLUDING THE ZERO
  IN THE VECTOR CMD_DATA;
  INTEGER I;
  I:=1;
  COMMENT %INPUT (CMD_DATA(I));
  WHILE CMD_DATA(I)~=0 DO
    BEGIN
    I:=I+1;
    COMMENT %INPUT (CMD_DATA(I));
    END;
  END READ_CMD_DATA;
COMMENT
;
COMMENT %EXTERNAL PLOT_COMMAND;
COMMENT VARIABLE DECLARATIONS FOLLOW
;
INTEGER ARRAY SCL,FRZ(1::FN_PAR);
LONG REAL ARRAY CREOBS_JOINTS(1::15);
LONG REAL ARRAY CREOBS_VAL(1::15);
LONG REAL ARRAY CREOBS(1::MAX_PTS);
LONG REAL ARRAY DCREOBS(1::MAX_PTS);
LONG REAL ARRAY FPAR,START_PAR,SIM_PAR,
INT_PAR,GRADIENT,PAR(1::FN_PAR);
LONG REAL ARRAY OBS_PLACE,RESIDUAL_VECTOR(1::N_OBS);
LONG REAL ARRAY PTS_TO_HIT(1::MAX_PTS);
INTEGER ARRAY STATES_OBS(1::N_STATE_OBS);
LONG REAL ARRAY OBS(1::N_OBS,1::N_STATE_OBS);
LONG REAL ARRAY WT(1::N_OBS,1::N_STATE_OBS);
REAL ARRAY WT_VEC(1::N_OBS*N_STATE_OBS);
LONG REAL ARRAY OBS_SMOOTH(1::MAX_PTS,1::N_STATE_OBS);
LONG REAL ARRAY JOINTS(1::17,1::N_STATE_OBS);
LONG REAL ARRAY SPLN_COEF(1::30,1::N_STATE_OBS);
LONG REAL ARRAY HERM_COEF(1::16,1::4,1::N_STATE_OBS);
INTEGER ARRAY NJOINTS,N_SPLN_PAR(1::N_STATE_OBS);
LONG REAL ARRAY STATE(1::MAX_PTS,1::N_STATE);
LONG REAL ARRAY JACOBIAN(1::MAX_PTS*N_STATE,1::FN_PAR);
INTEGER DFIT_LIN,NO_CREOBS_JOINTS,CMD_NO,I,N_PTS_HIT;
INTEGER ARRAY OBS_STATUS(1::MAX_PTS);
LONG REAL ARRAY CON(1::FN_PAR);
LONG REAL ARRAY COR,COV(1::FN_PAR,1::FN_PAR);
LONG REAL STD_DEV;
STRING(6) ARRAY COMMANDS(1::30);
STRING(6) CMD;STRING(6) UNIT;
LONG REAL FPROB;INTEGER SMF,WT_FLG;
LONG REAL EPS,HMIN,HMAX;
```

```
INTEGER PLOT_NUMBER,PERCENTILE,AUX_INT,OUTPUT,
KFLAG,METHOD_FLAG,OUTPUT_SUP,OUT_SEG,JAC_OPTION,
INT_PROC,N_PAR,JJ;
STRING(1) ANS;
LOGICAL PFRZ,PSCL,ECHO;
LONG REAL SUM_SQ_RES;
COMMENT
##################################################
################INITIALIZATION####################
##################################################;
ECHO:=FALSE;
OUTPUT:=0;
DFIT_LIN:=0;
INT_PROC:=2;
PLOT_NUMBER:=0;
EPS:=.01;
HMIN:=.00001;
HMAX:=5;
METHOD_FLAG:=1;
OUTPUT_SUP:=1;
OUT_SEG:=0;
JAC_OPTION:=0;
PFRZ:=FALSE;PSCL:=FALSE;
FOR I:=1 UNTIL FN_PAR DO
FRZ(I):=SCL(I):=0;
N_PAR:=FN_PAR;
COMMENT
DEFINE VECTOR OF COMMAND DESIGNATORS;
COMMANDS(1):="PLOT   ";
COMMANDS(2):="SET    ";
COMMANDS(3):="INTEG  ";
COMMANDS(4):="FIT    ";
COMMANDS(5):="QUIT   ";
COMMANDS(6):="REPORT";
COMMANDS(7):="CREATE";
COMMANDS(8):="SAMPLE";
COMMANDS(9):="SMOOTH";
COMMANDS(10):="DFIT   ";
COMMANDS(11):="SUSP   ";
COMMANDS(12):="OPTION";
COMMANDS(13):="STATS  ";
COMMANDS(14):="PROBE  ";
COMMANDS(15):="WEIGHT";
COMMANDS(16):="CONTIN";
COMMANDS(17):="CREOBS";
COMMANDS(18):="IFIT   ";
COMMANDS(19):="FREEZE";
COMMANDS(20):="SCALE  ";
COMMANDS(21):="ECHO   ";
COMMANDS(22):="LINEAR";
COMMANDS(23):="END    ";
```

```
COMMENT
;
COMMENT %OUTPUT TO USER -REPRT (MODEL);
IF MODEL(0|6) ~= "CREATE" THEN
  BEGIN
  COMMENT READ IN DATA;
  COMMENT %INPUT FROM DATA FILE
  (OBS),(OBS_PLACE),(START_PAR),(STATES_OBS),(WT),
  (WT_FLAG)--0 IF NO WEIGHTS IN DATA FILE;
  COMMENT SET ALL WEIGHTS TO 1 IF NOT PRESENT IN DATA FILE;
  IF WT_FLG=0 THEN
  FOR J:=1 UNTIL N_STATE_OBS DO
  FOR I:=1 UNTIL N_OBS DO
  WT(I,J):=1;
  COMMENT INITIALIZE PAR TO START_PAR;
  FOR I:=1 UNTIL FN_PAR DO PAR(I):=FPAR(I):=START_PAR(I);
  END;
COMMENT
;
COMMENT READ COMMAND DESIGNATOR AND IDENTIFY COMMAND;
READ_CMD:
COMMENT %OUTPUT TO USER
ENTER COMMAND;
COMMENT %INPUT (CMD);
I:=1;
WHILE COMMANDS(I)~="END    " DO
  BEGIN
  IF COMMANDS(I)=CMD THEN BEGIN CMD_NO:=I;GO TO OUT; END;
  I:=I+1;
  END;
COMMENT %OUTPUT TO USER
COMMAND IN ERROR, RESPECIFY;
GO TO READ_CMD;
COMMENT
;
OUT:IF ECHO=TRUE THEN ECHO1(2);
CASE CMD_NO OF
  BEGIN
    BEGIN
    COMMENT ### PLOT COMMAND ###;
    INTEGER KK;
    KK:=0;
    FOR I:=1 UNTIL FN_PAR DO
    IF FRZ(I)=0 THEN
      BEGIN
      KK:=KK+1;
      FPAR(I):=PAR(KK);
      END;
    PLOT_COMMAND( OBS_PLACE,
    PTS_TO_HIT, STATE, OBS, OBS_SMOOTH,FPAR,
    N_OBS,N_PTS_HIT,N_STATE_OBS,N_STATE,FN_PAR,
```

```
  STATES_OBS,PLOT_NUMBER,
  READ_CMD_DATA,CMD_AL,CREOBS);
  END;
COMMENT
;
  BEGIN
  COMMENT ### SET COMMAND ###;
  INTEGER I;
  COMMENT %OUTPUT
  GIVE LIST OF PARAMETERS AND VALUES (END WITH 0)
  SUBSCRIPTS CORRESPOND TO UNFROZEN PARAMETER VECTOR
  %INPUT (I);
  WHILE I~=0 DO
    BEGIN
    COMMENT %INPUT (FPAR(I)),(I);
    END;
  I:=0;
  FOR J:=1 UNTIL FN_PAR DO
  IF FRZ(J)=0 THEN
    BEGIN
    I:=I+1;
    PAR(I):=FPAR(J);
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### INTEGRATE COMMAND ###;
  LONG REAL ARRAY INITY(1::1);
  LONG REAL ARRAY INITYP(1::1,1::1);
  AUX_INT:=0;
  COMMENT %OUTPUT TO USER
  IS THE JACOBIAN DESIRED? Y OR N;
  COMMENT %INPUT (ANS);
  IF ANS="Y" THEN AUX_INT:=1;
    BEGIN
    LONG REAL ARRAY B(1::N_STATE,1::N_PTS_HIT);
    LONG REAL ARRAY JB(1::N_PAR,1::N_PTS_HIT*N_STATE);
    CASE INT_PROC OF
      BEGIN
      COMMENT %OUTPUT TO USER
      INTEGRATION METHOD AND PERTINENT CONTROL PARAMETERS;
      END;
    IF OUTPUT~=0 THEN
      BEGIN
      COMMENT %FILE EMPTY -SC1;
      COMMENT %OUTPUT TO -REPRT
      OUTPUT REFERENCE NUMBER (OUT_SEG);
      CASE INT_PROC OF
        BEGIN
        COMMENT %OUTPUT TO -REPRT
```

```
        INDICATE INTEGRATION METHOD AND
        PERTINENT PARAMETERS;
        END;
      END;
    CASE INT_PROC OF
      BEGIN
      GEAR(PAR,PTS_TO_HIT,B,JB,N_STATE,N_PTS_HIT,
      EPS,HMIN,HMAX,N_PAR,AUX_INT,EG_FUN,
      KFLAG,OUTPUT,METHOD_FLAG,0,INITY,INITYP);
      TRAP(PAR,PTS_TO_HIT,B,JB,N_STATE,N_PTS_HIT,
      N_PAR,AUX_INT,EG_FUN,KFLAG,OUTPUT)
      END;
    AUX_INT:=1;
    COPY_TRANS(STATE,B,JACOBIAN,JB,N_PTS_HIT,N_STATE,
    N_PTS_HIT*N_STATE,N_PAR,AUX_INT);
    END;
  JJ:=0;
  FOR I:=1 UNTIL FN_PAR DO
  IF FRZ(I)=0 THEN
    BEGIN
    JJ:=JJ+1;
    INT_PAR(I):=PAR(JJ);
    END
  ELSE
  INT_PAR(I):=FPAR(I);
  END;
COMMENT
;
  BEGIN
  COMMENT ### FIT COMMAND ###;
  INTEGER I;
  LONG REAL ARRAY INITY(1::1);
  LONG REAL ARRAY INITYP(1::1,1::1);
  PROCEDURE FUNC(LONG REAL ARRAY P(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
  LONG REAL ARRAY JAC(*,*);
  LONG REAL ARRAY GRAD(*);
  INTEGER RESULT EFLAG);
    BEGIN
    COMMENT JACOBIAN GRADIENT AND RESIDUAL INFORMATION;
    INTEGER M,N_RES,OUT1;
    LONG REAL SUM;
    COMMENT INTEGRATE;
    IF OUTPUT_SUP=1 THEN OUT1:=0 ELSE OUT1:=OUTPUT;
    EFLAG:=0;
      BEGIN
      LONG REAL ARRAY B(1::N_STATE,1::N_PTS_HIT);
      LONG REAL ARRAY JB(1::N_PAR,1::N_PTS_HIT*N_STATE);
      CASE INT_PROC OF
        BEGIN
```

```
        BEGIN
        GEAR(P,PTS_TO_HIT,B,JB,N_STATE,
        N_PTS_HIT,EPS,HMIN,HMAX,N_PAR,1,
        EG_FUN,KFLAG,OUT1,
        METHOD_FLAG,0,INITY,INITYP);
        IF KFLAG~=1 THEN BEGIN EFLAG:=1; GO TO OUT; END;
        END;
        BEGIN
        TRAP(P,PTS_TO_HIT,B,JB,N_STATE,
        N_PTS_HIT,N_PAR,AUX_INT,EG_FUN,EFLAG,OUT1);
        IF EFLAG=1 THEN GO TO OUT;
        END
      END;
    AUX_INT:=1;
    COPY_TRANS(STATE,B,JACOBIAN,JB,N_PTS_HIT,N_STATE,
    N_PTS_HIT*N_STATE,N_PAR,AUX_INT);
    END;
  EXTRACT_JACOBIAN(JAC);
  FOR J:=1 UNTIL N_STATE_OBS DO
  FOR I:=1 UNTIL N_PTS_HIT DO
  IF OBS_STATUS(I)~=0 THEN
    BEGIN
    INTEGER INDEX;
    INDEX:=(OBS_STATUS(I)-1)*N_STATE_OBS+J;
    RES(INDEX):=(STATE(I,STATES_OBS(J))-
    OBS(OBS_STATUS(I),J))*WT_VEC(INDEX);
    END;
  N_RES:=N_STATE_OBS*N_OBS;
  F:=0;
  FOR I:=1 UNTIL N_RES DO F:=F+RES(I)**2;
  SUM_SQ_RES:=F;
  COMMENT FORM GRADIENT;
  FOR I:=1 UNTIL N_PAR DO
    BEGIN
    SUM:=0.;
    FOR J:=1 UNTIL N_RES DO
    SUM:=SUM+JAC(J,I)*RES(J);
    GRAD(I):=SUM;
    END;
  OUT:
  END FUNC;
COMMENT %OUTPUT
CHOOSE METHOD
MARQUARDT----------1
INTERACTIVE ---------2;
COMMENT %INPUT (I);
CASE I OF
  BEGIN
    BEGIN
    COMMENT FIT USING MARQUARDT'S TECHNIQUE ;
    LONG REAL LAM,EPS_R,EPS_A;
```

```
      COMMENT %OUTPUT TO USER
      ENTER STARTING LAMBDA, RELATIVE TOLERENCE,
      AND ABSOLUTE TOLERENCE MARQUARDT;
      COMMENT %INPUT (LAM),(EPS_R),(EPS_A);
      COMMENT %FILE EMPTY -SC1;
      COMMENT %OUTPUT
      CURRENT OUTPUT REFERENCE NUMBER IS (OUT_SEG)
      MARQUARDT USED IN FIT COMMAND (LAM),(EPS_R),(EPS_A);
      MARQUARDT(EPS_R,EPS_A,N_OBS*N_STATE_OBS,N_PAR,
      FUNC,PAR,LAM,SVD_AL);
      END;
      BEGIN
      COMMENT %EXTERNAL INTERACTIVE_OPT;
      IF ECHO=TRUE THEN ECHO1(3);
      INTERACTIVE_OPT(N_OBS*N_STATE_OBS,N_PAR,FUNC,
      PAR,SVD_AL,OBS_PLACE(*), PTS_TO_HIT(*), STATE(*,*),
      OBS(*,*), OBS_SMOOTH(*,*), CREOBS(*),
      N_OBS,N_PTS_HIT,N_STATE_OBS,
      N_STATE, STATES_OBS(*), PLOT_NUMBER,
      READ_CMD_DATA, CMD_AL, PLOT_COMMAND);
      IF ECHO=TRUE THEN ECHO1(1);
      END
    END;
  JJ:=0;
  FOR I:=1 UNTIL FN_PAR DO
  IF FRZ(I)=0 THEN
    BEGIN
    JJ:=JJ+1;
    INT_PAR(I):=PAR(JJ);
    END
  ELSE
  INT_PAR(I):=FPAR(I);
  END;
COMMENT
;
  BEGIN
  COMMENT ### QUIT COMMAND ###;
  GO TO FINISHED;
  END;
COMMENT
;
  BEGIN
  COMMENT ### REPORT COMMAND ###;
  INTEGER ARRAY CMD_DATA(1::15);
  INTEGER II,UNI;
  COMMENT %OUTPUT TO USER
  ENTER LIST OF ITEMS TO PRINT (END WITH 0)
  GENERAL DATA -------1
  OBS----------------2
  PAR----------------3
  CREATION-----------4
```

```
PTS AND STATE VARS--5
SMOOTHING DATA------6
JACOBIAN------------7
INTEG./OPT. DETAILS-8
STATISTICAL DATA----9
OPTION SETTINGS----10
WEIGHTS------------11
GUESSED OBS--------12;
READ_CMD_DATA(CMD_DATA);
FOR IREP:=1 UNTIL 2 DO
  BEGIN
  COMMENT %FILE IF IREP=1 OUTPUT TO USER,
  IF IREP=2 OUTPUT TO -REPRT;
  II:=1;
  WHILE CMD_DATA(II)~=0 DO
    BEGIN
    CASE CMD_DATA(II) OF
      BEGIN
        BEGIN
        COMMENT OPTION 1, GENERAL SIZE DATA;
        COMMENT %OUTPUT (FN_PAR),(N_STATE),(N_STATE_OBS),
        (N_OBS),(STATES_OBS);
        END;
        BEGIN
        COMMENT OPTION 2, OBSERVATIONS;
        COMMENT %OUTPUT (OBS),(OBS_PLACE);
        END;
        BEGIN
        COMMENT OPTION 3, PARAMETER VALUES;
        INTEGER KK;
        KK:=0;
        FOR I:=1 UNTIL FN_PAR DO
        IF FRZ(I)=0 THEN
          BEGIN
          KK:=KK+1;
          FPAR(I):=PAR(KK);
          END;
        COMMENT %OUTPUT (START_PAR),(FPAR)
        (FRZ),(SCL);
        END;
        BEGIN
        COMMENT OPTION 4, DATA ON MODEL CREATION;
        COMMENT %OUTPUT RAMDOM ERROR USED,
        SIMULATION PARAMETERS
        (STD_DEV),(SIM_PAR);
        END;
        BEGIN
        COMMENT OPTION 5, TIMES, AND STATE VARIABLES;
        COMMENT %OUTPUT INTEGRATION PARAMETERS (INT_PAR),
        SAMPLE TIMES (PTS_TO_HIT),
        INTEGRATION PROCEDURE (INT_PROC)
```

```
INTEGRATION RESULTS (STATE);
END;
BEGIN
COMMENT OPTION 6, SMOOTHING INFORMATION;
CASE SMF OF
  BEGIN
    BEGIN
    COMMENT SPLINE SMOOTHING CASE;
    COMMENT %OUTPUT
    SPLINE SMOOTHING JOINTS,
    SMOOTHED OBSERVATIONS,
    SMOOTHED DERIVATIVES ON OBSERVED
    STATE VARIABLES;
    END;
    BEGIN
    COMMENT HERMITE CASE;
    COMMENT %OUTPUT
    HERMITE SMOOTHING JOINTS,
    SMOOTHED OBSERVATIONS,
    SMOOTHED DERIVATIVES ON OBSERVED
    STATE VARIABLES;
    END
  END;
END;
BEGIN
COMMENT OPTION 7, JACOBIAN OUTPUT;
COMMENT %OUTPUT
INTEGRATION PARAMETERS (INT_PAR);
IF JAC_OPTION=0 THEN
COMMENT %OUTPUT
FULL JACOBIAN ON ALL STATES AT ALL SAMPLE TIMES;
ELSE
COMMENT %OUTPUT
LEAST SQUARES JACOBIAN;
END;
BEGIN
COMMENT OPTION 8, OPTIMIZATION
AND/OR INTEGRATION DATA;
IF BATCH=FALSE THEN
  BEGIN
  COMMENT %FILE DISPLAY -SC1
  TO USER IF (UNIT) SPECIFIES
  OUTPUT TO USER;
  IF UNIT="-REPRT" THEN
    BEGIN
    STRING(1) ANS;
    COMMENT %OUTPUT TO USER
    IS INTEGRATION/OPTIMIZATION DATA TO BE
    ACCUMULATED FOR LATER OUTPUT Y OR N
    %INPUT (ANS);
    IF ANS="Y" THEN
```

```
      BEGIN
      COMMENT %OUTPUT TO -REPRT
      DATA ACCUMULATED IN -SC2 OUTPUT REFERENCE
      NUMBER IS (OUT_SEG)
      %FILE ACCUMULATE -SC1 AT END OF -SC2;
      OUT_SEG:=OUT_SEG+1;
      END;
    END;
  END;
END;
BEGIN
COMMENT OPTION 9, STATISTICAL DATA;
INTEGER N1;
N1:=N_OBS-N_PAR;
COMMENT %OUTPUT
F DISTRIBUTION VALUE AT (PERCENTILE) PERCENT WITH
(N_PAR) DEGREES OF FREEDOM IN NUMERATOR AND
(N1) DEGREES OF FREEDON IN DENOMINATOR
IS (FPROB),
SUM OF SQUARES OF RESIDUALS (SUM_SQ_RES),
CRRRELATION MATRIX(COR), COVARIANCE MATRIX (COV),
PARAMETERS (PAR), CONFIDENCE INTERVALS (CON);
END;
BEGIN
COMMENT OPTION 10 REPORT ON OPTION SETTINGS;
COMMENT %OUTPUT
OPTION SETTINGS:
OUTPUT FLAG (OUTPUT),
OUTPUT SUPPRESSION FLAG (OUTPUT_SUP),
JACOBIAN OUTPUT MODE (JAC_OPTION),
GEAR PROGRAM (METHOD_FLAG),
GEAR PROGRAM EPS (EPS),
GEAR PROGRAM MINIMUM AND
MAXIMUM STEP (HMIN), (HMAX),
INTEGRATION METHOD (INT_PROC);
END;
BEGIN
COMMENT OPTION 11 REPORT ON WEIGHTS;
COMMENT %OUTPUT
OBSERVATION TIMES (OBS_PLACE),
STATES OBSERVED (STATES_OBS),
WEIGHTING FACTORS (WT);
END;
BEGIN
COMMENT OPTION 12 CREATED OBSERVTIONS DATA;
COMMENT %OUTPUT
JOINTS OF INTERPOLATING SPLINE (CREOBS_JOINTS)
ORDINATES FOR INTERPOLATING SPLINE (CREOBS_VAL)
CREATED OBSERVATIONS (CREOBS)
AND DERIVATIVES (DCREOBS)
AT SAMPLE TIMES (PTS_TO_HIT);
```

```
          END
        END;
      II:=II+1;
      END;
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### CREATE COMMAND ###;
  INTEGER ARRAY INTDATA(1::7);
  COMMENT %EXTERNAL CREATE_DATA;
  INTDATA(1):=N_STATE;
  INTDATA(3):=N_OBS;
  INTDATA(4):=FN_PAR;
  INTDATA(5):=N_STATE_OBS;
  INTDATA(6):=MAX_PTS;
  INTDATA(7):=METHOD_FLAG;
  CREATE_DATA(SIM_PAR,
  START_PAR, FPAR, OBS_PLACE, PTS_TO_HIT, OBS,
  STATE, JACOBIAN, OBS_STATUS, STATES_OBS, INTDATA,
  MODEL, EG_FUN, GEAR, STANDARD_HIT,STD_DEV,
  INITIAL_TIME, KFLAG,OUTPUT);
  N_PTS_HIT:=INTDATA(2);
  FOR I:=1 UNTIL FN_PAR DO
  INT_PAR(I):=SIM_PAR(I);
  N_PAR:=FN_PAR;
  END;
COMMENT
;
  BEGIN
  COMMENT ### SAMPLE COMMAND ###;
  COMMENT %EXTERNAL HITPTS;
  HITPTS(STANDARD_HIT,
  READ_CMD_DATA,
  PTS_TO_HIT,
  N_PTS_HIT,MAX_PTS,
  OBS_STATUS,
  OBS_PLACE,
  N_OBS,
  INITIAL_TIME);
  END;
COMMENT
;
  BEGIN
  COMMENT ### SMOOTH COMMAND ###;
  INTEGER MET;
  COMMENT %OUTPUT
  SELECT METHOD
  CUBIC SPLINE--------1
  CUBIC HERMITE-------2
```

```
%INPUT (MET);
CASE MET OF
  BEGIN
    BEGIN
    COMMENT SPLINE CASE;
    INTEGER NJ,DEG;
    LONG REAL ARRAY W(1::N_OBS);
    FOR I:=1 UNTIL N_OBS DO W(I):=1.;
    FOR ST:=1 UNTIL N_STATE_OBS DO
      BEGIN
      COMMENT %OUTPUT
      ENTER NUMBER OF JOINTS (MAX 15),
      AND JOINT POSITIONS FOR
      STATE VARIABLE (STATES_OBS(ST))
      %INPUT (NJ);
      NJOINTS(ST):=NJ;
      FOR I:=1 UNTIL NJ DO
      COMMENT %INPUT (JOINTS(I,ST));
      DEG:=3;
      SPLINT_AL(OBS_PLACE,OBS(*,ST),W,N_OBS,
      SPLN_COEF(*,ST),
      DEG,JOINTS(*,ST),NJOINTS(ST));
      N_SPLN_PAR(ST):=DEG;
      COMMENT INSERT VALUES INTO OBS_SMOOTH;
      FOR I:=1 UNTIL N_PTS_HIT DO
      OBS_SMOOTH(I,ST):=SPLN_AL(PTS_TO_HIT(I),
      SPLN_COEF(*,ST),JOINTS(*,ST),
      NJOINTS(ST),N_SPLN_PAR(ST));
      END;
    SMF:=1;
    END;
    BEGIN
    COMMENT HERMITE CASE;
    INTEGER NJ;
    LONG REAL ARRAY COEF(1::40,1::4);
    FOR ST:=1 UNTIL N_STATE_OBS DO
      BEGIN
      INTEGER FLAG;
      REDO:
      COMMENT %OUTPUT
      ENTER NUMBER OF JOINTS (MAX 15),
      AND JOINT POSITIONS FOR
      STATE VARIABLE (STATES_OBS(ST))
      DO NOT INCLUDE FIRST OR LAST OBSERVATION
      %INPUT (NJ);
      NJOINTS(ST):=NJ;
      JOINTS(1,ST):=OBS_PLACE(1);
      FOR I:=1 UNTIL NJ DO
      COMMENT %INPUT (JOINTS(I+1,ST));
      JOINTS(NJ+2,ST):=OBS_PLACE(N_OBS);
      IF N_OBS<(2*NJ+4) THEN
```

```
            BEGIN
            COMMENT %OUTPUT TO USER
            EXCESS OF ((2*NJ+4-N_OBS)/2) JOINTS;
            GO TO REDO;
            END;
          HERMIT_AL(OBS_PLACE,OBS(*,ST),JOINTS(*,ST),
          N_OBS,NJ,COEF,40,FLAG);
          IF FLAG=1 THEN
            BEGIN
            COMMENT %OUTPUT
            SINGULAR PROBLEM--REDISTRIBUTE
            JOINTS OR REMOVE SOME JOINTS;
            GO TO REDO;
            END;
          FOR J:=1 UNTIL 4 DO
          FOR I:=1 UNTIL NJ+1 DO
          HERM_COEF(I,J,ST):=COEF(I,J);
          COMMENT INSERT VALUES INTO OBS SMOOTH;
          FOR I:=1 UNTIL N_PTS_HIT DO
          OBS_SMOOTH(I,ST):=HERM(PTS_TO_HIT(I),
          HERM_COEF(*,*,ST),
          JOINTS(*,ST),NJOINTS(ST));
          END;
        SMF:=2;
        END
      END;
    END;
  COMMENT
  ;
    BEGIN
    COMMENT ### DFIT COMMAND ###;
    INTEGER ARRAY INTDATA(1::14);
    INTEGER METHOD;
    COMMENT %EXTERNAL DATAFT_COMMAND, DFIT_CRE,
    DFITITER,SPRGN;
    INTDATA(1):=N_PTS_HIT;
    INTDATA(2):=N_STATE_OBS;
    INTDATA(3):=N_STATE;
    INTDATA(4):=N_PAR;
    INTDATA(5):=SMF;
    INTDATA(6):=OUTPUT;
    INTDATA(7):=METHOD_FLAG;
    INTDATA(9):=OUTPUT_SUP;
    INTDATA(10):=OUT_SEG;
    INTDATA(11):=INT_PROC;
    INTDATA(12):=DFIT_LIN;
    COMMENT %OUTPUT
    ENTER METHOD
    REGULAR----------------------1
    USING GUESSED OBS------------2
    ITERATED USING SPARSE GN-----3;
```

```
  CASE METHOD OF
    BEGIN
    DATAFT_COMMAND(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM, DHERM, INITIAL_TIME,EPS,HMIN,HMAX);
    DFIT_CRE(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM,DHERM,CREOBS,DCREOBS);
    SPRGN(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM,DHERM,CREOBS,DCREOBS,DFITITER)
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### SUSPEND COMMAND ###;
  COMMENT SUSPEND EXECUTION AND RETURN TO OPERATING SYSTEM;
  END;
COMMENT
;
  BEGIN
  COMMENT ### OPTION COMMAND ###;
  INTEGER I;
  COMMENT %OUTPUT
  DESIGNATE CONTROL PAR. AND NEW VALUE (0 TO END)"
  OUTPUT-------------------------------------------1
  INT. OUTPUT IN OPT. 0-YES, 1-NO------------------2
  JACOBIAN 0-FULL 1-LEAST SQUARES------------------3
  GEAR METHOD 0-ADAMS,1-STIFF----------------------4
  EPS FOR INTEGRATION------------------------------5
  HMIN FOR INTEGRATION-----------------------------6
  HMAX FOR INTEGRATION-----------------------------7
  INTEGRATION PROCEDURE 1-GEAR, 2-QUICK------------8;
  WHILE I~=0 DO
    BEGIN
    CASE I OF
      BEGIN
      COMMENT %INPUT DEPENDING ON (I)
      (OUTPUT),(OUTPUT_SUP),(JAC_OPTION),(METHOD_FLAG),
      (EPS),(HMIN),(HMAX),(INT_PROC);
      END;
    COMMENT %INPUT (I);
    END;
  END;
```

```
COMMENT
;
  BEGIN
  COMMENT ### STATS COMMAND ###
  THIS COMMAND ASSUMES
  THE JACOBIAN AND THE SUM OF THE
  SQUARES OF THE RESIDUAL ARE AVAILABLE AT THE
  OPTIMUM PARAMETERS OBTAINED;
  LONG REAL ARRAY INV,VD(1::N_PAR,1::N_PAR);
  LONG REAL ARRAY UD,JAC(1::N_STATE_OBS*N_OBS,1::N_PAR);
  LONG REAL ARRAY S(1::N_PAR);
  INTEGER M;
  LONG REAL SUM,SUMI,SUMJ,SIG_SQ,E;
  COMMENT %EXTERNAL FVALUE_AL;
  M:=N_STATE_OBS*N_OBS;
  COMMENT %OUTPUT
  ENTER (INTEGER) PERCENTILE FOR F DISTRIBUTION
  COMMENT %INPUT (PERCENTILE);
  FPROB:=FVALUE_AL(1.-PERCENTILE/100.,N_PAR,
  N_OBS*N_STATE_OBS-N_PAR);
  COMMENT %OUTPUT (FPROB);
  EXTRACT_JACOBIAN(JAC);
  SVD_AL(JAC,S,UD,VD,M,N_PAR,M,N_PAR,0,N_PAR,N_PAR);
  FOR J:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL N_PAR DO
  VD(I,J):=VD(I,J)/S(J);
  COMMENT FORM INVERSE OF (JACOBIAN TRANSPOSE
  TIMES JACOBIAN);
  FOR I:=1 UNTIL N_PAR DO
  FOR J:=1 UNTIL N_PAR DO
    BEGIN
    SUM:=0.;
    FOR K:=1 UNTIL N_PAR DO
    SUM:=SUM+VD(I,K)*VD(J,K);
    INV(I,J):=SUM;
    END;
  SIG_SQ:=SUM_SQ_RES/(M-N_PAR);
  E:=N_PAR*FPROB*SIG_SQ;
  COMMENT CALCULATE CONFIDENCE INTERVALS;
  FOR I:=1 UNTIL N_PAR DO
  CON(I):=SQRT(E*INV(I,I));
  COMMENT CALCULATE COVARIANCE MATRIX;
  FOR I:=1 UNTIL N_PAR DO
  FOR J:=1 UNTIL N_PAR DO
  COV(I,J):=SIG_SQ*INV(I,J);
  COMMENT CALCULATE CORRELATION MATRIX;
  FOR I:=2 UNTIL N_PAR DO
  FOR J:=1 UNTIL I-1 DO
    BEGIN
    SUMI:=0;SUMJ:=0;SUM:=0;
    FOR K:=1 UNTIL N_PAR DO
```

```
      BEGIN
      SUM:=SUM+VD(I,K)*VD(J,K);
      SUMI:=SUMI+VD(I,K)**2;
      SUMJ:=SUMJ+VD(J,K)**2;
      END;
    COR(I,J):=SUM/SQRT(SUMI*SUMJ);
    COR(J,I):=COR(I,J);
    END;
  FOR I:=1 UNTIL N_PAR DO COR(I,I):=1.;
  END;
COMMENT
;
  BEGIN
  COMMENT ### PROBE COMMAND ###;
  STRING(1) ANS;
  INTEGER M;
  LONG REAL ARRAY INITY(1::1);
  LONG REAL ARRAY INITYP(1::1,1::1);
  LONG REAL ARRAY UD,JAC(1::N_STATE_OBS*N_OBS,1::N_PAR);
  LONG REAL ARRAY VD(1::N_PAR,1::N_PAR);
  LONG REAL ARRAY S(1::N_PAR);
  COMMENT %OUTPUT
  IS AN INTEGRATION REQUIRED? Y OR N
  %INPUT (ANS);
  IF ANS="Y" THEN
    BEGIN
    IF OUTPUT~=0 THEN
      BEGIN
      COMMENT %FILE EMPTY -SC1
      %OUTPUT TO -REPRT INTEGRATION IN PROBE
      OUTPUT REFERENCE NUMBER IS (OUT_SEG);
      CASE INT_PROC OF
        BEGIN
        COMMENT %OUTPUT TO -REPRT
        INTEGRATION CONTROL PARAMETRERS;
        END;
      END;
      BEGIN
      LONG REAL ARRAY B(1::N_STATE,1::N_PTS_HIT);
      LONG REAL ARRAY JB(1::N_PAR,1::N_PTS_HIT*N_STATE);
      AUX_INT:=1;
      CASE INT_PROC OF
        BEGIN
          BEGIN
          GEAR(PAR,PTS_TO_HIT,B,JB,N_STATE,N_PTS_HIT,
          EPS,HMIN,HMAX,N_PAR,1,EG_FUN,KFLAG,OUTPUT,
          METHOD_FLAG,
          0,INITY,INITYP);
          IF KFLAG~=1 THEN GO TO OUT;
          END;
          BEGIN
```

```
          TRAP(PAR,PTS_TO_HIT,B,JB,N_STATE,N_PTS_HIT,
          N_PAR,1,
          EG_FUN,KFLAG,OUTPUT);
          IF KFLAG=1 THEN GO TO OUT;
          END
        END;
      AUX_INT:=1;
      COPY_TRANS(STATE,B,JACOBIAN,JB,N_PTS_HIT,N_STATE,
      N_PTS_HIT*N_STATE,N_PAR,AUX_INT);
      END;
    JJ:=0;
    FOR I:=1 UNTIL FN_PAR DO
    IF FRZ(I)=0 THEN
      BEGIN
      JJ:=JJ+1;
      INT_PAR(I):=PAR(JJ);
      END
    ELSE
    INT_PAR(I):=PAR(I);
    END;
M:=N_STATE_OBS*N_OBS;
EXTRACT_JACOBIAN(JAC);
SVD_AL(JAC,S,UD,VD,M,N_PAR,M,N_PAR,0,N_PAR,N_PAR);
COMMENT %OUTPUT TO USER
PARAMETERS (PAR) SINGULAR VALUES (S)
IS A REPORT OF PROBE REQUIRED?, Y OR N
%INPUT (ANS);
IF ANS="Y" THEN
  BEGIN
  COMMENT %OUTPUT TO -REPRT
  ###PROBE COMMAND###
  PARAMETERS (PAR) SINGULAR VALUES (S);
  IF OUTPUT~=0 THEN
    BEGIN
    COMMENT %OUTPUT
    IS INTEG OUTPUT TO BE DISPLAYED? Y OR N
    %INPUT (ANS);
    IF ANS="Y" THEN
      BEGIN
      COMMENT %FILE DISPLAY -SC1 TO USER
      %OUTPUT
      IS INTEG OUTPUT TO BE ACCUMULATED? Y OR N
      %INPUT (ANS);
      IF ANS="Y" THEN
        BEGIN
        COMMENT %OUTPUT TO -REPRT
        INTEGRATION DATA ACCUMULATED IN PROBE
        OUTPUT REFERENCE NUMBER IS (OUT_SEG)
        %FILE ACCUMULATE -SC1 AT END OF -SC2;
        OUT_SEG:=OUT_SEG+1;
        END;
```

```
        END;
      END;
    END;
  OUT:END;
COMMENT
;
  BEGIN
  COMMENT ### WEIGHTING COMMAND ###;
  INTEGER CAS;INTEGER ARRAY ST_WT(1::N_STATE_OBS+1);
  INTEGER ST,SUB,SUB1;
  COMMENT %OUTPUT
  INDICATE OPTION
  SET ALL WEIGHTS TO 1------------1
  INDIVIDUAL ENTRY OF WEIGHTS-----2
  %INPUT (CAS);
  CASE CAS OF
    BEGIN
      BEGIN
      FOR J:=1 UNTIL N_STATE_OBS DO
      FOR I:=1 UNTIL N_OBS DO
      WT(I,J):=1;
      END;
      BEGIN
      COMMENT %OUTPUT
      ENTER LIST OF STATE VARIABLES WHOSE
      OBSERVATIONS ARE TO BE WEIGHTED. 0 TO END;
      READ_CMD_DATA(ST_WT);
      ST:=1;
      WHILE ST_WT(ST)~=0 DO
        BEGIN
        FOR I:=1 UNTIL N_STATE_OBS DO
        IF STATES_OBS(I)=ST_WT(ST) THEN SUB1:=I;
        COMMENT %OUTPUT
        ENTER SEQUENCE OF OBSERVATION TIME SUBSCRIPTS
        AND NEW WEIGHTS FOR STATE VARIABLE
        (ST_WT(ST))
        ENTER A ZERO SUBSCRIPT TO END INPUT
        %INPUT (SUB);
        WHILE SUB~=0 DO
          BEGIN
          COMMENT %INPUT (WT(SUB,SUB1)),(SUB);
          END;
        ST:=ST+1;
        END;
      COMMENT %OUTPUT
      DO YOU WANT THE WEIGHTS STORED
      IN THE FILE -WEIGHT? Y,N
      %INPUT (ANS);
      IF ANS="Y" THEN
        BEGIN
        COMMENT %OUTPUT TO -WEIGHT (WT);
```

```
        END;
      END
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### CONTINUATION COMMAND ###;
  INTEGER ARRAY INTDATA(1::15);
  COMMENT %EXTERNAL CONTI;
  INTDATA(1):=N_PTS_HIT;
  INTDATA(2):=N_STATE_OBS;
  INTDATA(3):=N_STATE;
  INTDATA(4):=N_PAR;
  INTDATA(5):=SMF;
  INTDATA(6):=OUTPUT;
  INTDATA(7):=METHOD_FLAG;
  INTDATA(9):=OUTPUT_SUP;
  INTDATA(10):=OUT_SEG;
  INTDATA(11):=INT_PROC;
  INTDATA(12):=DFIT_LIN;
  CONTI(EG_FUN,
  PAR,0,INTDATA,OBS_SMOOTH,STATES_OBS,
  PTS_TO_HIT,JOINTS,SPLN_COEF,HERM_COEF,
  NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
  HERM,DHERM,EPS,HMIN,HMAX);
  END;
COMMENT
;
  BEGIN
  COMMENT ### CREOBS COMMAND ###
  THIS COMMAND ALLOWS THE USER
  TO CREATE OBSERVATIONS FOR AN
  UNOBSERVED STATE VARIABLE
  COMMENT %OUTPUT
  ENTER NO. OF JOINTS AND JOINT POSITIONS
  FOR STATE VARIABLE (3-STATES_OBS(1))
  INCLUDE FIRST AND LAST SAMPLE TIMES
  %INPUT (NO_CREOBS_JOINTS);
  FOR I:=1 UNTIL NO_CREOBS_JOINTS DO
  COMMENT %INPUT (CREOBS_JOINTS(I));
  COMMENT %OUTPUT
  ENTER CORRESPONDING ORDINATES FOR CREATED OBS;
  FOR I:=1 UNTIL NO_CREOBS_JOINTS DO
  COMMENT %INPUT (CREOBS_VAL(I));
    BEGIN
    REAL ARRAY X,Y(1::NO_CREOBS_JOINTS);
    REAL ARRAY P(1::7*NO_CREOBS_JOINTS);
    REAL ARRAY SI(1::2);
    REAL ARRAY T,S,S1,S2(1::N_PTS_HIT);
    COMMENT %EXTERNAL SMT;
```

```
    FOR I:=1 UNTIL NO_CREOBS_JOINTS-1 DO P(I):=0;
    FOR I:=1 UNTIL NO_CREOBS_JOINTS DO
      BEGIN
      X(I):=CREOBS_JOINTS(I);
      Y(I):=CREOBS_VAL(I);
      END;
    SI(1):=(Y(2)-Y(1))/(X(2)-X(1));
    SI(2):=(Y(NO_CREOBS_JOINTS)-Y(NO_CREOBS_JOINTS-1))/
    (X(NO_CREOBS_JOINTS)-X(NO_CREOBS_JOINTS-1));
    FOR I:=1 UNTIL N_PTS_HIT DO
    T(I):=PTS_TO_HIT(I);
    SMT(X,Y,P,SI,T,S,S1,S2,NO_CREOBS_JOINTS,
    1,N_PTS_HIT);
    FOR I:=1 UNTIL N_PTS_HIT DO
      BEGIN
      CREOBS(I):=S(I);
      DCREOBS(I):=S1(I);
      END;
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### IFIT COMMAND ###;
  INTEGER ARRAY INTDATA(1::14);
  INTEGER METHOD;
  COMMENT %EXTERNAL CONT,IFIT_CRE,IFITI,SPRGN;
  INTDATA(1):=N_PTS_HIT;
  INTDATA(2):=N_STATE_OBS;
  INTDATA(3):=N_STATE;
  INTDATA(4):=N_PAR;
  INTDATA(5):=SMF;
  INTDATA(6):=OUTPUT;
  INTDATA(7):=METHOD_FLAG;
  INTDATA(9):=OUTPUT_SUP;
  INTDATA(10):=OUT_SEG;
  INTDATA(11):=INT_PROC;
  INTDATA(12):=DFIT_LIN;
  COMMENT %OUTPUT
  ENTER METHOD
  REGULAR----------------------1
  USING GUESSED OBS------------2
  ITERATED INTEG. SUBSYSTEMS---3
  ITERATED USING SPARSE GN-----4
  %INPUT (METHOD);
  CASE METHOD OF
    BEGIN
    CONTI(EG_FUN,
    PAR,1,INTDATA,OBS_SMOOTH,STATES_OBS,
    PTS_TO_HIT,JOINTS,SPLN_COEF,HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
```

```
    HERM,DHERM,EPS,HMIN,HMAX);
    IFIT_CRE(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM,DHERM,CREOBS,DCREOBS);
    IFITI(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM,DHERM,CREOBS,DCREOBS,IFIT_CRE);
    SPRGN(EG_FUN,
    PAR, INTDATA, OBS_SMOOTH, STATES_OBS,
    PTS_TO_HIT, JOINTS, SPLN_COEF, HERM_COEF,
    NJOINTS, N_SPLN_PAR,SPLN_AL,DSPLN_AL,
    HERM,DHERM,CREOBS,DCREOBS,IFIT_CRE)
    END;
  END;
COMMENT
;
  BEGIN
  COMMENT ### FREEZE COMMAND ###;
  INTEGER SUB,KK;
  KK:=0;
  FOR I:=1 UNTIL FN_PAR DO
  IF FRZ(I)=0 THEN
    BEGIN
    KK:=KK+1;
    FPAR(I):=PAR(KK);
    END;
  COMMENT %OUTPUT
  ENTER LIST OF SUBSCRIPTS FOR FROZEN PARAMETERS (0 TO END)
  (AN ENTRY OF 0 REMOVES ALL FREEZING)
  %INPUT (SUB);
  IF SUB=0 THEN
    BEGIN
    PFRZ:=FALSE;
    FOR I:=1 UNTIL FN_PAR DO FRZ(I):=0;
    FOR I:=1 UNTIL FN_PAR DO PAR(I):=FPAR(I);
    N_PAR:=FN_PAR;
    END
  ELSE
    BEGIN
    PFRZ:=TRUE;
    WHILE SUB~=0 DO
      BEGIN
      FRZ(SUB):=1;
      COMMENT %INPUT (SUB);
      END;
    END;
  KK:=0;
```

```
  FOR I:=1 UNTIL FN_PAR DO
  IF FRZ(I)=0 THEN
    BEGIN
    KK:=KK+1;
    PAR(KK):=FPAR(I);
    END;
  N_PAR:=KK;
  END;
COMMENT
;
  BEGIN
  COMMENT ### SCALE COMMAND ###;
  INTEGER SUB,KK;
  COMMENT UPDATE FPAR FOR RESCALING;
  KK:=0;
  FOR I:=1 UNTIL FN_PAR DO
  IF FRZ(I)=0 THEN
    BEGIN
    KK:=KK+1;
    FPAR(I):=PAR(KK);
    END;
  COMMENT DESCALE FPAR IN PREPARATION FOR RESCALING;
  FOR I:=1 UNTIL FN_PAR DO
  IF SCL(I)=1 THEN FPAR(I):=LONGEXP(FPAR(I))
  ELSE
  IF SCL(I)=2 THEN FPAR(I):=FPAR(I)**2;
  COMMENT %OUTPUT
  ENTER LIST OF PARAMETER SUBSCRIPTS AND
  SCALING INDICATORS (0 SUBSCRIPT TO END)
  INDICATOR OF 1 GIVES LOG SCALING
  INDICATOR OF 2 GIVES SQUARE ROOT SCALING
  %INPUT (SUB);
  IF SUB=0 THEN
    BEGIN
    PSCL:=FALSE;
    FOR I:=1 UNTIL FN_PAR DO SCL(I):=0;
    END
  ELSE
    BEGIN
    PSCL:=TRUE;
    WHILE SUB~=0 DO
      BEGIN
      COMMENT %INPUT (SCL(SUB)),(SUB);
      END;
    END;
  COMMENT RESCALE PARAMETER VALUES;
  FOR I:=1 UNTIL FN_PAR DO
  IF SCL(I)=1 THEN FPAR(I):=LONGLN(FPAR(I))
  ELSE
  IF SCL(I)=2 THEN FPAR(I):=LONGSQRT(FPAR(I));
  COMMENT UPDATE PAR;
```

```
      KK:=0;
      FOR I:=1 UNTIL FN_PAR DO
      IF FRZ(I)=0 THEN
        BEGIN
        KK:=KK+1;
        PAR(KK):=FPAR(I);
        END;
      END;
    COMMENT
    ;
      BEGIN
      COMMENT ### ECHO COMMAND ###;
      INTEGER INT;
      COMMENT %OUTPUT
      ENTER 1 TO BEGIN ECHO, 0 TO END ECHO
      %INPUT (INT);
      IF INT=1 THEN ECHO:=TRUE ELSE ECHO:=FALSE;
      IF INT=1 THEN ECHO1(1)
      ELSE ECHO1(3);
      END;
    COMMENT
    ;
      BEGIN
      COMMENT ### LINEAR COMMAND ###
      TO INDICATE LINEARITY IN IFIT, DFIT;
      COMMENT %OUTPUT
      ENTER 1 IF IFIT, DFIT GIVE LINEAR PROBLEMS
      ENTER 0 TO REMOVE LINEARITY SETTING
      %INPUT (DFIT_LIN);
      END
    END;
  GO TO READ_CMD;
  FINISHED: IF ECHO=TRUE THEN ECHO1(3);
  END PARFIT.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                Procedure to modify sample times

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE HITPTS(PROCEDURE STANDARD_HIT, READ_CMD_DATA;
LONG REAL ARRAY PTS_TO_HIT(*);
INTEGER VALUE RESULT N_PTS_HIT;
INTEGER VALUE MAX_PTS;
INTEGER ARRAY OBS_STATUS(*);
LONG REAL ARRAY OBS_PLACE(*);
INTEGER VALUE N_OBS;
LONG REAL VALUE INITIAL_TIME);
  BEGIN
```

```
COMMENT MODIFICATION OF SAMPLE TIMES
(DEFAULT, UNIFORM MESH, OR INSERTION);
INTEGER HIT_OPTION;
COMMENT %OUTPUT TO USER
PICK OPTION
OBS PTS+INITIAL TIME--1
UNIFORM MESH----------2
INTERACTIVE INSERTION-3
%INPUT SELECTION IN (HIT_OPTION);
CASE HIT_OPTION OF
  BEGIN
    BEGIN
    COMMENT STANDARD SCHEME--OBSERVATION POINTS PLUS INITIAL
    TIME;
    STANDARD_HIT(INITIAL_TIME,OBS_STATUS,OBS_PLACE,N_OBS,
    PTS_TO_HIT,N_PTS_HIT);
    END;
    BEGIN
    COMMENT A UNIFORM MESH STARTING WITH INITIAL_TIME;
    INTEGER N;LONG REAL DEL;
    COMMENT %OUTPUT TO USER
    ENTER NUMBER OF POINTS AND POINT SPACING
    %INPUT (N),(DEL);
    PTS_TO_HIT(1):=INITIAL_TIME;
    FOR I:=1 UNTIL N DO
    PTS_TO_HIT(I+1):=INITIAL_TIME+I*DEL;
    N_PTS_HIT:=N+1;
    END;
    BEGIN
    COMMENT INTERACTIVE INSERTION OF POINTS BETWEEN EXISTING
    POINTS;
    INTEGER L1,L2,K,J;
    LONG REAL ARRAY NEW_PTS_TO_HIT(1::MAX_PTS);
    INTEGER ARRAY NEW_OBS_STATUS(1::MAX_PTS);
    INTEGER ARRAY CMD_DATA(1::21);
    COMMENT %OUTPUT TO USER
    IS A LISTING OF SAMPLE TIMES REQUIRED?
    IF NOT ENTER 0, IF YES ENTER SUBSCRIPT LIMITS
    %INPUT (L1);
    IF L1~=0 THEN
      BEGIN
      COMMENT %INPUT (L2)
      %OUTPUT TO USER
      LIST OF SAMPLE TIMES BETWEEN SUBSCRIPTS L1 AND L2;
      END;
    COMMENT %OUTPUT TO USER
    ENTER SEQUENCE OF UPPER INDICIES AND NUMBER OF
    TIMES TO INSERT BETWEEN INDICATED TIME AND
    PREVIOUS TIME (END WITH 0).;
     READ_CMD_DATA(CMD_DATA);
     K:=1;
```

```
      J:=0;
      FOR I:=1 UNTIL N_PTS_HIT DO
      IF CMD_DATA(K)=I THEN
        BEGIN
        COMMENT INSERTION OF POINTS;
        LONG REAL DEL;
        INTEGER N_INS;
        N_INS:=CMD_DATA(K+1);
        DEL:=(PTS_TO_HIT(I)-PTS_TO_HIT(I-1))/(N_INS+1);
        FOR R:=1 UNTIL N_INS DO
          BEGIN
          J:=J+1;
          NEW_PTS_TO_HIT(J):=PTS_TO_HIT(I-1)+R*DEL;
          NEW_OBS_STATUS(J):=0;
          END;
        J:=J+1;
        NEW_PTS_TO_HIT(J):=PTS_TO_HIT(I);
        NEW_OBS_STATUS(J):=OBS_STATUS(I);
        K:=K+2;
        END
      ELSE
        BEGIN
        COMMENT COPYING OF OLD POINT;
        J:=J+1;
        NEW_PTS_TO_HIT(J):=PTS_TO_HIT(I);
        NEW_OBS_STATUS(J):=OBS_STATUS(I);
        END;
      N_PTS_HIT:=J;
      COMMENT COPY NEW ARRAYS;
      FOR I:=1 UNTIL N_PTS_HIT DO
        BEGIN
        OBS_STATUS(I):=NEW_OBS_STATUS(I);
        PTS_TO_HIT(I):=NEW_PTS_TO_HIT(I);
        END;
      END
    END;
  END HITPTS.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

              Trapezoidal integration procedure

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE TRAP(LONG REAL ARRAY PAR(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY STATE(*,*);
LONG REAL ARRAY JACOBIAN(*,*);
INTEGER VALUE N_STATE,N_PTS_HIT,N_PAR,AUX_INT;
PROCEDURE FUN;
```

```
INTEGER RESULT EFLAG;
INTEGER VALUE OUTPUT);
  BEGIN
  LONG REAL ARRAY PREV_G,G,PREV_Y,Y(1::N_STATE);
  LONG REAL ARRAY PREV_DGP,DGP,PREV_SENSE,
  SENSE(1::N_STATE,1::N_PAR);
  LONG REAL ARRAY PREV_DGY,DGY(1::N_STATE,1::N_STATE);
  LONG REAL T,PREV_T;
  INTEGER OUT_COUNT,M;
  LONG REAL CRIT;
  PROCEDURE INT_STEP;
    BEGIN
    COMMENT INTEGRATE WITH TRAP. RULE FROM PREV_T TO T;
    COMMENT IF AUX_INT IS NOT 0, INTEGRATE SENSITIVITY EQNS;
    LONG REAL ARRAY RES(1::N_STATE);
    LONG REAL ARRAY X,DGY_DYP,B(1::N_STATE,1::N_PAR);
    LONG REAL ARRAY TMP,A(1::N_STATE,1::N_STATE);
    INTEGER ARRAY IPERM(1::2*N_STATE);
    LONG REAL H,SUM,SSQD,SSQ;
    INTEGER MAXIT;
    COMMENT %EXTERNAL FSLE_AL;
    MAXIT:=12;
    H:=T-PREV_T;
    COMMENT NEWTON METHOD ON NONLINEAR SYSTEM FOR TIME STEP;
    FOR I:=1 UNTIL N_STATE DO Y(I):=PREV_Y(I);
    FOR I:=1 UNTIL MAXIT DO
      BEGIN
      FUN(T,Y,PAR,1,G,DGY,DGP,SENSE);
      COMMENT FORM JACOBIAN FOR NONLINEAR SYSTEM;
      FOR I:=1 UNTIL N_STATE DO
      FOR J:=1 UNTIL N_STATE DO
      A(I,J):=-DGY(I,J)*H/2.;
      FOR I:=1 UNTIL N_STATE DO A(I,I):=1.+A(I,I);
      FOR I:=1 UNTIL N_STATE DO
      B(I,1):=-Y(I)+PREV_Y(I)+H/2.*(G(I)+PREV_G(I));
      FSLE_AL(N_STATE,N_STATE,A,1,N_STATE,
      B,X,IPERM,N_STATE,TMP);
      COMMENT UPDATE AND GET SUP NORM;
      SSQD:=0.;
      SSQ:=0.;
      FOR I:=1 UNTIL N_STATE DO
        BEGIN
        SSQD:=SSQD+X(I,1)**2;
        Y(I):=X(I,1)+Y(I);
        SSQ:=SSQ+Y(I)**2;
        END;
      IF SSQ<1.'-3L THEN SSQ:=1.'-3L;
      IF (SSQD/SSQ)<(CRIT**2) THEN GO TO OUT;
      END;
    COMMENT %OUTPUT TO USER
    ABNORMAL EXIT IN NEWTON IN INTEGRATION STEP;
```

```
  EFLAG:=1;
  GO TO STOP;
  OUT:IF AUX_INT~=0 THEN
    BEGIN
    COMMENT THIS BLOCK SOLVES AUXILIARY LINEAR PROBLEMS;
    FUN(T,Y,PAR,2,G,DGY,DGP,SENSE);
    FOR I:=1 UNTIL N_STATE DO
    FOR J:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0.;
      FOR K:=1 UNTIL N_STATE DO
      SUM:=SUM+PREV_DGY(I,K)*PREV_SENSE(K,J);
      DGY_DYP(I,J):=SUM;
      END;
    COMMENT SET UP RIGHT HAND SIDES FOR DISCRETE LINEAR PROB;
    FOR I:=1 UNTIL N_STATE DO
    FOR J:=1 UNTIL N_PAR DO
    B(I,J):=PREV_SENSE(I,J)+H/2.*(PREV_DGP(I,J)
    +DGY_DYP(I,J)+DGP(I,J));
    COMMENT SOLVE SYSTEMS;
    FSLE_AL(N_STATE,N_STATE,A,N_PAR,N_STATE,B,SENSE,IPERM,
    N_STATE,TMP);
    END;
  END INT_STEP;
COMMENT MAIN PROGRAM STARTS HERE;
CRIT:=1.'-5L;
EFLAG:=0;
OUT_COUNT:=0;
M:=0;
COMMENT SET UP INITIAL CONDITIONS;
PREV_T:=PTS_TO_HIT(1);
FUN(PREV_T,PREV_Y,PAR,3,G,DGY,DGP,PREV_SENSE);
FOR I:=1 UNTIL N_STATE DO STATE(I,1):=PREV_Y(I);
FUN(PREV_T,PREV_Y,PAR,1,PREV_G,
PREV_DGY,DGP,SENSE);
IF AUX_INT~=0 THEN
  BEGIN
  FUN(PREV_T,PREV_Y,PAR,2,PREV_G,
  PREV_DGY,PREV_DGP,SENSE);
  FOR J:=1 UNTIL N_STATE DO
    BEGIN
    M:=M+1;
    FOR L:=1 UNTIL N_PAR DO
    JACOBIAN(L,M):=PREV_SENSE(J,L);
    END;
  END;
FOR I:=2 UNTIL N_PTS_HIT DO
  BEGIN
  T:=PTS_TO_HIT(I);
  INT_STEP;
  COMMENT OUTPUT OPTION;
```

```
    IF OUTPUT~=0 THEN
      BEGIN
      OUT_COUNT:=OUT_COUNT+1;
      IF OUT_COUNT=OUTPUT THEN
        BEGIN
        COMMENT OUTPUT TO -SC1
        TIME AND STATE VARIABLES;
        OUT_COUNT:=0;
        END;
      END;
    COMMENT INSERT DATA INTO STATE AND UPDATE;
    FOR J:=1 UNTIL N_STATE DO
      BEGIN
      STATE(J,I):=PREV_Y(J):=Y(J);
      PREV_G(J):=G(J);
      PREV_T:=T;
      FOR K:=1 UNTIL N_STATE DO PREV_DGY(J,K):=DGY(J,K);
      END;
    COMMENT INSERT DATA INTO JACOBIAN IF REQUIRED;
    IF AUX_INT~=0 THEN
      BEGIN
      FOR J:=1 UNTIL N_STATE DO
      FOR K:=1 UNTIL N_PAR DO
        BEGIN
        PREV_SENSE(J,K):=SENSE(J,K);
        PREV_DGP(J,K):=DGP(J,K);
        END;
      FOR J:=1 UNTIL N_STATE DO
        BEGIN
        M:=M+1;
        FOR L:=1 UNTIL N_PAR DO
        JACOBIAN(L,M):=SENSE(J,L);
        END;
      END;
    END;
  STOP:END TRAP.


     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

   Sensitivity equation solution and interface to Gear's code

     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE GEAR(LONG REAL ARRAY P(*);
LONG REAL ARRAY PTS(*);
LONG REAL ARRAY Y_PTS(*,*);
LONG REAL ARRAY JAC(*,*);
INTEGER VALUE N,N_PTS;
LONG REAL VALUE EPS,HMIN,HMAX;
INTEGER VALUE N_PAR,AUX_INT;
```

```
PROCEDURE FUN;
INTEGER RESULT KFLAG;
INTEGER VALUE OUTPUT;
INTEGER VALUE METHOD_FLAG;
INTEGER VALUE INIT;
LONG REAL ARRAY INITY(*);
LONG REAL ARRAY INITYP(*,*));
  BEGIN
  REAL ARRAY PW(1::N*N);
  LONG REAL ARRAY LY(1::8,1::N);
  LONG REAL ARRAY LSAVE(1::10,1::N+3);
  COMMENT ASSUMING N IS AT MOST 15 IN DECLARATION OF SAVE;
  LONG REAL ARRAY LYMAX,LERROR(1::N);
  LONG REAL LT,LH,LHMIN,LHMAX,LEPS,LNEXT_PT,HP1,HP2,LTS;
  INTEGER JSTART,MAXDER,MAX_STEP,STEP_COUNT,PT;
  INTEGER OUT_COUNT;
  LONG REAL ARRAY G(1::N);
  LONG REAL ARRAY DGY(1::N,1::N);
  LONG REAL ARRAY DGP,SENSE(1::N,1::N_PAR);
  LONG REAL ARRAY Z(1::N);
  COMMENT DECLARATIONS FOR HERMITE INTERPOLATION;
  LONG REAL ARRAY HERM_SAVE(1::2,1::N,1::1+N_PAR);
  LONG REAL ARRAY INTERP(1::4,1::N,1::1+N_PAR);
  COMMENT DECLARATIONS FOR AUXILIARY INTEGRATIONS;
  LONG REAL ARRAY STORE_DER(1::8,1::N,1::N_PAR);
  LONG REAL ARRAY AA(1::8);
  INTEGER PASS,ORDER;
  INTEGER MJAC;
  COMMENT %EXTERNAL DIFF;
  LONG REAL PROCEDURE POLYVAL(LONG REAL VALUE X;
  LONG REAL ARRAY COEF(*);
  INTEGER VALUE N_COEF);
    BEGIN
    LONG REAL Y;
    Y:=0.;
    FOR I:=N_COEF STEP -1 UNTIL 2 DO
    Y:=X*(COEF(I)+Y);
    Y+COEF(1)
    END POLYVAL;
  PROCEDURE INTERPOLATE_COEF;
    BEGIN
    LONG REAL ARRAY COEF(1::4);
    LONG REAL ARRAY DATA(1::6);
    PROCEDURE HERMITE(LONG REAL ARRAY DATA(*);
    LONG REAL ARRAY COEF(*));
      BEGIN
      COMMENT CUBIC HERMITE INTERPOLATION. DATA
      CONTAINS ABSCISSAE, ORDINATES, AND
      SLOPES IN PAIRS.  COEF CONTAINS THE
      COEFFICIENTS STARTING WITH THE CONSTANT
      TERM IN THE CUBIC;
```

```
      COEF(4):=(DATA(5)+DATA(6))/(DATA(2)-DATA(1))**2
      -2.0L*(DATA(4)-
      DATA(3))/(DATA(2)-DATA(1))**3;
      COEF(3):=(DATA(6)-DATA(5))/(2.0L*(DATA(2)-DATA(1)))
      -1.5L*(DATA(2)+DATA(1))*COEF(4);
      COEF(2):=DATA(5)-3.0L*DATA(1)**2*COEF(4)
      -2.0L*DATA(1)*COEF(3);
      COEF(1):=DATA(3)-DATA(1)**3*COEF(4)-DATA(1)**2*COEF(3)
      -DATA(1)*COEF(2);
      END HERMITE;
    DATA(1):=LT-HP1;
    DATA(2):=LT;
    FOR I:=1 UNTIL N DO
      BEGIN
      DATA(3):=HERM_SAVE(1,I,1);
      DATA(4):=LY(1,I);
      DATA(5):=HERM_SAVE(2,I,1);
      DATA(6):=LY(2,I)/HP1;
      HERMITE(DATA,INTERP(*,I,1));
      IF AUX_INT~=0 THEN
      FOR K:=1 UNTIL N_PAR DO
        BEGIN
        COMMENT INTERPOLATION ON AUXILIARY PROBLEMS;
        DATA(3):=HERM_SAVE(1,I,K+1);
        DATA(4):=STORE_DER(1,I,K);
        DATA(5):=HERM_SAVE(2,I,K+1);
        DATA(6):=STORE_DER(2,I,K)/HP1;
        HERMITE(DATA,INTERP(*,I,K+1));
        END;
      END;
    END INTERPOLATE_COEF;
COMMENT
MANAGEMENT PROGRAM STARTS HERE;
OUT_COUNT:=0; PASS:=1; MJAC:=0;
MAX_STEP:=200;
COMMENT ONLY USE UP TO FOURTH ORDER METHOD
CONSISTENT WITH INTERPOLATION;
MAXDER:=4; LHMAX:=HMAX; LHMIN:=HMIN;
LEPS:=EPS; PT:=2;
LH:=(PTS(2)-PTS(1))/3;
COMMENT MAKE SURE STARTING H IS LESS THAN .1;
IF LH>.1 THEN LH:=.1;
COMMENT FOR USE WHEN AUX_INT~=0;
HP2:=LH;
COMMENT INITIALIZE;
IF INIT=1 THEN
  BEGIN
  FOR I:=1 UNTIL N DO Z(I):=INITY(I);
  END
ELSE
FUN(PTS(1),Z,P,-3,G,DGY,DGP,SENSE);
```

```
FOR I:=1 UNTIL N DO
  BEGIN
  LY(1,I):=Z(I);
  Y_PTS(I,1):=Z(I);
  LYMAX(I):=1.;
  END;
COMMENT INITIALIZE NON JACOBIAN PART OF HERM_SAVE;
FOR I:=1 UNTIL N DO HERM_SAVE(1,I,1):=Z(I);
FUN(PTS(1),Z,P,-1,G,DGY,DGP,SENSE);
FOR I:=1 UNTIL N DO HERM_SAVE(2,I,1):=G(I);
LT:=PTS(1);
STEP_COUNT:=0;
LNEXT_PT:=PTS(PT);
JSTART:=0;
COMMENT USE MULTISTEP METHOD SUITABLE FOR STIFF PROBLEMS;
WHILE PT<=N_PTS DO
  BEGIN
  STEP_COUNT:=STEP_COUNT+1;
  IF STEP_COUNT>MAX_STEP THEN
    BEGIN
    COMMENT %OUTPUT TO USER
    OVER (MAX_STEP) STEPS USED IN GEAR--GAVE UP;
    GO TO STOP;
    END;
  COMMENT FOR USE WITH AUX_INT~=0;
  LTS:=LT;
  DIFF(N,LT,LY,LSAVE,LH,LHMIN,LHMAX,LEPS,METHOD_FLAG,LYMAX,
  LERROR,KFLAG,JSTART,MAXDER,AA,ORDER,FUN,P,PW);
  COMMENT PROCESS COMPLETION CODE;
  IF KFLAG<0 THEN
    BEGIN
    CASE ABS KFLAG OF
      BEGIN
      COMMENT %OUTPUT
      PRINT TO USER LINE INDEXED BY KFLAG
      H=HMIN USED, ERROR NOT ATTAINED
      MAX ORDER SPECIFIED IS TOO LARGE
      NO CORRECTOR COMVERGENCE FOR H>HMIN
      REQUESTED ERROR TOO SMALL FOR PROB.;
      END;
    IF OUTPUT~=0 THEN
    CASE ABS KFLAG OF
      BEGIN
      COMMENT %OUTPUT TO -SC1 THE SAME MESSAGE
      AS SENT TO USER IN LAST OUTPUT STATEMENT;
      END;
    GO TO STOP;
    END;
  HP1:=LT-LTS;
  LTS:=LT;
  IF AUX_INT~=0 THEN
```

```
BEGIN
INTEGER ARRAY IPERM(1::2*N);
LONG REAL ARRAY TMPR,U(1::N,1::N);
LONG REAL ARRAY B,F_PRED(1::N,1::N_PAR);
COMMENT %EXTERNAL FSLE_AL;
COMMENT AUXILIARY SYSTEMS BEING INTEGRATED;
IF PASS=1 THEN
  BEGIN
  COMMENT FIRST PASS---INITIALIZE VARIABLES;
  IF INIT=1 THEN
    BEGIN
    FOR J:=1 UNTIL N DO
    FOR K:=1 UNTIL N_PAR DO
    SENSE(J,K):=INITYP(J,K);
    END
  ELSE
  FUN(PTS(1),Y_PTS(1,*),P,3,G,DGY,DGP,SENSE);
  FOR J:=1 UNTIL N DO
  FOR K:=1 UNTIL N_PAR DO
  STORE_DER(1,J,K):=SENSE(J,K);
  FOR J:=1 UNTIL N DO
    BEGIN
    MJAC:=MJAC+1;
    FOR L:=1 UNTIL N_PAR DO
    JAC(L,MJAC):=STORE_DER(1,J,L);
    END;
  COMMENT INITIALIZE FIRST DERIVATIVE PART OF STORE_DER;
  FUN(PTS(1),Y_PTS(1,*),P,1,G,DGY,DGP,SENSE);
  FUN(PTS(1),Y_PTS(1,*),P,2,G,DGY,DGP,SENSE);
  FOR K:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL N DO
    BEGIN
    LONG REAL SUM;
    SUM:=0.;
    FOR J:=1 UNTIL N DO
    SUM:=SUM+DGY(I,J)*STORE_DER(1,J,K);
    STORE_DER(2,I,K):=HP2*(SUM+DGP(I,K));
    END;
  COMMENT INITIALIZE PART OF HERM_SAVE FOR JACOBIAN;
  FOR J:=1 UNTIL N DO
  FOR K:=1 UNTIL N_PAR DO
    BEGIN
    HERM_SAVE(1,J,K+1):=STORE_DER(1,J,K);
    HERM_SAVE(2,J,K+1):=STORE_DER(2,J,K)/HP2;
    END;
  PASS:=2;
  END;
IF HP1~=HP2 THEN
  BEGIN
  COMMENT STEP HAS CHANGED AND STORED DERIVATIVES
  NEED RESCALING;
```

```
  LONG REAL RATIO,FACTOR;
  FACTOR:=1.L;
  RATIO:=HP1/HP2;
  FOR I:=2 UNTIL ORDER+1 DO
    BEGIN
    FACTOR:=FACTOR*RATIO;
    FOR J:=1 UNTIL N DO
    FOR K:=1 UNTIL N_PAR DO
    STORE_DER(I,J,K):=STORE_DER(I,J,K)*FACTOR;
    END;
  END;
COMMENT UPDATE HP2;
HP2:=HP1;
COMMENT FIND PREDICTED VALUE BY MULTIPLYING STORE_DER BY
PASCAL TRIANGLE MATRIX;
FOR K:=1 UNTIL N_PAR DO
FOR J:=2 UNTIL ORDER+1 DO
FOR J1:=J UNTIL ORDER+1 DO
  BEGIN
  INTEGER J2;
  J2:=ORDER-J1+J;
  FOR I:=1 UNTIL N DO
  STORE_DER(J2,I,K):=STORE_DER(J2,I,K)
  +STORE_DER(J2+1,I,K);
  END;
COMMENT SAVE HP1 TIMES PREDICTED
R.H.S. TO DE'S IN F_PRED;
FOR J:=1 UNTIL N DO
FOR K:=1 UNTIL N_PAR DO
F_PRED(J,K):=STORE_DER(2,J,K);
COMMENT SET UP R.H. SIDES FOR ALGEBRAIC LINEAR SYSTEMS;
FOR J:=1 UNTIL N DO
FOR K:=1 UNTIL N_PAR DO
B(J,K):=STORE_DER(1,J,K)-AA(1)*STORE_DER(2,J,K);
COMMENT ADD IN INHOMOGENEOUS TERMS;
FOR I:=1 UNTIL N DO Z(I):=LY(1,I);
FUN(LT,Z,P,2,G,DGY,DGP,SENSE);
FOR K:=1 UNTIL N_PAR DO
FOR J:=1 UNTIL N DO
B(J,K):=B(J,K)+HP1*AA(1)*DGP(J,K);
COMMENT SET UP MATRIX FOR LINEAR PROBLEMS;
FOR I:=1 UNTIL N DO Z(I):=LY(1,I);
FUN(LT,Z,P,1,G,DGY,DGP,SENSE);
FOR I:=1 UNTIL N DO
FOR J:=1 UNTIL N DO
U(I,J):=-HP1*AA(1)*DGY(I,J);
FOR I:=1 UNTIL N DO U(I,I):=U(I,I)+1.;
COMMENT SOLVE LINEAR SYSTEMS;
FSLE_AL(N,N,U,N_PAR,N,B,SENSE,IPERM,N,TMPR);
COMMENT STORE SOLUTION AT NEW TIME IN STORE_DER;
FOR J:=1 UNTIL N DO
```

```
  FOR K:=1 UNTIL N_PAR DO
  STORE_DER(1,J,K):=SENSE(J,K);
  COMMENT FIND R.H.S. OF DE'S AT NEW POINT;
  FOR K:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL N DO
    BEGIN
    LONG REAL SUM;
    SUM:=0.;
    FOR J:=1 UNTIL N DO
    SUM:=SUM+DGY(I,J)*STORE_DER(1,J,K);
    STORE_DER(2,I,K):=HP1*(SUM+DGP(I,K));
    END;
  COMMENT CORRECT HIGHER DERIVATIVES;
  IF ORDER>1 THEN
  FOR I:=3 UNTIL ORDER+1 DO
  FOR J:=1 UNTIL N DO
  FOR K:=1 UNTIL N_PAR DO
  STORE_DER(I,J,K):=STORE_DER(I,J,K)+AA(I)
  *(STORE_DER(2,J,K)-F_PRED(J,K));
  COMMENT IF REQUIRED INSERT NEXT HIGHER DERIVATIVES
  IN STORE_DER;
  IF JSTART>ORDER THEN
  FOR J:=1 UNTIL N DO
  FOR K:=1 UNTIL N_PAR DO
  STORE_DER(ORDER+2,J,K):=AA(ORDER+1)*(STORE_DER(2,J,K)
  -F_PRED(J,K))/(ORDER+1);
  END;
COMMENT OUTPUT OPTION;
IF OUTPUT~=0 THEN
  BEGIN
  OUT_COUNT:=OUT_COUNT+1;
  IF OUT_COUNT=OUTPUT THEN
    BEGIN
    LONG REAL ARRAY V(1::N);
    LONG REAL TIME;
    FOR I:=1 UNTIL N DO V(I):=LY(1,I);
    TIME:=LT;
    COMMENT %OUTPUT TO -SC1
    TIME AND STATE VARIABLES;
    OUT_COUNT:=0;
    END;
  END;
COMMENT CHECK IF PASSED A POINT;
IF LT>=LNEXT_PT THEN
  BEGIN
  LONG REAL E;
  INTERPOLATE_COEF;
  WHILE LT>=LNEXT_PT DO
    BEGIN
    E:=(PTS(PT)-LT)/HP1;
    COMMENT INSERT DATA INTO Y_PTS;
```

```
        FOR I:=1 UNTIL N DO
        Y_PTS(I,PT):=POLYVAL(PTS(PT),INTERP(*,I,1),4);
        COMMENT INSERT DATA INTO JACOBIAN IF REQUIRED;
        IF AUX_INT~=0 THEN
        FOR J:=1 UNTIL N DO
          BEGIN
          MJAC:=MJAC+1;
          FOR L:=1 UNTIL N_PAR DO
          JAC(L,MJAC):=POLYVAL(PTS(PT),INTERP(*,J,L+1),4);
          END;
        PT:=PT+1;
        IF PT>N_PTS THEN GO TO STOP;
        LNEXT_PT:=PTS(PT);
        END;
      STEP_COUNT:=0;
      END;
    COMMENT UPDATE HERM_SAVE;
    FOR J:=1 UNTIL N DO
      BEGIN
      HERM_SAVE(1,J,1):=LY(1,J);
      HERM_SAVE(2,J,1):=LY(2,J)/HP1;
      END;
    IF AUX_INT~=0 THEN
    FOR J:=1 UNTIL N DO
    FOR K:=1 UNTIL N_PAR DO
      BEGIN
      HERM_SAVE(1,J,K+1):=STORE_DER(1,J,K);
      HERM_SAVE(2,J,K+1):=STORE_DER(2,J,K)/HP1;
      END;
    COMMENT RESET JSTART;
    JSTART:=1;
    END;
  STOP:END GEAR.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                        Marquardt procedure

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE MARQUARDT(LONG REAL VALUE EPS_R,EPS_A;
INTEGER VALUE M,N; PROCEDURE FUNC;
LONG REAL ARRAY P(*); LONG REAL VALUE LAM;
PROCEDURE SVD_AL);
  BEGIN
  COMMENT MARQUARDT-LEVENBERG TECHNIQUE--AN ADAPTATION OF
  THE VERSION USED BY BARD (1970), SIAM J. NUMER. ANAL. 7,
  157-186--OPTIONAL SCALING OF PROBLEM IF EPS_R, EPS_A ARE
  BOTH NEGATIVE;
  LONG REAL ARRAY UD,JAC(1::M,1::N);
```

```
LONG REAL ARRAY RES,RES1(1::M);
LONG REAL ARRAY P1,S,GRAD,GRAD1,AUX_VEC,Z,DELTA,C(1::N);
LONG REAL ARRAY VD(1::N,1::N);
INTEGER ERROR_COUNT,MAXIT,MAXINT,MAXERR,EFLAG,DECLAM;
LONG REAL F,F1,SUM; LOGICAL CONV,SCALE;
LONG REAL PROCEDURE COSINE(LONG REAL ARRAY V1(*);
LONG REAL ARRAY V2(*));
  BEGIN
  COMMENT FIND COSINE OF ANGLE BETWEEN V1 AND V2;
  LONG REAL S1,S2,S3;
  S1:=0.L; S2:=0.L; S3:=0.L;
  FOR I:=1 UNTIL N DO
    BEGIN
    S1:=S1-V1(I)*V2(I);
    S2:=S2+V1(I)**2;
    S3:=S3+V2(I)**2;
    END;
  S1/LONGSQRT(S2*S3)
  END COSINE;
COMMENT INITIALIZE AND SET BOUNDS ON EFFORT IN OPTIMIZATION
ATTEMPT;
MAXIT:=25; MAXINT:=4; MAXERR:=3; CONV:=FALSE;
ERROR_COUNT:=0;
DECLAM:=0;
IF EPS_R<0 AND EPS_A<0 THEN
  BEGIN
  SCALE:=TRUE;
  EPS_R:=-EPS_R;EPS_A:=-EPS_A;
  COMMENT %OUTPUT TO USER, -SC1
  SCALING USED IN MARQUARDT;
  END
ELSE
  BEGIN
  SCALE:=FALSE;
  FOR I:=1 UNTIL N DO C(I):=1.L;
  END;
COMMENT INITIAL LEAST SQUARES FUNCTION EVALUATION;
FUNC(P,F,RES,JAC,GRAD,EFLAG);
IF EFLAG=1 THEN
  BEGIN
  COMMENT HANDLE ERROR RETURN
  %OUTPUT TO USER, -SC1
  ERROR IN FIRST FUNCTION CALL IN MARQUARDT;
  WRITE("ERROR IN FIRST FUNC CALL IN MARQUARDT");
  GO TO STOP;
  END;
COMMENT
%OUTPUT TO USER, -SC1
INITIAL SUM OF SQUARES (F);
COMMENT %OUTPUT TO USER
INITIAL SUM OF SQUARES IS (F);
```

```
FOR ITER:=1 UNTIL MAXIT DO
  BEGIN
  IF SCALE=TRUE THEN
  BEGIN
  LONG REAL SUM;
  FOR J:=1 UNTIL N DO
    BEGIN
    SUM:=0.L;
    FOR I:=1 UNTIL M DO
    SUM:=SUM+JAC(I,J)**2;
    C(J):=LONGSQRT(SUM);
    END;
    FOR J:=1 UNTIL N DO
    FOR I:=1 UNTIL M DO
    JAC(I,J):=JAC(I,J)/C(J);
    FOR I:=1 UNTIL N DO
    GRAD(I):=GRAD(I)/C(I);
    END;
  COMMENT FIND SINGULAR VALUE DECOMPOSITION OF JAC;
  SVD_AL(JAC,S,UD,VD,M,N,M,N,0,N,N);
  IF ITER=1 THEN
    BEGIN
    COMMENT INITIALIZE LAM;
    LAM:=(IF LAM<0 THEN .01L ELSE LAM);
    END;
  COMMENT PREPARE FOR AN ITERATION;
  FOR I:=1 UNTIL N DO
    BEGIN
    SUM:=0.L;
    FOR J:=1 UNTIL M DO SUM:=SUM+UD(J,I)*RES(J);
    AUX_VEC(I):=SUM*S(I);
    END;
  COMMENT DETERMINE LAM;
  REDO:FOR I:=1 UNTIL N DO
  Z(I):=AUX_VEC(I)/(S(I)**2+LAM);
  FOR I:=1 UNTIL N DO
    BEGIN
    SUM:=0.L;
    FOR J:=1 UNTIL N DO SUM:=SUM+VD(I,J)*Z(J);
    DELTA(I):=-SUM;
    END;
  FOR I:=1 UNTIL N DO P1(I):=P(I)+DELTA(I)/C(I);
  COMMENT
  %OUTPUT TO -SC1
  TRIAL LAMBDA (LAM), TRIAL PARAMETER VECTOR (P1);
  COMMENT FIND LEAST SQUARES FUNCTION AT TRIAL PARAMETERS;
  FUNC(P1,F1,RES1,JAC,GRAD1,EFLAG);
  IF EFLAG=1 THEN
    BEGIN
    COMMENT HANDLE ERROR RETURN;
    ERROR_COUNT:=ERROR_COUNT+1;
```

```
   IF ERROR_COUNT=MAXERR THEN
     BEGIN
     COMMENT
     %OUTPUT TO USER, -SC1
     (MAXERR) FUNCTION ERRORS;
     GO TO STOP;
     END
   ELSE
     BEGIN
     LAM:=LAM*10.L;
     DECLAM:=0;
     GO TO REDO;
     END;
   END;
ERROR_COUNT:=0;
COMMENT
%OUTPUT TO  -SC1
TRIAL SUM OF SQUARES (F1);
IF F1<F THEN
COMMENT DECREASE LAM AFTER TWO SUCCESSFUL
FUNCTION REDUCTIONS;
  BEGIN
  DECLAM:=DECLAM+1;
  IF DECLAM=2 THEN
    BEGIN
    LAM:=(IF (.1L*LAM>1.'-10L) THEN .1L*LAM ELSE 1.'-10L);
    DECLAM:=0;
    END;
  COMMENT CHECK FOR CONVERGENCE;
  IF (F-F1<EPS_R*F1+EPS_A) OR (F1<EPS_A) THEN CONV:=TRUE;
  GO TO UPDATE;
  END
ELSE
  BEGIN
  IF (COSINE(DELTA,GRAD))<.707 THEN
    BEGIN
    COMMENT INCREASE LAM;
    LAM:=10.L*LAM;
    DECLAM:=0;
    GO TO REDO;
    END
  ELSE
    BEGIN
    COMMENT DIRECTION TOO CLOSE TO STEEPEST DESCENT,
    THEREFORE DO NOT INCREASE LAM, BUT INTERPOLATE;
    LONG REAL GA,R0,R1,W1;
    COMMENT
    %OUTPUT TO USER, -SC1
    INTERPOLATING;
    GA:=0.L;
    R0:=1.L;
```

```
      FOR I:=1 UNTIL N DO
      GA:=GA+DELTA(I)*GRAD(I);
      FOR INTERP:=1 UNTIL MAXINT DO
        BEGIN
        R1:=GA*R0**2/(2.L*(GA*R0+F-F1));
        W1:=(IF (.75L*R0<R1) THEN .75L*R0 ELSE R1);
        R0:=(IF (.25L*R0>W1) THEN .25L*R0 ELSE W1);
        REPEAT:FOR I:=1 UNTIL N DO
        P1(I):=P(I)+R0*DELTA(I)/C(I);
        COMMENT
        %OUTPUT TO -SC1
        TRIAL PARAMETER VECTOR (P1);
        COMMENT FUNCTION EVALUATION AT TRIAL PARAMETERS;
        FUNC(P1,F1,RES1,JAC,GRAD1,EFLAG);
        IF EFLAG=1 THEN
          BEGIN
          COMMENT HANDLE ERROR RETURN;
          ERROR_COUNT:=ERROR_COUNT+1;
          IF ERROR_COUNT=MAXERR THEN
            BEGIN
            COMMENT
            %OUTPUT TO USER, -SC1
            (MAXERR) FUNCTION ERRORS IN INTERPOLATION PART;
            GO TO STOP;
            END
          ELSE
            BEGIN
            R0:=R0*.5L;
            GO TO REPEAT;
            END;
          END;
        ERROR_COUNT:=0;
        COMMENT
        %OUTPUT TO USER, -SC1
        TRIAL SUM OF SQUARES (F1);
        IF F1<F THEN
          BEGIN
          IF (F-F1<EPS_R*F1+EPS_A)
          OR (F1<EPS_A) THEN CONV:=TRUE;
          GO TO UPDATE;
          END;
        END INTERP;
      COMMENT
      %OUTPUT TO USER, -SC1
      (MAXINT) INTERPOLATIONS TRIED, NO REDUCTION
      IN SUM OF SQUARES;
      GO TO STOP;
      END;
    END;
  UPDATE:
  FOR I:=1 UNTIL N DO
```

```
      BEGIN
      P(I):=P1(I);
      GRAD(I):=GRAD1(I);
      END;
    F:=F1;
    COMMENT %OUTPUT TO USER
    NEW SUM OF SQUARES IS (F);
    FOR I:=1 UNTIL M DO RES(I):=RES1(I);
    IF CONV=TRUE THEN GO TO FINISHED;
    END ITER;
  COMMENT
  %OUTPUT TO USER, -SC1
  OVER (MAXIT) ITERATIONS REQUIRED;    GO TO STOP;
  FINISHED:
  COMMENT
  %OUTPUT TO USER, -SC1
  FINAL PARAMETERS (P), FINAL GRADIENT (GRAD),
  FINAL SUM OF SQUARES (F);
  STOP:END MARQUARDT.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                    Interactive optimization

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE INTERACTIVE_OPT(INTEGER VALUE M,N;
PROCEDURE FUNC;
LONG REAL ARRAY P(*);
PROCEDURE SVD_AL;
LONG REAL ARRAY OBS_PLACE(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY STATE(*,*);
LONG REAL ARRAY OBS(*,*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
LONG REAL ARRAY CREOBS(*);
INTEGER VALUE N_OBS,N_PTS_HIT,N_STATE_OBS
,N_STATE;
INTEGER ARRAY STATES_OBS(*);
INTEGER VALUE RESULT_PLOT_NUMBER;
PROCEDURE READ_CMD_DATA;
PROCEDURE CMD_AL;
PROCEDURE PLOT_COMMAND);
  BEGIN
  STRING(6) ARRAY COMMANDS(1::20);
  STRING(6) CMD;
  INTEGER CMD_CTR,CMD_NO,TECH,NP,EFLAG,ECT;
  LONG REAL LAMDA,DEL,F,TRIAL_F,SUM;
  LONG REAL ARRAY AUX_VEC,S,SP,TRIAL_SP,
  TRIAL_P,GRAD,TRIAL_GRAD(1::N);
```

```
INTEGER ARRAY FREEZE(1::N);
LOGICAL NUM,INIT,SUCCESS;
LONG REAL ARRAY RES,TRIAL_RES(1::M);
LONG REAL ARRAY TRIAL_JAC,JAC,AJAC,UD(1::M,1::N);
LONG REAL ARRAY VD(1::N,1::N);
COMMENT %EXTERNAL ECHO;
PROCEDURE ECHO_CHECK;
  BEGIN
  COMMENT ECHO 3270 CONVERSATION BUFFER;
  IF ECT>80 THEN
    BEGIN
    ECHO(2);
    END;
  ECT:=0;
  END ECHO_CHECK;
PROCEDURE MARQ_PREP;
  BEGIN
  COMMENT EXTRACT JACOBIAN AND PARAMETERS
  ON NONFROZEN COMPONENTS;
  INTEGER JJ;
  LONG REAL SUM;
  JJ:=0;
  FOR J:=1 UNTIL N DO
  IF FREEZE(J)=0 THEN
    BEGIN
    JJ:=JJ+1;
    FOR I:=1 UNTIL M DO AJAC(I,JJ):=JAC(I,J);
    SP(JJ):=P(J);
    END;
  NP:=JJ;
  SVD_AL(AJAC,S,UD,VD,M,N,M,NP,0,NP,NP);
  FOR I:=1 UNTIL NP DO
    BEGIN
    SUM:=0;
    FOR J:=1 UNTIL M DO
    SUM:=SUM+UD(J,I)*RES(J);
    AUX_VEC(I):=SUM;
    END;
  FOR I:=1 UNTIL NP DO
  AUX_VEC(I):=AUX_VEC(I)*S(I);
  END MARQ_PREP;
PROCEDURE INITIALIZE;
  BEGIN
  COMMENT INITIALIZE;
  FUNC(P,F,RES,JAC,GRAD,EFLAG);
  IF EFLAG=1 THEN
    BEGIN
    INIT:=FALSE;
    NUM:=FALSE;
    GO TO READ_CMD;
    END;
```

```
  CASE TECH OF
    BEGIN
      BEGIN
      COMMENT STEEPEST DESCENT CASE;
      COMMENT INITIALIZATION AUTOMATIC;
      END;
      BEGIN
      COMMENT MARQUARDT;
      MARQ_PREP;
      END
    END;
  INIT:=FALSE;
  END INITIALIZE;
PROCEDURE UPDATE;
  BEGIN
  CASE TECH OF
    BEGIN
      BEGIN
      COMMENT UPDATE FOR STEEPEST DESCENT;
      FOR I:=1 UNTIL N DO
        BEGIN
        P(I):=TRIAL_P(I);
        GRAD(I):=TRIAL_GRAD(I);
        END;
      F:=TRIAL_F;
      END;
      BEGIN
      COMMENT UPDATE FOR MARQUARDT;
      F:=TRIAL_F;
      FOR I:=1 UNTIL N DO
        BEGIN
        P(I):=TRIAL_P(I);
        GRAD(I):=TRIAL_GRAD(I);
        END;
      FOR I:=1 UNTIL M DO
      RES(I):=TRIAL_RES(I);
      FOR J:=1 UNTIL N DO
      FOR I:=1 UNTIL M DO
      JAC(I,J):=TRIAL_JAC(I,J);
      MARQ_PREP;
      END
    END;
  SUCCESS:=FALSE;
  END UPDATE;
COMMENT INITIALIZE;
ECT:=0;
ECHO(1);
COMMENT %OUTPUT TO USER -REPRT
######INTERACTIVE OPTIMIZATION ATTEMPT######;
INIT:=TRUE;
LAMDA:=.01;
```

```
TECH:=2;
SUCCESS:=FALSE;
NUM:=FALSE;
FOR I:=1 UNTIL N DO FREEZE(I):=0;
COMMENT SET UP COMMAND DESIGNATORS;
COMMANDS(1):="T      ";
COMMANDS(2):="M      ";
COMMANDS(3):="N      ";
COMMANDS(4):="V      ";
COMMANDS(5):="F      ";
COMMANDS(6):="DF     ";
COMMANDS(7):="SET    ";
COMMANDS(8):="Q      ";
COMMANDS(9):="PLOT   ";
COMMANDS(10):="END    ";
COMMENT READ COMMAND DESIGNATOR AND IDENTIFY;
READ_CMD:
COMMENT %OUTPUT TO USER
****ENTER OPTIMIZATION COMMAND****;
ECHO_CHECK;
COMMENT %INPUT (CMD);
CMD_CTR:=1;
WHILE COMMANDS(CMD_CTR)~="END    " DO
  BEGIN
  IF COMMANDS(CMD_CTR)=CMD THEN
    BEGIN CMD_NO:=CMD_CTR;
    GO TO OUT;
    END;
  CMD_CTR:=CMD_CTR+1;
  END;
COMMENT %OUTPUT TO USER
COMMAND IN ERROR RESPECIFY;
GO TO READ_CMD;
OUT:CASE CMD_NO OF
  BEGIN
    BEGIN
    COMMENT CHOICE OF TECHNIQUE COMMAND;
    STRING(6) MET;
    INTEGER TECH1;
    ECT:=ECT+1;
    TECH1:=TECH;
    COMMENT %INPUT (MET);
    IF MET="SD     " THEN TECH:=1
    ELSE
    IF MET="MARQ   " THEN TECH:=2
    ELSE
      BEGIN
      COMMENT %OUTPUT TO USER
      ERROR IN TECHNIQUE  SPECIFICATION;
      GO TO READ_CMD;
      END;
```

```
IF TECH~=TECH1 THEN INIT:=TRUE;
END;
BEGIN
COMMENT MULTIPLY OPTIMIZATION PARAMETER BY A FACTOR;
LONG REAL FACTOR;
ECT:=ECT+3;
COMMENT %INPUT (FACTOR);
CASE TECH OF
  BEGIN
  DEL:=DEL*FACTOR;
  LAMDA:=LAMDA*FACTOR
  END;
NUM:=TRUE;
END;
BEGIN
COMMENT NEW OPTIMIZATION PARAMETER;
ECT:=ECT+3;
CASE TECH OF
  BEGIN
    BEGIN
    COMMENT %INPUT (DEL);
    END;
    BEGIN
    COMMENT %INPUT (DEL);
    END
  END;
NUM:=TRUE;
END;
BEGIN
COMMENT VIEW COMMAND;
STRING(6) UNIT;
ECT:=ECT+10;
IF INIT=TRUE THEN INITIALIZE
ELSE IF SUCCESS=TRUE THEN UPDATE;
  BEGIN
  COMMENT %OUTPUT TO USER
  PARAMETERS, GRADIENT, FREEZING;
  CASE TECH OF
    BEGIN
      BEGIN
      COMMENT STEEPEST DESCENT;
      COMMENT %OUTPUT TO USER
      STEEPEST DESCENT CURRENT STEP (DEL);
      END;
      BEGIN
      COMMENT MARQUARDT DATA;
      COMMENT %OUTPUT TO USER
      MARQUARDT, (LAMDA), (S);
      END
    END;
  COMMENT %OUTPUT TO USER
```

```
  CURRENT SUM OF SQUARES (F);
  END;
END;
BEGIN
COMMENT FREEZE COMMAND;
INTEGER CMP;
IF SUCCESS=TRUE THEN UPDATE;
COMMENT %INPUT (CMP);
WHILE CMP~=0 DO
  BEGIN
  ECT:=ECT+1;
  FREEZE(CMP):=1;
  COMMENT %INPUT (CMP);
  END;
INIT:=TRUE;
END;
BEGIN
COMMENT DEFREEZE COMMAND;
INTEGER CMP;
IF SUCCESS=TRUE THEN UPDATE;
COMMENT %INPUT (CMP);
IF CMP=0 THEN
  BEGIN
  COMMENT REMOVE ALL FREEZING;
  FOR I:=1 UNTIL N DO FREEZE(I):=0;
  END
ELSE
WHILE CMP~=0 DO
  BEGIN
  ECT:=ECT+1;
  FREEZE(CMP):=0;
  COMMENT %INPUT (CMP);
  END;
INIT:=TRUE;
END;
BEGIN
COMMENT SET COMMAND;
INTEGER CMP;
IF SUCCESS=TRUE THEN UPDATE;
COMMENT %INPUT (CMP);
WHILE CMP~=0 DO
  BEGIN
  COMMENT %INPUT (P(CMP)), (CMP);
  ECT:=ECT+1;
  END;
INIT:=TRUE;
END;
BEGIN
COMMENT QUIT COMMAND;
IF SUCCESS=TRUE THEN UPDATE;
ECHO(3);
```

```
    GO TO STOP;
    END;
    BEGIN
    COMMENT PLOT COMMAND;
    PLOT_COMMAND( OBS_PLACE,
    PTS_TO_HIT,STATE,OBS,OBS_SMOOTH,P,
    N_OBS,N_PTS_HIT,N_STATE_OBS,N_STATE,N,
    STATES_OBS,PLOT_NUMBER,
    READ_CMD_DATA,
    CMD_AL,CREOBS);
    ECT:=ECT+25;
    END
  END;
COMMENT CARRY OUT NUMERICAL WORK IF REQUIRED;
IF NUM=TRUE THEN
  BEGIN
  IF INIT=TRUE THEN
  INITIALIZE
  ELSE IF SUCCESS=TRUE THEN
  UPDATE;
  CASE TECH OF
    BEGIN
      BEGIN
      COMMENT STEEPEST DESCENT ITERATION ATTEMPT;
      FOR I:=1 UNTIL N DO
      IF FREEZE(I)=0 THEN
      TRIAL_P(I):=P(I)-DEL*GRAD(I)
      ELSE
      TRIAL_P(I):=P(I);
      ECT:=ECT+3;
      COMMENT %OUTPUT
      TRIAL PARAMETERS (TRIAL_P);
      FUNC(TRIAL_P,TRIAL_F,TRIAL_RES,TRIAL_JAC
      ,TRIAL_GRAD,EFLAG);
      IF EFLAG=1 THEN
        BEGIN
        NUM:=FALSE;
        GO TO READ_CMD;
        END;
        BEGIN
        IF TRIAL_F<F THEN
          BEGIN
          COMMENT REDUCED RESIDUAL--TELL USER;
          ECT:=ECT+3;
          COMMENT %OUTPUT TO USER
          STEEPEST DESCENT STEP SUCCESSFUL
          NEW SUM OF SQUARES (TRIAL_F),
          CHANGE IN SUM OF SQUARES (F-TRIAL_F);
          SUCCESS:=TRUE;
          END
        ELSE
```

```
      BEGIN
      COMMENT STEP UNSUCCESSFUL;
      ECT:=ECT+3;
      COMMENT %OUTPUT TO USER
      STEEPEST DESCENT DID NOT REDUCE RESIDUAL
      NEW SUM OF SQUARES (TRIAL_F);
      END;
    COMMENT %OUTPUT TO USER
    STEP WAS (DEL);
    END;
END;
BEGIN
COMMENT MARQUARDT ITERATION ATTEMPT;
LONG REAL ARRAY Z(1::N);
INTEGER JJ;
FOR I:=1 UNTIL NP DO
Z(I):=AUX_VEC(I)/(S(I)**2+LAMDA);
FOR I:=1 UNTIL NP DO
  BEGIN
  SUM:=0;
  FOR J:=1 UNTIL NP DO SUM:=SUM+VD(I,J)*Z(J);
  TRIAL_SP(I):=SP(I)-SUM;
  END;
JJ:=0;
FOR I:=1 UNTIL N DO
  BEGIN
  IF FREEZE(I)=0 THEN
    BEGIN
    JJ:=JJ+1;
    TRIAL_P(I):=TRIAL_SP(JJ)
    END
  ELSE TRIAL_P(I):=P(I);
  END;
  BEGIN
  COMMENT %OUTPUT TO USER
  TRIAL PARAMETERS (TRIAL_P);
  ECT:=ECT+3;
  END;
FUNC(TRIAL_P,TRIAL_F,TRIAL_RES,
TRIAL_JAC,TRIAL_GRAD,EFLAG);
IF EFLAG=1 THEN
  BEGIN
  NUM:=FALSE;
  GO TO READ_CMD;
  END;
  BEGIN
  IF TRIAL_F<F THEN
    BEGIN
    COMMENT REDUCED RESIDUAL--TELL USER;
    ECT:=ECT+3;
    COMMENT %OUTPUT TO USER
```

```
            MARQUARDT STEP SUCCESSFUL
            NEW SUM OF SQUARES (TRIAL_F)
            CHANGE IN SUM OF SQUARES (T-TRIAL_F);
            SUCCESS:=TRUE;
            END
          ELSE
            BEGIN
            ECT:=ECT+3;
            COMMENT %OUTPUT TO USER
            MARQUARDT DID NOT REDUCE RESIDUAL
            NEW SUM OF SQUARES (TRIAL_F);
            END;
          COMMENT %OUTPUT TO USER
          LAMBDA IS (LAMDA);
          ECT:=ECT+1;
          END;
        END
      END;
    NUM:=FALSE;
    END;
  GO TO READ_CMD;
  STOP:END INTERACTIVE_OPT.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                        Plotting procedure

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE PLOT_COMMAND(LONG REAL ARRAY LOBS_PLACE(*);
LONG REAL ARRAY LPTS_TO_HIT(*);
LONG REAL ARRAY LSTATE(*,*);
LONG REAL ARRAY LOBS(*,*);
LONG REAL ARRAY LOBS_SMOOTH(*,*);
LONG REAL ARRAY PAR(*);
INTEGER VALUE N_OBS,N_PTS_HIT,N_STATE_OBS
,N_STATE,N_PAR;
INTEGER ARRAY STATES_OBS(*);
INTEGER VALUE RESULT PLOT_NUMBER;
PROCEDURE READ_CMD_DATA;
PROCEDURE CMD_AL;
LONG REAL ARRAY LCREOBS(*));
  BEGIN
  COMMENT PLOT COMMAND FOR STATE VARIABLES, OBSERVATIONS,
  SMOOTHING, GUESSED OBSERVATIONS, AND PHASE PLOTS;
  INTEGER ARRAY STATE_PLOT(1::N_STATE+1);
  INTEGER ARRAY OBS_PLOT,SMOOTH_PLOT,SCRATCH(1::N_STATE_OBS+1);
  INTEGER ARRAY PLOT_ITEMS(1::5);
  INTEGER ARRAY CRE_PLOT,PH_PLOT(1::2);
  STRING(1) ANS;
```

```
INTEGER I;
LONG REAL XSIZE,YSIZE;
STRING(10) LABELX,LABELY;
COMMENT DECLARE SINGLE PRECISION ARRAYS TO PASS DATA
TO GRAPHICS PROCEDURES;
REAL ARRAY OBS_PLACE(1::N_OBS);
REAL ARRAY PTS_TO_HIT(1::N_PTS_HIT);
REAL ARRAY STATE(1::N_PTS_HIT,1::N_STATE);
REAL ARRAY OBS(1::N_OBS,1::N_STATE_OBS);
REAL ARRAY OBS_SMOOTH(1::N_PTS_HIT,1::N_STATE_OBS);
REAL ARRAY CREOBS(1::N_PTS_HIT,1::1);
PROCEDURE MAX_MIN(REAL ARRAY X(*);INTEGER VALUE N;
REAL RESULT MAX,MIN);
  BEGIN
  COMMENT FIND MAXIMUM AND MINIMUM OF X;
  MAX:=X(1);MIN:=X(1);
  FOR I:=1 UNTIL N DO
    BEGIN
    IF X(I)>MAX THEN MAX:=X(I);
    IF X(I)<MIN THEN MIN:=X(I);
    END;
  END MAX_MIN;
PROCEDURE PLOT;
  BEGIN
  COMMENT THIS PROCEDURE CONTAINS THE INTERFACE TO ALL
  GRAPHICS PROCEDURES;
  REAL MAX_X,MAX_Y,MAX_XT,MAX_YT,
  MIN_X,MIN_Y,MIN_XT,MIN_YT;
  INTEGER I,NEW;
  COMMENT %EXTERNAL ALGRAF_AL, PLOT_AL
  ALSIZE_AL, ALSCAL_AL, ALAXIS_AL;
  PROCEDURE MAX_MIN_MTX(INTEGER ARRAY POS(*);
  REAL ARRAY X(*,*);
  INTEGER VALUE N);
    BEGIN
    INTEGER J;
    J:=1;
    WHILE POS(J)~=0 DO
      BEGIN
      MAX_MIN(X(*,POS(J)),N,MAX_YT,MIN_YT);
      IF MAX_YT>MAX_Y THEN MAX_Y:=MAX_YT;
      IF MIN_YT<MIN_Y THEN MIN_Y:=MIN_YT;
      J:=J+1;
      END;
    END MAX_MIN_MTX;
  PROCEDURE PLOT_HELP(INTEGER ARRAY POS(*);REAL ARRAY X(*);
  REAL ARRAY Y(*,*);
  INTEGER VALUE N,NS);
    BEGIN
    INTEGER J;
    J:=1;
```

```
  WHILE POS(J)~=0 DO
    BEGIN
    ALGRAF_AL(X,Y(*,POS(J)),NEW*N,-5*NS-POS(J));
    IF NEW>0 THEN NEW:=-1;
    J:=J+1;
    END;
  END PLOT_HELP;
COMMENT DETERMINE BOUNDS FOR SCALING;
MAX_X:=-1.'50;MIN_X:=1.'50;
IF PLOT_ITEMS(1)=5 THEN
MAX_MIN(STATE(*,1),N_PTS_HIT,MAX_X,MIN_X)
ELSE
  BEGIN
  FOR I:=1 UNTIL 5 DO
  IF PLOT_ITEMS(I)=2 THEN
  MAX_MIN(OBS_PLACE,N_OBS,MAX_X,MIN_X);
  MAX_MIN(PTS_TO_HIT,N_PTS_HIT,MAX_XT,MIN_XT);
  IF MAX_XT>MAX_X THEN MAX_X:=MAX_XT;
  IF MIN_XT<MIN_X THEN MIN_X:=MIN_XT;
  END;
MAX_Y:=-1.'50;MIN_Y:=1.'50;
I:=1;
WHILE PLOT_ITEMS(I)~=0 DO
  BEGIN
  CASE PLOT_ITEMS(I) OF
    BEGIN
    MAX_MIN_MTX(STATE_PLOT,STATE,N_PTS_HIT);
    MAX_MIN_MTX(OBS_PLOT,OBS,N_OBS);
    MAX_MIN_MTX(SMOOTH_PLOT,OBS_SMOOTH,N_PTS_HIT);
    MAX_MIN_MTX(CRE_PLOT,CREOBS,N_PTS_HIT);
    MAX_MIN(STATE(*,2),N_PTS_HIT,MAX_Y,MIN_Y)
    END;
  I:=I+1;
  END;
COMMENT DO THE PLOTTING;
ALSIZE_AL(XSIZE-1.'-5,YSIZE-1.'-5);
ALSCAL_AL(MIN_X,MAX_X,MIN_Y,MAX_Y);
ALAXIS_AL(LABELX,50,LABELY,50);
NEW:=1;
I:=1;
WHILE PLOT_ITEMS(I)~=0 DO
  BEGIN
  CASE PLOT_ITEMS(I) OF
    BEGIN
    PLOT_HELP(STATE_PLOT,PTS_TO_HIT,STATE,N_PTS_HIT,0);
    PLOT_HELP(OBS_PLOT,OBS_PLACE,OBS,N_OBS,1);
    PLOT_HELP(SMOOTH_PLOT,PTS_TO_HIT,OBS_SMOOTH,
    N_PTS_HIT,2);
    PLOT_HELP(CRE_PLOT,PTS_TO_HIT,CREOBS,N_PTS_HIT,3);
    PLOT_HELP(PH_PLOT,STATE(*,1),STATE,N_PTS_HIT,4)
    END;
```

```
    I:=I+1;
    END;
  COMMENT PREPARE FOR NEXT PLOT;
  PLOT_AL(12.0,0,-3);
  END PLOT;
COMMENT EXECUTION BEGINS HERE
FIRST COPY ARRAYS TO SINGLE PRECISION COUNTERPARTS;
FOR I:=1 UNTIL N_OBS DO
OBS_PLACE(I):=LOBS_PLACE(I);
FOR I:=1 UNTIL N_PTS_HIT DO
PTS_TO_HIT(I):=LPTS_TO_HIT(I);
FOR J:=1 UNTIL N_STATE DO
FOR I:=1 UNTIL N_PTS_HIT DO
STATE(I,J):=LSTATE(I,J);
FOR J:=1 UNTIL N_STATE_OBS DO
FOR I:=1 UNTIL N_OBS DO
OBS(I,J):=LOBS(I,J);
FOR J:=1 UNTIL N_STATE_OBS DO
FOR I:=1 UNTIL N_PTS_HIT DO
OBS_SMOOTH(I,J):=LOBS_SMOOTH(I,J);
FOR I:=1 UNTIL N_PTS_HIT DO
CREOBS(I,1):=LCREOBS(I);
XSIZE:=5.;
YSIZE:=3.;
COMMENT %OUTPUT TO USER
SEQUENCE OF ITEMS TO PLOT(END WITH 0)
STATE VARIABLES----------------------1
OBSERVATIONS-------------------------2
SMOOTHED OBSERVATIONS----------------3
GUESSED OBSERVATIONS-----------------4
PHASE PLOT---------------------------5;
READ_CMD_DATA(PLOT_ITEMS);
LABELX:="TIME";LABELY:="STATE";
I:=1;
WHILE PLOT_ITEMS(I)~=0 DO
  BEGIN
  CASE PLOT_ITEMS(I) OF
    BEGIN
      BEGIN
      COMMENT %OUTPUT TO USER
      STATE VARIABLES(END WITH 0);
      READ_CMD_DATA(STATE_PLOT);
      END;
      BEGIN
      INTEGER II;
      COMMENT %OUTPUT TO USER
      OBSERVED VARIABLES TO PLOT (END WITH 0);
      READ_CMD_DATA(SCRATCH);
      II:=1;
      WHILE SCRATCH(II)~=0 DO
        BEGIN
```

```
        FOR J:=1 UNTIL N_STATE_OBS DO
        IF SCRATCH(II)=STATES_OBS(J) THEN OBS_PLOT(II)
        :=J;
        II:=II+1;
        END;
      OBS_PLOT(II):=0;
      END;
      BEGIN
      INTEGER II;
      COMMENT %OUTPUT
      SMOOTHED OBSERVATIONS TO PLOT (END WITH 0);
      READ_CMD_DATA(SCRATCH);
      II:=1;
      WHILE SCRATCH(II)~=0 DO
        BEGIN
        FOR J:=1 UNTIL N_STATE_OBS DO
        IF SCRATCH(II)=STATES_OBS(J) THEN SMOOTH_PLOT(II)
        :=J;
        II:=II+1;
        END;
      SMOOTH_PLOT(II):=0;
      END;
      BEGIN
      COMMENT GUESSED OBSERVATIONS CASE;
      CRE_PLOT(1):=1;CRE_PLOT(2):=0;
      END;
      BEGIN
      COMMENT PHASE PLOT CASE;
      PH_PLOT(1):=2;PH_PLOT(2):=0;
      LABELX:="Y1";LABELY:="Y2";
      END
    END;
  I:=I+1;
  END;
COMMENT %FILE EMPTY FILE -GRAPH;
PLOT;
COMMENT %FILE DISPLAY -GRAPH TO USER
%OUTPUT TO USER
IS A PERMANENT PLOT REQUIRED ANS. Y OR N
%INPUT (ANS);
IF ANS="Y" THEN
  BEGIN
  XSIZE:=10;
  YSIZE:=10;
  COMMENT %FILE EMPTY -GRAPH;
  PLOT;
  COMMENT %FILE ACCUMULATE -GRAPH IN -GRAPHSTORE;
  PLOT_NUMBER:=PLOT_NUMBER+1;
  COMMENT %OUTPUT TO USER, -REPRT
  CURRENT PLOT_NUMBER (PLOT_NUMBER)
```

```
    CURRENT PARAMETERS (PAR)
    DESCRIPTION OF PLOT CONTENTS;
    END;
  END PLOT_COMMAND.

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

          Sparse Gauss-Newton for iterated methods

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE SPRGN(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,DSPLN_AL,HERM,DHERM;
LONG REAL ARRAY CREOBS(*);
LONG REAL ARRAY DCREOBS(*);
PROCEDURE ITER_PAR);
  BEGIN
  COMMENT ITERATED IFIT AND DFIT IN THE TWO STATE VARIABLE CASE
  USING A SPARSE GAUSS-NEWTON METHOD TO IMPROVE
  GUESSED OBSERVATIONS
  (2 OPTIONS, FIXED INITIAL CONDITIONS ON GUESSED OBSERVATIONS,
  AND VARIABLE INITIAL CONDITIONS FOR GUESSED OBSERVATIONS;
  PROCEDURE ITER_FUNC(LONG REAL ARRAY C(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY AHES(*);
  LONG REAL ARRAY GRAD(*);
  INTEGER VALUE FULL);
    BEGIN
    COMMENT LEAST SQUARES FUNCTION FOR
    ITERATIVELY IMPROVING THE
    GUESSED OBSERVATIONS USING A SPARSE GAUSS NEWTON METHOD;
    LONG REAL ARRAY A,JA(1::N_PTS_HIT);
    LONG REAL ARRAY B(1::N_PTS_HIT-1);
    LONG REAL ARRAY JB(1::N_PTS_HIT-1,1::2);
    LONG REAL ARRAY DGY,PREV_DGY(1::2,1::2);
    LONG REAL ARRAY DGP,SENSE(1::1,1::1);
    LONG REAL ARRAY G,PREV_G,Y(1::2);
    LONG REAL HI;
    Y(SM_Y):=OBS_SMOOTH(1,1);
    Y(CR_Y):=C(1);
```

```
G_FUN(PTS_TO_HIT(1),Y,PAR,1,PREV_G,PREV_DGY,DGP,SENSE);
A(1):=PREV_G(SM_Y)
-(CASE SMF OF(DSPLN_AL(PTS_TO_HIT(1),
SPLN_COEF(*,1),JOINTS(*,1),
NJOINTS(1),N_SPLN_PAR(1)),
DHERM(PTS_TO_HIT(1),HERM_COEF(*,*,1),JOINTS(*,1),
NJOINTS(1))));
JA(1):=PREV_DGY(SM_Y,CR_Y);
FOR I:=2 UNTIL N_PTS_HIT DO
  BEGIN
  Y(SM_Y):=OBS_SMOOTH(I,1);
  Y(CR_Y):=C(I);
  G_FUN(PTS_TO_HIT(I),Y,PAR,1,G,DGY,DGP,SENSE);
  A(I):=G(SM_Y)
  -(CASE SMF OF(DSPLN_AL(PTS_TO_HIT(I),
  SPLN_COEF(*,1),JOINTS(*,1),
  NJOINTS(1),N_SPLN_PAR(1)),
  DHERM(PTS_TO_HIT(I),HERM_COEF(*,*,1),JOINTS(*,1),
  NJOINTS(1))));
  JA(I):=DGY(SM_Y,CR_Y);
  HI:=PTS_TO_HIT(I)-PTS_TO_HIT(I-1);
  B(I-1):=.5L*(G(CR_Y)+PREV_G(CR_Y))-(C(I)-C(I-1))/HI;
  JB(I-1,1):=.5L*PREV_DGY(CR_Y,CR_Y)+1.L/HI;
  JB(I-1,2):=.5L*DGY(CR_Y,CR_Y)-1.L/HI;
  FOR J:=1 UNTIL 2 DO
  PREV_G(J):=G(J);
  FOR J:=1 UNTIL 2 DO
  FOR K:=1 UNTIL 2 DO
  PREV_DGY(J,K):=DGY(J,K);
  END;
IF FULL=1 THEN
  BEGIN
  COMMENT FORM AHES TO CONFORM TO DFBAND;
  AHES(1):=JB(1,1)**2+JA(1)**2;
  FOR I:=2 UNTIL N_PTS_HIT-1 DO
  AHES(2*(I-1)+1):=JA(I)**2+JB(I-1,2)**2+JB(I,1)**2;
  AHES(2*(N_PTS_HIT-1)+1):=JA(N_PTS_HIT)**2
  +JB(N_PTS_HIT-1,2)**2;
  FOR I:=1 UNTIL N_PTS_HIT-1 DO
  AHES(2*I):=JB(I,1)*JB(I,2);
  COMMENT FORM GRADIENT;
  FOR I:=1 UNTIL N_PTS_HIT DO
    BEGIN
    GRAD(I):=JA(I)*A(I);
    IF I=1 THEN GRAD(I):=GRAD(I)+JB(1,1)*B(1)
    ELSE
    IF I=N_PTS_HIT THEN GRAD(I):=GRAD(I)+
    JB(N_PTS_HIT-1,2)*B(N_PTS_HIT-1)
    ELSE
    GRAD(I):=GRAD(I)+JB(I-1,2)*B(I-1)+JB(I,1)*B(I);
    END;
```

```
    END;
  COMMENT FORM F;
  F:=0.L;
  FOR I:=1 UNTIL N_PTS_HIT DO
  F:=F+A(I)**2;
  FOR I:=1 UNTIL N_PTS_HIT-1 DO
  F:=F+B(I)**2;
  END ITR_FUNC;
COMMENT
;
PROCEDURE ITER_FUNC_FIXIC(LONG REAL ARRAY C(*);
LONG REAL RESULT F;
LONG REAL ARRAY AHES(*);
LONG REAL ARRAY GRAD(*);
INTEGER VALUE FULL);
  BEGIN
  COMMENT LEAST SQUARES FUNCTION FOR
  ITERATIVELY IMPROVING THE
  GUESSED OBS FIXED IC ON GUESSED OBS;
  LONG REAL ARRAY A,JA(1::N_PTS_HIT-1);
  LONG REAL ARRAY B(1::N_PTS_HIT-1);
  LONG REAL ARRAY JB(1::N_PTS_HIT-1,1::2);
  LONG REAL ARRAY DGY,PREV_DGY(1::2,1::2);
  LONG REAL ARRAY DGP,SENSE(1::1,1::1);
  LONG REAL ARRAY G,PREV_G,Y(1::2);
  LONG REAL HI;
  G_FUN(PTS_TO_HIT(1),Y,PAR,-3,G,DGY,DGP,SENSE);
  Y(CR_Y):=CREOBS(1);
  G_FUN(PTS_TO_HIT(1),Y,PAR,1,PREV_G,PREV_DGY,DGP,SENSE);
  FOR I:=2 UNTIL N_PTS_HIT DO
    BEGIN
    Y(SM_Y):=OBS_SMOOTH(I,1);
    Y(CR_Y):=C(I-1);
    G_FUN(PTS_TO_HIT(I),Y,PAR,1,G,DGY,DGP,SENSE);
    A(I-1):=G(SM_Y)
    -(CASE SMF OF(DSPLN_AL(PTS_TO_HIT(I),
    SPLN_COEF(*,1),JOINTS(*,1),
    NJOINTS(1),N_SPLN_PAR(1)),
    DHERM(PTS_TO_HIT(I),HERM_COEF(*,*,1),JOINTS(*,1),
    NJOINTS(1))));
    JA(I-1):=DGY(SM_Y,CR_Y);
    HI:=PTS_TO_HIT(I)-PTS_TO_HIT(I-1);
    B(I-1):=.5L*(G(CR_Y)+PREV_G(CR_Y))-(C(I-1)
    -(IF I>2 THEN C(I-2) ELSE CREOBS(1)))/HI;
    JB(I-1,1):=.5L*PREV_DGY(CR_Y,CR_Y)+1.L/HI;
    JB(I-1,2):=.5L*DGY(CR_Y,CR_Y)-1.L/HI;
    FOR J:=1 UNTIL 2 DO
    PREV_G(J):=G(J);
    FOR J:=1 UNTIL 2 DO
    FOR K:=1 UNTIL 2 DO
    PREV_DGY(J,K):=DGY(J,K);
```

```
    END;
  IF FULL=1 THEN
    BEGIN
    COMMENT FORM AHES TO CONFORM TO DFBAND;
    FOR I:=1 UNTIL N_PTS_HIT-2 DO
    AHES(2*(I-1)+1):=JA(I)**2+JB(I,2)**2+JB(I+1,1)**2;
    AHES(2*(N_PTS_HIT-2)+1):=JA(N_PTS_HIT-1)**2
    +JB(N_PTS_HIT-1,2)**2;
    FOR I:=1 UNTIL N_PTS_HIT-2 DO
    AHES(2*I):=JB(I+1,1)*JB(I+1,2);
    COMMENT FORM GRADIENT;
    FOR I:=1 UNTIL N_PTS_HIT-1 DO
      BEGIN
      GRAD(I):=JA(I)*A(I);
      IF I=(N_PTS_HIT-1) THEN GRAD(I):=GRAD(I)+
      JB(N_PTS_HIT-1,2)*B(N_PTS_HIT-1)
      ELSE
      GRAD(I):=GRAD(I)+JB(I,2)*B(I)+JB(I+1,1)*B(I+1);
      END;
    END;
  COMMENT FORM F;
  F:=0.L;
  FOR I:=1 UNTIL N_PTS_HIT-1 DO
  F:=F+A(I)**2+B(I)**2;
  END ITER_FUNC_FIXIC;
COMMENT
;
PROCEDURE SPARSE_GN(LONG REAL ARRAY C(*);
INTEGER VALUE NC;
PROCEDURE ITER_FUNC);
  BEGIN
  COMMENT ITERATIVE UPDATING OF GUESSED OBSERVATIONS
  USING A SPARSE GAUSS NEWTON METHOD;
  LONG REAL F,F1,RATIO,DET,GA,R0,R1,W1;
  LONG REAL ARRAY AHES(1::2*NC);
  LONG REAL ARRAY C1,GRAD,DELTA(1::NC);
  INTEGER JEXP,JAY;
  LOGICAL CONV;
  COMMENT %EXTERNAL DFBAND;
  CONV:=FALSE;
  FOR ITER:=1 UNTIL 25 DO
    BEGIN
    ITER_FUNC(C,F,AHES,GRAD,1);
    IF ITER=1 THEN
    COMMENT %OUTPUT TO USER
    STARTING SUM OF SQUARES IN SPARSE GAUSS NEWTON IS (F);
    IF ITER=1 THEN JAY:=0;
    JAY:=JAY DIV 2;
    R0:=1.L/2**(JAY);
    COMMENT SOLVE FOR DELTA;
    FOR I:=1 UNTIL NC DO DELTA(I):=-GRAD(I);
```

```
      RATIO:=1.'-7L;
      DFBAND(AHES,DELTA,NC,2,1,RATIO,DET,JEXP,0);
      COMMENT FIND STEP LENGTH;
      FOR I:=1 UNTIL NC DO
      C1(I):=C(I)+R0*DELTA(I);
      COMMENT CHECK FOR CONVERGENCE;
      FOR I:=1 UNTIL NC DO
      IF (ABS(C(I)-C1(I))>(TOL*ABS(C1(I))+.001L))
      THEN GO TO CON;
      CONV:=TRUE;
      GO TO UPDATE;
      CON:ITER_FUNC(C1,F1,AHES,GRAD,0);
      IF F1>F THEN
        BEGIN
        COMMENT INTERPOLATE;
        GA:=0.L;
        FOR I:=1 UNTIL NC DO
        GA:=GA+DELTA(I)*GRAD(I);
        FOR INTERP:=1 UNTIL 5 DO
          BEGIN
          JAY:=JAY+1;
          R1:=GA*R0**2/(2.L*(GA*R0+F-F1));
          W1:=(IF (.75L*R0<R1) THEN .75L*R0 ELSE R1);
          R0:=(IF (.25L*R0>W1) THEN .25L*R0 ELSE W1);
          FOR I:=1 UNTIL NC DO
          C1(I):=C(I)+R0*DELTA(I);
          COMMENT CHECK FOR CONVERGENCE;
          FOR I:=1 UNTIL NC DO
          IF (ABS(C(I)-C1(I))>(TOL*ABS(C1(I))+.001L))
          THEN GO TO CONT;
          CONV:=TRUE;
          GO TO UPDATE;
          CONT:ITER_FUNC(C1,F1,AHES,GRAD,0);
          IF F1<F THEN GO TO UPDATE;
          END INTERP;
        COMMENT %OUTPUT TO USER
        TERMINATING--OVER 5 INTERPOLATIONS REQUIRED;
        GO TO FINISHED;
        END;
      UPDATE:FOR I:=1 UNTIL NC DO
      C(I):=C1(I);
      COMMENT %OUTPUT TO USER
      SUM OF SQUARES IS (F1);
      IF CONV=TRUE THEN GO TO FINISHED;
      END ITER;
    COMMENT %OUTPUT TO USER
    TERMINATING--SPARSE GAUSS NEWTON NEEDS
    MORE THAN 25 ITERATIONS TO MEET ERROR CRITERION;
    FINISHED: END SPARSE_GN;
  INTEGER N_PTS_HIT,N_PAR,SMF,OUTPUT,METHOD_FLAG,OUTPUT_SUP,
  OUT_SEG,INT_PROC,SM_Y,CR_Y,NO_ITER,EFLAG;
```

```
LONG REAL ARRAY B(1::1,1::INTDATA(1));
LONG REAL ARRAY BJAC(1::1,1::1);
COMMENT COMMAND PROPER FOLLOWS;
STRING(1) ANS,ANS1;
LONG REAL TOL;
N_PTS_HIT:=INTDATA(1);
N_PAR:=INTDATA(4);
SMF:=INTDATA(5);
OUTPUT:=INTDATA(6);
METHOD_FLAG:=INTDATA(7);
OUTPUT_SUP:=INTDATA(9);
OUT_SEG:=INTDATA(10);
INT_PROC:=INTDATA(11);
SM_Y:=STATES_OBS(1);
CR_Y:=3-SM_Y;
COMMENT %OUTPUT TO USER
IS INITIAL VALUE OF GUESSED OBSERVATIONS FIXED? Y OR N
%INPUT (ANS1)
%OUTPUT
ENTER RELATIVE TOLERANCE ON CHANGE IN ITERATES FOR
TERMINATION OF SPARSE GAUSS NEWTON
%INPUT (TOL);
REPT:ITER_PAR(G_FUN,PAR,INTDATA,OBS_SMOOTH,STATES_OBS,
PTS_TO_HIT,JOINTS,SPLN_COEF,HERM_COEF,NJOINTS,N_SPLN_PAR,
SPLN_AL,DSPLN_AL,HERM,DHERM,
CREOBS,DCREOBS);
COMMENT SET UP AND OPTIMIZE SPARSE PROB;
REDO:IF ANS1~="Y" THEN
SPARSE_GN(CREOBS,N_PTS_HIT,ITER_FUNC)
ELSE
  BEGIN
  LONG REAL ARRAY C(1::N_PTS_HIT-1);
  FOR I:=1 UNTIL N_PTS_HIT-1 DO C(I):=CREOBS(I+1);
  SPARSE_GN(C,N_PTS_HIT-1,ITER_FUNC_FIXIC);
  FOR I:=1 UNTIL N_PTS_HIT-1 DO CREOBS(I+1):=C(I);
  END;
COMMENT %OUTPUT
IS A FURTHER REFINEMENT OF THE GUESSED OBSERVATIONS
DESIRED? Y OR N
%INPUT (ANS);
IF ANS="Y" THEN
  BEGIN
  COMMENT %OUTPUT
  ENTER NEW TOLERANCE FOR SPARSE PROBLEM
  %INPUT (TOL);
  GO TO REDO;
  END;
COMMENT %OUTPUT
```

```
  IS ANOTHER ITERATION DESIRED? Y OR N
  %INPUT (ANS);
  IF ANS="Y" THEN GO TO REPT;
  OUT:END SPRGN.


     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

                    Derivative fitting procedure

     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE DATAFT_COMMAND(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,
DSPLN_AL,HERM,DHERM;
LONG REAL VALUE INITIAL_TIME,EPS,HMIN,HMAX);
  BEGIN
  COMMENT DATAFIT COMMAND;
  INTEGER N_PTS_HIT,N_STATE_OBS,N_STATE,N_PAR,SMF,OUTPUT;
  INTEGER OUTPUT_SUP,OUT_SEG;
  INTEGER METHOD_FLAG,INT_PROC;
  LONG REAL LAM,EPS_R,EPS_A;
  LONG REAL ARRAY DJAC(1::1,1::1);
  INTEGER KFLAG;
  LONG REAL ARRAY STATE_DATA(1::INTDATA(1),1::INTDATA(3));
  COMMENT %EXTERNAL GEAR, TRAP, SVD_AL, MARQUARDT, CMD_AL;
  PROCEDURE DATA_FUNC(LONG REAL ARRAY P(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
  LONG REAL ARRAY JAC(*,*);
  LONG REAL ARRAY GRAD(*);
  INTEGER RESULT EFLAG);
    BEGIN
    COMMENT  THIS PROCEDURE PROVIDES THE
    PERTINENT INFORMATION TO A NONLINEAR
    LEAST SQUARES ALGORITHM USED WITH THE DFIT METHOD;
    INTEGER K,M,OUT1;
    LONG REAL SUM;
    LONG REAL ARRAY INITY(1::1);
    LONG REAL ARRAY  INITYP(1::1,1::1);
    LONG REAL ARRAY G(1::N_STATE);
```

```
LONG REAL ARRAY DGY(1::N_STATE,1::N_STATE);
LONG REAL ARRAY DGP,SENSE(1::N_STATE,1::N_PAR);
K:=0;
IF N_STATE_OBS~=N_STATE THEN
  BEGIN
  COMMENT NOT ALL STATE VARIABLES ARE OBSERVED;
  INTEGER ARRAY SCRATCH(1::N_STATE);
  INTEGER ARRAY STATES_NOT_OBS(1::N_STATE-N_STATE_OBS);
  LONG REAL ARRAY STATE_INTEG(1::N_PTS_HIT,
  1::N_STATE-N_STATE_OBS);
  INTEGER L,N_STATE_NOT_OBS;
  PROCEDURE G_DATA(LONG REAL VALUE T;
  LONG REAL ARRAY Y(*);
  LONG REAL ARRAY P(*);
  INTEGER VALUE OPTION;
  LONG REAL ARRAY G(*);
  LONG REAL ARRAY DGY(*,*);
  LONG REAL ARRAY D2(*,*);
  LONG REAL ARRAY D3(*,*));
    BEGIN
    COMMENT INTERFACE TO G_FUN FOR DE SOLVER WHEN A
    SUBSET OF THE STATE VARIABLES ARE BEING INTEGRATED;
    LONG REAL ARRAY Y_A,DY_A(1::N_STATE);
    LONG REAL ARRAY DGY_A(1::N_STATE,1::N_STATE);
    LONG REAL ARRAY D2,D3(1::1,1::1);
    FOR I:=1 UNTIL N_STATE_NOT_OBS DO
    Y_A(STATES_NOT_OBS(I)):=Y(I);
    FOR I:=1 UNTIL N_STATE_OBS DO
    CASE SMF OF
      BEGIN
      Y_A(STATES_OBS(I)):=
      SPLN_AL(T,SPLN_COEF(*,I),JOINTS(*,I)
      ,NJOINTS(I),N_SPLN_PAR(I));
      Y_A(STATES_OBS(I)):=
      HERM(T,HERM_COEF(*,*,I),JOINTS(*,I)
      ,NJOINTS(I))
      END;
    G_FUN(T,Y_A,P,OPTION,DY_A,DGY_A,D2,D3);
    IF OPTION=1 THEN
    COMMENT CONSTRUCT SUB JACOBIAN;
    FOR I:=1 UNTIL N_STATE_NOT_OBS DO
    FOR J:=1 UNTIL N_STATE_NOT_OBS DO
    DGY(I,J):=DGY_A(STATES_NOT_OBS(I),STATES_NOT_OBS(J));
    FOR I:=1 UNTIL N_STATE_NOT_OBS DO
    G(I):=DY_A(STATES_NOT_OBS(I));
    IF OPTION=-3 THEN
    FOR I:=1 UNTIL N_STATE_NOT_OBS DO
    Y(I):=Y_A(STATES_NOT_OBS(I));
    END G_DATA;
  COMMENT FORM VECTOR OF STATES NOT OBSERVED;
  FOR I:=1 UNTIL N_STATE DO SCRATCH(I):=I;
```

```
  FOR I:=1 UNTIL N_STATE_OBS DO
  SCRATCH(STATES_OBS(I)):=0;
  N_STATE_NOT_OBS:=N_STATE-N_STATE_OBS;
  L:=0;
  FOR I:=1 UNTIL N_STATE DO
  IF SCRATCH(I)~=0 THEN
    BEGIN
    L:=L+1;
    STATES_NOT_OBS(L):=SCRATCH(I);
    END;
  EFLAG:=0;
  COMMENT INTEGRATION;
  IF OUTPUT_SUP=1 THEN OUT1:=0
  ELSE OUT1:=OUTPUT;
    BEGIN
    LONG REAL ARRAY B(1::N_STATE_NOT_OBS,1::N_PTS_HIT);
    CASE INT_PROC OF
      BEGIN
        BEGIN
        GEAR(P,PTS_TO_HIT,B,DJAC,N_STATE_NOT_OBS,
        N_PTS_HIT,EPS,HMIN,HMAX,
        N_PAR,0,G_DATA,KFLAG,OUT1,METHOD_FLAG,
        0,INITY,INITYP);
        IF KFLAG~=1 THEN
          BEGIN
          EFLAG:=1;
          GO TO OUT;
          END;
        END;
        BEGIN
        TRAP(P,PTS_TO_HIT,B,DJAC,N_STATE_NOT_OBS,
        N_PTS_HIT,N_PAR,0,G_DATA,EFLAG,OUT1);
        IF EFLAG=1 THEN GO TO OUT;
        END
      END;
    FOR J:=1 UNTIL N_STATE_NOT_OBS DO
    FOR I:=1 UNTIL N_PTS_HIT DO
    STATE_INTEG(I,J):=B(J,I);
    END;
  COMMENT COPY INTEGRATION DATA INTO STATE DATA;
  FOR I:=1 UNTIL N_PTS_HIT DO
  FOR J:=1 UNTIL N_STATE_NOT_OBS DO
  STATE_DATA(I,STATES_NOT_OBS(J)):=STATE_INTEG(I,J);
  END;
FOR I:=1 UNTIL N_PTS_HIT DO
  BEGIN
  G_FUN(PTS_TO_HIT(I),STATE_DATA(I,*),P,1,G,DGY,DGP,SENSE);
  G_FUN(PTS_TO_HIT(I),STATE_DATA(I,*),P,2,G,DGY,DGP,
  SENSE);
  FOR J:=1 UNTIL N_STATE_OBS DO
    BEGIN
```

```
      K:=K+1;
      CASE SMF OF
        BEGIN
        RES(K):=G(STATES_OBS(J))-DSPLN_AL(PTS_TO_HIT(I),
        SPLN_COEF(*,J),
        JOINTS(*,J),NJOINTS(J),N_SPLN_PAR(J));
        RES(K):=G(STATES_OBS(J))-DHERM(PTS_TO_HIT(I),
        HERM_COEF(*,*,J),
        JOINTS(*,J),NJOINTS(J));
        END;
      FOR L:=1 UNTIL N_PAR DO
      JAC(K,L):=DGP(STATES_OBS(J),L);
      END;
    END;
  COMMENT FORM SUM OF SQUARES OF RESIDUAL;
  SUM:=0.;
  M:=K;
  FOR I:=1 UNTIL M DO SUM:=SUM+RES(I)**2;
  F:=SUM;
  COMMENT FORM GRADIENT IF REQUIRED;
    BEGIN
    FOR I:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0.;
      FOR J:=1 UNTIL N_PTS_HIT*N_STATE_OBS DO
      SUM:=SUM+JAC(J,I)*RES(J);
      GRAD(I):=SUM;
      END;
    END;
  END DATA_FUNC;
COMMENT
COMMAND PROPER FOLLOWS;
N_PTS_HIT:=INTDATA(1);
N_STATE_OBS:=INTDATA(2);
N_STATE:=INTDATA(3);
N_PAR:=INTDATA(4);
SMF:=INTDATA(5);
OUTPUT:=INTDATA(6);
METHOD_FLAG:=INTDATA(7);
OUTPUT_SUP:=INTDATA(9);
OUT_SEG:=INTDATA(10);
INT_PROC:=INTDATA(11);
FOR I:=1 UNTIL N_PTS_HIT DO
FOR J:=1 UNTIL N_STATE_OBS DO
STATE_DATA(I,STATES_OBS(J)):=OBS_SMOOTH(I,J);
COMMENT %OUTPUT
ENTER STARTING LAMBDA RELATIVE
AND ABSSOLUTE ERROR TOLERENCES FOR
MARQUARDT PROCEDURE
%INPUT (LAM),(EPS_R),(EPS_A)
%FILE EMPTY TEMPORARY FILE -SC1
```

```
  %OUTPUT TO -REPRT
  MARQUARDT USED IN DFIT COMMAND--OPTION 1
  LAMBDA, RELATIVE, AND ABSOLUTE
  ERROR TOLERENCES ARE (LAM),(EPS_R),(EPS_A);
  MARQUARDT(EPS_R,EPS_A,N_PTS_HIT*N_STATE_OBS,N_PAR,DATA_FUNC,
  PAR,LAM,SVD_AL);
  OUT:END.


     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

          Derivative fitting with guessed observations

     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE DFIT_CRE(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,
DSPLN_AL,HERM,DHERM;
LONG REAL ARRAY CREOBS(*);
LONG REAL ARRAY DCREOBS(*));
  BEGIN
  COMMENT DFIT ON 2 STATE VARIABLES WHEN OBSERVATIONS ON
  ONE HAVE
  BEEN GUESSED AT. SPECIAL HANDLING OF LINEAR PROBLEM;
  INTEGER N_PTS_HIT,N_PAR,SMF,DFIT_LIN,KK;
  LONG REAL LAM,EPS_R,EPS_A;
  INTEGER SM_Y,CR_Y,M,OUT_SEG;
  COMMENT %EXTERNAL SVD_AL, MARQUARDT, CMD_AL;
  PROCEDURE CRE_FUNC(LONG REAL ARRAY P(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
  LONG REAL ARRAY JAC(*,*);
  LONG REAL ARRAY GRAD(*);
  INTEGER RESULT EFLAG);
    BEGIN
    COMMENT THIS PROCEDURE PROVIDES INFORMATION TO THE LEAST
    SQUARES MINIMIZER WHEN GUESSED OBSERVATIONS ARE USED;
    LONG REAL ARRAY SENSE,DGY(1::1,1::1);
    LONG REAL ARRAY Y,G(1::2);
    LONG REAL ARRAY DGP(1::2,1::N_PAR);
    INTEGER K,KK;
```

```
    LONG REAL SUM;
    EFLAG:=0;
    K:=0;
    FOR I:=1 UNTIL N_PTS_HIT DO
      BEGIN
      Y(SM_Y):=OBS_SMOOTH(I,1);
      Y(CR_Y):=CREOBS(I);
      G_FUN(PTS_TO_HIT(I),Y,P,-1,G,DGY,DGP,SENSE);
      G_FUN(PTS_TO_HIT(I),Y,P,2,G,DGY,DGP,SENSE);
      FOR J:=1 UNTIL 2 DO
        BEGIN
        K:=K+1;
        IF SM_Y=J THEN
          BEGIN
          CASE SMF OF
            BEGIN
            RES(K):=G(J)-DSPLN_AL(PTS_TO_HIT(I),
            SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
            N_SPLN_PAR(1));
            RES(K):=G(J)-DHERM(PTS_TO_HIT(I),
            HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))
            END;
          END
        ELSE RES(K):=G(J)-DCREOBS(I);
        FOR L:=1 UNTIL N_PAR DO
        JAC(K,L):=DGP(J,L);
        END;
      END;
    COMMENT FORM F;
    F:=0;
    FOR I:=1 UNTIL M DO F:=F+RES(I)**2;
    COMMENT FORM GRADIENT;
    FOR I:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0;
      FOR J:=1 UNTIL M DO
      SUM:=SUM+JAC(J,I)*RES(J);
      GRAD(I):=SUM;
      END;
    END CRE_FUNC;
  COMMENT COMMAND PROPER FOLLOWS;
  N_PTS_HIT:=INTDATA(1);
  N_PAR:=INTDATA(4);
  SMF:=INTDATA(5);
  OUT_SEG:=INTDATA(10);
  DFIT_LIN:=INTDATA(12);
  SM_Y:=STATES_OBS(1);
  CR_Y:=3-SM_Y;
  M:=2*N_PTS_HIT;
  COMMENT %FILE EMPTY TEMPORARY FILE -SC1;
  IF DFIT_LIN~=1 THEN
```

```
  BEGIN
  COMMENT %OUTPUT
  ENTER STARTING LAMBDA, RELATIVE AND ABSOLUTE TOLERENCES FOR
  THE MARQUARDT PROCEDURE
  %INPUT (LAM),(EPS_R),(EPS_A)
  %OUTPUT TO -SC1
  MARQUARDT CALLED IN DFIT OPTION 2
  OUTPUT REFERENCE NUMBER IS (OUT_SEG)
  LAMBDA, RELATIVE, AND ABSOLUTE ERROR TOLERENCES ARE
  (LAM),(EPS_R),(EPS_A);
  MARQUARDT(EPS_R,EPS_A,M,N_PAR,CRE_FUNC,PAR,LAM,SVD_AL);
  END
ELSE
  BEGIN
  COMMENT LINEAR LEAST SQUARES USING SVD;
  LONG REAL SING_CUTOFF;
  LONG REAL ARRAY JAC(1::M,1::N_PAR);
  LONG REAL ARRAY RES(1::M);
  LONG REAL ARRAY SP,S,GRAD(1::N_PAR);
  LONG REAL ARRAY V(1::N_PAR,1::N_PAR);
  LONG REAL ARRAY A(1::M,1::N_PAR+1);
  LONG REAL SUM,F;
  INTEGER K;
  LONG REAL ARRAY U(1::M,1::N_PAR);
  INTEGER EFLAG;
  SING_CUTOFF:=.00001L;
  CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
  FOR J:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL M DO
  A(I,J):=JAC(I,J);
  K:=0;
  FOR I:=1 UNTIL N_PTS_HIT DO
    BEGIN
    FOR J:=1 UNTIL 2 DO
      BEGIN
      K:=K+1;
      IF SM_Y=J THEN
        BEGIN
        CASE SMF OF
          BEGIN
          A(K,N_PAR+1):=DSPLN_AL(PTS_TO_HIT(I),
          SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
          N_SPLN_PAR(1));
          A(K,N_PAR+1):=DHERM(PTS_TO_HIT(I),
          HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))
          END;
        END
      ELSE A(K,N_PAR+1):=DCREOBS(I);
      END;
    END;
  SVD_AL(A,S,U,V,M,N_PAR,M,N_PAR,1,0,N_PAR);
```

```
    FOR I:=1 UNTIL N_PAR DO
    IF (S(I)/S(1)>SING_CUTOFF) THEN SP(I):=A(I,N_PAR+1)/S(I)
    ELSE SP(I):=0.;
    FOR I:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0.;
      FOR J:=1 UNTIL N_PAR DO
      SUM:=SUM+V(I,J)*SP(J);
      PAR(I):=SUM;
      END;
    COMMENT %OUTPUT TO USER
    LINEAR DFIT-OPTION 2 SINGULAR VALUE REJECTION LEVEL IS
    (SING_CUTOFF);
    CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
    COMMENT %OUTPUT TO USER
    IN LINEAR PART SUM OF SQUARES OF RESIDUALS IS (F);
    END;
  END DFIT_CRE.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

            Iterated derivative fitting procedure

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE DFITITER(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,DSPLN_AL,HERM,DHERM;
LONG REAL ARRAY CREOBS(*);
LONG REAL ARRAY DCREOBS(*));
  BEGIN
  COMMENT DFIT ON 2 STATE VARIABLES TO GO WITH
  SPARSE GAUSS NEWTON TO IMPROVE GUESSED STATE;
  INTEGER N_PTS_HIT,N_PAR,SMF,DFIT_LIN,KK;
  LONG REAL LAM,EPS_R,EPS_A;
  INTEGER SM_Y,CR_Y,M,OUT_SEG;
  LONG REAL ARRAY V0(1::2);
  COMMENT %EXTERNAL SVD_AL,MARQUARDT,CMD_AL;
  PROCEDURE CRE_FUNC(LONG REAL ARRAY P(*));
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
```

```
LONG REAL ARRAY JAC(*,*);
LONG REAL ARRAY GRAD(*);
INTEGER RESULT EFLAG);
  BEGIN
  COMMENT THIS PROCEDURE PROVIDES INFORMATION TO THE LEAST
  SQUARES MINIMIZER WHEN GUESSED OBSERVATIONS ARE USED;
  LONG REAL ARRAY SENSE,DGY(1::1,1::1);
  LONG REAL ARRAY Y,G,INT_G,PREV_G(1::2);
  LONG REAL ARRAY DGP,INT_DGP,PREV_DGP(1::2,1::N_PAR);
  INTEGER K,KK;
  LONG REAL SUM,DEL;
  EFLAG:=0;
  K:=0;
  Y(SM_Y):=OBS_SMOOTH(1,1);
  Y(CR_Y):=CREOBS(1);
  G_FUN(PTS_TO_HIT(1),Y,P,-1,PREV_G,DGY,DGP,SENSE);
  G_FUN(PTS_TO_HIT(1),Y,P,2,G,DGY,PREV_DGP,SENSE);
  FOR I:=2 UNTIL N_PTS_HIT DO
    BEGIN
    Y(SM_Y):=OBS_SMOOTH(I,1);
    Y(CR_Y):=CREOBS(I);
    G_FUN(PTS_TO_HIT(I),Y,P,-1,G,DGY,DGP,SENSE);
    G_FUN(PTS_TO_HIT(I),Y,P,2,G,DGY,DGP,SENSE);
    DEL:=(PTS_TO_HIT(I)-PTS_TO_HIT(I-1));
    FOR J:=1 UNTIL 2 DO
      BEGIN
      K:=K+1;
      IF SM_Y=J THEN
        BEGIN
        CASE SMF OF
          BEGIN
          RES(K):=G(J)-DSPLN_AL(PTS_TO_HIT(I),
          SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
          N_SPLN_PAR(1));
          RES(K):=G(J)-DHERM(PTS_TO_HIT(I),
          HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))
          END;
        END
      ELSE RES(K):=.5L*(G(J)+PREV_G(J))-
      (CREOBS(I)-CREOBS(I-1))/DEL;
      KK:=0;
      FOR L:=1 UNTIL N_PAR DO
        BEGIN
        KK:=KK+1;
        IF SM_Y=J THEN
        JAC(K,KK):=DGP(J,L)
        ELSE
        JAC(K,KK):=.5L*(DGP(J,L)+PREV_DGP(J,L));
        END;
      END;
    COMMENT UPDATE PREV_G,PREV_DGP;
```

```
    FOR I:=1 UNTIL 2 DO PREV_G(I):=G(I);
    FOR I:=1 UNTIL 2 DO
    FOR J:=1 UNTIL N_PAR DO
    PREV_DGP(I,J):=DGP(I,J);
    END;
  COMMENT FORM F;
  F:=0;
  FOR I:=1 UNTIL M DO F:=F+RES(I)**2;
  COMMENT FORM GRADIENT;
  FOR I:=1 UNTIL N_PAR DO
    BEGIN
    SUM:=0;
    FOR J:=1 UNTIL M DO
    SUM:=SUM+JAC(J,I)*RES(J);
    GRAD(I):=SUM;
    END;
  END CRE_FUNC;
COMMENT COMMAND PROPER FOLLOWS;
N_PTS_HIT:=INTDATA(1);
N_PAR:=INTDATA(4);
SMF:=INTDATA(5);
OUT_SEG:=INTDATA(10);
DFIT_LIN:=INTDATA(12);
SM_Y:=STATES_OBS(1);
CR_Y:=3-SM_Y;
M:=2*(N_PTS_HIT-1);
COMMENT %FILE EMPTY TRMPORARY FILE -SC1;
IF DFIT_LIN~=1 THEN
  BEGIN
  COMMENT %OUTPUT
  ENTER STARTING LAMBDA, RELATIVE AND ABSOLUTE TOLERENCES
  FOR MARQUARDT
  %INPUT (LAM),(EPS_R),(EPS_A)
  %OUTPUT TO USER IF IN BATCH, TO -SC1 IF NOT,
  MARQUARDT USED IN DFIT FOR SPARSE GAUSS NEWTON
  OUTPUT REFERENCE NUMBER IS (OUT_SEG), LAMBDA,
  RELATIVE AND ABSOLUTE TOLERENCES ARE
  (LAM),(EPS_R),(EPS_A);
  MARQUARDT(EPS_R,EPS_A,M,N_PAR,CRE_FUNC,PAR,LAM,SVD_AL)
  END
ELSE
  BEGIN
  COMMENT LINEAR LEAST SQUARES USING SVD;
  LONG REAL SING_CUTOFF;
  LONG REAL ARRAY JAC(1::M,1::N_PAR);
  LONG REAL ARRAY RES(1::M);
  LONG REAL ARRAY SP,S,GRAD(1::N_PAR);
  LONG REAL ARRAY V(1::N_PAR,1::N_PAR);
  LONG REAL ARRAY A(1::M,1::N_PAR+1);
  LONG REAL SUM,F;
  INTEGER K;
```

```
  LONG REAL ARRAY U(1::M,1::N_PAR);
  INTEGER EFLAG;
  SING_CUTOFF:=.00001L;
  CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
  FOR J:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL M DO
  A(I,J):=JAC(I,J);
  K:=0;
  FOR I:=2 UNTIL N_PTS_HIT DO
    BEGIN
    FOR J:=1 UNTIL 2 DO
      BEGIN
      K:=K+1;
      IF SM_Y=J THEN
        BEGIN
        CASE SMF OF
          BEGIN
          A(K,N_PAR+1):=DSPLN_AL(PTS_TO_HIT(I),
          SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
          N_SPLN_PAR(1))-V0(J);
          A(K,N_PAR+1):=DHERM(PTS_TO_HIT(I),
          HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))-V0(J)
          END;
        END
      ELSE
      A(K,N_PAR+1):=(CREOBS(I)-CREOBS(I-1))/
      (PTS_TO_HIT(I)-PTS_TO_HIT(I-1));
      END;
    END;
  SVD_AL(A,S,U,V,M,N_PAR,M,N_PAR,1,0,N_PAR);
  FOR I:=1 UNTIL N_PAR DO
  IF (S(I)/S(1)>SING_CUTOFF) THEN SP(I):=A(I,N_PAR+1)/S(I)
  ELSE SP(I):=0.;
  FOR I:=1 UNTIL N_PAR DO
    BEGIN
    SUM:=0.;
    FOR J:=1 UNTIL N_PAR DO
    SUM:=SUM+V(I,J)*SP(J);
    PAR(I):=SUM;
    END;
  COMMENT %OUTPUT TO USER
  SOLUTION IN LINEAR DFIT FOR SPARSE GAUSS NEWTON IS (PAR)
  SINGULAR VALUES ARE (S)
  SINGULAR VALUE REJECTION RATIO IS (SING_CUTOFF);
  CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
  COMMENT %OUTPUT TO USER
  SUM OF SQUARES OF RESIDUALS IS (F);
  END;
END DFITITER.
```

```
      %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

           Integral fitting using created observations

      %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE IFIT_CRE(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,DSPLN_AL,HERM,DHERM;
LONG REAL ARRAY CREOBS(*);
LONG REAL ARRAY DCREOBS(*));
  BEGIN
  COMMENT IFIT ON 2 STATE VARIABLES WHEN OBSERVATIONS
  ON ONE HAVE
  BEEN GUESSED AT--LINEAR OPTION AVAILABLE;
  INTEGER N_PTS_HIT,N_PAR,SMF,DFIT_LIN,KK;
  LONG REAL LAM,EPS_R,EPS_A;
  INTEGER SM_Y,CR_Y,M,OUT_SEG;
  LONG REAL ARRAY V0(1::2);
  COMMENT %EXTERNAL SVD_AL,MARQUARDT,CMD_AL;
  PROCEDURE CRE_FUNC(LONG REAL ARRAY P(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
  LONG REAL ARRAY JAC(*,*);
  LONG REAL ARRAY GRAD(*);
  INTEGER RESULT EFLAG);
    BEGIN
    COMMENT THIS PROCEDURE PROVIDES INFORMATION TO THE LEAST
    SQUARES MINIMIZER WHEN GUESSED OBSERVATIONS ARE USED;
    LONG REAL ARRAY SENSE,DGY(1::1,1::1);
    LONG REAL ARRAY Y,G,INT_G,PREV_G(1::2);
    LONG REAL ARRAY DGP,INT_DGP,PREV_DGP(1::2,1::N_PAR);
    INTEGER K,KK;
    LONG REAL SUM,DEL2;
    EFLAG:=0;
    K:=0;
    Y(SM_Y):=OBS_SMOOTH(1,1);
    Y(CR_Y):=CREOBS(1);
    G_FUN(PTS_TO_HIT(1),Y,P,-1,PREV_G,DGY,DGP,SENSE);
    G_FUN(PTS_TO_HIT(1),Y,P,2,G,DGY,PREV_DGP,SENSE);
    G_FUN(PTS_TO_HIT(1),INT_G,P,3,G,DGY,DGP,INT_DGP);
    FOR I:=1 UNTIL 2 DO V0(I):=INT_G(I);
```

```
    FOR I:=2 UNTIL N_PTS_HIT DO
      BEGIN
      Y(SM_Y):=OBS_SMOOTH(I,1);
      Y(CR_Y):=CREOBS(I);
      G_FUN(PTS_TO_HIT(I),Y,P,-1,G,DGY,DGP,SENSE);
      G_FUN(PTS_TO_HIT(I),Y,P,2,G,DGY,DGP,SENSE);
      DEL2:=(PTS_TO_HIT(I)-PTS_TO_HIT(I-1))/2.L;
      FOR I:=1 UNTIL 2 DO
      INT_G(I):=INT_G(I)+DEL2*(G(I)+PREV_G(I));
      FOR I:=1 UNTIL 2 DO
      FOR J:=1 UNTIL N_PAR DO
      INT_DGP(I,J):=INT_DGP(I,J)+DEL2*(DGP(I,J)+PREV_DGP(I,J));
      FOR J:=1 UNTIL 2 DO
        BEGIN
        K:=K+1;
        IF SM_Y=J THEN
          BEGIN
          CASE SMF OF
            BEGIN
            RES(K):=INT_G(J)-SPLN_AL(PTS_TO_HIT(I),
            SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
            N_SPLN_PAR(1));
            RES(K):=INT_G(J)-HERM(PTS_TO_HIT(I),
            HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))
            END;
          END
        ELSE RES(K):=INT_G(J)-CREOBS(I);
        FOR L:=1 UNTIL N_PAR DO
        JAC(K,L):=INT_DGP(J,L);
        END;
      COMMENT UPDATE PREV_G,PREV_DGP;
      FOR I:=1 UNTIL 2 DO PREV_G(I):=G(I);
      FOR I:=1 UNTIL 2 DO
      FOR J:=1 UNTIL N_PAR DO
      PREV_DGP(I,J):=DGP(I,J);
      END;
    COMMENT FORM F;
    F:=0;
    FOR I:=1 UNTIL M DO F:=F+RES(I)**2;
    COMMENT FORM GRADIENT;
    FOR I:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0;
      FOR J:=1 UNTIL M DO
      SUM:=SUM+JAC(J,I)*RES(J);
      GRAD(I):=SUM;
      END;
    END CRE_FUNC;
  COMMENT COMMAND PROPER FOLLOWS;
  N_PTS_HIT:=INTDATA(1);
  N_PAR:=INTDATA(4);
```

```
SMF:=INTDATA(5);
OUT_SEG:=INTDATA(10);
DFIT_LIN:=INTDATA(12);
SM_Y:=STATES_OBS(1);
CR_Y:=3-SM_Y;
M:=2*(N_PTS_HIT-1);
COMMENT %FILE E,PTY TEMPORARY FILE -SC1;
IF DFIT_LIN~=1 THEN
  BEGIN
  COMMENT %OUTPUT TO USER
  ENTER STARTING LAMBDA, RELATIVE AND ABSOLUTE TOLERENCES
  FOR MARQUARDT PROCEDURE
  %INPUT (LAM),(EPS_R),(EPS_A)
  %OUTPUT TO -SC1
  MARQUARDT USED IN IFIT OPTION 2
  LAMBDA, RELATIVE AND ABSOLUTE TOLERENCES ARE
  (LAM),(EPS_R),(EPS_A);
  MARQUARDT(EPS_R,EPS_A,M,N_PAR,CRE_FUNC,PAR,LAM,SVD_AL);
  END
ELSE
  BEGIN
  COMMENT LINEAR LEAST SQUARES USING SVD;
  LONG REAL SING_CUTOFF;
  LONG REAL ARRAY JAC(1::M,1::N_PAR);
  LONG REAL ARRAY RES(1::M);
  LONG REAL ARRAY SP,S,GRAD(1::N_PAR);
  LONG REAL ARRAY V(1::N_PAR,1::N_PAR);
  LONG REAL ARRAY A(1::M,1::N_PAR+1);
  LONG REAL SUM,F;
  INTEGER K;
  LONG REAL ARRAY U(1::M,1::N_PAR);
  INTEGER EFLAG;
  SING_CUTOFF:=.00001L;
  CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
  FOR J:=1 UNTIL N_PAR DO
  FOR I:=1 UNTIL M DO
  A(I,J):=JAC(I,J);
  K:=0;
  FOR I:=2 UNTIL N_PTS_HIT DO
    BEGIN
    FOR J:=1 UNTIL 2 DO
      BEGIN
      K:=K+1;
      IF SM_Y=J THEN
        BEGIN
        CASE SMF OF
          BEGIN
          A(K,N_PAR+1):=SPLN_AL(PTS_TO_HIT(I),
          SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
          N_SPLN_PAR(1))-V0(J);
          A(K,N_PAR+1):=HERM(PTS_TO_HIT(I),
```

```
            HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1))-V0(J)
            END;
          END
        ELSE A(K,N_PAR+1):=CREOBS(I)-V0(J);
        END;
      END;
    SVD_AL(A,S,U,V,M,N_PAR,M,N_PAR,1,0,N_PAR);
    FOR I:=1 UNTIL N_PAR DO
    IF (S(I)/S(1)>SING_CUTOFF) THEN SP(I):=A(I,N_PAR+1)/S(I)
    ELSE SP(I):=0.;
    FOR I:=1 UNTIL N_PAR DO
      BEGIN
      SUM:=0.;
      FOR J:=1 UNTIL N_PAR DO
      SUM:=SUM+V(I,J)*SP(J);
      PAR(I):=SUM;
      END;
    COMMENT %OUTPUT TO USER
    LINEAR OPTION IN IFIT OPTION 2 PARAMETERS FOUND ARE
    (PAR), SINGULAR VALUES ARE (S), SINGULAR VALUE
    REJECTION LEVEL IS (SING_CUTOFF);
    CRE_FUNC(PAR,F,RES,JAC,GRAD,EFLAG);
    COMMENT %OUTPUT TO USER
    SUM OF SQUARES OF RESIDUALS IS (F);
    END;
  END IFIT_CRE.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

      Iterated integral fitting (subsystem integration)

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE IFITI(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,
DSPLN_AL,HERM,DHERM;
LONG REAL ARRAY CREOBS(*);
LONG REAL ARRAY DCREOBS(*);
PROCEDURE IFIT_CRE);
  BEGIN
```

```
COMMENT ITERATED IFIT ON 2 STATE VARIABLES
INTEGRATION OF SUBSYSTEMS TO UPDATE GUESSED OBSERVATIONS;
COMMENT %EXTERNAL TRAP;
PROCEDURE G(LONG REAL VALUE T;
LONG REAL ARRAY Y(*);
LONG REAL ARRAY P(*);
INTEGER VALUE OPTION;
LONG REAL ARRAY G(*);
LONG REAL ARRAY DGY(*,*);
LONG REAL ARRAY DGP(*,*);
LONG REAL ARRAY ISEN(*,*));
  BEGIN
  COMMENT INTERFACE TO G_FUN WHEN ONLY 1 STATE VAR
  IS INTEGRATED.;
  LONG REAL ARRAY Y1,G1(1::2);
  LONG REAL ARRAY DGP1,ISEN1(1::2,1::N_PAR);
  LONG REAL ARRAY DGY1(1::2,1::2);
  Y1(CR_Y):=Y(1);
  Y1(SM_Y):=CASE SMF OF
  (SPLN_AL(T,SPLN_COEF(*,1),JOINTS(*,1),NJOINTS(1),
  N_SPLN_PAR(1)),
  HERM(T,HERM_COEF(*,*,1),JOINTS(*,1),NJOINTS(1)));
  G_FUN(T,Y1,P,OPTION,G1,DGY1,DGP1,ISEN1);
  IF (OPTION=1) OR (OPTION=-1) THEN G(1):=G1(CR_Y);
  IF OPTION=1 THEN DGY(1,1):=DGY1(CR_Y,CR_Y);
  IF OPTION=3 THEN Y(1):=Y1(CR_Y);
  END G;
INTEGER N_PTS_HIT,N_PAR,SMF,OUTPUT,METHOD_FLAG,OUTPUT_SUP,
OUT_SEG,INT_PROC,SM_Y,CR_Y,NO_ITER,EFLAG,OUT1;
STRING(1) ANS;
LONG REAL ARRAY B(1::1,1::INTDATA(1));
LONG REAL ARRAY BJAC(1::1,1::1);
COMMENT COMMAND PROPER FOLLOWS;
N_PTS_HIT:=INTDATA(1);
N_PAR:=INTDATA(4);
SMF:=INTDATA(5);
OUTPUT:=INTDATA(6);
METHOD_FLAG:=INTDATA(7);
OUTPUT_SUP:=INTDATA(9);
OUT_SEG:=INTDATA(10);
INT_PROC:=INTDATA(11);
SM_Y:=STATES_OBS(1);
CR_Y:=3-SM_Y;
REPT:IFIT_CRE(G_FUN,PAR,INTDATA,OBS_SMOOTH,STATES_OBS,
PTS_TO_HIT,JOINTS,SPLN_COEF,HERM_COEF,NJOINTS,N_SPLN_PAR,
SPLN_AL,DSPLN_AL,HERM,DHERM,
CREOBS,DCREOBS);
COMMENT INTEGRATION OF SUBSYSTEM (TRAPEZOIDAL RULE
IMPLEMENTED);
IF OUTPUT_SUP=1 THEN OUT1:=0 ELSE OUT1:=OUTPUT;
TRAP(PAR,PTS_TO_HIT,B,BJAC,1,N_PTS_HIT,N_PAR,0,G,EFLAG,0);
```

```
  IF EFLAG=1 THEN GO TO OUT;
  FOR I:=1 UNTIL N_PTS_HIT DO
  CREOBS(I):=B(1,I);
  COMMENT %OUTPUT
  IS ANOTHER ITERATION DESIRED? Y OR N
  %INPUT (ANS);
  IF ANS="Y" THEN GO TO REPT;
  OUT:END IFITI.


     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

            Continuation and quasi-multiple shooting

     %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE CONTI(PROCEDURE G_FUN;
LONG REAL ARRAY PAR(*);
INTEGER VALUE MODE;
INTEGER ARRAY INTDATA(*);
LONG REAL ARRAY OBS_SMOOTH(*,*);
INTEGER ARRAY STATES_OBS(*);
LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY JOINTS(*,*);
LONG REAL ARRAY SPLN_COEF(*,*);
LONG REAL ARRAY HERM_COEF(*,*,*);
INTEGER ARRAY NJOINTS(*);
INTEGER ARRAY N_SPLN_PAR(*);
LONG REAL PROCEDURE SPLN_AL,
DSPLN_AL,HERM,DHERM;
LONG REAL VALUE EPS,HMIN,HMAX);
  BEGIN
  COMMENT CONTINUATION FORM IFIT WITH BREAK POINTS
  AND WEIGHTING AT BREAK POINTS;
  INTEGER N_PTS_HIT,N_PAR,SMF,DFIT_LIN,KK;
  LONG REAL LAM,EPS_R,EPS_A;
  LONG REAL ARRAY INITY(1::1);
  LONG REAL ARRAY INITYP(1::1,1::1);
  INTEGER KFLAG;
  INTEGER NP,SM_Y,CR_Y,M,OUT_SEG;
  LONG REAL ARRAY V0(1::2);
  COMMENT %EXTERNAL TRAP, GEAR, SVD_AL, MARQUARDT,
  CMD_AL;
  PROCEDURE C_FUNC(LONG REAL ARRAY P(*);
  LONG REAL RESULT F;
  LONG REAL ARRAY RES(*);
  LONG REAL ARRAY JAC(*,*);
  LONG REAL ARRAY GRAD(*);
  INTEGER RESULT EFLAG);
    BEGIN
    COMMENT OPTIMIZATION FUNCTION FOR CONTINUATION METHOD;
```

```
INTEGER M,N_RES,OUT1;
LONG REAL SUM;
INTEGER I_BREAK;
PROCEDURE CG_FUN(LONG REAL VALUE T;
LONG REAL ARRAY Y(*);
LONG REAL ARRAY P(*);
INTEGER VALUE OPTION;
LONG REAL ARRAY G(*);
LONG REAL ARRAY DGY(*,*);
LONG REAL ARRAY DGP(*,*);
LONG REAL ARRAY ISEN(*,*));
  BEGIN
  COMMENT G_FUN INTERFACE FOR CONTINUATION METHOD USING
  QUASI MULTIPLE SHOOTING;
  LONG REAL ARRAY CY(1::N_STATE);
  IF (ABS OPTION)~=3 THEN
    BEGIN
    FOR I:=1 UNTIL N_STATE DO CY(I):=Y(I)*GA(I);
    FOR I:=1 UNTIL N_STATE_OBS DO
    CY(STATES_OBS(I)):=(CASE SMF OF
    (SPLN_AL(T,SPLN_COEF(*,I),JOINTS(*,I),NJOINTS(I),
    N_SPLN_PAR(I)),
    HERM(T,HERM_COEF(*,*,I),JOINTS(*,I),NJOINTS(I))))
    *(1.L-GA(STATES_OBS(I)))+CY(STATES_OBS(I));
    END;
  CASE ABS OPTION OF
    BEGIN
      BEGIN
      G_FUN(T,CY,P,OPTION,G,DGY,DGP,ISEN);
      IF OPTION>0 THEN
        BEGIN
        FOR J:=1 UNTIL N_STATE DO
        FOR K:=1 UNTIL N_STATE DO
        DGY(J,K):=DGY(J,K)*GA(K);
        END;
      END;
      BEGIN
      G_FUN(T,CY,P,OPTION,G,DGY,DGP,ISEN);
      END;
      BEGIN
      IF I_BREAK=1 THEN
      G_FUN(T,Y,P,OPTION,G,DGY,DGP,ISEN)
      ELSE
        BEGIN
        INTEGER INDX;
        INDX:=BREAK(I_BREAK-1);
        FOR I:=1 UNTIL
        N_STATE DO Y(I):=STATE(INDX,I)*GA1(I);
        FOR I:=1 UNTIL N_STATE_OBS DO
        Y(STATES_OBS(I)):=(CASE SMF OF (
        SPLN_AL(T,SPLN_COEF(*,I),JOINTS(*,I),
```

```
        NJOINTS(I),N_SPLN_PAR(I)),
        HERM(T,HERM_COEF(*,*,I),JOINTS(*,I),NJOINTS(I))))*
        (1.L-GA1(STATES_OBS(I)))+Y(STATES_OBS(I));
        IF OPTION>0 THEN
        FOR I:=1 UNTIL N_STATE DO
        FOR J:=1 UNTIL N_PAR DO
        ISEN(I,J):=GA1(I)*JACOBIAN((INDX-1)*N_STATE+I,J);
        END;
      END
    END;
  END CG_FUN;
FOR I:=1 UNTIL N_BREAK DO
  BEGIN
  COMMENT INTEG TO BREAK(I);
  LONG REAL ARRAY SAMPLE(1::BREAK(I)-BREAK(I-1)+1);
  INTEGER N_PTS;
  LONG REAL ARRAY B(1::N_STATE,1::BREAK(I)-BREAK(I-1)+1);
  LONG REAL ARRAY
  JB(1::N_PAR,1::(BREAK(I)-BREAK(I-1)+1)*N_STATE);
  INTEGER INDX,JI;
  I_BREAK:=I;
  N_PTS:=BREAK(I)-BREAK(I-1)+1;
  FOR J:=1 UNTIL N_PTS DO
  SAMPLE(J):=PTS_TO_HIT(BREAK(I)-N_PTS+J);
  COMMENT INTEGRATE;
  IF OUTPUT_SUP=1 THEN OUT1:=0 ELSE OUT1:=OUTPUT;
  EFLAG:=0;
  CASE INT_PROC OF
    BEGIN
      BEGIN
      GEAR(P,SAMPLE,B,JB,N_STATE,N_PTS,EPS,HMIN,HMAX,
      N_PAR,1,CG_FUN,KFLAG,OUT1,METHOD_FLAG,0,
      INITY,INITYP);
      IF KFLAG~=1 THEN BEGIN EFLAG:=1;GO TO OUT;END;
      END;
      BEGIN
      TRAP(P,SAMPLE,B,JB,N_STATE,N_PTS,N_PAR,
      1,CG_FUN,EFLAG,OUT1);
      IF EFLAG=1 THEN GO TO OUT;
      END
    END;
  JI:=1;
  INDX:=BREAK(I_BREAK-1);
  IF INDX~=1 THEN JI:=2;
  FOR J:=JI UNTIL N_PTS DO
  FOR K:=1 UNTIL N_STATE DO
    BEGIN
    STATE(INDX-1+J,K):=B(K,J);
    FOR L:=1 UNTIL N_PAR DO
    JACOBIAN((INDX-2+J)*N_STATE+K,L):=
    JB(L,(J-1)*N_STATE+K);
```

```
      END;
    END I;
  COMMENT EXTRACT NON WEIGHTED JACOBIAN
  AND FORM RESIDUAL;
  FOR K:=1 UNTIL N_PAR DO
    BEGIN
    M:=0;
    FOR I:=1 UNTIL N_PTS_HIT DO
    FOR J:=1 UNTIL N_STATE_OBS DO
      BEGIN
      M:=M+1;
      JAC(M,K):=JACOBIAN((I-1)*N_STATE+STATES_OBS(J),K);
      RES(M):=STATE(I,STATES_OBS(J))-
      (CASE SMF OF(
      SPLN_AL(PTS_TO_HIT(I),SPLN_COEF(*,J),JOINTS(*,J)
      ,NJOINTS(J),
      N_SPLN_PAR(J)),
      HERM(PTS_TO_HIT(I),HERM_COEF(*,*,J),JOINTS(*,J),
      NJOINTS(J))))
      END;
    END;
  N_RES:=M;
  COMMENT SPECIAL WEIGTING OF BREAK POINTS;
  FOR I:=1 UNTIL N_BREAK DO
    BEGIN
    INTEGER INDX;
    INDX:=(BREAK(I)-1)*N_STATE_OBS;
    FOR J:=1 UNTIL N_STATE_OBS DO
    RES(INDX+J):=RES(INDX+J)*W(I);
    FOR J:=1 UNTIL N_STATE_OBS DO
    FOR K:=1 UNTIL N_PAR DO
    JAC(INDX+J,K):=JAC(INDX+J,K)*W(I);
    END;
  F:=0.L;
  FOR I:=1 UNTIL N_RES DO F:=F+RES(I)**2;
  COMMENT FORM GRADIENT;
  FOR I:=1 UNTIL N_PAR DO
    BEGIN
    SUM:=0.L;
    FOR J:=1 UNTIL N_RES DO
    SUM:=SUM+JAC(J,I)*RES(J);
    GRAD(I):=SUM;
    END;
  OUT:END C_FUNC;
INTEGER N_STATE,N_STATE_OBS,OUTPUT_SUP,OUTPUT,INT_PROC,METHOD
,METHOD_FLAG;
LONG REAL ARRAY STATE(1::INTDATA(1),1::INTDATA(3));
LONG REAL ARRAY JACOBIAN(1::INTDATA(1)
*INTDATA(3),1::INTDATA(4));
INTEGER N_BREAK;
INTEGER ARRAY BREAK(0::50);
```

```
LONG REAL ARRAY W(0::50);
LONG REAL ARRAY GA,GA1(1::INTDATA(3));
N_PTS_HIT:=INTDATA(1);
N_STATE_OBS:=INTDATA(2);
N_STATE:=INTDATA(3);
N_PAR:=INTDATA(4);
SMF:=INTDATA(5);
OUTPUT:=INTDATA(6);
METHOD_FLAG:=INTDATA(7);
OUTPUT_SUP:=INTDATA(9);
OUT_SEG:=INTDATA(10);
INT_PROC:=INTDATA(11);
IF MODE=0 THEN
  BEGIN
  INTEGER N_BRK; LONG REAL GAM;
  COMMENT %OUTPUT TO USER
  ENTER NUMBER OF BREAK POINTS
  ENTER 0 FOR NO BREAK POINTS
  %INPUT (N_BRK);
  IF N_BRK~=0 THEN
    BEGIN
    COMMENT %OUTPUT TO USER
    ENTER SAMPLE TIME SUBSCRIPTS FOR BREAK POINTS
    DO NOT INCLUDE FIRST OR LAST SAMPLE TIME;
    FOR I:=1 UNTIL N_BRK DO
    COMMENT %INPUT BREAK(I);
    BREAK(0):=1;BREAK(N_BRK+1):=N_PTS_HIT;
    N_BREAK:=N_BRK+1;
    END
  ELSE
    BEGIN
    BREAK(0):=1;
    BREAK(1):=N_PTS_HIT;
    N_BREAK:=1;
    END;
  COMMENT %OUTPUT
  ENTER CONTINUATION PARAMETER
  FOR INITIAL VALUE PROBLEM
  %INPUT (GAM);
  FOR I:=1 UNTIL N_STATE DO GA(I):=GAM;
  IF N_BRK~=0 THEN
    BEGIN
    COMMENT %OUTPUT
    ENTER CONTINUATION PARAMETER
    FOR BREAK POINTS.  EACH COMPONENT
    CORRESPONDS TO THE STATE VARIABLE
    WITH THE SAME SUBSCRIPT;
    FOR I:=1 UNTIL N_STATE DO
    COMMENT %INPUT (GA1(I))
    %OUTPUT
    ENTER WEIGHTS AT BREAK POINTS
```

```
      ONE ENTRY FOR EACH BREAK POINT;
      FOR I:=1 UNTIL N_BRK DO
      COMMENT %INPUT (W(I));
      W(0):=1.L;
      W(N_BREAK):=1.L;
      END
    ELSE
    W(1):=1.L;
    END
  ELSE
    BEGIN
    COMMENT MODE NOT 0. IFIT OPTION;
    N_BREAK:=1;
    BREAK(0):=1; BREAK(1):=N_PTS_HIT;
    FOR I:=1 UNTIL N_STATE DO GA(I):=0.L;
    W(1):=1.L;
    END;
  COMMENT %FILE EMPTY TEMPORARY FILE -SC1
  %OUTPUT
  ENTER STARTING LAMBDA, REL AND ABS TOL FOR MARQUARDT
  %INPUT (LAM),(EPS_R),(EPS_A);
  COMMENT %OUTPUT TO USER IF IN BATCH, TO -SC1 IF NOT,
  MARQUARDT USED IN CONTINUATION PROCEDURE
  OUTPUT REFERENCE NUMBER (OUT_SEG),
  LAMBDA (LAM), RELATIVE ERROR TOLERENCE (EPS_R),
  ABSOLUTE ERROR TOLERENCE (EPS_A);
  MARQUARDT(EPS_R,EPS_A,N_PTS_HIT*N_STATE_OBS,
  N_PAR,C_FUNC,PAR,LAM,SVD_AL);
  END CONTI.


    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

         Description of externally defined procedures

    %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%% %%%%

PROCEDURE CMD_AL(STRING(80) VALUE CMD;
INTEGER VALUE N);
  BEGIN
  COMMENT EXECUTE AN MTS COMMAND;
  END.
PROCEDURE CHECK_BATCH(LOGICAL RESULT BATCH);
  BEGIN
  COMMENT CHECK IF IN BATCH MODE;
  END.
PROCEDURE SVD_AL(LONG REAL ARRAY AD(*,*);
LONG REAL ARRAY S(*);
LONG REAL ARRAY UD(*,*);
LONG REAL ARRAY VD(*,*);
INTEGER VALUE MDIM,NDIM,M,N,NRHS,NU,NV);
```

```
  BEGIN
  COMMENT SINGULAR VALUE DECOMPOSITION
  USES U.B.C. FORTRAN VERSION OF
  PROCEDURE BY G. GOLUB AND C. REINSCH,
  NUMER. MATH. 14 (1970) 403-420;
  END.
PROCEDURE G_FUN(LONG REAL VALUE T;
LONG REAL ARRAY Y(*);LONG REAL ARRAY P(*);
INTEGER VALUE OPTION;LONG REAL ARRAY G(*);
LONG REAL ARRAY DGY(*,*);LONG REAL ARRAY DGP(*,*);
LONG REAL ARRAY PREV_SENSE(*,*));
  BEGIN
  COMMENT MODEL DEFINITION ;
  END.
PROCEDURE ECHO1(INTEGER VALUE CAS);
  BEGIN
  COMMENT ECHO IBM 3270 CONVERSATION BUFFER
  (AS USED BY MTS AT U.B.C.) AND RESET POINTERS SO
  NO OVERLAP OCCURS;
  END.
PROCEDURE SPLINT_AL(LONG REAL ARRAY X(*);
LONG REAL ARRAY Y(*);LONG REAL ARRAY W(*);
INTEGER VALUE N;LONG REAL ARRAY P(*);
INTEGER VALUE RESULT M;LONG REAL ARRAY XJOINTS(*);
INTEGER VALUE NJOINT);
  BEGIN
  COMMENT LEAST SQUARES SPLINE APPROXIMATION;
  END.
LONG REAL PROCEDURE SPLN_AL(LONG REAL VALUE X;
LONG REAL ARRAY P(*);LONG REAL ARRAY XJOINT(*);
INTEGER VALUE NJOINT,M);
  BEGIN
  COMMENT CALCULATE SPLINE APPROXIMATION USING
  SPLINT_AL RESULTS;
  END.
LONG REAL PROCEDURE DSPLN_AL(LONG REAL VALUE X;
LONG REAL ARRAY P(*);LONG REAL ARRAY XJOINT(*);
INTEGER VALUE NJOINT,M);
  BEGIN
  COMMENT CALCULATE DERIVATIVE APPROXIMATION USING
  SPLINT_AL RESULTS;
  END.
PROCEDURE HERMIT_AL(LONG REAL ARRAY X(*);
LONG REAL ARRAY Y(*);LONG REAL ARRAY KNOTS(*);
INTEGER VALUE N,NKNOTS;LONG REAL ARRAY COEF(*,*);
INTEGER VALUE NC1;INTEGER RESULT FLAG);
  BEGIN
  COMMENT LEAST SQUARES PIECEWISE CUBIC HERMITE
  APPROXIMATION;
  END.
LONG REAL PROCEDURE HERM(LONG REAL VALUE X;
```

```
LONG REAL ARRAY COEF(*,*);LONG REAL ARRAY JOINTS(*);
INTEGER VALUE NJOINTS);
  BEGIN
  COMMENT CALCULATE HERMITE APPROXIMATION USING
  RESULTS OF HERMIT_AL;
  END.
LONG REAL PROCEDURE DHERM(LONG REAL VALUE X;
LONG REAL ARRAY COEF(*,*);LONG REAL ARRAY JOINTS(*);
INTEGER VALUE NJOINTS);
  BEGIN
  COMMENT CALCULATE DERIVATIVE APPROXIMATION USING
  RESULTS OF HERMIT_AL;
  END.
PROCEDURE CREATE_DATA(LONG REAL ARRAY SIM_PAR(*);
LONG REAL ARRAY START_PAR(*);LONG REAL ARRAY PAR(*);
LONG REAL ARRAY OBS_PLACE(*);LONG REAL ARRAY PTS_TO_HIT(*);
LONG REAL ARRAY OBS(*,*);LONG REAL ARRAY STATE(*,*);
LONG REAL ARRAY JACOBIAN(*,*);INTEGER ARRAY OBS_STATUS(*);
INTEGER ARRAY STATES_OBS(*);INTEGER ARRAY INTDATA(*);
STRING(31) VALUE MODEL;PROCEDURE EG_FUN,GEAR,STANDARD_HIT;
LONG REAL RESULT STD_DEV;LONG REAL VALUE INITIAL_TIME;
INTEGER VALUE KFLAG,OUTPUT);
  BEGIN
  COMMENT SIMULATE OBSERVATIONS;
  END.
LONG REAL PROCEDURE FVALUE_AL(LONG REAL VALUE P;
INTEGER VALUE N1,N2);
  BEGIN
  COMMENT STATISTICAL F DISTRIBUTION;
  END.
PROCEDURE SMT(REAL ARRAY X(*);
REAL ARRAY Y(*);REAL ARRAY P(*);
REAL ARRAY SI(*);REAL ARRAY T(*);
REAL ARRAY S(*);REAL ARRAY S1(*);
REAL ARRAY S2(*);INTEGER VALUE N,IFF,M);
  BEGIN
  CUBIC SPLINE INTERPOLATION;
  END.
PROCEDURE FSLE_AL(INTEGER VALUE N,NDIMA;
LONG REAL ARRAY A(*,*);INTEGER VALUE NSOL,NDIMBX;
LONG REAL ARRAY B(*,*);LONG REAL ARRAY X(*,*);
INTEGER ARRAY IPERM(*);INTEGER VALUE NDIMT;
LONG REAL ARRAY TMP(*,*));
  BEGIN
  COMMENT SOLVE A LINEAR SYSTEM OF EQUATIONS;
  END.
PROCEDURE DIFF(INTEGER VALUE N;
LONG REAL VALUE RESULT LT;LONG REAL ARRAY LY(*,*);
LONG REAL ARRAY LSAVE(*,*);LONG REAL VALUE RESULT LH;
LONG REAL VALUE LHMIN,LHMAX,LEPS;INTEGER VALUE METHOD_FLAG;
LONG REAL ARRAY LYMAX(*);LONG REAL ARRAY LERROR(*);
```

```
INTEGER VALUE RESULT KFLAG;
INTEGER VALUE RESULT JSTART;
INTEGER VALUE MAXDER;LONG REAL ARRAY AA(*);
INTEGER RESULT ORDER;PROCEDURE FUN;
LONG REAL ARRAY P(*);REAL ARRAY PW(*));
  BEGIN
  COMMENT ALGOL FORTRAN INTERFACE TO GEAR'S CODE;
  END.
PROCEDURE ECHO(INTEGER VALUE CAS);
  BEGIN
  COMMENT SIMILAR TO ECHO1--USED IN INTERACTIVE FIT;
  END.
PROCEDURE ALGRAF_AL(REAL ARRAY X(*);
REAL ARRAY Y(*);
INTEGER VALUE N,NS);
  BEGIN
  COMMENT PLOT A SET OF DATA POINTS;
  END.
PROCEDURE PLOT_AL(REAL VALUE X,Y;INTEGER VALUE IPEN);
  BEGIN
  COMMENT MOVE PLOTTING PEN TO (X,Y), CAN BE UP OR
  DOWN DEPENDING ON IPEN;
  END.
PROCEDURE ALSIZE_AL(REAL VALUE XSIZE,YSIZE);
  BEGIN
  COMMENT SET SIZE OF PLOT;
  END.
PROCEDURE ALSCAL_AL(REAL VALUE XMIN,XMAX,YMIN,YMAX);
  BEGIN
  COMMENT SCALE DATA TO FIT PLOT;
  END.
PROCEDURE ALAXIS_AL(STRING(50) VALUE LABELX;
INTEGER VALUE NX;STRING(50) VALUE LABELY;
INTEGER VALUE NY);
  BEGIN
  COMMENT DRAW AXES;
  END.
PROCEDURE DFBAND(LONG REAL ARRAY DA(*);
LONG REAL ARRAY DB(*);INTEGER VALUE N,LHB,NRHS;
LONG REAL VALUE RESULT RATIO;LONG REAL RESULT DET;
INTEGER RESULT JEXP;INTEGER VALUE NSCALE);
  BEGIN
  COMMENT SOLVE A BANDED SYSTEM OF LINEAR EQUATIONS
  WITH A POSITIVE DEFINITE MATRIX USING
  A CHOLESKY DECOMPOSITION;
  END.
```