Title

ESTIMATION OF THE DYNAMICAL PARAMETERS OF THE CALVIN PHOTOSYNTHESIS CYCLE, OPTIMIZATION, and Ill CONDITIONED INVERSE PROBLEMS

Author

Milstein, Jaime.

Publication Date

1975-09-01

Thesis/dissertation

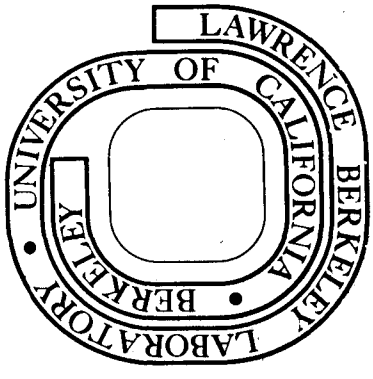
LBL-4264

ESTIMATION OF THE DYNAMICAL PARAMETERS OF THE CALVIN PHOTOSYNTHESIS CYCLE, OPTIMIZATION, AND I11 CONDITIONED INVERSE PROBLEMS

Jaime Milstein
(Ph.D. thesis)

September 1975

Prepared for the U.S. Energy Research and Development Administration under Contract W-7405-ENG-48



LBL-4264

DISCLAIMER

In memory of my mother, Rachel, and my father, Samuel whose eternal light has guided me through the labyrinths of life.

Acknowledgments

It is a pleasure to express my deeply felt gratitude to my mentor and friend, Professor Hans J. Bremermann, for his warm encouragement, thoughtful guidance, countless insights, and unlimited generosity throughout the making of this thesis.

I wish to thank the other members of the committee, Professors B. Parlett and Dr. A. J. Bassham, for their patience, constructive criticism, and helpful discussions.

I am indebted to Professors J. A. Bassham and M. Calvin and the members of the Biodynamic Laboratory for their excellent assistance and their many hours spent to obtain experimental data. Dr. Clifford Risk and Dr. Joel Swartz have provided invaluable suggestions and encouraging support throughout this work.

I extend many thanks to Myron Katz, who was especially helpful in making the introductions and general text easier to understand, and to Said Doss, Pat Gilmer, David Krumme, Stan Zietz and Bob Miller for their valuable comments and suggestions.

I am very grateful to my sister, Monica, my brother-in-law, Chaim, and their children, Naama, Eial and Yuval, who have been my inspiration in all personal endeavors and whose affection has always given me personal strength to overcome many difficult years. I wish to thank my aunts and

uncles, Helena, Piri, Simon and Moshe, for their moral and spiritual support. I give special thanks to my close friend, Shoshana, for her great understanding.

The acknowledgments would not be complete without extending thanks to many close friends, Carmela and Sam, Tova and Avi, Naftaly, and Dalia and Shmulik, to name a few, for their understanding and companionship throughout the difficult moments of my work.

Last but not least, I want to express my deep appreciation to Nora Lee, for her patience, warmth, and the wonderful job she has done in typing this manuscript.

INTRODUCTION

This thesis is concerned with the determination of the dynamic rate constants of the carbon reduction cycle (Calvin Cycle) and with the mathematical and computational tools required. The rate constant problem has been an open problem for over twenty years.

The dynamics of photosynthesis can be described by a system of eighteen ordinary non-linear differential equations which contain twenty-two unknown parameters -- the rate constants.

Determination of these constants from observable data is the concern of this thesis. Mathematically, it reduces to a non-linear fitting problem which in turn reduces to optimizing a transcendental function that is not known in closed form, but it can be evaluated pointwise through (laborious) computation.

The literature on optimization is vast and many algorithms are in use. The problem at hand can only be solved if a powerful, computationally efficient, non-stagnating optimization algorithm is used.

Thus the first part of the thesis is concerned with the comparison of different optimization algorithms and their performance on a variety of test problems. This is a mathematical-numerical problem in its own right which in turn is closely related to the problem of rootfinding for systems of polynomials and transcendental functions in many

variables. The research showed that the optimization algorithm by H. Brmermann outperformed all others tested.

In applying the optimization algorithm to the photosynthesis problem another problem was encountered. The dynamic equations that describe photosynthesis turned out to be very stiff and standard integration algorithms (Runge-Kutta) failed. Rate constants determination is an iterative process that requires repeated integration of the dynamic equations and a special stiff method (Gear's method) had to be adapted to the problem.

The earlier work of J. Swartz [83] which solves similar but smaller and simpler parameter determination problems showed the importance of an error analysis. Due to the intrinsic nature of the parameter determination problem, noise in the (experimental) data can result in a percentage error that can vary greatly (by orders of magnitude) between parameters. Hence an error analysis of the expected variance of the parameters is required if the results are to be meaningful. In the case of photosynthesis, this analysis involves matrices of partial derivatives of the system's equations with so many terms, that even a very small probability of making a mistake during symbol manipulation, results in a virtual certainty of erroneous terms in the complete matrices. Since, in this case, human error in symbol manipulation cannot be made small enough, automation of symbol manipulation proved necessary. To this end we

used a new experimental language called ALTRAN (Algebra Translator). ALTRAN is a language and system for performing symbolic computations on algebraic data. The basic capability of the language is to perform operations on rational expressions in one or more indeterminants. The system is designed to handle very large problems involving such data, with considerable efficiency.

Our mathematical description of the kinetic of the Calvin Cycle is a non-linear system of ordinary differential equations. Determination of the rate constants (kinetic parameters) is based on the knowledge of the trajectories of the intermediates of the system (the concentration of the intermediates change with time). A problem for which the trajectories are known and the parameters have to be determined is called an "inverse problem". Experimental error can cause errors in the observation of the trajectories which in turn can cause large errors in the determination of the parameters. Such a problem is called "ill-conditioned". Parameter identification of the Calvin Cycle turned out to be an ill-conditioned problem if only one set of initial conditions is used. To overcome the ill-conditioned nature of our model it is necessary to obtain accurate experimental data using several initial conditions, that is, performing several experiments with different initial concentration.

The experiments to obtain accurate data are elaborate, costly and require a substantial amount of time. M. Calvin,

J. A. Bassham and co-workers at the Chemical Biodynamics Laboratory at the University of California, Berkeley, will furnish sufficient experimental data for use in the parameter identification problem.

Table of Contents

Page

Tables

Abstract

This thesis is concerned with the determination of kinetic parameters of the Calvin photosynthesis cycle and the numerical tools required. A mathematical model with seventeen non-linear ordinary differential equations describing this cycle is presented; its unknown parameters are to be determined using data from observations of the state variables and an optimization technique developed herein. This method for parameter identification involves a non-linear optimization algorithm, first developed by Hans Bremermann, a computer routine for numerical integration of stiff systems and an error analysis technique, based on a method of Rosenbrook and Storey which was implemented and tested by Joel Swartz. Bremermann's optimization algorithm is tested and compared to other techniques frequently used on optimization and root-finding problems. Finally, in order to test both the mathematical model and parameter identification technique, an arbitrary choice of parameter-values was designated as the correct or exact parameter values and the technique was implemented using simulated "observational" data.

PART I: NON-LINEAR OPTIMIZATION

1.0 INTRODUCTION TO PART I

The first chapter of this thesis will center on discussing the problem of non-linear optimization. A survey on different methods is carried out with the aim of obtaining an understanding of the difficulties encountered while optimizing a real value function of many variables. Of central importance to this study is the performance of an algorithm as it depends on the dimensionality, i.e., the number of independent variables of the function whose extremum is sought. There is no theory which determines how the number of variables will affect the performance of an algorithm. Moreover the preparation of an algorithm to be used in a computer may turn out to be a very laborious task. Some algorithms require a subroutine containing the Jacobian or the Hessian matrix; if a real valued function of three variables is being optimized then the calculation of the Jacobian matrix will involve three formal differentiation of the function with respect to the variables, in that case, the Hessian will require nine differentiations. This computation can accurately be performed in a reasonable amount of time, but a fifty dimensional problem requires fifty formal differentiations to obtain the Jacobian and 2500 to obtain the Hessian. With only fifty variables this task is humanly impossible. As the dimensionality of a function increases the calculation of a Hessian matrix grows quadratically. Furthermore, some algorithms require the inversion or diagonalization of large dimensional linear

systems and these tasks are cumbersome and require a significant amount of computing time.

Because of the difficulties mentioned above there exists a gap in the understanding of the performance of algorithms on problems of large dimensionality. As the field of non-linear optimization evolved, the performance of an algorithm was tested not on theoretical grounds but rather by applying the algorithm to some "test" problems for which success or failure could be easily determined.

In this Chapter a comparison of Bremermann's optimizer, which has evolved as an algorithm to simulate the process of evolution, and some of the prominent algorithms found in the literature is made. Initially we compared their performances on functions having up to four variables and then on fifty dimensional problems.

The preparation of subroutines for use in an algorithm has to be considered. Since Bremermann's algorithm does not require any special subroutine except one that evaluates a function, we will see that Bremermann's optimizer is easy to implement.

The literature of non-linear optimization is vast and therefore it is difficult to describe all the methods which deal with the optimization problem. We extracted from the literature a representative set of algorithms and discussed their main features with the sole purpose of acquiring a basic knowledge of the methods and difficulties encountered in their use.

1.1 OPTIMIZATION: KEY TO PARAMETER DETERMINATION

In the next Chapter, non linear optimization will be used in determining the kinetic parameters of the carbon reduction cycle (Calvin Cycle). Because of the many kinetic parameters included in the Calvin Cycle and because derivatives cannot readily be computed, many optimization algorithm fail on this problem, in any case an optimization process for this many parameters is costly and computationally demanding Thus, the principal purpose of this Chapter is to explore different algorithms considered prominent in the literature. We shall discuss the main features of these methods and compare their actual performance with Bremermann's optimization algorithm in solving several "tests" problems.

1.2 UNCONSTRAINED OPTIMIZATION

The unconstrained minimization problem can be stated as follows:

$$\text{Minimize } F[\bar{x}] \text{ for } \bar{x} \in \mathbb{R}^n$$

where $F[\bar{x}]$ is a nonlinear function in n-variables; $\bar{x}$ is an n-dimensional vector, i.e., $\bar{x} = (x_1, \ldots, x_n)$ and $\mathbb{R}^n$ is the n-dimensional Euclidean space.

Definition: The function $F[\bar{x}]$ for which a minimum value is sought is called the objective function.

Definition: A point $\bar{x}^*$ is said to be a solution of the unconstrained minimization problem if

$$F[\bar{x}^*] \leq F[\bar{x}] \quad \text{for} \quad \forall\, \bar{x} \in \mathbb{R}^n$$

and $\bar{x}^*$ does not have to satisfy any constraints.

1.3 COMPUTATIONAL DIFFICULTIES IN OPTIMIZATION

Given an objective function $F[\bar{x}]$ the task of finding a global optimum (a solution to the optimization problem), in general is nontrivial. The computional difficulties can be classified into several categories:

1) Convergence properties of the method
2) Sensitivity of the method to initial guesses
3) Computational cost

In the next section we shall survey the main features of methods in optimization which are most widely used.

1.4 SURVEY OF METHODS OF UNCONSTRAINED OPTIMIZATION

The methods most widely used can be classified into two broad categories:

1) Direct search methods
2) Descent techniques

1.4.1 Direct Search Methods

Direct search methods do not require the calculation of derivatives but rather the evaluations of the objective function. Some examples of search methods follow:

1.4.2 Pattern Search Method: Hooke-Jeeves [1961]

This method consists of two kinds of moves: exploratory and pattern moves. In an exploratory move, the algorithm examines the local behavior of the function trying to locate a decrease in the objective function. When the value of the objective function decreases, there is an indication that a "valley" is present. A pattern move utilizes the information obtained from an exploratory move by progressing along such a "valley".

The Exploratory Move. This move consists of the following procedure: A steplength λ is chosen arbitrarily. Starting from the current iterate, $\bar{x} = (x_1, \ldots, x_n)$, a single step is taken along the direction of one coordinate (by adding preset increment, λ, to the particular coordinate considered). If the value of the objective function at this point decreased or remained unchanged compared with its initial value, then this step is considered successful, if its value increases the step is considered unsuccessful and that choice is not used. When an unsuccessful step is present, replace λ by $-\lambda$ and check if the new value of the objective function is smaller. If it increases the value then it is unsuccessful; choose another coordinate and perform the previous procedure. If in this exploration a point $\bar{y} = (y_1, \ldots, y_n)$ is found for which $F[\bar{y}] \leq F[\bar{x}]$ then $\bar{y}$ will be retained as a new initial point. When all n-coordinate directions have been tried the exploratory move is complete.

We should note that the point $\bar{y}$ obtained by the exploratory move may or may not be distinct from the initial point $\bar{x}$. If this is the case, it can be understood that either we are very close to the minimum or that the steplenth λ is too big to succeed in reducing the value of the objective function F. Thus a reduction of the steplenth is desirable and further explorations can be continued.

Mathematically the exploratory move is as follows: Let $\bar{x}_0 = (x_1, \ldots, x_n)$ be an initial guess; $a = x_j$ for $1 \leq j \leq n$, j an integer, be the current coordinate being considered; $\hat{\bar{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ the vector sought by an exploratory move; $F(x_1, \ldots, x_n)$ is our initial value of the objective function:

The basic iteration of the exploratory move is:

1) Initialize J i.e. $J = 1$
2) $a = x_j$
3) if $[F(\hat{x}_1, \ldots, \hat{x}_{j-1}, x_j+\lambda, \ldots, x_n) \leq F(\hat{x}_1, \ldots, \hat{x}_{j-1}, x_j, \ldots, x_n)]$ then set $a = x_j + \lambda$, go to Step 5. Otherwise go to Step 4 but if you have been to Step 4 for the same j go to Step 5.
4) Set λ to $-\lambda$ and go to Step 3.
5) $\hat{x}_j = a$ go to Step 6.
6) $j = j+1$ go to Step 2.

When all n-coordinates directions x_j $j = 1, \ldots, n$ have been tried the exploratory move is complete and we will have arrived at a new point $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)$.

The Pattern Move: This move consists of a single step from the present point $\hat{x}$. From the point $\hat{x}$ obtained in the exploratory move, obtain a new point $\bar{w} = (w_1,\ldots,w_n)$, by taking a step along the approximate gradient direction $\bar{s}$; i.e.

$$\bar{w} = \hat{x} + \bar{s}$$

where $\bar{s} = (\hat{x} - \bar{x})$

$$= \{(\hat{x}_1 - x_1), (\hat{x}_2 - x_2) \ldots (\hat{x}_n - x_n)\}$$

The interaction of Exploratory and Pattern Moves.

The method is iterative:

1) Exploratory Move → 2) Pattern Move → 3) Exploratory Move.

Each time that such a sequence of move is performed, a comparison of the value of the objective function at the points obtained by consecutive exploratory moves is made. The algorithm is described below:

Let $\hat{x}$ be the point obtained in 1)
Let $\overset{*}{x}$ be the point obtained in 3)

Then

a) If $F[\overset{*}{x}] \leq F[\hat{x}]$ go to b. Otherwise go to Step C

b) Perform a new Pattern Move (using the Point $\overset{*}{x}$), followed by an Exploratory Move. Return to Step 1.

c) $\lambda = \lambda/2$. Start an Exploratory Move around the Point $\hat{x}$ and follow it by a Pattern and Exploratory Move. Then go to a).

The iterative scheme is terminated when the size of the steplength λ is less than a given value γ.

1.4.3 METHOD OF ROTATING COORDINATES

It was first suggested by Rosenbrook [1960] that the coordinate system be rotated, from the current point of iteration, in such a way that one axis is oriented toward the "locally estimated direction of a local minimum" and the other axes are normal to it and mutually orthogonal.

The iteration steps of the algorithm

1) Let $S = \{s_1, s_2, s_3, \ldots s_n\}$ be an arbitrary orthogonal set of vectors in $\mathbb{R}^n$. Let $\Lambda = (\lambda_1, \ldots, \lambda_n)$ be an arbitrary n-tuple of numbers which will be used as the initial steplengths. Let $\bar{x} = (x_1, \ldots, x_n)$ be an n-tuple of real numbers representing a vector in $\mathbb{R}^n$ where $\bar{x} = x_1 s_1 + x_2 s_2 + \ldots + x_n s_n$. $\bar{x}$ is the initial guess. Let F: $\mathbb{R}^* \rightarrow \mathbb{R}$ be the function to be minimized.
2) This step is called "a sequence of minimizations". Its primary object is to obtain a new vector $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ which satisfies $F[\hat{x}] \leq F[\bar{x}]$. A secondary effect of this step is to provide information about "successful steplengths" ("successful" will be defined below).

If $F\left[(x_1 + \lambda_1)s_1 + \sum_{j=2}^{n} x_j s_j\right] \leq F\left[\sum_{j=1}^{n} x_j s_j\right]$ then λ_1 is successful. Try $\lambda_1 = 2\lambda_1$, repeat until you are no longer successful then $\hat{x}_1 = x_1 + \lambda_1$. Change to the next coordinate. Otherwise (i.e.

$\left(\text{i.e. } F\left[(x_1 + \lambda_1)s_1 + \sum_{j=2}^{n} x_j s_j\right] > F\left[\sum_{j=1}^{n} x_j s_j\right]\right)$ 1 is unsuccessful. Try $\lambda_1 = 1/2\ \lambda_1$ repeat until you are successful. Then $\hat{x} = x_1 + \lambda_1$ change to next coordinate.

For the j-th coordinate: if

$$F\left[\sum_{i=1}^{j-1} \hat{x}_i s_i + (x_j + \lambda_j)s_j + \sum_{k=j+1}^{n} x_k s_k\right] \leq$$

$$F\left[\sum_{i=1}^{j-1} x_i s_i + \sum_{k=j}^{n} \hat{x}_j s_j\right]$$

Then λ_j is successful. Try $\lambda_j = 2\lambda_j$ repeat until you are no longer successful then $\hat{x}_j = x_j + \lambda_j$, change to next coordinate. Otherwise λ_j is unsuccessful. Try $\lambda_j = 1/2\ \lambda_j$ repeat until you are successful then $\hat{x}_j = x_j + \lambda_j$, change to next coordinate.

3) Let θ_i denote the sum of all the successful steplengths in the direction s_i (<u>note</u>: $\theta_i \neq 0$).

4) Calculate a set of independent directions.

$P = (p_1, p_2, \ldots, p_n)$

$$(p_1,p_2,\ldots,p_n)=(s_1,s_2,\ldots,s_n)\begin{pmatrix}\theta_1 & & & \\ \theta_2 & \theta_2 & & \\ \vdots & & & \\ \theta_n & \theta_n & \cdots & \theta_n\end{pmatrix}$$

Note: If $\exists\ i \ni \theta_i = 0$ the procedure is terminated.

5) Using Gram-Schmidt orthogomalization procedure generate a new set of orthogonal directions.

Let $w_1 = p_1$

and $w_i = p_i - \sum_{j=1}^{i} \left[(P_i)^T T_j\right] T_j \quad i = 1, \ldots, n$

where $T_j = \frac{w_i}{\|w_i\|} \quad i = 1, \ldots, n$

will be the new search directions.

6) Repeat the procedure from Step 1, using the point $\hat{x}$ found in Step 2 and the new orthogonal directions, T, found in Step 5.

The search for the minimum is terminated when:

1) Either λ is smaller than a predetermined tolerance γ

or

2) The magnitude $\|p_0\|$ of the progress of several steps is less than a predertimined minimum value.

1.4.4 Simplex Method: Nelder & Mead [1965]

For the purpose of this method we use the following definition*:

Definition: "simplex" in $\mathbb{R}^n$ is a set of $n+1$ vectors

$$\underline{X} = \left\{ \bar{x}^1, \bar{x}^2, \bar{x}^3, \ldots, \bar{x}^n, \bar{x}^{n+1} \right\} \quad \text{in} \quad \mathbb{R}^n$$

The main feature of the simplex method is that at each iteration we change a simplex (the current iterate) by operations called: reflection, expansion, contraction (defined below). These operations change only one of the vectors at a time.

The one changed is usually that for which the objective function $F[\bar{x}]$ is the greatest.

Let $\bar{x}^h$ be the vector corresponding to the maximum value of F on $\underline{X}$; i.e.

$$F[\bar{x}^h] = \max_i F[\bar{x}^i]$$

Let $\bar{x}^\ell$ be the vector corresponding to the minimum value of F on $\underline{X}$, i.e.:

$$F[\bar{x}^\ell] = \min_i F[\bar{x}^i]$$

---

*Within the context of topology a simplex is defined differentely, see Spanier, Algebraic Topology [80]. In the context of optimization theory (constrained and unconstrained) the above definition is sufficient.

Let $\bar{x}^0$ be the centroid of $\{\bar{x}^i \mid i \neq h\}$, i.e.:

$$x^0 = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq h}}^{n+1} \bar{x}^i$$

The three basic operations of the simplex method are defined below, where α, γ, β are arbitrary real numbers.

Reflection: $\bar{x}^h$ is replaced by $\bar{x}^r$ defined by

$$\bar{x}^r = (1+\alpha)\,\bar{x}^0 - \alpha\bar{x}^n \qquad \alpha > 0$$

(This corresponds to reflection through an opposite face*)

Expansion: $\bar{x}^r$ is replaced by $\bar{x}^e$ defined by:

$$\bar{x}^e = \gamma\bar{x}^r + (1-\gamma)\,\bar{x}^0 \qquad \gamma > 1$$

(This corresponds to expanding the simplex in the direction $\bar{x}^r - \bar{x}^0$)

Contraction: $\bar{x}^h$ is replaced by $\bar{x}^c$ defined by:

$$\bar{x}^c = \beta\bar{x}^h + (1-\beta)\,\bar{x}^0 \qquad 0 < \beta < 1$$

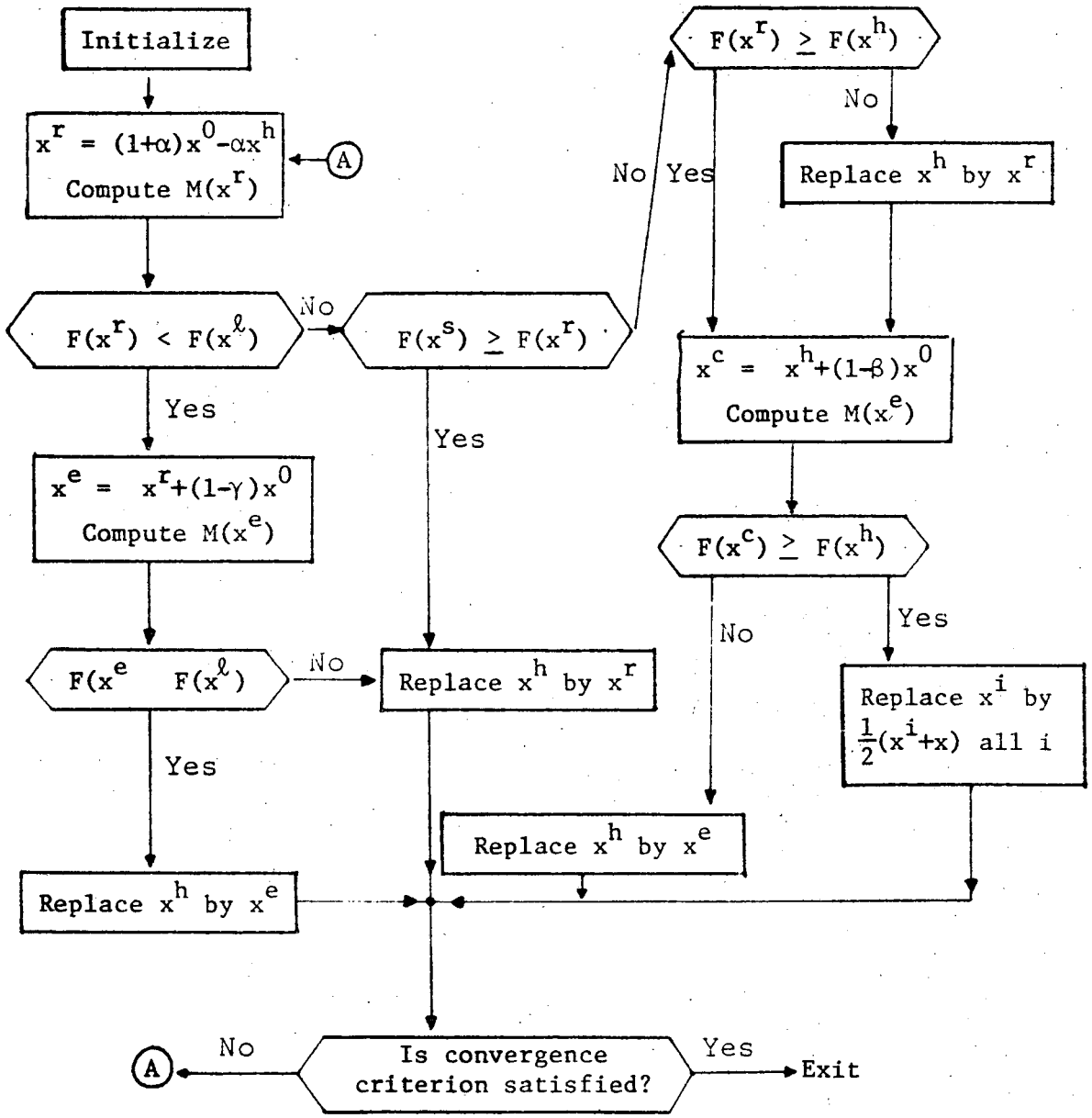
(This corresponds to contracting the simplex by effectively reducing $\|\bar{x}^h - \bar{x}^0\|$).

Melder and Mead found that "good" values for the constants are $\alpha = 1$, $\gamma = 2$, $\beta = .5$.

The stopping criterion is based on the comparison of the standard deviation of F at the $(n+1)$ vectors with a preselected tolerance $\tau > 0$ i.e.

$$\left\{ \sum_{i=1}^{n+1} \frac{[F[\bar{x}^i] - F[\bar{x}^0]]^2}{n} \right\}^{1/2} \leq \tau$$

The following flowchart describes the entire process:



Further development on this method was done by Paviani, Himmelblau [1969].

1.4.5 Random Methods

These methods have the property that random searches do not require derivatives which may be difficult to compute if functional evaluation is a bit noisy.

Random Jumping

Let $F: \mathbb{R}^n \to \mathbb{R}$ be an objective function for which a minimum $\bar{x}^* \in \mathbb{R}^n$ is sought. Let $\bar{x}^1 = (x_1, \ldots, x_n)$ be the initial guess.

Define a hypercube by

$$a_i \leq x_i \leq b_i$$

where $a_i, b_i \in \mathbb{R}$ and are lower and upper bound of x_i, for each i. In what follows, $\bar{s} = (s_1, \ldots, s_n)$ is called a random vector because each component, s_i, is a random number uniformly distributed between 0 and 1.

The basic iteration is as follows:

1) $k = 1$

2) Find a point $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n) \in \mathbb{R}^n$; pick a random vector, s, then

$$x_i = (b_i - a_i)s_i + a_i \qquad i = 1, \ldots, n$$

3) If $F[\hat{x}] \leq F[\bar{x}^k]$ go to Step 4. Otherwise find a new random vector $\hat{S} = (\hat{s}_1, \ldots, \hat{s}_n)$. Go to Step 2.

4) Set $\bar{x}^{k+1} = \hat{x}$ and $\bar{x}^{k+1}$ is the new estimate of $\bar{x}^*$. Go to Step 5.

5) If $\|F[\bar{x}^{k+1}] - F[\bar{x}^k]\| \leq \varepsilon$ where, ε is a predetermined tolerance. Terminate the procedure. Otherwise go to Step 1.

Random Direction Stepping

The iteration step for this method is:

1) $k = 0$, $\bar{x}^0 \in \mathbb{R}^n$ is the initial guess
2) Choose a set $\Lambda = (\lambda^0, \ldots, \lambda^n)$ of steplengths, n any positive integer.
3) Find a vector $s^k = (s_1^k, \ldots, s_n^k)$ whose components are random numbers between 0 and 1.
4) A new point is determined by

$$\bar{x}^{k+1} = \bar{x}^k + \lambda \bar{s}^k \quad \text{for} \quad \lambda \in \{\lambda^0, \ldots, \lambda^k\}$$

where $F[\bar{x}^k + \lambda \bar{s}^k] = \min_{1 \leq l \leq k} F[\bar{x}^k \; \lambda^i \bar{s}^k]$

5) If two consecutive values of F differ by less than a predetermined value τ terminate the procedure. Otherwise go to Step 6.
6) If $k = n$ stop. Otherwise go to Step 3.

1.4.6 Descent Techniques

These techniques use the gradient of $F[\bar{x}]$, which is the vector normal to the local contour surface and is denoted by $\nabla F[\bar{x}]$. Its components are the first order partial derivatives of a differentiable function $F[\bar{x}] \in C^1$; that is

$$\nabla F[\bar{x}] = \left(\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_n}\right)^T$$

where T indicates the transpose.

Since the gradient vector $\nabla F[\bar{x}]$ points in the direction in which the function decreases most rapidly. It seems reasonable to follow the gradient in order to reach a minimum. Gradient methods implement this idea. They require repeated evaluation of partial derivatives of the objective function $F[\bar{x}]$.

Let k denote the k^{th} iteration of a process; $\bar{s}^k \in \mathbb{R}^n$ be a direction in $\mathbb{R}^n$; $\bar{x}^k \in \mathbb{R}^n$ a point in the n-dimensional space.

A unit vector $\frac{\bar{s}^k}{\|\bar{s}^k\|}$ is said to be a <u>descent direction</u> with respect to $F[\bar{x}^k]$ at the point $\bar{x}^k \in \mathbb{R}^n$ if $\exists$ a scalar $\lambda_0 > 0$ $\forall$ scalars λ satisfying $0 \leq \lambda < \lambda_0$ we have

$$F[\bar{x}^{k+1}] = F[\bar{x}^k + \lambda \bar{s}^k] \leq F[\bar{x}^k]$$

if F is differentiable, $\bar{s}^k$ is a descent direciton if

$$\lim_{\alpha \to 0} \frac{F[\bar{x}^k + \alpha \bar{s}^k] - F[\bar{x}^k]}{\alpha} = dF_{\bar{s}^k}(\bar{x}^k)$$

$$= \bar{s}^k \cdot \nabla F[\bar{x}^k] < 0$$

where $\nabla F[\bar{x}^k]$ denotes the gradient of $F[\bar{x}]$ evaluated at the point $(\bar{x}^k)$. Note that by definition the product

$(\bar{s}^k)\cdot\nabla F[\bar{x}^k]$ is nothing other than the directional derivative of $F[\bar{x}]$ in the direction of $\bar{s}^k$ evluated at $\bar{x}^k$. If this directional derivative exists and is negative then $\bar{s}^k$ is a descent direction.

Now we are ready to demonstrate a typical descent iteration:

Let $\bar{x}^k$ be the point obtained from the k-th iteration

a) compute, according to some rule, a descent direction $\bar{s}^k = (s_1^k, \ldots, s_n^k)$

b) Compute, according to some rule, a descent steplength λ^k

c) obtain a new point using

$$\bar{x}^{k+1} = \bar{x}^k + \lambda^k \bar{s}^k$$

A sequence of k-descent steps starting from a point $\bar{x}_0$ to the point $\bar{x}^k$ is given by

$$\bar{x}^k = \bar{x}_0 + \sum_{i=0}^{k-1} \lambda^i s^i = \bar{x}_0 + \sum_{i=0}^{k-1} (\bar{x}^{i+1} - \bar{x}^i)$$

$$= \bar{x}_0 + \sum_{i=0}^{k-1} \Delta\bar{x}^i$$

where $\Delta\bar{x}^i = (\bar{x}^{i+1} - \bar{x}^i)$.

At the k-th iteration defines the matrix Δx_k

$$\Delta x_k = [\Delta\bar{x}^0, \Delta\bar{x}^1, \ldots, \Delta\bar{x}^{k-1}]$$

i.e. the columns of Δx_k are the k-descent steps $\Delta\bar{x}^0$, $\Delta\bar{x}^1$, ..., $\Delta\bar{x}^{k-1}$ preceding $\Delta\bar{x}^k$.

The literature describes numerous descent techniques which differ in the rules for computing s^k and λ^k and which differ in the use of s^k, λ^k and Δx_k. We shall consider some representatives of this class, but before this, we shall discuss some preliminary concepts.

1.4.7 Lagrangian Multipliers

Suppose we want to find the minimum of the objective function $F[\bar{x}]$ constrained by $g[\bar{x}]$ (a constraint is a condition that an objective function must satisfy).

$L(\bar{x},\bar{u}) = F[\bar{x}] - \sum_{j=1}^{m} u_j\, g_j(\bar{x})$ is defined to be Lagrangian Function corresponding to the constrained minimization problem

$$\min_x \{F[\bar{x}] \parallel [g_j(\bar{x}) \geq 0 \quad j = 1, \ldots, m]\}$$

The components $u_1, \ldots, u_n$ of $\bar{u}$ are called the Lagrange Multipliers.

Definition. A pair of points $(\bar{x}^*, \bar{u}^*)$ such that

$$L(\bar{x}^*,\bar{u}) \leq L(\bar{x}^*,\bar{u}^*) \leq L(\bar{x},\bar{u}^*) \quad \forall\ \bar{x},\bar{u} \in \mathbb{R}^n$$

will be called a saddle point of the Lagrangian. In particular if our problem is:

Find the min $F[\bar{x}]$

subject to the equality constraint

$g(\bar{x}) = 0$

i.e. min $\{F[\bar{x}] \parallel [g(\bar{x}) = 0]\}$, then the saddle point $(\bar{x}^*, \bar{u}^*)$ is characterized by:

$$\nabla_x L(\bar{x}^*, \bar{u}^*) = 0 \quad \text{and} \quad \nabla_u L(\bar{x}^*, \bar{u}^*) = 0$$

where ∇_x and ∇_u are the gradients of $\bar{x}$ and $\bar{u}$ respectively. These results will be used in the following section. [1.4.8]

1.4.8 Descent Directions

Definition: The distance between two points in $\mathbb{R}^n$, the n-dimensional space, is defined as:

$$d(\bar{x}_1, \bar{x}_2) = [(\bar{x}_1 - \bar{x}_2)^T A(\bar{x}_1 - \bar{x}_2)]^{1/2}$$

where the superscript T on the vector $(\bar{x}_1 - \bar{x}_2)$ denotes the transpose and A is a positive definite n×n symmetric matrix. We assume A to be positive definite to assure

$$d(\bar{x}_1, \bar{x}_2) > 0 \quad \text{for any} \quad \bar{x}_1 \neq \bar{x}_2 .$$

Once distance, d, is defined on $\mathbb{R}^n$ a metric is established then the role of the matrix A is to introduce a new metric relative to the old coordinate system. Keeping the matrix A fixed during an iterative process will fix the metric. For example, if A is the n-dimensional

identity matrix $I_{n\times n}$ then $d(\bar{x}_1, \bar{x}_2)$ = Euclidean Metric. Changing the matrix A is equivalent to rescaling the variables, and by doing so we generate a new metric relative to the old coordinates.

Clearly, the locus of points at a distance d from a point $\bar{x}^k \in \mathbb{R}^n$ is given by the n-dimensional ellipsoid

$$d^2 = [(\bar{x} - \bar{x}^k)^T A(\bar{x} - \bar{x}^k)]$$

with center $\bar{x}^k$.

Now let us consider a step $\Delta\bar{x}^k$ from $\bar{x}^k$ onto this ellipsoid such that the value of F is the least, in other words:

$$\{\text{MIN } F[\bar{x}^k+\Delta\bar{x}^k] \parallel (\Delta\bar{x}^k)^T A(\Delta\bar{x}^k) = d^2\}$$

F can be expanded in a first order Taylor approximation around $\bar{x}^k$ to get

$$F[\bar{x}^k+\Delta\bar{x}^k] \approx F[\bar{x}^k] + [\Delta\bar{x}^k]^T \nabla F[\bar{x}^k]$$

since $F[\bar{x}^k]$ is constant, we can reduce our problem to finding

$$\text{MIN}\{[\Delta\bar{x}^k]^T \nabla F[\bar{x}^k] \parallel d^2 = [\Delta\bar{x}^k]^T A[\Delta\bar{x}^k]\}$$

The saddle point can be found by the method of Lagrange multipliers:

$$L[\bar{x},\mu] = [\Delta\bar{x}^k]^T \nabla F[\bar{x}^k] - \mu\{[\Delta\bar{x}^k]A[\Delta\bar{x}^k] - d^2\}$$

and at the saddle point

$$(1) \qquad \frac{\partial L}{\partial \bar{x}} = \nabla F[\bar{x}^k] - 2\mu A[\Delta \bar{x}^k] = 0$$

$$(2) \qquad \frac{\partial L}{\partial \mu} = [\Delta \bar{x}^k]^T A[\Delta \bar{x}^k] - d^2 = 0$$

From the first condition

$$\Delta \bar{x}^k = \frac{A^{-1}}{2\mu} \nabla F[\bar{x}^k]$$

which indicates that the step $\Delta \bar{x}^k$ is taken in the direction of $A^{-1}\nabla F[\bar{x}^k]$. The Lagrangian multiplier can be found by solving (1) and (2).

Thus the direction of locally steepest descent is given by

$$(3) \qquad \bar{s}^k = - [A_k]^{-1} \nabla F[\bar{x}^k]$$

The matrix A has been denoted by A_k to indicate that this matrix may change from step to step. Now A_k was taken to be positive definite and symmetric, and if $\nabla F[\bar{x}^k] \neq 0$ then

$$[\bar{s}^k]^T \nabla F[\bar{x}^k] = - [\nabla F[\bar{x}^k]]^T [A_k{}^{-1}]^T \nabla F[\bar{x}^k] < 0$$

Since the inverse of a positive definite matrix is also positive definite, the directions $\bar{s}^k$ as defined by (3) are descent directions.

1.4.9 Gradient Descent Methods

Gradient descent techniques differ in the choice of A_k

The simplest choice of A_k is the $n \times n$ dimensional identity matrix $I_{n\ n}$, that is, $A_k = I \ \forall_k$. For this choice of A_k we have

$$\bar{s}^k = - \nabla F[\bar{x}^k]$$

This clearly is a descent direction since

$$[\bar{x}^k]^T \nabla F[\bar{x}^k] = - [F[\bar{x}^k]]^T \nabla F[\bar{x}^k] < 0$$

This method is termed the steepest descent technique, and its implementation involves only first order differentiation.

For further discussion on some specific algorithms see Forsythe and Motzkim [1951] and Gradient Partan ("Parallel Tangents") Shah, et al [1964].

Second Order Gradient Technique

This technique is very well known as Newton's Method. In this method the matrix A_k is chosen to be the Hessian Matrix (Hess $[\bar{x}^k]$). The Hess $[\bar{x}^k]$ is defined as the matrix whose elements are the second partial derivatives of a twice differentiable function (i.e. $F \in C^2$):

$$\text{Hess}[x] = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ \vdots & & \\ h_{n1} & & h_{nn} \end{bmatrix} \qquad \text{where } h_{ij} = \frac{\partial^2 F[x]}{\partial x_i \partial x_j}$$

Note: The Hessian is the second order term of the Taylor series expansion of F at $\bar{x}^k$.

Now, since F is a C^2 function the Hessian is a symmetric matrix. If in addition the Hessian is positive definite and non singular, then we have:

$$\bar{s}^k = - [\text{Hess}[\bar{x}^k]]^{-1} \nabla F[\bar{x}^k]$$

The direction, $\bar{s}^k$, defines in this way a descent direction since

$$[\bar{s}^k]^T \nabla F[\bar{x}^k] = - [\nabla F[\bar{x}^k]]^T [\text{Hess}[\bar{x}^k]]^{-1} \nabla F[\bar{x}^k] \quad 0$$

The vector $\bar{s}^k$ is called a "second order gradient direction".

Among the vast number of methods using second derivatives we shall mention some of the prominent techniques which are most often quoted in the literature (for reference see [30], [36], [53]). Some examples follow: Greenstadt's Method [1967], Fiacco and MacCormick [1968], Marquardt-Levenberg [1963], [1944], Mathews and Davies [1969] and [1971].

1.4.10 Techniques Using Conjugate Directions

The search directions in second order gradient techniques are generated using the Hessian matrix, but in many cases the derivatives of the objective function are not available in an explicit form and in such cases we would like to generate the search direction without the use of derivatives. To this end we shall consider the conjugate method without calculating derivatives.

Conjugate Direction Techniques

Definition. Given two vectors $\bar{v}$ and $\bar{w}$ we say that $\bar{v}$ and $\bar{w}$ are "conjugate" with respect to some positive

definite symmetric matrix A if:

$$\bar{v}^T A\bar{w} = 0$$

Definition. A set Λ of nonzero vectors which are pairwise conjugate with respect to the same matrix A will be called a set of conjugate directions, i.e. $\Lambda = \{v_1, \ldots, v_n\}$, where $v_i A v_j = 0$ for $i \neq j$. Note that the n-conjugate directions are linearly independent.

Definition. If an objective function is quadratic then the method is said to have quadratic convergence if it finds the exact minimum in a finite number of iterations.

Note. Second order gradient techniques, discussed previously, have quadratic convergence.

The theory of conjugate methods deals primarily with quadratic functions. This particular characteristic is due to the fact that if an objective function $F[\bar{x}]$ is quadratic i.e.

$$F[\bar{x}] = \bar{x}^T A\bar{x} - 2b^T \bar{x} + c \qquad A \in GL(n),\ \bar{x} \in \mathbb{R}^n \qquad [1.0]$$

where A is any positive definite symmetric matrix, b^T is an n-vector and c is a constant.

It is possible to find the exact minimum by using conjugate directions in a finite number of iterations.

The conjugate directions are obtained as follows: pick $\bar{v}$, $\bar{x}_0^*$, $\bar{x}_1^*$ arbitrarily in $\mathbb{R}^n$, find $\bar{x}_i$ satisfying

$$\bar{x}_i \in s_i = \{x \mid x = \bar{x}_i^* + \alpha\bar{v}\} \quad \text{and} \quad F[\bar{x}_i] \leq F[\bar{x}] \quad \forall\, \bar{x} \in s_i$$

then α_i is a steplength satisfying $\bar{x}_i = \bar{x}_i^* + \alpha_i\bar{v}$ for $i = 0,1$ thereby obtaining $\bar{x}_0$, $\bar{x}_1$. Then the direction $\bar{w}$ defined by

$$\bar{w} = (\bar{x}_1 - \bar{x}_0)$$

is conjugate to $\bar{v}$, i.e. $\bar{v}\, A\, \bar{w} = 0$. Since:

Theorem: If the minimum of $F[\bar{x}]$ in the subspace

$$s_i = \{\bar{x} \in \mathbb{R}^n \mid \bar{x} = \bar{x}_i^* + \alpha\bar{v}_2 \,,\ \alpha \in \mathbb{R}\}$$

is at $\bar{x}_i$, for $i = 0,1$ then $(\bar{x}_1 - \bar{x}_0)$ is conjugate to $\bar{v}$, with respect to A.

Proof. For $i = 0,1$ and the definition of $\bar{x}_i$ we obtain

$$\frac{\partial F}{\partial \lambda}\,[\bar{x}_i + \lambda\bar{v}] = 0$$

Taking the partial derivatives with respect to λ and substituing $x_i + \lambda\bar{v}$ for $\bar{x}$ in Equation [1.0] we obtain

$$\frac{\partial F}{\partial \lambda}\,[\bar{x}_i + \lambda\bar{v}] = \frac{\partial}{\partial \lambda}\,[(\bar{x}_i + \lambda\bar{v})^T A(\bar{x}_i + \lambda\bar{v}) - 2b^T(\bar{x}_i + \lambda\bar{v}) + c]$$

The right expression implies

$$\bar{v}^T(A\bar{x}_i - b) = 0$$

and considering this expression for $i = 0$ and 1 we get $\bar{v}^T(A\bar{x}_0 - b) = 0$ and $\bar{v}^T(A\bar{x}_1 - b)$ respectively.

Subtracting the first expression from the second we obtain the desired result, i.e.

$$\bar{v}^T A(\bar{x}_1 - \bar{x}_0) = 0$$

This shows (by definition) that $\bar{v}$ is conjugate to $\bar{w} = (\bar{x}_1 - \bar{x}_0)$. Q.E.D.

In the next section we discuss an example.

1.4.11 Powell's Algorithm Without Using Derivatives

Description of the basic procedure of the algorithm.

This method starts with an initial guess $\bar{x}^1$ to the minimum, initially the set of conjugate directions $\bar{v}^1, \ldots, \bar{v}^n$ are chosen to be the columns of the identity matrix $I_{n\times n}$. Let $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ be a set of given steplengths. The basic iteration of the method is as follows:

1) initialize $k = 1$
2) evaluate $F[\bar{x}^k]$, save this value
3) solve the problem of minimzations along a line; i.e., find λ_k that minimize $F[x^k + \lambda_k \bar{v}^k]$; where λ_k is the steplength in the k-th iteration, $\bar{v}^k$ is the k^{th} search direction and $\bar{x}^k$ is the current iterate.
4) with the λ_k found in Step 3 perform the current descent step, i.e., $\bar{x}^{k+1} = \bar{x}^k + \lambda_k \bar{v}^k$
5) if $k < n$, set $k = k+1$ and repeat from Step 2. Otherwise continue to Step 6.

6) if $|x_i^n - x_i| < \varepsilon_i$, $i = 1, \ldots, n$ where ε_i are predetermined minimum steplength values, terminate the iteration. Otherwise continue to Step 7.

7) find the integer j , $1 \leq j \leq n$ such that

$$\phi = \max_{1 \leq j \leq n} (F[\bar{x}^{j-1}] - F[\bar{x}^j])$$

8) set $F_1 = F[\bar{x}^1]$; $F_2 = F[\bar{x}^n]$ and obtain F_3 by

$$F_3 = F[2\bar{x}^n - x^1]$$

9) if either

$$F_3 \geq F_1 \quad \text{or} \quad (F_1-2F_2+F_3)(F_1-F_2-\phi)^2 \geq \tfrac{1}{2}\phi(F_1-F_3)^2$$

The present n-directions $\bar{v}^1, \ldots, \bar{v}^n$ will be retained and used in the next iteration. Set $\bar{x}^1 = \bar{x}^n$, $k = 1$ repeat from Step 2. Otherwise continue to Step 10.

10) set $\hat{v} = (\bar{x}^n - \bar{x}^1)$, $\hat{v} = (\hat{v}_1, \ldots, \hat{v}_n)$ and find $\hat{\lambda}_s$ such that $F[\bar{x}^n + \hat{\lambda}_s \hat{v}_s]$ is a minimum with this value of $\hat{\lambda}_s$ set

$$\bar{x}^1 = x^n + \hat{\lambda}_s \hat{v}_s$$

11) determine the new set of conjugate directions, i.e.

$$\{\bar{v}^1, \ldots, v^{j-1}, v^{j+1}, \ldots, v^{n-1}, \hat{v}\}$$

The current direction v^j for j found in Step 7 is discarded, and a new direction, $\hat{v}$ obtained in Step 10 is added set $k = 1$ and repeat from Step 2.

The above method was first implemented by Powell [1964] It turned out that this procedure generates linearly dependent directions, which causes difficulties in the implementation of the algorithm. Powell [1944], Zangwill [1967] and Brent [1973] modified the algorithm to overcome this problem.

1.4.12 Conjugate Gradient Techniques

The conjugate gradient techniques result from combining the conjugate method with first order gradient methods.

These techniques use a sequence of descent steps rather than individual steps. The gradient of $F[\bar{x}]$, $\nabla F[\bar{x}]$, is used to generate conjugate directions. The method is designed for quadratic objective functions or for algebraic functions that can be approximated by a quadratic function. Thus the method will generate mutually conjugate directions with respect to any positive definite matrix A, corresponding to the quadratic function

$$F[\bar{x}] = a + \bar{b}^T\bar{x} + \frac{1}{2}\bar{x}^TA\bar{x}$$

where a is a constant $\bar{x},\bar{b} \in \mathbb{R}^n$, $A \in GL(n)$. The minimum sought $\bar{x}^*$ will have to satisfy

$$\nabla F[\bar{x}] = \bar{b} + A\bar{x}^* = 0$$

Hence the problem of finding the unique solution $\bar{x}^*$ of $Ax + b = 0$ is equivalent to finding the

$$\min \left[a+\bar{b}^T\bar{x} + \frac{1}{2}\bar{x}^T A\bar{x}\right] \qquad [1.1]$$

It turned out that finding $\bar{x}^*$ by solving the linear system is much more computational demanding than to minimize [1.1], since [1.1] is only a local approximation of F and a and b as well as A vary with $\bar{x}$.

We shall examine now the process of solving the minimization problem when $F[\bar{x}]$ is quadratic, using conjugate and gradient directions. There exist several methods using this general approach: first a method developed by Fletcher and Reeves [1964], second, the "supermemory gradient method", Miele and Cantrell [1969], Gragg and Levy [1969] and third, the "projected gradient method", Myers [1968], Pearson [1969], Sorenson [1969].

Since the method of Fletcher and Reeves [1964] has been tested intensively we will outline the basic iteration of the technique.

1.4.13 Fletcher and Reeves Algorithm [1964]

The descent direction of the method:

Let $\bar{s}^0, \bar{s}^1, \ldots, \bar{s}^{k-1}, \bar{s}^k$ be conjugate directions defined by the recurrence formula:

$$\bar{s}^o = - \nabla F[\bar{x}]$$

$$\bar{s}^k = - \nabla F[\bar{x}] + \sum_{i=1}^{k} \beta^i s^{i-1} \quad ,$$

where β^i are chosen to be scalars such that $\bar{s}^k$ is conjugate to all previously used descent directions,

$\bar{s}^0, \ldots, \bar{s}^{k-1}$. The formula for β^k is:

$$\beta^k = \frac{[\nabla F[\bar{x}^k]]^T \nabla F[\bar{x}^k]}{[\nabla F[x^{k-1}]]^T \nabla F[\bar{x}^{k-1}]} \quad . \qquad [1.2]$$

The iteration of the method is:

Let $\Lambda^k = \{\lambda_1^k, \ldots, \lambda_n^k\}$ be a set of given steplengths,

1) given $\bar{x}^0$, evaluate $F[\bar{x}^0]$, set $k = 0$
2) find the gradient at $\bar{x}^k$, i.e. $\nabla F[\bar{x}^k]$
3) if $\|\nabla F[\bar{x}^k]\|$ less than the predetermined tolerance. Terminate the iteration. Otherwise continue to Step 4.
4) find the descent direction for the k^{th} iteration, i.e.

$$\bar{s}^k = - \nabla F[\bar{x}^k] + \beta^k s^{k-1}$$

β^k is found from formula [1.2]
normalize $\bar{s}^k$, i.e. $\|\bar{s}^k\| = 1$

5) find λ^k that minimize $F[\bar{x}^k + \lambda \bar{s}^k]$
6) use $\lambda^k \bar{s}^k$ found in Step 5 to perform a descent steep i.e.

$$\bar{x}^{k+1} = \bar{x}^k + \lambda^k \bar{s}^k$$

7) evaluate $F[\bar{x}^{k+1}]$
8) if $F[\bar{x}^{k+1}] - F[\bar{x}^k] < \varepsilon$; ε a predetermined improvement value and $\|\lambda^k \bar{s}^k\| < \tau$; τ a predetermined steplength. Terminate the iteration. Otherwise go to Step 9.
9) Accept the point $\bar{x}^{k+1}$ and if $k < n$ set $k = k+1$; otherwise set $\bar{x}^0 = \bar{x}^{k+1}$; $k = 0$; in both cases repeat from Step 2.

1.4.14 Variable Metric Techniques

These methods involve finding conjugate directions that under certain conditions approach second order gradient directions (see page 23). They are referred to in the literature as quasi-second order or quasi-Newton techniques. An important contribution in the area of descent techniques was made by Davidon [1959] and extended by Fletcher and Powell [1963]. Good references dealing with the theoretical and practical aspects of these techniques have been published by Adachi [1971], Huang [1970], Huang and Levy [1970], Pearson [1969].

As we mentioned earlier, the gradient techniques differ basically in the choice of the matrix A_k. The variable metric techniques start with an arbitrary positive definite matrix; in most cases $A_0 = I$, the n×n dimensional identity matrix and in each of the following steps $A_k^{-1} = A_{k+1}^T$ of the matrix is updated.

The basic iteration procedure is:

a) given a point $\bar{x}^0$ and a positive definite matrix. A_0 (in most cases $A_0 = I$), set $k = 0$ and compute $\nabla F[\bar{x}^0]$.

b) Obtain a new point $\bar{x}^{k+1}$ from the point $\bar{x}^k$ obtained from the k-th iteration according to

$$\bar{x}^{k+1} = \bar{x}^k - \lambda^k A_K^T \nabla F[\bar{x}^k]$$

where we have to find λ^k that minimizes

$F[\bar{x}^k - \lambda A_K^T \nabla F[\bar{x}^k]]$.

c) Compute the gradient at $\bar{x}^{k+1}$, i.e. $\nabla F[\bar{x}^{k+1}]$.

d) Update inverse of the matrix to obtain A_{k+1} . The various methods in this class differ in the manner in which they update A_k .

e) Set $k = k+1$ repeat from Step b.

The descent direction $\bar{s}^k$ used in the variable metric techniques are computed by

$$\bar{s}^k = - A_k^T \nabla F[\bar{x}^k]$$

As we mentioned earlier the method by Davidon Fletcher and Powell is of major importance hence we will outline the main iteration step of their technique.

1.4.15 Davidon-Fletcher-Powell Variable Metric Technique

The basic iteration of the method consists of:

a) Obtain the value of the objective function $F[\bar{x}^0]$ at a given point $\bar{x}^0$; set $k = 0$

b) Compute the gradient at the point $\bar{x}^k$, i.e. $\nabla F[\bar{x}^k]$

c) Compute the matrix H_k , (where H_k is the inverse of the matrix A_k)

c_1) For the initial step $H_0 = I$, the n n identity matrix. Go to Step d. For $k > 0$ go to c_2 .

c_2) The updating of H_k is obtained using

$$H_{k+1} = H_k + \bar{Y}_{k+1}$$

where

$$\gamma_{k+1} = \frac{\Delta\bar{x}^k\,(\Delta\bar{x}^k)^T}{(\Delta\bar{x}^k)^T\bar{y}^k} - \frac{(H_k\bar{y}^k)(H_k\bar{y}^k)^T}{(\bar{y}^k)^T\,H_k\bar{y}^k}$$

and where

$$\Delta\,\bar{x}^k = \bar{x}^{k+1} - \bar{x}^k \quad \text{and} \quad \bar{y}^k = \nabla\,F[\bar{x}^{k+1}] - F[\bar{x}^k]$$

d) Compute the descent direction $\bar{s}^k$ according to

$$\bar{s}^k = -\,H_k^T\,\nabla\,F[\bar{x}^k]$$

and normalize $\bar{s}^k$, i.e. $\|\bar{s}^k\| = 1$

e) Calculate the normalized derivative of the objective function $F[\bar{x}^k]$ in the descent direction

$$\tau^k = \frac{|(\bar{s}^k)^T\,\nabla F[\bar{x}^k]|}{\|\nabla\,F[\bar{x}^k]\|}$$

f) If $\tau^k \leq \varepsilon_1$, $\|\nabla\,F[\bar{x}^k]\| < \varepsilon_2$ terminate iteration (ε_1, ε_2 are predetermined tolerances). Otherwise go to g.

g) If $(\bar{s}^k)^T\,\nabla\,F[\bar{x}^k] \geq 0$ set $\bar{s}^k = -\,\bar{s}^k$ and reset H_k to $H_k = I_{n\times n}$

h) Solve the problem of minimization along the line $\bar{x}^k + \lambda\bar{s}^k$ i.e. find λ^k that minimizes $F[\bar{x}^k + \lambda\bar{s}^k]$

i) Define $\Delta\bar{x}^k = \bar{\lambda}^k\bar{s}^k$

j) Obtain a new point according to

$$\bar{x}^{k+1} = \bar{x}^k + \Delta\bar{x}^k$$

k) Evaluate the objective function at the point $\bar{x}^{k+1}$ i.e. $F[\bar{x}^{k+1}]$.

l) If $F[\bar{x}^{k}] - F[\bar{x}^{k+1}] < \theta_1$ and $\|\Delta\bar{x}^{k}\| < \theta_2$ terminate the iteration. (θ_1 is predetermined minimal improvement of the function and θ_2 is a predetermined steplength). Otherwise go to m.

m) Accept the point $\bar{x}^{k+1}$ and if $k < n$ set $k = k+1$; otherwise set $k = 0$, $\bar{x}^{0} = \bar{x}^{k+1}$. Repeat from Step b.

There exist variants of the variables metric methods and some of those have been proposed by Broyden [1967], [1970], Huang [1970], Pearson [1969], Greenstadt [1970a,b], Goldfarb [1970], and Murtach and Sargeant [1970].

In general the variable metric techniques perform better for general non-quadratic functions than many other quadratically convergent methods.

These methods have the advantage of fast convergence near the minimum.

1.4.16 Other Algorithms Considered

A) Two prominent algorithms have recently been the subject of a paper by E. Polack [70]. In that paper a comparison of his method, a gradient secant method, with the Brents-Shananskii discrete Newton algorith is made. Two fifty dimensional problems are discussed. It is of interest to compare their results with the performance of Bremermann's optimizer on problems having many variables. To this end we are introducing these two algorithms. The

full descriptions of these methods can be found via the aforementioned paper.

B) Since root-finding for algebraic objective functions is a special case of minimization we would like to mention a new technique developed by S. Smale. He has developed "A Global Newton Ralphson" method for finding a zero of a system of non-linear algebraic equations. The algorithm is still in its early developments and it is a very promising method on the basis of its performance on some test problems. The algorithm views a system of non-linear algebraic equations as a system of non-linear ordinary differential equations, and using concepts of global analysis it is able to follow the trajectories which will lead to a zero of the system.

1.5 BREMERMANN'S ALGORITHM (THE "OPTIMIZER") [1970]

This method was developed by Bremermann and grew out of simulation of biological evolution as a search and optimization process.

Bremermann observed that the computational cost of elaborate choices of search or descent directions often exceeds the benefits derived from it. He observed that by searching along randomly chosen directions (rather than along computed directions) the overall speed of convergence (in computer time) is faster than when he searched along gradients. In the following material we will investigate

this phenomena, not only in comparison with gradients but with respect to a representative sample of all the methods described so far.

This method finds the global maximum or minimum of an objective function with a polynomial of degree four or less of many variables. The method is iterative and theoretically guaranteed to converge for polynomials of several variables up to the fourth degree. A detailed theoretical analysis of the optimizer's convergence properties, and other theoretical considerations can be found in [11].

Description of the Method

1) F is evaluated for the initial estimate $\bar{x}^{(0)}$.
2) A random direction R is chosen. The probability distribution of the R is an n-dimensional Gaussian with $\sigma_1 = \sigma_2 = \ldots = \sigma_n = 1$. σ_i is the standard deviation of the i^{th} coordinate.
3) On the line determined by $\bar{x}^{(0)}$ and R the restriction of F to this line is approximated by five-point Lagrangian interpolation, centered at $\bar{x}^{(0)}$ and equidistant with distance H , the preset steplength parameter.
4) The Lagrangian interpolation of the restriction of F is a fourth-degree polynomial in a parameter λ describing the line $\bar{x}^0 + \lambda\bar{R}$. (It describes F exactly, up to round-off errors, if F is a fourth-order function.) The five coefficients of the Lagrangian

interpolation polynomial are determined.

5) The derivative of the interpolation polynomial is a third-degree polynomial. It has one or three real roots. The roots are computed by Cardan's formula.

6) If there is one root λ_0, the procedure is iterated from the point $\bar{x}^{(0)} + \lambda_0 \bar{R}$ with a new random direction provided that $F(\bar{x}^{(0)} + \lambda_0 R) \leq F(\bar{x}^{(0)})$. If the latter inequality does not hold, then the method is iterated from $\bar{x}^0$ with a new random direction.

7) When there are three real roots λ_1, λ_2, λ_3, then the polynomial (or F) is evaluated at $\bar{x}^{(0)} + \lambda_1 \bar{R}, \bar{x}^{(0)} + \lambda_2 \bar{R}$, and $\bar{x}^{(0)} + \lambda_3 R$. Also considering the value $F(\bar{x}^{(0)})$, the procedure is iterated from the point where F has the smallest value (if F has a minimum value at more than one point, then the procedure chooses one of them).

8) When a predetermined number of iterations has been run, the method is stopped and the value of F and the value of $\bar{x}$ are printed.

A FORTRAN program implementing the procedure is listed in the Appendix.

Features of the optimizer

1) Preparation of a problem for use with Bremermann's optimizer is easy. It consists of:

   a) the optimizer requires a subroutine that evaluates F at any desired point.

b) Very few changes have to be made to optimize different functions (i.e., number of variables, number of iterations, a steplength parameter) besides providing a routine that computes the objective function $F[\bar{x}]$.

2) It does not require close initial estimates for convergence to the global minimum.

3) It does not require the gradient or Hessian of an objective function. Hence the optimizer can be applied with a minimum of effort.

1.6 TEST PROBLEMS

When developing an algorithm we have to be concerned with the theoretical as well as with the practical properties of the method. The best way to verify how well it performed is to actually try to solve specific problems.

In the field of optimization some functions having pathological properties were formualted with the intention of determining how well an algorithm is able to overcome various difficulties. Examples of difficulties include: local minima, number of variables, slow speed of convergence, accuracy of the result obtained, singular Jacobian or Hessian and ill conditioned problems.

Historically these functions carry the name of their originators or the name of the particular difficulty inherent

in the problem. (e.g. Rosenbrook 1960, Singular Powell 1964). The purpose of formulating these functions is to test how robust is an algorithm in a wide range of different problems. To this end we have compiled eleven known test problems which, in the literature, are considered difficult to optimize.

The test problems in this Chapter were obtained from a) D. M. Himmelblau's paper "A uniform evaluation of unconstrained optimization techniques"; b) Richard Brent's book, "Algorithm for minimization without derivatives"; and c) E. Polak's paper, "A modified secant method for unconstrained minimization".

The Test Problems are:

1. Rosenbrook 1960: $F[\bar{x}] = 100(x_2-x_1^2)^2 + (1-x_1)^2$

   Descent methods tend to fall into a parabolic valley before reaching the true minimum at (1,1).

2. Beale 1956: $F[\bar{x}] = \sum_{i=1}^{3} [c_i - x_1(1-x_2^i)]^2$

   where $c_1 = 1.5$, $c_2 = 2.25$, $c_3 = 2.626$. The global minima is $F[\bar{x}] = 0$ at $\bar{x} = (3, 1/2)$.

3. Engwall 1966: $F[\bar{x}] = x_1^4 + x_2^4 + 2x_1^2x_2^2 - 4x_1 + 3$

   The global minima is $F[\bar{x}] = 0$ at $\bar{x} = (1,0)$.

4. Zangwill 1967: $F[\bar{x}] = (1/15)\,[16x_1^2 + 16x_2^2 - 8x_1x_2 - 56x_1 - 256x_i + 991]$

   The global minima is $F[\bar{x}] = -18.2$ at the point $\bar{x} = (4,9)$.

5. Zangwill 1967: $F[x] = (x_1-x_2+x_3)^2 + (-x_1+x_2+x_3)^2 + (x_1+x_2-x_3)^2$

   The global minima is $F[\bar{x}] = 0$ at $\bar{x} = (0,0,0)$.

6. Engvall 1966: $F[x] = \sum_{i=1}^{5} f_i^2(\bar{x})$ where

$$f_1(\bar{x}) = x_1^2 + x_2^2 + x_3^2 - 1$$

$$f_2(\bar{x}) = x_1^2 + x_2^2 + (x_3-2)^2 - 1$$

$$f_3(\bar{x}) = x_1 + x_2 + x_3 - 1$$

$$f_4(\bar{x}) = x_1 + x_2 - x_3 + 1$$

$$f_5(\bar{x}) = x_1^3 + 3x_2^2 + (5x_3-x_1+1)^2 - 36$$

Global minima is $F[\bar{x}] = 0$ at $\bar{x} = (0,0,1)$.

7. Fletcher and Powell 1964:

$$F[\bar{x}] = \frac{1}{1+(x_1-x_2)^2} + \sin\left(\frac{\pi x_2+x_3}{2}\right) + \exp\left[\left(\frac{x_1+x_2}{x_2} - 2\right)^2\right]$$

x global minima is $F[\bar{x}] = 3$ at $\bar{x} = (0.78547, 0.78547, 0.78547)$.

8. Wood: $100(x_2-x_1)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2 + (1-x_3)^3 +$

$$+ 10(x_2-1)^2 + (x_4-1)^2 + 19.8(x_2-1)(x_4-1) .$$

This function has a local minima that may interfere in finding the global one. Global minima is $F[\bar{x}] = 0$ at $\bar{x} = (1,1,1,1)$.

9. Singular (Powell 1962):

$$F(\bar{x}) = (x_1+10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4$$

In this function most of the known algorithms failed to pass the stopping criterion since the Hessian at the minimum value $\bar{0}$ is doubly singular. Global minimum is $F[\bar{x}] = 0$ $x = (0,0,0,0)$.

10. Rosenbrook: 50 dimensional "banana"

$$F(x) = \sum_{i=1}^{19} \left[x_i + \left(\frac{i-10.4}{2} \right)^{(-1)^i} x_{i+1} \right]^2 +$$

$$+ 2 \sum_{i=20}^{24} (x_i - x_{i+1})^2 +$$

$$+ \sum_{i=25}^{29} (x_i - 0.5x_{i+1})^2 +$$

$$+ 3 \sum_{i=30}^{49} \left[x_i - \left(\frac{i-40.5}{3} \right)^{(-1)^i} x_{i+1} \right]^2 +$$

$$+ 2 \sum_{i=1}^{20} (x_i - x_{51-i})^4 + \sum_{i=21}^{25} (x_i - x_{51-i})^4$$

Global minima is $F[\bar{x}] = 0$ at $\bar{x} = 0$

11. "Bowl" Type. 50 dimensional

$$F(\bar{x}) = \left(1 - e^{-\|x\|^2 / 100} \right)$$

Global minima is $F[\bar{x}] = 0$ at $\bar{x} = 0$.

1.6.1 Algorithms Used in the Comparison with the Optimizer

For the purpose of comparing the optimizer performance on the test problem, having up to 4-variables we have chosen 15 prominent algorithms. Our basis for the comparison will be the results obtained by D. M. Himmelblau in his paper, "A uniform evaluation of unconstrained optimization techniques"

Detailed descriptions of each technique are not included and the reader is referred to proper references at the end of this chapter.

The 15 algorithms will be classified into two categories:

a) Algorithms using analytical derivatives

b) Derivative free algorithms

Algorithms using analytical derivatives

The algorithm used were:

1) DPF Davidon-Fletcher-Powell Rank 2 (Fletcher, Powell 1963)
2) B Broyden (Rank 1) 1965
3) P2 Pearson No. 2 (1969) without reset
4) P3 Pearson No. 3 (1969) without reset
5) PN Projected Newton (Pearson) 1969
6) FR Fletcher Reeves (1964) reset each n+1 iteration
7) CP Continued Partan (Shah et. a. 1964)
8) IP Iterated Partan (Shah et. al. 1964)
9) GP Goldstein-Price 1967
10) F Fletcher 1970

Derivatives Free Algorithms

11) HJ Hooke-Jeeves (1961)

12) NM Nelder-Mead (1965)

13) P Powell (1964)

14) R Rosenbrook (1960)

15) S Stewart (DPF with numerical derivatives) 1967

All algorithms in the comparison performed by Himmelblau w were tested in a CDC 6600 computer. Bremermann's optimizer performance on the test problems was also tested on a CDC 6600 computer, though in a different facility.

In using the optimizer on large dimensional problems, we chose to compare it with the results obtained by E. Polak in his paper "A modified secant method for unconstrained minimization" [1973]. In this paper a new gradient-secant method is presented, and its performance on two 50-dimensional problems is compared with Brent-Shamanskii's discrete Newton Algorithm

E. Polak shows that his algorithm is superior to most of the various conjugate gradient methods described in the literature; he also has a heuristic argument to justify his method's superiority over the variable metric techniques. Morever, the paper compares the Brent-Shamanskii algorithm with his method and he concludes that the new gradient-secant method will emerge "as one of the more efficient methods for the solution of certain classes of unconstrained optimization problems". A full theoretical discussion of both methods is given in

The methods compared by E. Polak were tested on a CDC 6400 in the computer facility of U.C. Berkeley.

Our comparison with Polak's results were obtained using the same computer and facilities.

1.6.2 Discussion

In order to compare Bremermann's optimizer with other algorithms, it is necessary to determine a criterion by which all the algorithms can be fairly compared.

In the literature [58], [59], [71], the most common points for comparison are:

1) The number of function evaluations required to obtain the minimum within a predetermined accuracy
2) How robust is the method (correct solution on a wide number of test problems).
3) Number of iterations.
4) Total computational time required to obtain the desired optimum value.

We shall elaborate on these points.

1) Function Evaluations. This criterion has different meanings for different authors. Some consider the evaluation of a Jacobian or Hessian as one function evaluation, while in fact a $n \times 1$ Jacobian requires n function evaluations and a $n \times 1$ Hessian requires n^2 function evaluation. Furthermore, it is possible to reduce the number of functions evaluation by a different time-consuming test, such as: matrix operations, heuristic operations, numerical derivatives, etc. Therefore

we should be very cautious when considering function evaluation as the sole criterion for a comparison.

2) To determine how robust an algorithm is, it is necessary to test it in a wide range of problems. Each of these problems exhibits a particular difficulty which an algorithm must overcome. The algorithms used for the comparison including the optimizer, were applied to eleven test problems, which are considered as "classics" in the literature. It is important to emphasize that even though an algorithm might solve all eleven of the test problems, it is very possible that there are pathological objective functions for that algorithm.

3) The criterion for comparison based on number of iterations is very misleading since it means search directions in variable metric techniques and conjugate methods, but has a different meaning in the others. In particular an iteration step in methods using conjugate directions will have a substep of minimizing along a line $x^k + \lambda \bar{s}^k$, i.e. find λ^k such that minimize $F[\bar{x}^k + \lambda \bar{s}^k]$, while other methods, like Bremermann's optimizer, consider such a line minimization to be a single iteration of the procedure, therefore, when we interprete results in the literature we should always understand what is meant by the"number of iterations" required to obtain the global minimum in any comparison between algorithms.

4) By the total computational time: we mean the actual time required to run an algorithm on a specified computer. This time will include the time for: function evaluations,

derivatives evaluations, matrix inversions, and Heuristic operations, etc. It is of interest to note that in 1968, A. R. Colville after comparing thirty different algorithms on eight optimization problems found that totoal computational time is a more valid performance index than the number of function evaluations alone.

1.6.3 Effects of the Computer on the Results

When comparing distinct algorithms, we should be aware that the numerical results were possibly obtained on different computers having different machine precision. To determine the effect on the results, due to such computer differences we tested Bremermann's optimizer on a CDC 6400, CDC 6600 and CDD 7600 computers. All of these have the same precision. The results of this comparison are recorded in Table [1.0].

Central processing time is used for comparing the performance of various algorithms on different computers. The purpose of this comparison is to show that the time i.e. central processing time, can be varied over a large range simply by varying the computer; in later sections, we use central processing time on the same computer as a tool for comparison of optimization algorithms.

1.6.4 Test Procedure

a) Dimensionality of the Problems

In the literature of optimization comparison among algorithms has concentrated primarilly on test functions of

Table 1.0

Comparison of Central Processing Time (in seconds) when Bremermann's Optimizer is Used on a Variety of Problems with Various Computers

COMPUTER	PROBLEM									
	50 Dimensional "Banana" Type	50 Dimensional "Bowl" Type	2 DIM Zangwill (1967)	Rosenbrook (1960)	Beale (1958)	2 DIM Engwall (1966)	Zangwill (1967)	Engwall (1966)	Singular (1962)	Fletcher & Powell (1963)
CDC 6400 Computer	15.345	9.809	0.032	0.295	0.196	0.167	0.136	0.923	0.056	0.041
CDC 6600 Computer	4.110	2.378	0.008	0.076	0.039	0.038	0.023	0.180	0.023	0.013
CDC 7600 Computer	0.688	0.432								

The termination criteria are given in page 50.

The number of iterations for a particular problem did not vary from computer to computer.

a few variables, usually less than five. When a particular algorithm is applied to problems of a few variables we elude the difficulty of dimensionality. For example, in methods where second derivatives are required, the process for obtaining analytical derivatives may be humanly impossible, (for $n = 100$ the Hessian has 10.000 terms) while numerical differentiation may introduce errors. Furthermore, the inversion of a Hessian matrix can be time consuming and inaccurate, in particular, if the process is iterative and at each iteration an inversion of $\text{Hess}[\bar{x}]$ is performed.

To take into account the problem of dimensionality we shall consider the following two classes of optimization problems:

a) Problems with up to four variables

b) 50 dimensional problems

b) Termination Criteria

Each class of problems will have the same termination criteria. For functions of up to four variables termination will occur when both of the following conditions are satisfied:

if $\bar{x}^*$ is the true minimum then

$$1) \quad |F[\bar{x}^*] - F[\bar{x}^k]| \leq 10^{-5}$$

and

$$2) \quad |\bar{x}^* - \bar{x}^k| \leq 10^{-5}$$

where k denotes the k-th iteration.

These conditions were set by Himmelblau when comparing its leading algorithms on test problems with up to four variations. For functions of 50 variables, termination will occur when:

$$1) \quad F[\bar{x}^k] \leq 10^{-9}$$

and

$$2) \quad \bar{x}^k \leq 10^{-6}$$

These conditions were set by Polak when he compared his method, "a new secant method", with Brent-Shamanskii algorithm.

1.6.5 Results Obtained with Bremermann's Optimizer

The results on functions of 50-variables were obtained using a CDC-6400 computer in the Computer Center of the University of California at Berkeley.

The results are recorded in the following tables.

Table 1.1

RESULTS OBTAINED ON TWO 50 DIMENSIONAL TEST PROBLEMS
USING BREMERMANN'S OPTIMIZER
ON THE CDC 6400 COMPUTER

ROSENBROOK 50-DIMENSIONAL BANANA TYPE

INITIAL FUNCTION VALUE = $F[\bar{x}_0]$ = 1019,004			
FINAL FUNCTION VALUE OBTAINED = $F[\bar{x}]$ = 65,736 E-20 *			
GLOBAL MAXIMUM SOUGHT = $F[\bar{x}]$ = 0			
Iteration #	$F[\bar{x}]$	Iteration #	$F[\bar{x}]$
20	49,400	180	2,3092 E-8
40	2,492	200	9,611 E-10
60	1,690 E-1	220	1,654 E-11
80	2,575 E-2	240	1,526 E-12
100	3,566 E-3	260	4,500 E-15
120	4,235 E-4	280	3,175 E-18
140	4,917 E-5	300	6,573 E-20
160	1,641 E-6		

*In this and the following tables E-n, n any integer, will denote 10^{-n} for example 65,736×-20 = $65,736 \times 10^{-20}$

Table 1.1 (continued)

50-DIMENSIONAL BOWL TYPE

INITIAL FUNCTION = $F[\bar{x}_0]$ = 0.116312			
FINAL FUNCTION VALUES OBTAINED = $F[\bar{x}]$ = 1.3509 E-10			
GLOBAL MINIMUM SOUGHT = $F[\bar{x}]$ = 0			
Iteration #	$F[\bar{x}]$	Iteration #	$F[\bar{x}]$
20	1,2851 E-3	120	1,4866 E-9
40	7,0999 E-5	140	1,4866 E-9
60	2,4575 E-6	160	1,3509 E-10
80	1,0753 E-7	180	1,3509 E-10
100	3,0119 E-9	200	1,3509 E-10

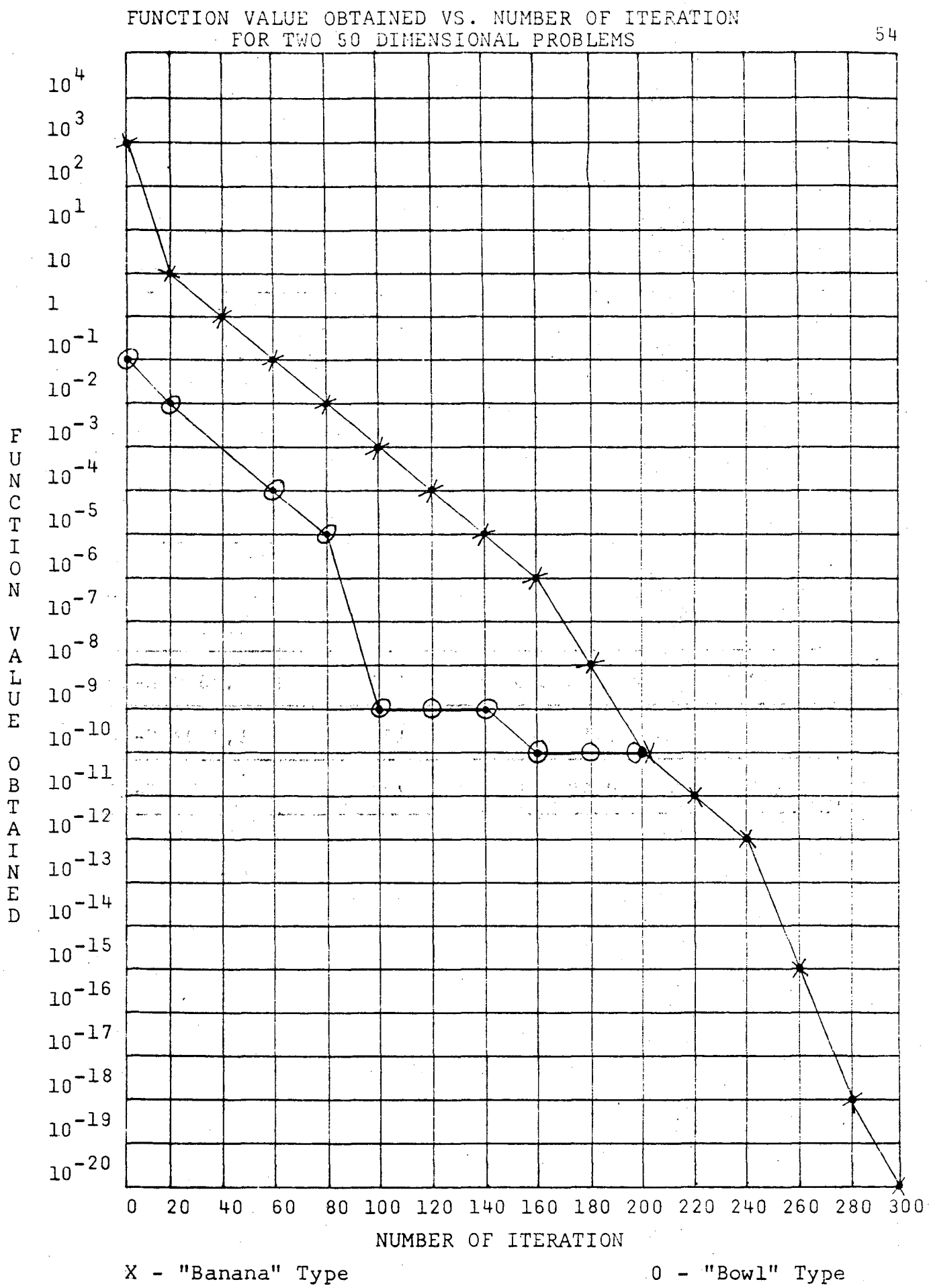


Table 1.2 Graphic Representation of Table 1.1

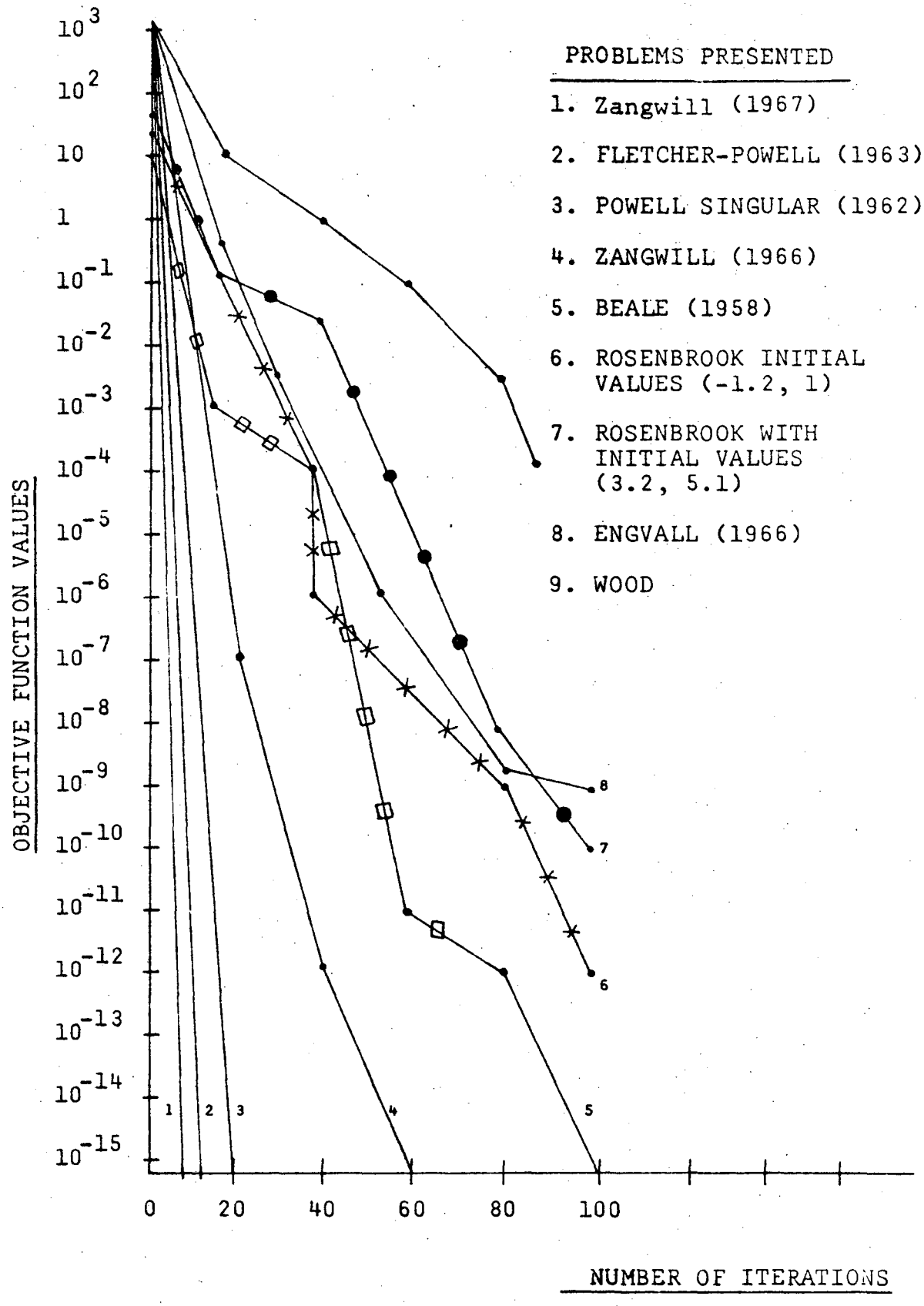


Table 1.2: Graphic Representation of Table 1.1.

Results obtained using Bremermann's optimizer on several test problems.

Table 1.3

Rosenbrook (1960)

Initial Values	$\bar{x}$ = (-1.2,1)
Final Values Obtained	x_1 = 1.0000000240 x_2 = 1.0000004770
True Values	x_1 = 1 x_2 = 1

Iteration No.	Objective Function Value $F[\bar{x}]$	
0	24.2	
20	0.534	
40	3.621	E-3
60	5.576	E-6
80	4.771	E-9
100	5.863	E-12

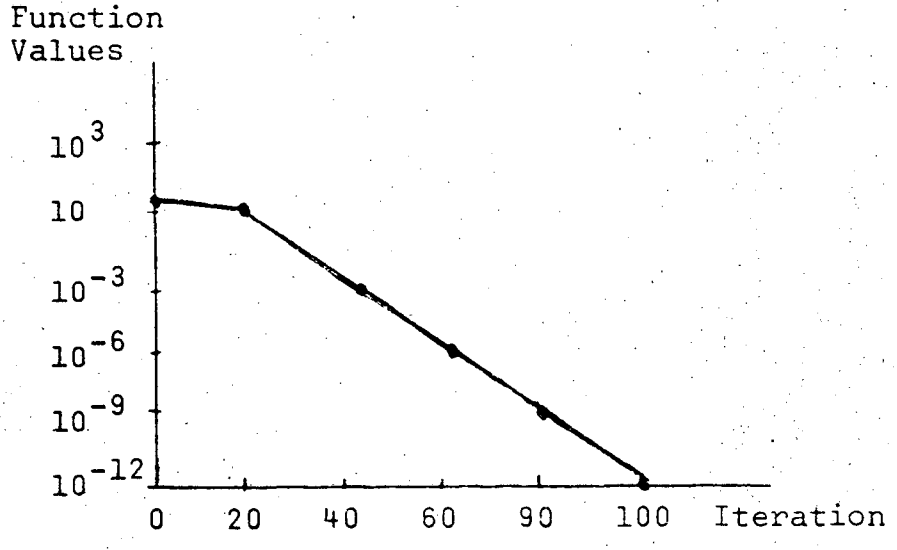


Table 1.4

Rosenbrook (1960)

Initial Value	$\bar{x}$ = (3.2,5.1)
Final Values Obtained	x_1 = 0.99999995710 x_2 = 099999986980
True Values	$x_1 = 1 \quad x_2 = 1$
Iteration No.	Objective Function Value $F[\bar{x}]$
0	2646.8
20	0.512
40	1.467 E-2
60	5.931 E-4
80	1.522 E-8
100	1.867 E-11

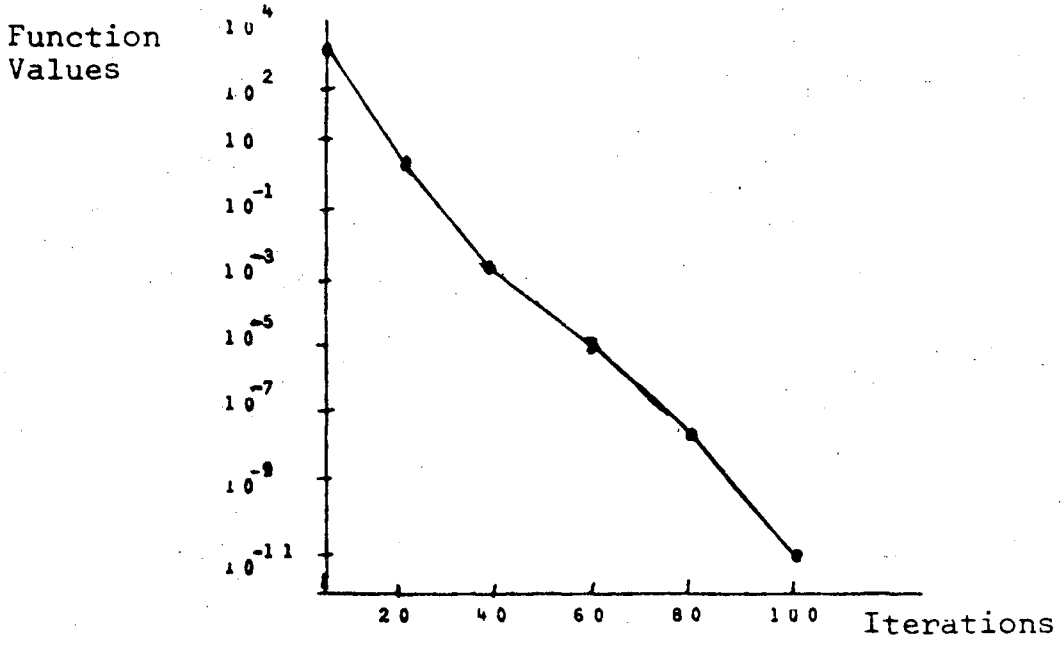


Table 1.5

Beale (1958)

Initial Values	$\bar{x} = (1,8)$
Final Values Obtained	$x_1 = 3.0000000026$ $x_2 = 4.9999999922$
True Values	$x_1 = 3 \quad x_2 = 2.5$
Iteration No.	Objective Function Value $F[\bar{x}]$
0	9.828869
20	0.149
40	7.027
60	1.796
80	2.117
100	2.111

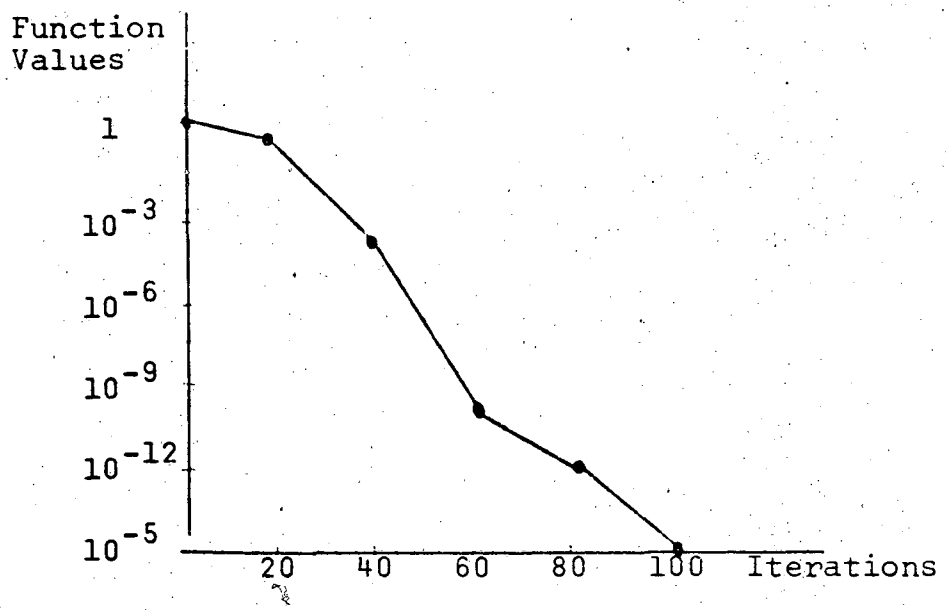


Table 1.6

Engvall (1966)

Initial Values	$\bar{x}$ = (5,2)
Final Values Obtained	x_1 = 10000003008 x_2 = 0.0000640257
True Values	x_1 = 1 x_2 = 0

Iteration Number	Objective Function Value $F[\bar{x}]$	
0	19.0625	
20	3.375	E-3
40	3.165	E-4
60	1.647	E-4
80	8.295	E-6
100	8.199	E-6

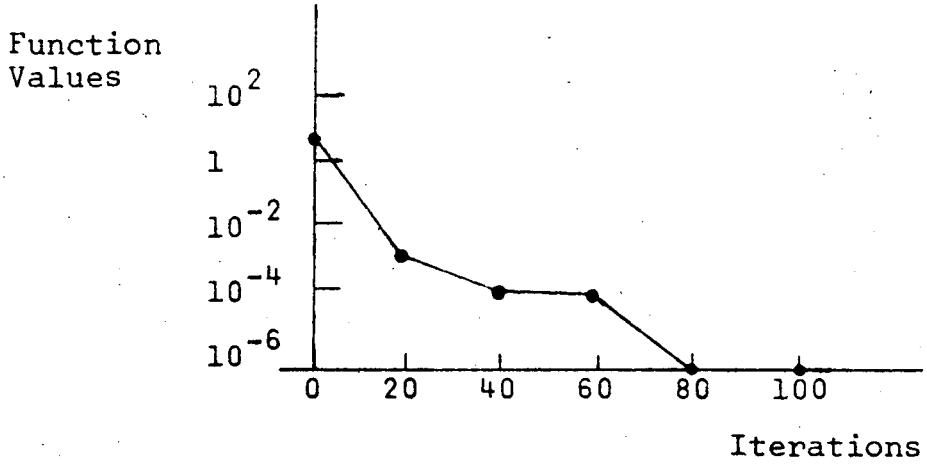


Table 1.7

Zangwill (1967)

Initial Values	$\bar{x}$ = (3,8)
Final Values Obtained	x_1 = 3.9999992813 x_2 = 9.000003345
True Values	x_1 = 4 x_2 = 9

Iteration Number	Objective Function Value
0	-16.6
10	-18.2

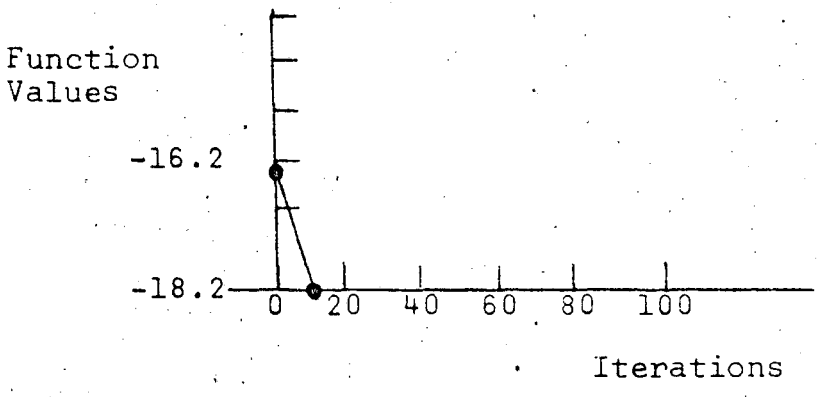


Table 1.8

Zangwill (1967)

Initial Values	$\bar{x}$ = (100,-1,2.5)
Final Values Obtained	x_1 = -4.5963931176 E-8 x_2 = -3.9523165246 E-9 x_3 = 7.6403466058 E-9
True Values	x_1 = 0 x_2 = 0 x_3 = 0

Iteration Number	Objective Function Value $F[\bar{x}]$
0	29,726
20	9,496 E-7
40	1,455 E-12
60	6,959 E-15

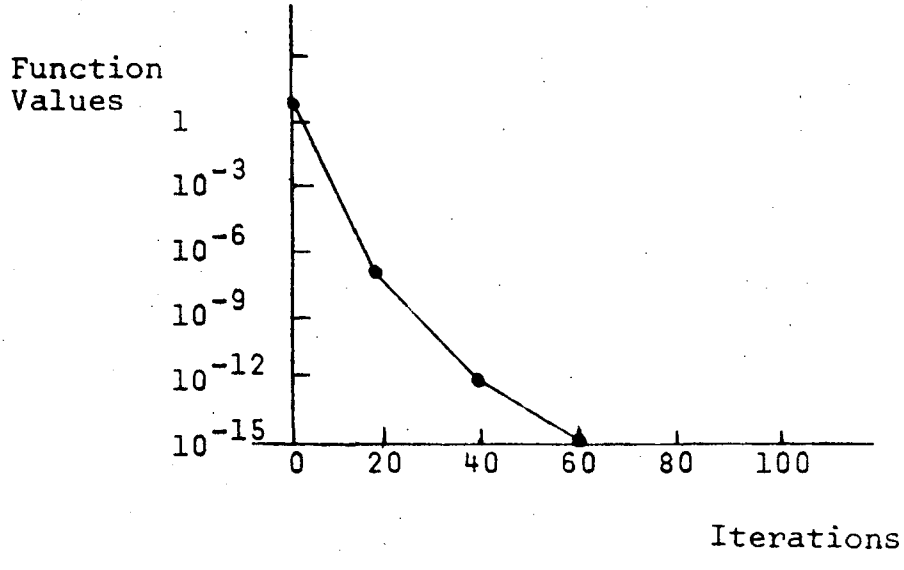


Table 1.9

Engvall (1966)

Initial Values	$\bar{x}$ = (1,2,0)	
Final Values Obtained	x_1 = 0.0000063 x_2 = 0.0000007 x_3 = 1.0000032	
True Values	$\bar{x}$ = (0,0,1)	
J	F[$\bar{x}$]	
0	2.424	E-2
40	8.302	E-3
80	1.482	E-3
120	4.101	E-5
160	7.891	E-8
200	1.007	E-11

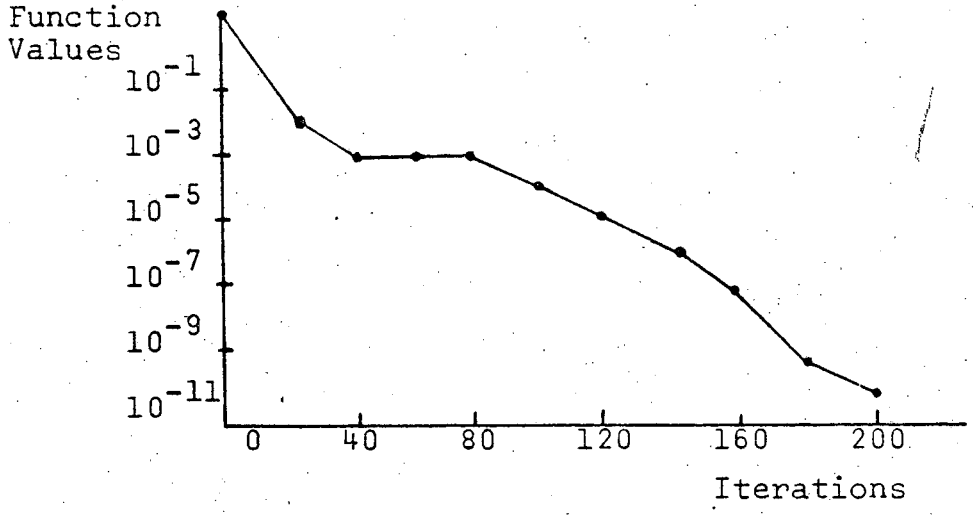


Table 1.10

Fletcher & Powell (1963)

Initial Values	$x_1 = -1$ $x_2 = 0$ $x_3 = 0$
Final Values Obtained	$x_1 = 1$ $x_2 = 0$ $x_3 = 0$
True Values	$\bar{x} = (1,0,0)$
J	$F[\bar{x}]$
0	2500
9	0

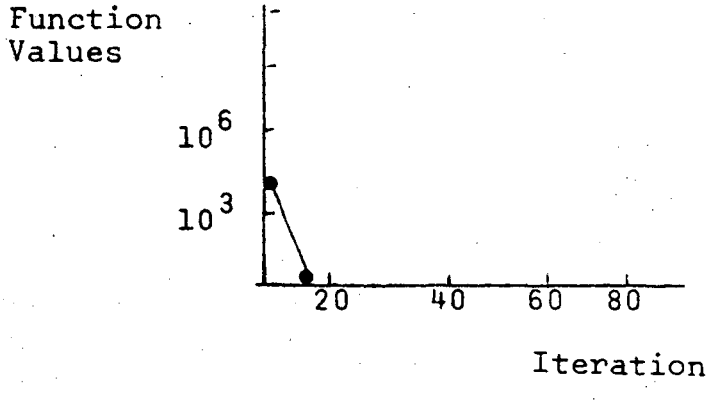
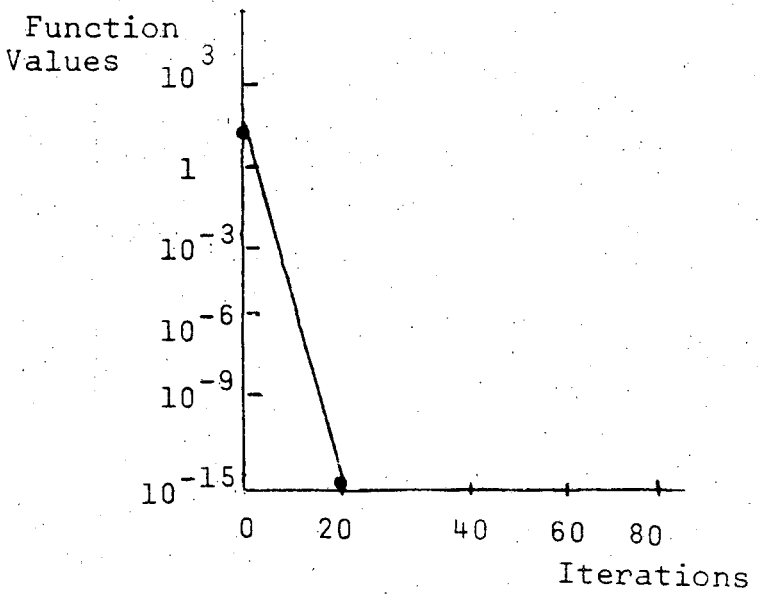


Table 1.11

Powell Singular (1964)

Initial Values	$\bar{x} = (3,-1,0,1)$	
Final Values Obtained	$x_1 = -1{,}00065692829$	E-4
	$x_2 = 1{,}00065692869$	E-5
	$x_3 = 0$	
	$x_4 = -2{,}7535001022$	E-8
True Values	$x_1 = 0$ $x_2 = 0$	
	$x_3 = 0$ $x_4 = 0$	

J	$F[\bar{x}]$
0	215
20	4.816



1.6.6 The Comparison of Bremermann's Optimizer with other Algorithms

A. In the comparison on the 50-dimensional problems, Bremermann's optimizer clearly performed better than the Brent-Shamanskii method and the new gradient-secant method.

The Rosenbrook Banana Type (Tables 1.12, 1.13) in central processing time (C.P.U.) on the CDC 6400 computer, the optimizer is twice as fast as the Brent Shamanskii method, and three times faster than the new gradient-secant method.

The optimizer needed 16,044 fewer function evaluations than the Secant method and 8,050 less than the Brent-Shamanskii method. (Note: Here clearly function evaluation is not a good criteria for comparison.)

The 50-Dimensional Bowl Type (Tables 1.12, 1.14)

Here again the optimizer performed better in terms of both time and number of function evaluations. The optimizer needed 1,043 function evaluations compared to 9,093 and 9,771 for the Brent Shamanskii method and the new Secant method, respectively.

B. In the problems with up to 4 variables the optimizer proved itself a robust method on the given test problems. It was a little slower than Fletcher's method in terms of time comparison, but in all problems was comparable to those algorithms which are generally classified in the literature as "good algorithms", Himmelblau [1973]. The results are

Table 1.12 RESULTS OF OPTIMIZER ON 50-DIMENSIONAL PROBLEMS TESTED ON A CDC 6400 COMPUTER
All problems satisfied the stopping criteria, i.e. $F[\bar{x}] \leq 10^{-10} |x_k - x_{k+1}| \leq 10^{-6}$

Function	Number of Variables	Initial Function Value	Number of Iterations	Computational Processing Time (in seconds)	Solution
Rosenbrook 50-dimensional type "Banana"	50	$F[\bar{x}] = 1019$	199	15.3	$\bar{x}_i = \bar{0}$, $i = 1,\ldots,50$
50-dimensional Bowl type	50	$F[\bar{x}] = 0.1163$	149	9.8	$\bar{x}_i = \bar{0}$, $i = 1,\ldots,50$

Table 1.13 COMPARATIVE RESULTS ON THE 50-DIMENSIONAL ROSENBROOK TYPE "BANANA"

Method	Number of Iterations	Number of Function Evaluations	C.P.U. Second
Brent-Shamanskii (B-S)	58	9,443	37.5
New Secant Method (SEC)	170	17,537	48.25
Bremermann's Optimizer (OPTIM)	199	1,393	15.34

Method	Initial Value of $\|F[\bar{x}_0]\|$	Computation was Terminated when
B-S	441	$\|\nabla F[\bar{x}_i]\| < 10^{-9}$ $i = 1, \ldots, 50$
SFC	441	$\|\nabla F[\bar{x}_i]\| < 10^{-9}$ $i = 1, \ldots, 50$
OPTIM	1.019×10^3	$\|F[\bar{x}]\| < 10^{-9}$ and $\|x_k - x_{k+1}\| < 10^{-6}$

Table 1.14 COMPARATIVE RESULTS ON THE 50-DIMENSIONAL FUNCTION "BOWL" TYPE

Method	Number of Iterations	Number of Function Evaluations	C.P.U. Second
B-S	98	9,093	39.77
SECANT	93	9,771	16.5
OPTIM	149	1,043	9.809

Method	Initial Value of $\|F[\bar{x}_0]\|$	Computation was Terminated when
B-S	0.0047	$\|\nabla F[\bar{x}_i]\| < 10^{-9}$
SEC	0.0047	$\|\nabla F[\bar{x}_i]\| < 10^{-9}$
OPTIM	0.0116	$\|F[\bar{x}_i]\| < 10^{-10}$ and $\|x_k - x_{k+1}\| < 10^{-6}$

	PROBLEM CHARACTERISTICS:				
Function Name	Number of Variables	Initial Values Chosen	Number of Iterations	Total Time in Seconds	Exact Valves Sought
Rosenbrook	2	(-1.2,1)	91	0.066	(1,1)
	2	(3.2,5.1)	94	0.071	(1,1)
Engvall (1966)	2	(5,2)	46	0.032	(1,0)
Zangwill (1967)	2	(3,8)	11	0.008	(4,9)
Beale (1958)	2	(1,8)	53	0.030	(3,5)
Zangwill (1967)	3	(100,-1,2.5)	35	0.020	(0,0,0)
Fletcher & Powell (1967)	3	(-1,0,0)	9	0.013	(1,0,0)
Engvall	3	(1,2,0)	115	0.098	(0,0,1)
Powell (1962)	4	(3,-1,01)	25	0.023	(0,0,0,0)
Wood	4	(-3,-1,-3,-1)		0.5	(1,1,1,1)

Table 1.15

Results for Test Functions with up to 4-variables using Bremermann's Optimizer on a CDC-6600 Computer. All the problems satisfied the stopping criteria, i.e., $F[\bar{x}] \leq 10^{-5}$ and $|\bar{x}_k - x_{k+1}| \leq 10^{-5}$, (where k denotes the k^{th} iteration)

Table 1.16 Central Processing Time of the Different Algorithms on Various Test Problems With a CDC-6600 Computer (Dimension $\leq$ 4)*

Method Using Derivatives	PROBLEM Zangwill 1967	Rosenbrook 1960	Beale 1958	Engvall 1966	Zangwill 1967	Engvall 1966	Powell Singular 1963	Fletcher & Powell 1963	Wood
	Number of Variables								
	2	2	2	2	3	3	4	4	4
Davidon Fletcher Powell	.006	.030	F**	.016	.010	.028	.079	.047	.085
Broyden	.008	.031	F**	.017	.017	.056	.112	.047	.089
Projected Newton (Pearson)	.007	.043	.031	.017	.032	.073	.125	.134	F**
Fletcher Reeves	.010	.047	.033	.019	.021	.154	.691	.222	.330
Continued Spartan (Shah et al)	.009	.043	.041	.020	.017	.489	.536	.256	.410
Goldstein Price	.007	.027	.024	.020	.019	.044	.065	.061	.089
Fletcher	.004	.022	.013	.008	.004	.022	.062	.036	.024

*All the algorithms except Bremermann's optimizer were tested by H. Himmelblau at another location. [59]
** F = Failed

Table 1.16 (Continued)

Derivatives Free Methods	Zangwill 1967	Rosenbrook 1960	Beale 1958	Engvall 1966	Zangwill 1967	Engvall 1966	Powell Singular 1963	Fletcher & Powell 1967	Wood
	Number of Variables								
	2	2	2	2	3	3	4	4	4
Hooke & Teeves	.009	.056	.024	.008	.014	.012	.015	.214	0.152
Melder & Mead	.024	.035	.029	.025	.096	.087	.153	.118	0.154
Powell	.010	.038	.021	.017	.014	.052	.114	.017	0.041
Rosenbrook	.018	.053	.052	.026	.067	.114	.183	.146	0.378
Stewart	.008	.026	.024	.016	.020	.048	.230	.090	0.171
Bremermann's Optimizer	.008	.066	.030	.032	.020	.098	.023	.013	0.151

recorded in Tables

1.7 ROOT-FINDING

Given an objective function $F: \mathbb{R}^n \rightarrow \mathbb{R}$; a vector $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Finding a minimum of F involves solving

$$\frac{\partial F}{\partial x_i} = 0 \quad i = 1, \ldots, n$$

To find the critical point of this system, in particular if this system consists of non-linear algebraic equations, the critical point is a "zero" of the system. Hence optimization is closely related to root-finding of non-linear algebraic equations.

The root finding problem can be stated as follows:

Given n-non-linear algebraic equations f_i $i = 1, \ldots, n$ find the value $\bar{x}^*$ of the n-dimensional vector $\bar{x}$ such that

$$f_i[\bar{x}^*] = 0 \qquad i = 1, \ldots, n$$

There exist an extensive and detailed literature dealing with the root-finding problem on non-linear algebraic equations. In particular, all methods which minimize a function in n-variables can be used for solving such systems by minimizing an objective $F[\bar{x}]$ such that

$$F[\bar{x}] = \sum_{i=1}^{n} (f_i[x_1, \ldots, x_n])^2$$

where the global minimum is obtained where $F[\bar{x}^*] = 0$, i.e., $f_i[\bar{x}^*] = 0 \quad i = 1, \ldots, n$. Good references can be found in [19], and [73].

In previous sections we have compared Bremermann's optimizer performance with fifteen prominent minimization algorithms. We can consider a minimization procedure a root finding algorithm.

We have further investigated the performance of Bremermann's optimizer on various test problems including root finding.

The following test problems were kindly furnished by S. Smale.

1.7.1 The Test Problems were:

$$f_1[x_1,x_2,x_3,x_4,x_5] = x_1^3 + a_1x_1x_5 + a_2$$

$$f_2[x_1,x_2,x_3,x_4,x_5] = x_2^3 + b_1x_2x_5 + b_2$$

$$f_3[x_1,x_2,x_3,x_4,x_5] = x_3^3 + c_1x_2x_3 + c_2x_5$$

$$f_4[x_1,x_2,x_3,x_4,x_5] = x_4^3 + d_1(x_1^2 + x_3^2) + d_2x_5$$

$$f_5[x_1,x_2,x_3,x_4,x_5] = x_5^3 + p_1x_1x_3 + p_2x_2x_4$$

The initial guesses and coefficients values are as follows:

Case I:

Coefficients: $a_1 = -10$, $a_2 = 1$, $b_1 = 10$, $b_2 = -1$,
$c_1 = 7$, $c_2 = 0$, $d_1 = -2$, $d_2 = 0$,
$p_1 = -7$, $p_2 = 8$

Initial Values:

Set 1: $(x_1,x_2,x_3,x_4,x_5) = (1,2,3,4,5)$

Set 2: " $= (5,4,3,2,1)$

Case II:

Coefficients: $a_1 = -11$, $a_2 = 1$, $b_1 = -12$, $b_2 = 2$,

$c_1 = -13$, $c_2 = 3$

$d_1 = -14$, $d_2 = 4$, $p_1 = -15$, $p_2 = 5$

Initial Values:

Set 1: $(x_1,x_2,x_3,x_4,x_5) = (40,40,30,40,30)$

" 2: " $= (5, -5, 5, -5, 5)$

" 3: " $= (-5, -5, -5, -5, -5)$

" 4: " $= (-1, 1, -1, 1, -1)$

" 5: " $= (-11, -12, -13, -14, -15)$

" 6: " $= (-1, 2, -4, 7, 36)$

" 7: " $= (1, 1, 1, 1, 1)$

Case III:

Coefficients: $a_1 = -0.11$, $a_2 = 0.1$, $b_1 = -0.12$, $b_2 = 0.2$

$c_1 = -0.13$, $c_2 = 0.3$, $d_1 = -0.14$, $d_2 = 0.4$

$p_1 = -0.15$, $p_2 = 0.5$

Initial Values: Set 1 to Set 7 will have the same initial values as the ones in Case II.

1.7.2 Root Finding Results

The results obtained by using the optimizer are recorded on Table 1.17.

	Total Execution Time (in seconds)	Final Values of the Objective Function	Root Found
Case I:			
Set 1	2.2	$F[\bar{x}] \leq 10^{-25}$ and $\lvert\bar{x}_{k+1}-\bar{x}_k\rvert \leq 10^{-14}$	$x_1 = -0.054$, $x_2 = 4.31$, $x_3 = 0$, $x_4 = .18$, $x_5 = -1.84$
Set 2	2.1		
Case II:			
Set 1	All sets together took 13.1 sec	$F[\bar{x}] \leq 10^{-25}$ and $\lvert\bar{x}_{k+1}-\bar{x}_k\rvert \leq 10^{-12}$	$\bar{x} = (0.18, 0.34, 0.33, 0.47, 0.49)$
Set 2			$\bar{x} = (10.5, 11.02, 11.8, 15.1, 10.1)$
Set 3			$\bar{x} = (-8.9, 0.023, -2.8, 10.6, 7.2)$
Set 4			$\bar{x} = (0.29, -2.1, -0.03, 0.016, 0.3)$
Set 5			$\bar{x} = (0.18, 0.34, 0.33, 0.47, 0.49)$
Set 6			$\bar{x} = (10.5, 11.02, 11.8, 15.1, 10.1)$
Set 7			$\bar{x} = (-8.9, 0.023, -2.8, 10.6, 7.2)$
Case III			
Set 1	All sets together took 4.4 sec	$F[\bar{x}] \leq 10^{-25}$ and $\lvert\bar{x}_{k+1} - \bar{x}_k\rvert \leq 10^{-12}$	$x_1 = -0.47$, $x_2 = -0.59$, $x_3 = -0.22$, $x_4 = -0.05$, $x_5 = 0.09$
Set 2			
Set 3			
Set 4			
Set 5			
Set 6			
Set 7			

Table 1.17 RESULTS ON ROOT-FINDING

When optimization of F finds a root, this, in general, will be one of many roots (a system of p-th order polynomials in n-variables has up to p^n roots). In one variable, when a root x_o has been found, then the polynomial can be divided by $x-x_0$ and a root of the remaining polynomial can be found, etc. This method is not available for several variables. However, multiple roots can be found by optimization combined with the method of deflation. We shall now discuss the different aspects of finding multiple roots of a non-linear algebraic system.

1.7.3 Finding Multiple Roots

Bremermann's optimizer was used to find multiple roots of a given non-linear system. In this context, we use the deflation techniques that have been studied in the one dimensional case by J. H. Wilkinson [1963].

Basically deflation techniques work so that once a root r of the system Q has been found, a new system is formed. This new system has the same roots of the original system except for the root r. (A good discussion of deflation techniques can be found in the paper of K. Brown and W. Gearhart [19].

Deflation Matrices

Let F be a system of N real non-linear algebraic equations in n-unknowns. Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space.

Definition:

For each element $\bar{P} \in \mathbb{R}^n$, let $N = N(\bar{x},p): \bar{P}\times\mathcal{O} \subset \mathbb{R}^n \to \mathbb{R}^{n2}$ matrix on $\mathbb{R}^n$ where $\bar{x}$ belongs to an open set $\mathcal{O}$ of $\mathbb{R}^n$ and $\bar{P}$ is the closure of such a set. We will call the function, N, a Deflation Matrix if for any differentiable function $F: \mathbb{R}^n \to \mathbb{R}^n$ such that $F[P] = 0$ and $F'[\bar{P}]$ is nonsingular, we have

$$\lim_{\to\infty} \text{Inf } \|N(\bar{x}_i,\bar{P})\cdot F[\bar{x}_i]\| > 0$$

for any sequence $\bar{x}_i \to \bar{P}$ where $\bar{x}_i \in \mathcal{O}$

Definition:

Let $G = N(\bar{x},\bar{P})\ F[\bar{x}]$ then we shall call G the "Deflated Function" or the function obtained from F by "deflating out" the simple zero $\bar{x} = \bar{r}$.

If we apply an iterative procedure to find additional roots the function G will have the following representation:

$$G[\bar{x}] = N^k(\bar{x},\bar{P}_k)\times \ldots \times N^2(x,P_2)\times N^1(\bar{x},P_1)\times F[\bar{x}] \ ;$$

where we deflated-out K-simple roots $P_1, \ldots, P_k$.

Three basic methods of deflation are found in the literature:

1) norm deflation

2) inner product deflation

3) gradient deflation

(In the following, the subscripted vector P_i is the ith root found.)

1) In the norm deflation the ith deflation matrix $N^i(\bar{x},\bar{P}_i)$ is taken to be:

$$N^i(\bar{x},\bar{P}_i) = \frac{A}{\|\bar{x}-\bar{P}_i\|}$$

where A is nonsingular matrix of $\mathbb{R}^n$ and the norm $\|\ \|$ can be any well defined vector norm.

2) In the inner product deflation the ith $N^i(\bar{x},\bar{P})$ matrix is always a diagonal matrix and the j-th diagonal element of $N^i(\bar{x},\bar{P}_i)$ is given by

$$N^i_{jj} = (< a_j^{(i)}, \bar{x}-\bar{P}_i >)^{-1}$$

where $a_1^{(i)}, \ldots, a_n^{(i)}$ are nonzero vectors belonging to $\mathbb{R}^n$, $i = 1, \ldots, k$ and $< >$ denotes the inner product.

3) If the above $a_j^{(i)}$ are chosen to be

$$a_j^{(i)} = \nabla F[\bar{P}_i] \qquad i = 1, \ldots, k$$

The method is called gradient deflation. And the j-th components of the function G will be denoted by:

$$G_j[\bar{x}] = \frac{F_j[\bar{x}]}{\prod_{i=1}^{k} < \nabla F_j[\bar{P}_i], \bar{x}-\bar{P}_i >} \qquad j = 1, \ldots, k$$

1.7.4 Methods Used in the Comparison for Multiple Root Finding:

In the paper of K. M. Brown and W. B. Gearhart [19] two methods were used to find multiple roots of 4 different systems. The methods were

1) discretized form of Newton's method

2) a quadratically convergent method due to K. M. Brown

In the comparison all three deflation techniques were used by each of the above algorithms.

For the purpose of comparing the algorithms performance with the optimizer, the best of the deflation technique for each method was chosen, while the optimizer used the same norm, i.e. $\|(a_1, \ldots, a_n)\| = \max(|a_1|, |a_2|, \ldots, |a_n|)$ for three of the test problems and the euclidean norm for the fourth problem.

1.7.5 The Test Problems for Multiple Root-Finding

The test problems used for finding multiple roots were formulated by K. M. Brown and W. B. Gearhart

The test problems used were

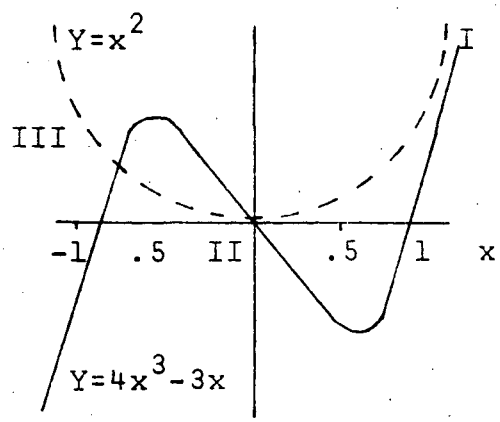
1) The cubic parabola

$$F[x,y] = yx^3 - 3x - y$$

$$G[x,y] = x^2 - y$$

The system has three roots:

$P_1 = (1,1)$ $P_2 = (0,0)$ $P_3 = (-0.75, 0.5625)$



This system has a "magnetic zero", with respect to Newton's method, which is defined as a zero for which a particular method tends to converge, irrespectively of the starting guess.

The existence of magnetic zero together with the technique used in finding multiple roots will determine a region of convergence. For different methods the region of convergence can vary, hence it is possible that one technique will be able to find all the roots of the system while another will fail to obtain some of these roots.

2) <u>The four cluster</u>

$$F(x,y) = (x-y^2)\ (x-\sin y)$$

$$G(x,y) = (\cos y - x)\ (y-\cos x)$$

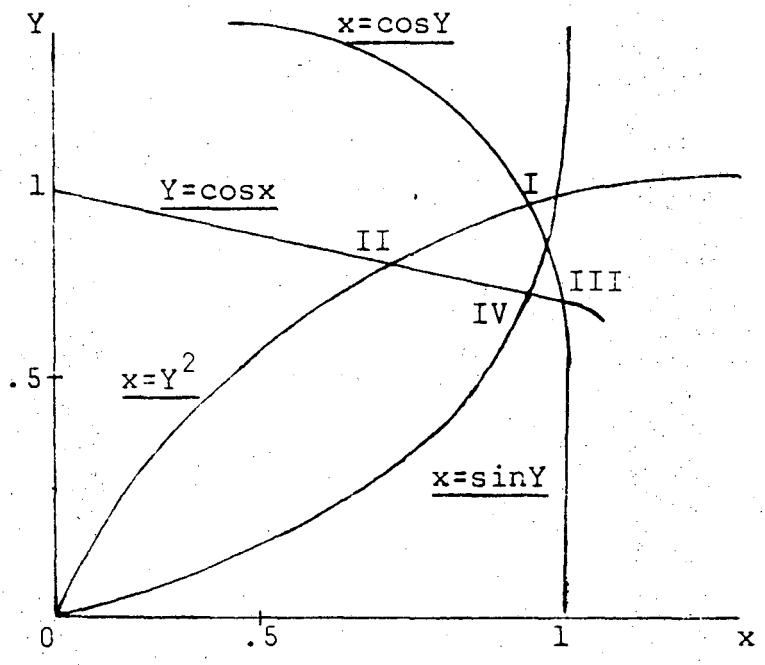
has four roots:

$$P_1 \simeq (0.68,\ 0.82)$$

$$P_2 \simeq (0.64,\ 0.80)$$

$$P_3 \simeq (0.71,\ 0.79)$$

$$P_4 \simeq (0.69,\ 0.77)$$



This system has distinct roots nearly close together in the first quadrant of the x-y plane. Here round-off errors and the region of convergence of a particular technique may cause that not all of the roots can be found.

3) The Hyperbola Circle

$$F[x,y] = xy - 1$$

$$G[x,y] = x^2 + y^2 - 4$$

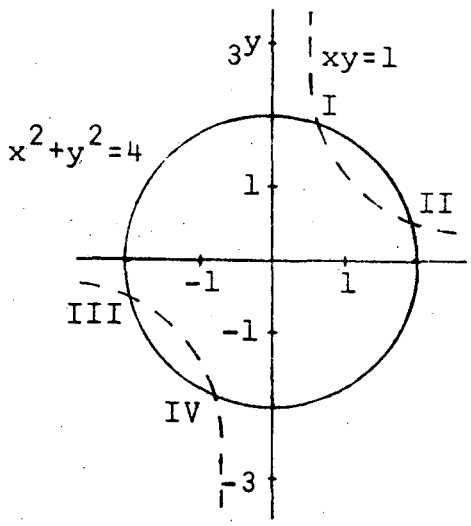
has four zeros,

$P_1 = (0.517, 1.93)$

$P_2 = (1.93, 0.517)$

$P_3 = (-1.93, -0.517)$

$P_4 = (-0.517, -1.93)$



This system has the "bad" property, that some algorithms might diverge to infinity on the first attempt to find a root. The "right" combination of both, the deflation technique and the method used in finding the roots will determine the extent of success in finding all roots of this system.

4) The 3×3 system

$$F[x,y,z] = x^2 + 2y^2 - 4$$

$$G[x,y,z] = x^2 + y^2 + z - 8$$

$$H[x,y,z] = (x-1)^2 + (2y-2)^2 + (z-5)^2 - 4$$

has two roots:

$P_1 = (0, \sqrt{2}, 6)$

$P_2 = (2, 0, 4)$

Table 1.18 ROOT FINDING: RESULTS OF COMPARISON

	PROBLEMS	# OF ROOTS	BROWN'S METHOD		DISCRETIZE NEWTON'S METHOD		BREMERMANN'S OPTIMIZER	
			Number of Roots Found	Deflation Method	Number of Roots Found	Deflation Method	Number of Roots Found	Deflation Method
1	Cubic Parabola	3	3	Uniform Norm	2	With all 3 types of deflection only 2 roots were found	3	IN ALL PROBLEMS
2	The Four Cluster	4	2	Euclidean Norm	2	Euclidean Norm	4	WE USED
3	Hyperbola Circle	4	4	Euclidean & Uniform Norm	2	Gradient Norm	3 or 4*	THE MAXIMUM
4	The 3×3 System	2	2	Euclidean Norm	2	Euclidean Norm	2	NORM

INITIAL POINTS FOR ALL PROBLEMS IN THE COMPARISON WERE THE SAME; THE POINTS WERE:

PROBLEM	STARTING POINT
1	$\bar{X}_0 = (0.8, 0.55)$
2	$\bar{X}_0 = (0.9, 1)$
3	$\bar{X}_0 = (0, 1)$
4	$X_0 = (1, 0.7, 5)$

*DEPENDS ON INITIAL POINT; FOUND 3 ROOTS FROM SAME INITIAL POINT AS OTHER METHODS BUT FOUND ALL 4 ROOTS WHEN ALLOWED TO START AT ANOTHER INITIAL POINT.

These systems will increase the computational difficulty and will cause difficulties when applying the inner product or gradient norms.

1.7.6 Results on Multiple Root-Finding

All four problems were tested with a CDC 6400 computer and all roots were found to within 12 digits accuracy. The results of the comparison are recorded in Table [1.18].

1.8 CONCLUSIONS

In this Chapter we tried for the first time to evaluate the performance of the Bremermann optimizer. We can conclude the following:

1) The Bremermann optimizer has performed well compared to 15 leading algorithms in solving test problems involving 4 or fewer variables.
2) The Bremermann optimizer performed very well on n-dimensional problems; in our case, 50-dimensional.
3) The Bremermann optimizer is a good algorithm to be used in finding a root as well as multiple roots of nonlinear systems.
4) There is a need to continue to investigate the different multipliers and steplength strategies.
5) Execution time is small and preparation of a problem for use with Bremermann's optimizer is easy, facts that few algorithms can claim.

6) The optimizer can be applied with a minimum of effort since it does not require the Hessian or Jacobian of an objective function. Furthermore very few changes have to be made to optimize different functions.

1.9 NOTATION

A = matrix

A_K = matrix (the K index indicates that this matrix may change from step to step.

I = identity matrix

$\bar{J}$ = Jacobian matrix

J = iteration

L = Lagrangian function

F = objective function

T = superscript refering to the transpose of a vector or matrix

i,j = indexes used for variables

K = superscript refering to iteration number

n = number of variables

$\bar{S}$ = vector $(S_1, \ldots, S_n)$ indicating direction

$\bar{X}$ = vector $(X_1, \ldots, X_n)$ of variables

λ = steplenght

∇ = gradient [e.g. $\nabla F(X_1,\ldots,X_n) = \left(\frac{\partial F}{\partial X_1}, \frac{\partial F}{\partial X_2}, \ldots, \frac{\partial F}{\partial X_n} \right)^T$

d = distance in n-dimensional space

h = elements of Hessian

r,p = root

$\bar{U},\bar{V}$ = vectors

$N(\bar{X},P)$ = matrix on E^{N^2} used for finding multiple roots

optim. = Bremermann's optimizer

PART II: DETERMINATION OF THE KINETIC PARAMETERS OF THE PHOTOSYNTHESIS CALVIN CYCLE

2.0 INTRODUCTION TO PART II

All life on earth is dependent on photosynthesis by plants, the process by which carbon dioxide, CO_2, in the presence of light is fixed and reduced to organic matter.

Photosynthesis is carried out by both procaryotic and eurayotic cells. The procaryotes include the blue green algae and some bacteria; the eucaryotes include higher green plants, red, green and brown algae, dimoflagellates, and diatons.

Photosythesis consists of two processes, a light-dependent phase (light reactions), and a light-independent phase (dark reactions). In the light reactions, light energy, photons, are converted into chemical energy, stored in ATP (ademosine-5'-triphosphate) and NADPH (reduced nicotinamide adenine dinucleotide phosphate). The dark reactions, on the other hand, refer to the enzymatic reactions in which CO_2 is incorporated into reduced carbon compounds previously encountered in carbohydrate metabolism.

The pathway whereby CO_2 is reduced to carbohydrates and other organic compounds was first discovered by M. Calvin, J. A. Bassham et al [3].

Although the pathway of the pentose phosphate cycle (Calvin cycle) has been known since 1954, knowledge of the dynamic changes (kinetics) of its intermediates has remained incomplete. The cycle involves seventeen intermediates which conversion to each other is mediated by different enzymes. Each step from one intermediate to another can be described

mathematically by seventeen non-linear, first order differential equations which contain twenty-two kinetic parameters. These parameters determine the kinetics; their calculation will be the object of this Chapter. There are major difficulties in this calculation, since the parameters must be determined simultaneously for the following reasons:

a) some intermediates are involved in several reactions
b) some reactions reach equilibrium too fast for accurate measurement when tested in isolation from the rest of the reaction of the cycle
c) the reactions depend on the enzymes present whose activities, in turn, are affected by other metabolite pool sizes in the cycle.

Though the data of the cycle has been available for a number of years, the mathematical computation has remained beyond reach for five main reasons:

a) the system was too large for computational techniques;
b) powerful computers to handle large scale problems were not available;
c) the set of available data was incomplete to determine the kinetics of pool size changes;
d) there were no adequate techniques for labelling the intermediates of the cycle;
e) only recently have good techniques for handling stiff equations (differential equations that are difficult to integrate numerically) become available.

However, recent advances in experimental and computational techniques have brought the problem within reach.

The determination of the kinetic parameters is known mathematically as "systems identification" or "parameter" identification.

Previously to our attempt, Heinmets [45] evolved a large non-linear model formualted for an enzyme induction system through automatic computarized parameter identification. Roth and Roth [76] used Bellman's method of quasi-linearization for testing Heinmets model. The attempt was unsuccessful and it pointed to the need for investigating the numerical problems involved.

Such an investigation of these numerical problems was carried out by Joel Swartz [83]. He implemented many methods proposed by Bremermann [10], [9] and, based upon the method of Rosenbrook and Storey [75] , developed and tested a theory of how experimental errors affect the accuracy of determination of parameters.

With work done in the area of numerical analysis one of our tasks was the gathering of reliable biochemical data. We were interested in determining the pool sizes of intermediates of the Calvin cycle under certain biochemical conditions and after disturbance of such a system. With this information we should be able to determine simultaneously the kinetic parameters of all reactions involved in the cycle.

2.1 EXPERIMENTAL DATA

With the discovery of C^{14}, a radioactive isotope of carbon, it was recognized the usefulness of this isotope as a label for the identification of compounds in biological processes. In 1946, M. Calvin, J. A. Bassham et al set out to trace the path of carbon in photosynthesis using $C^{14}O_2$ as one of their principal tools.

Early experiments were conducted with leaves of plants in a closed chamber containing carbon dioxide labelled with C^{14}. After photosynthesis was allowed in a leaf for a given period, biochemical activity was halted by immersion in alcohol. The alcohol inactivates the enzymes which are needed for the creation of various intermediates, without which the reactions must stop. Because it was recognized that in this technique even a very small delay in alcohol immersion would lead to misinterpretation of the results, a new method was needed through which more reliable data could be obtained.

To obtain greater precision in this respect a single-cell green algae -- chlorella pyrenoidoisa -- was adapted as the subject for many experiments. Data about metabolic changes under various conditions were obtained as follows: chlorella was grown under steady-state conditions. At time zero, unlabelled CO_2 was replaced by $C^{14}O_2$ and C^{14}-labelled bicarbonate was injected into the culture medium to effect a step function switch from unlabelled to labelled carbon in the CO_2 uptake of the system. Aliquots were killed, i.e.

immersed in alcohol, at an interval of one to several minutes after addition of $C^{14}O_2$. The aliquots were subsequently analyzed by paper chromatography and autoradiography (pp

These experiments successfully determined the pathways of CO_2 from its fixation to the reduced state (sugar phosphates) and the regeneration of the acceptor for CO_2 in the Calvin Cycle.

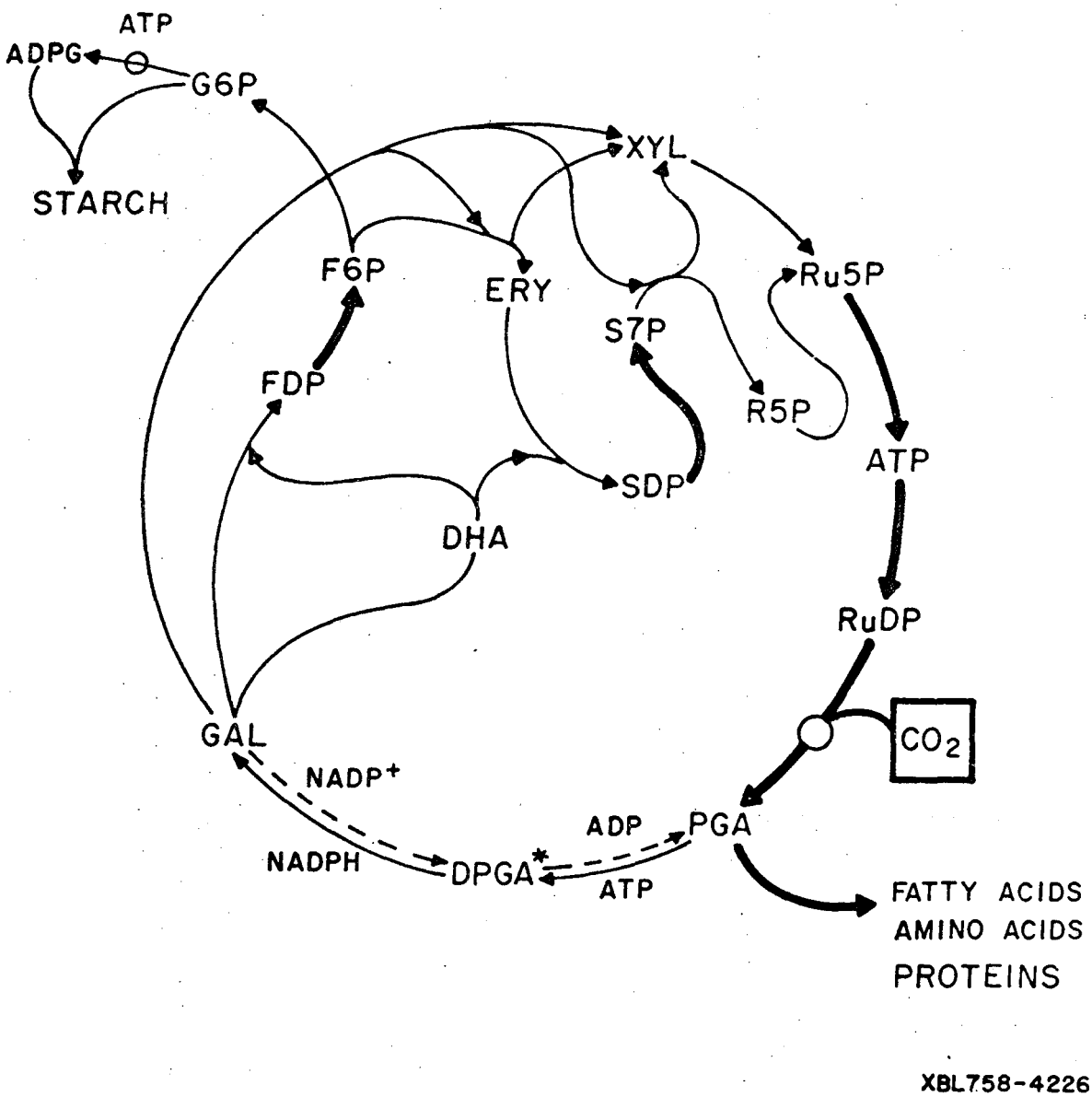
The first steps along the path are the reaction of ribulose diphosphate (RUDP) with CO_2, the resulting products are two molecules of 3-phosphoglyceric acid, PGA. This step is called the carboxylation phase. The reduction phase, is the step along the path where PGA combines with adenosine-5'-triphosphate to produce diphosphoglycbrate, (DPGA), and adenosine-5'-dyphosphate, ADP. DPGA in turn combines with reduced nicotinanide adenine dinucleotide phosphate, NADPH, and the resulting products are glyceraldehyde-3-phosphate, GAL, and nicotinanide adenine dinucleotide phosphate, $NADP^+$. The chloroplasts of higher green plants contain the NADP+ specific enzyme. Moreover in these two reactions the NADPH and ATP are utilized to drive the carbon reduction cycle.

The remainder of the reactions accomplishes the regeneration of ribulose diphosphate, RUDP, necessary to keep the cycle operating, this phase is called the regeneration phase.

2.1.1 Abbreviations

The following are the intermediates of the Calvin Cycle and the abbreviations that will be used in the state equations;

CO_2	Carbon-Dioxide
RUDP	Ribulose - 1,5 - Diphosphate
PGA	3 - Phosphoglyceric Acid
GAL	Glyceraldehyde-3 - Phosphate
FDP	Fructose - 1,6 - Diphosphate
F6P	Fructose - 6 - Phosphate
ERY	Erythrose - 4 - Phosphate
SDP	Sedoheptulose - 1,7 - Diphosphate
S7P	Sedoheptulose - 7 - Phosphate
XYL	Xylulose - 5 - Phosphate
Ru5P	Ribulose - 5 - Phosphate
ADP	Adenosine - 5' - Diphosphate
ATP	Adenosine - 5' - Triphosphate
NADP+	Nicotinamide Adenine Dinucleotide Phosphate
NADPH	Reduced Nicotinamide Adenine Dinucleotide Phosphate
G6P	Glucose - 6 - Phosphate
DHAP	Dihydroxy Acetone Phosphate
R5P	Ribose - 5 - Phosphate
P_i^{2-}	Inorganic Phosphate
H^+	Hydrogen Proton



*This intermediate is not being considered in the mathematical model (see page 104)

The dark lines represent regulatory points of the cycle where the reaction proceeds only in the direction of the arrow. The rest of the reactions are considered to be reversible.

As a result of many experiments, it was found that carbohydrates are not the sole organic product of photosynthesis but serve for further biosynthesis of fats, amino acids and proteins.

After the main intermediates and enzymes of the Calvin Cycle were found, interest concentrated on the regulation of the enzymes involved in this cycle. Since whole leaves and chlorella turned out to be a very complex system to study regulatory sites, attempts were made to isolate the choroplast, the compartment where the reductive pentose phosphate cycle is located and operating.

In 1966 J. A. Bassham and R. G. Jensen [54] developed a new technique for isolating chloroplast. The isolated chlorophast became the subject for study regulatory sites of the Calvin Cycle. In this new system photosynthetic reactions could be isolated from reactions of the cytoplasm, and the cell wall could be eliminated as a barrier to the assimilation of various added metabolites and chemicals. This new characteristic greatly facilitated the study of the mechanisms of enzymic transformation and metabolic control in the reactions of photosynthesis.

When isolated chloroplasts were studied a special difficulty arose. Initially the measurements of CO_2 fixation of whole spinach leaves was far greater than the corresponding rate in isolated chloroplats. However, as a result of improvements made in the isolation procedures, the rate of CO_2 fixation by isolated chloroplats gradually rose until

it reached 50% or more of the rate for intact spinach leaves [6].

To further eliminate problems of chemical transport across the chloroplast membrane and to enhance more metabolic control over the Calvin Cycle, J. Bassham and co-workers developed a new technique useful in studies of regulatory mechanisms of photosynthesis. The new technique consists of fracturing the chloroplast and combining its contents with soluble components from lysed chloroplast. This is called a reconstituted chloroplast system. When phtosynthesis is allowed in this sytem the resulting products are principally intermediate compound of the Calvin Cycle.

When whole leaves or chloroplasts are used the observed radioactivity comes from a mixture of unlabelled, partially labelled and fully labelled compounds. This mixture cannot be untangled for conversion into concentration data necessary for solving the parameter identification problem. However the new technique of fracturing chloroplasts and reconstituting the photosynthetic system in vitro eliminates the problem of partial labelling because of the following reasons:

a) After CO_2 is removed (nitrogen gas is flushed through the flask, for several minutes), the system is reactivated by introduction of one or several of the following substances:

1) $NaHC^{14}O_3$, radioactive sodium bicarbonate

2) $C^{14}O_2$ radioactive carbon dioxide

3) P^{32} radioactive phosphate

4) labelled primers such as, labelled PGA, and labelled glucose

b) Endogenous metabolite concentration are small due to dilution of chloroplast volume into flask volume and therefore the specific activity by adding a primer with known S.A. (μc/μ mole) does not change hence by knowing the radioactivity in a labelled compound the metabolites is directly convertible into concentrations. New methods of counting and automatic collection of radioactivity counts have contributed to a greater accuracy of the data. This data can be used for determination of the kinetic parameters.

The enzymes in the reconstituted system are also diluted in the flask, but the concentrations are sufficiently high to maintain high rates of photosynthesis.

For the cycle to be operative at satisfactory level 1/10 of a millimolar of ADP and NADP is added to the flask. Furthermore in the flask, the ratio of the soluble part of chloroplasts to the chloroplast membrane system (lamellae) is increased by fourteen to one (14:1) compared to whole chloroplasts.

2.1.3 Design of an Experiment

The following technique was applied to obtain kinetic data from the reconstituted system.

Two batches of spinach chloroplasts were isolated according to the method of Jensen and Bassham [54] .

The leaves were cut into small pieces and placed in a solution. The leave pieces and the solution are cooled to 0°, and are blended for five seconds at high speed. The slurry was filtered through several layers of cheesecloth and the resulting juice was centrifuged for fifty seconds.

Each chloroplast was suspended in a solution and cooled to 0°.

To fracture the chloroplasts the pellets of both batches were suspened in a solution cooled to 0°. After ten minutes the suspension was centrifuged and the supernatant solution containing soluble components from the chloroplast was stored at 0° until the reconstitution of the system.

The pellet was resuspended in a solution and washed. After ten minutes was centrifuged and stored at 0°.

The resuspended green pellet and the soluble solution were placed into a flask.

Further additions were made, i.e., PGA as the primer, NADP, ADP. Photosynthesis is carried out in flasks stoppered with a serum cap. The flasks are mounted on a rack which holds 16 flasks and moves in a circular motion in the horizontal plane. This swirling distributes the suspension of chloroplast uniformly in the flask.

The flasks are held in a water bath and are illuminated through the transparent bottom of the bath.

Nitrogen is flushed through the flask to remove any air CO_2. The system was "starved" of CO_2 for five minutes. Then

labelled primer (C^{14}, PGA, or C^{14}-glucose and P^{32})is injected through the serum cap. Lamellae are injected and a pre-illumination period follows. After preillumination $NaHC^{14}O_3$ or $C^{14}O_2$ is injected to the system and photosynthesis proceeds. The shaking device is frequently stopped to allow sampling from the flasks. These samples are killed in a methanol solution and thus stopping the biochemical reaction.

The aliquots are analyzed by using two dimensional paper chromatography and radioautography.

2.1.4 Paper Chromatography and Autoradiography

For the identification of the intermediate of the Calvin Cycle the technique of two dimensional paper chromatography was used. This technique allows to separate individual components of a mixture by their size and charge. The principle of this technique is based on the distribution of compounds between a stationary phase (the paper) and a moving phase (organic, inorganic solvents which move by capillar forces or gravity through the paper). The different affinity and solubility to the stationary phase and the moving phase, respectively, separate the individual components from each other. The method is described as folllows:

A sample of the solution killed in mehanol and containing the labelled intermediates of the Calvin Cycle is applied to a sheet of chromatography paper (stationary phase) near one corner. An edge of the paper adjacent to the corner is immersed in an organic solvent (moving phase); the whole

assembly is placed in a vapor tight box. While the solvents are moving through the paper separation occurs. As a result, the compounds will be distributed in a row in one dimensional. Depending on the solubility of the compounds and the nature of the solvent used, some compounds may still overlap one another. Repeating the same procedure with a different solvent travelling orthogonal to the first direction will separate the compounds in a second dimension.

Since most of the compounds are colorless, special techniques are required to locate them on the paper. And since the compounds are labelled with either C^{14}, P^{32} or both, the technique of radioautography is applied.

The chromatography paper is placed in contact with a sheet of X-ray film for a few days. The compounds that are radioactive will appear as "foot prints" on the X-ray film. Further comparison between chromatographs and radiographs will identify the compounds and their exact location on the paper. The spots that contain a specific compound are then cut out from the paper and mounted on a tape for automatic counting. Counts of the radioactive decay of C^{14} or P^{32} are recorded automatically and entered into a computer, where the counts per compound are converted as amounts in μ moles/mg chlorophyl.

2.1.5 Data Used For Determining The Kinetic Parameters

The data used for determining the kinetic parameters is obtained from experiments with the reconstituted system.

The success in obtaining "reliable data" has not been able to be predicted from experimental experiments. (Some experiments have led to consistent results which in turn define what is meant by reliable data.) Therefore it is not unusual to perform several experiments until satisfactory kinetic data can be obtained. One difficulty in obtaining such data lies in the fact that such experiments are very costly and require substantial amounts of time (sometimes one or two months).

In this thesis one set of data will be used. In a later section (2.5.1) we shall discuss the need for several sets of data to determine accurately the kinetic parameters.

2.1.6 Derivation of the Dynamic Equations Describing the Calvin Cycle

Given any arbitrary chemical reaction

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C + D$$

where A,B,C and D are compounds; k_1, k_2 are kinetic parameters of this reaction; the arrows indicate the direction of the flux. It is possible to describe the rate of change of a particular compound in the following way:

The rate of formation of a compound, say A, is given by the flux entering the pool of this compound, i.e.

$$k_2[C][D]$$

The rate of decomposition is given by the flux leaving the pool of A and is given by

$$k_1[A][B]$$

The total rate of change of compound A is the net result of the forward flux given by $k_1[A][B]$ and the backward flux given by $k_2[C][D]$, i.e.

$$\frac{dA}{dt} = k_2[C][D] - k_1[A][B]$$

where $\frac{dA}{dt}$ denotes the rate of change of compound A with respect to time. By a similar way we obtain, $\frac{dA}{dt} = \frac{dB}{dt} = -\frac{dC}{dt} = -\frac{dD}{dt}$. Using the schematic representation (on page 102) of the Calvin Cycle we can apply the same procedure over all the reactions. Thus obtaining the rate of change with respect to time of each of the intermediates of the cycle.

The calculation of these equations is crucial for the success and validity of our **results.** Therefore it is important not to err while deriving the equations since a missing term or a wrong sing can give equations that belong to a completely different system. We derived the equations

several times with great care, but afterwards, when checking we found each time a missing term or a wrong sign. To be sure that our description is without mistakes we decided to generate the equations automatically by computer. This is possible because of the logical structure inherent in the derivations of the equations. To this end I adapted a computer program written in machine language (COMPASS), by Keith Davenport as a joint term project for a course in computer science and a course in biomathematics, to derive the rate equations (differential equations), describing the Calvin Cycle. This program will generate any system of differential equations which satisfies the mass action laws.

2.1.7 Schematic Representation of the Pathways of the Calvin Cycle and the Kinetic Parameters

(The arrows denote the direction of the reaction)

a) $CO_2 \oplus RUDP \xrightarrow{k_1} 2\ PGA$

b) $PGA \oplus ATP + NADPH \underset{k_3}{\overset{k_2}{\rightleftharpoons}} ADP + GAL + NADP^{\oplus} + P_i^{2-}$

c) $GAL \underset{k_4}{\overset{k_5}{\rightleftharpoons}} DHA$

d) $GAL \oplus DHA \underset{k_6}{\overset{k_7}{\rightleftharpoons}} FDP$

e) $FDP \xrightarrow{k_8} F6P$

f) $F6P \oplus GAL \underset{k_{10}}{\overset{k_9}{\rightleftharpoons}} ERY \oplus XYL$

g) $ERY \oplus DHA \underset{k_{11}}{\overset{k_{12}}{\rightleftharpoons}} SDP$

h) $SDP \xrightarrow{k_{13}} S7P$

i) $S7P \oplus GAL \underset{k_{15}}{\overset{k_{14}}{\rightleftharpoons}} R5P \oplus XYL$

j) $R5P \underset{k_{17}}{\overset{k_{16}}{\rightleftharpoons}} RU5P$

k) XYL $\underset{k_{18}}{\overset{k_{19}}{\rightleftharpoons}}$ RU5P

l) RU5P $\oplus$ ATP $\xrightarrow{k_{20}}$ RUDP $\oplus$ ADP

m) F6P $\underset{k_{22}}{\overset{k_{21}}{\rightleftharpoons}}$ G6P

2.1.8 Chemical Kinetics

Given any chemical reactions we can describe them by rate equations of the form:

$$\dot{x}_i = f_i(x_1, \ldots, x_n) \qquad i = 1 = 1, \ldots, n$$

where x_i is the concentration of the i^{th} species; and the f_i are polynomials in the x_i. The degree of the polynomial f_i is the order of the reaction. When two molecules unite to form a product, a second order term arises. When a molecule decomposes into parts, a linear term arises. When three molecules combine a third order term appears in f_i; however, usually three molecules first combine in a pair, forming an intermediate product which then reacts with the third molecule. Thus third order rate equations are a simplification of second order equations with an additional intermediate product.

The Calvin Cycle will be described by seventeen non-linear, first-order differential equations, which contain

twenty-two kinetic parameters. The polynomials in the rate equations are mostly of second degree. The equations have the form

$$\frac{dx_1}{dt} = \sum_i K_i^1 x_i + \sum_{i,j} K_{ij}^1 x_i x_j$$

$$\vdots$$

$$\frac{dx_n}{dt} = \sum_i K_i^n x_i + \sum_{i,j} K_{ij}^n x_i x_j$$

where x_i represents the state variables (measured as concentrations) of the system; i.e. the seventeen intermediates of the cycle, and K_i^n and K_{ij}^n are the kinetic parameter of the system. The concentrations x_i $i = 1, \ldots, n$ vary with time and will be of major importance in the determination of the parameters K_i and K_{ij}.

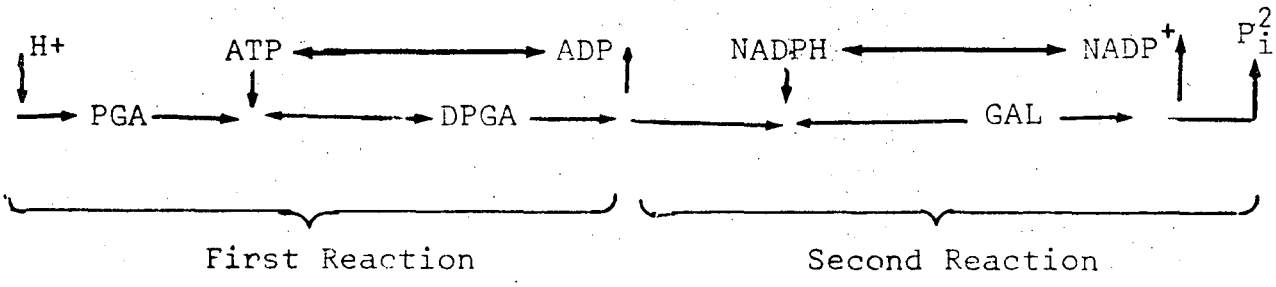
Note: Some of the equations will have third or fourth order reactions due to a simplification of the path from the intermediate PGA to the intermediate GAL. The reason for this simplification is that there exists an intermediate, 1-3 phosphoglycerate, (DPGA), whose half-life is very small; therefore, it is difficult to measure its concentration. Hence the reactions

$$H^+ \oplus PGA \oplus ATP \rightleftharpoons DPGA \oplus ADP + P_i^{2-}$$

and

$$DPGA \oplus NADPH \rightleftharpoons GAL \oplus NADP^+$$

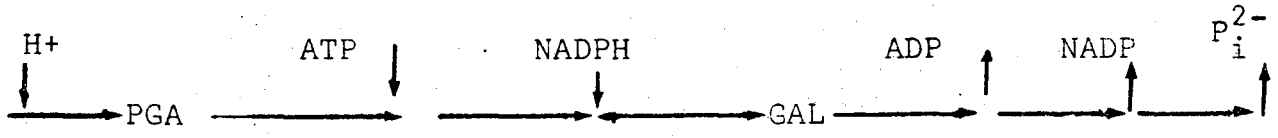
i.e.



can be considered equivalent to

$$H^+ \oplus PGA \oplus ATP \oplus NADPH \rightleftharpoons GAL \oplus ADP \oplus NADP^+ + P_i^{2-}$$

i.e.



This last reaction will be presented in some differential equations as a term having three or four interacting state variables.

As of now we have considered the biochemical reactions of the cycle without introducing enzyme kinetics. Since the enzymes have a kinetic system of their own which considerably affect the rate of biochemical reactions, we should discuss

discuss the basic characteristics of such a system.

2.1.9 Some Aspects of Enzyme Kinetics Michaelis-Menten

An enzyme is a protein that functions as a catalyst for a biochemical reaction. In any enzymatic reaction we encounter the phenomenon of saturation with substrate (compound which is being transformed). This phenomenon can be described as follows: with a fixed enzyme concentration, an increase of substrate will result at first in a very rapid rise in reaction rate. As the substrate concentration continues to increase, the reaction rate begins to decrease until, with a large substrate concentration, no further change in the reaction rate is observed. The reaction rates at low substrate concentrations are known as the "first order kinetics"; when the reaction rate starts to decrease due to the increase of substrate we have a "mixture of zero and first-order kinetics"; and finally when no change in the rate of reaction occurs this stage is called "zero order kinetics".

The mathematical equation that defines the quantitative relationship between the rate of an enzyme reaction and the substrate concentration is the Michaelis-Menten equation.

$$V = \frac{V_{max}\,[S]}{K_M + [S]} \qquad [2.1]$$

where V is the observed velocity (rate of change) at a given substrate concentration [S]; K_M is the Michaelis constant expressed in units of concentration (mole/liter); and V_{max} is the maximum velocity at saturating concentration of substrate.

Any typical enzyme-reaction involves the formation of an enzyme-substrate complex (ES) which eventually breaks down to form the enzyme E and the product P. Thus

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

where k_1, k_2, k_3, k_4 are the rate constants for each reaction.

The Michaelis Menten constant K_m is defined to be:

$$K_M = \frac{k_2 + k_3}{k_1}$$

The maximum velocity V_{max} is attained when the total enzyme concentration [E] in the system is present as the [ES] complex. In that case, the initial rate V of an enzymatic reaction

is proportional to the concentration of the ES complex we can write

$$V = k_3 [ES]$$

In particular when the total enzyme concentration, [E], is in ES complex form, then the rate of reaction is at its maximum and

$$V_{max} = k_3 [E] \quad .$$

If we have the case that $V = 1/2\ V_{max}$, then from the Michaelis-Menten equation we can easily derive that

$$K_M = [S]$$

which implies that K_M is numerically equal to the substrate concentration at which the velocity is half the maximal velocity.

Furthermore, if we let the concentration of the substrate be very large in Equation [2.1], K_M can be neglected and the rate of change V is equal to the maximal velocity, i.e.

$$V = V_{max}$$

which is an indication of a zero order reaction.

So far, we have considered enzymes that possess independent substrate binding sizes, i.e. the binding of one molecule of substrate has no effect on the structural or electronic changes of the vacant site. If such changes

occur, the velocity curve will not follow the Michaelis-Menten kinetics, and the enzyme will be classified as an "allosteric" or "regulatory" enzyme. These enzymes show signoid kinetics (and S-shaped curve).

Some of the enzymes reacting in the Calvin Cycle are allosteric, [5]. Hence along the pathways of the cycle we will encounter regulatory points. At these points the enzymes act as "valves" regulating the speed of the biochemical reactions.

In the reconsituted system, however, it was possible to obtain steady state conditions [4] , i.e. the "valve" maintains the flux through the regulatory points at constant rate. Hence the kinetic parameters, k_1, ..., k_{22} , of the mathematical model [2.1.10] will be assumed to be constant throughout the cycle. Having discussed the enzymes and the general biochemical aspects of the Calvin Cycle, we shall now introduce our mathematical model.

2.1.10 The Following are the Differential Equations that Describe the kinetics of the Calvin Cycle

K_1 to K_{22} are the parameters that are being determined, H^+ is a constant and its value is 10^{-1} in pH 7.

$$d(RUDP)/dt = +K(20)(ATP)(RU5P) - K(1)(RUDP)$$

$$d(PGA)/dt = +K(3)(GAL)(ADP)(NADP+)(P_i^{2-}) - K(2)(PGA)(ATP)(NADPH)(H^+) + K(1)(RUDP)(CO_2)$$

$$d(ATP)/dt = -K(20)(ATP)(RU5P) + K(3)(GAL)(ADP)(NADP+)(P_i^{2-}) - K(2)(PGA)(ATP)(NADPH)(H^+)$$

$$d(ADP)/dt = +K(20)(ATP)(RU5P) - K(3)(GAL)(ADP)(NADP+)(P_i^{2-})$$
$$+ K(2)(PGA)(ATP)(NADPH)(N^+)$$

$$d(NADPH)dt = +K(3)(GAL)(ADP)(NADP+)(P_i^{2-})$$
$$-K(2)(PGA)(ATP)(NADPH)(H^+)$$

$$d(NADP+)/dt = -K(3)(GAL)(ADP)(NADP+)(P_i^{2-})$$
$$+K(2)(PGA)(ATP)(NADPH)(H^+)$$

$$d(GAL)/dt = +K(6)(FDP) - K(5)(GAL) + K(4)(DHA)$$
$$- K(3)(GAL)(ADP)(NADP+)(P_i^{2-})+K(2)(PGA)(ATP)(NADPH)(H^+)$$
$$+ K(15)(XYL)(R5P) - K(14)(S7P)(GAL)$$
$$+ K(10)(XYL)(ERY) - K(9)(GAL)(F6P) - K(7)(GAL)(DHA)$$

$$d(DHA)/dt = +K(11)(SDP) - K(7)(GAL)(DHA) + K(6)(FDP)$$
$$+K(5)(GAL) - K(7)(DHA) - K(12)(DHA)(ERY)$$

$$d(FDP)/dt = -K(8)(FDP) + K(7)(GAL)(DHA) - K(6)(FDP$$

$$d(f6P)/dt = +K(22)(G6P) - K(21)(F6P) + K(10)(XYL)(ERY)$$
$$-K(9)(GAL)(F6P)$$

$$d(ERY)/dt = -K(12)(DHA)(ERY) + K(11)(SDP)$$
$$-K(10)(XYL)(ERY) + K(9)(GAL)(F6P)$$

$$d(XYL)/dt = +K(18)(RU5P) - K(15)(XYL)(R5P) + K(14)(S7P)(GAL)$$
$$-K(10)(XYL)(ERY) + K(9)(GAL)(F6P) - K(19)(XYL)$$

$$d(SPD)/dt = - K(13)(SDP) + K(12)(DHA)(ERY) - K(11)(SDP)$$

$$d(S7P)/dt = + K(15)(XYL)(R5P) - K(14)(S7P)(GAL) + K(13)(SPD)$$

$$d(R5P)/dt = - K(17)(R5P) + K(16)(RU5P) - K(15)(XYL)(R5P)$$
$$+ K(14)(S7P)(GAL)$$

$$d(RU5P)/dt = K(17)(R5P) - K(16)(RU5P) + K(19)(XYL)$$
$$- K(18)(RU5P) - K(20)(RU5P)(AYP)$$

$$d(G6P)/dt = - K(22)(G6P) + K(21)(F6P)$$

2.2 DESCRIPTION OF THE METHODOLOGY FOR DETERMINING THE KINETIC PARAMETERS

The following assumptions and notations will be used

a) We have a system of non-linear ordinary differential equations

$$\dot{\bar{x}} = f(\bar{x},\bar{k},t) \qquad \bar{x}(0) = c$$

where $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, is the vector of the state variables; $f = (f_1, \ldots, f_n)$ is the n-dimensional vector function describing the system; $\bar{k} = (k_1, \ldots, k_p) \in \mathbb{R}^p$ is the vector of parameters; $t \in \mathbb{R}$ is a real variable representing time. A solution, x, of the system is a function of $\bar{k}$ and t , i.e. $x = x(\bar{k},t)$.

b) The points observed in the experiments will be denoted by $\bar{y}(t) = (y_1(t), \ldots, y_n(t)) \in \mathbb{R}^n$, $t = (t_1, \ldots, t_n)$ These points are subject to an error η , i.e.

$$\bar{y}(t) = \bar{x}(t) + \eta$$

where $y_i(t_j)$ is the observed concentration of the i^{th} intermediate at time $t = t_j$.

The methodology

Let us integrate, numerically, the system of differential equations $\dot{\bar{x}} = f(\bar{x},\bar{k},t)$, $\bar{x}(0) = c$, using an assumed set of parameters, k . Let us define a new function F[k] where F is obtained by summing over the square of the discrepancies between the measured values $\bar{y}(t) = \bar{x}(t) + \eta$ and the

computed values obtained in the above integration, then:

$$F[K] = \sum_{r=1}^{M} \left\{ \bar{y}^T(t_r) - \bar{x}^T(K,t_r) \right\} W_r \left\{ \bar{y}(t_r) - \bar{x}(K,t_r) \right\} \qquad [2.2]$$

where $\bar{y}(t_r)$ is the measured value at time t_r ; $\bar{x}(K,t_r)$ the computed value at time t_r and W_r is a symmetric positive matrix (for which there are different choices, the choice of W_r used in our technique will be discussed in Section 2.4). We are interested in minimizing F[K] over the space of parameters, i.e. we wish to obtain a value, $\hat{\bar{K}}$, such that

$$F[\hat{\bar{K}}] \leq F[\bar{K}] \qquad \forall\ \bar{K} \in \mathbb{R}^p$$

The procedure will consist of several parts:

a) Make an initial guess, $K = (k_1, \ldots, k_p)$, of the parameters $\bar{K} = (\bar{k}_1, \ldots, \bar{k}_p)$.

b) Integrate numerically the non-linear system [2.2], using the parameters values obtained in Step a, and one set of initial conditions $\bar{x}(0) = c^1$ ($c^1 \in \mathbb{R}^n$ representing the first set of initial values).

c) Apply Bremermann's optimizer discussed in Chapter I. To minimize the function F[K] as described previously the new parameters values, say $K^0 = (k_1^0, \ldots, k_r^0)$, obtained after the optimization will be considered a <u>new</u> first approximation to the value $\hat{\bar{K}}$ sought.

d) For the system $\dot{\bar{x}} = f(\bar{x}, k^0, t)$, choose several different initial values $\bar{x}(0) = C^1$, $\bar{x}(0) = C^2$, $\bar{x}(0) = C^3$, $\bar{x}(0) = C^4$, $\bar{x}(0) = C^5$, $\bar{x}(0) = C^6$. (The number of initial values were chosen arbitrarily. For example we can choose different values for some intermediates of the Calvin Cycle by taking twice the initial concentration of one, half the concentration of another and son on.)

e) Integrate the system mentioned in Step d from each of the different initial values.

f) Obtain the experimental data of the Calvin Cycle $\bar{y}^i$, $i = 1,2,3,4,5,6$ using each of the initial values mentioned in Step d.

g) Reform the function F:

$$F[K^0] = \sum_{i=1}^{6} \sum_{r=1}^{M} \left\{ \bar{y}^i(t_r) - (\bar{x}^i(k,t_r))^T \right\} W_r \left\{ \bar{y}^i(t_r) - (\bar{x}^i(k,t_r)) \right\}$$

h) Minimize $F[K^0]$ using Bremermann's optimizer over the space of parameters and obtain the best value of $\hat{\bar{K}}$, $\bar{K}^*$.

i) Determine the accuracy of the parameters in relation to the assumed errors η in the data, using an error analysis technique, a la J. Schwartz and based upon the methods of Rosenbrook and Storey.

The procedure described above has three major numerical tasks:

1. Non-linear optimization
2. Numerical integration
3. Error analysis evaluation

1. In our first Chapter we discussed many aspects of non-linear optimization. We observed that many of the prominent techniques use descent techniques for which, first and second partial derivatives are required. This requirement implicitly assumes that the objective function is in closed form and derivatives can be obtained. But since the objective function F of our procedure is given point-wise rather than in closed form, this characteristic eliminates the use of techniques involving derivatives. Furthermore the minimization of F has to be performed over twenty-two kinetic parameters. Very few algorithms which do not require derivatives have been used in large scale problems, and those that had gave unsatisfactory results [58]. Therefore it was a natural choice to use Bremermann's optimizer since it does not require derivatives of the objective function F. Our comparison, in Chapter I, has shown that the algorithm is for large dimensional problems (50-dimensional) faster convergent than any other algorithm tested.

2. The function F depends on the solution of the system $\dot{\bar{x}} = f(\bar{x},\bar{k},t)$. The solution, $\bar{x} = \bar{x}(\bar{k},t)$, is

obtained by integrating the system numerically since no closed form solution can be obtained.

The choice of the method used to integrate numerically a system of ordinary differential equations depends on the behavior of the system. If a linear approximation of the system has eigenvalues of the same order of magnitude then the system is called a "well-posed system". However, if the ratio of the largest to the smallest eigenvalue is very large, the system is called a "stiff system". It turns out that our system is stiff. The difficulty invited by integrating a stiff system is illustrated by the following example: Let

$$\dot{u} = 11u + 9/2\ v$$
$$\dot{v} = 18u - 11v$$

with initial values $x(0) = x_0 = (2,3) = (u_0, v_0)$ with solution

$$u(t) = 2e^{-2t} + e^{-20t}$$
$$v(t) = 4e^{-2t} - 2e^{-20t}$$

Numerical integration of this system requires a very small stepsize because the first terms in the expression for u and v decay much slower than the second terms. Integration is unstable, i.e. when too large a stepsize is chosen the solution oscillates in a way different from the correct solution. For more explanation see [40].

We have chosen an elaborate integration technique, Gear's Method, since it is accurate and stable over a larger domain than the standard methods. We shall discuss with more details the aspects of integration in Section [2.3].

3. Error analysis of the results is a very important feature of the method. Because we want to find a measure of the error on the best set of parameters found, in relationship to errors in the experimental data.

An inverse problem for which small errors in the data can lead to very large errors in the parameters is called an "ill-conditioned system" (and example of a linear ill-conditioned system is given in Section

Since the inverse problem associated with the Calvin Cycle is potentially ill-conditioned we should be very careful when interpreting our results. Therefore, an error analysis is required to check the validity of the kinetic parameters obtained by our method.

The error analysis performed herein, based upon the methods of Storey and Rosenbrook and implemented and tested for small systems by J. Swartz, Section [2.4] is entirely devoted to a theoretical analysis of this technique.

2.2.1 Characteristics of the Objective Function F

The objective function F will have to satisfy the following criteria:

a) The state variables of the system are non-negative

This is due to the fact that the state variables

X_i, i = 1, ..., n of the mathematical model represent the concentrations of the intermediates of the Calvin Cycle. These concentrations have to be non-negative to be meaningful.

b) The system (modelled by F) satisfies nine linear relations between the parameters.

By considering the thermodynamics of the Calvin Cycle we are able to find a linear relation between the parameters for the forward direction, K_F, and backward direction, K_B, of a particular reversible reaction. (In the schematic representation in Section [2.1.7] the forward reaction was denoted by an arrow pointing to the right, $\longrightarrow$, and the backward reaction by an arrow pointing to the left, $\longleftarrow$.) These linear relations allow us to optimize over fewer parameters -- thirteen, instead of twenty-two. Once we have determined thirteen of them we obtain the other nine from the linear relations.

In the next section we shall discuss thermodynamical properties used to obtain the linear relations between reversible reactions.

2.2.2 Thermodynamic Considerations of the Cycle

Standard free energy of formation:

It is customary to define, for each element at one atmosphere of pressure, an arbitrary standard state in either the solid, liquid, or diatomic gas phase. When this is defined, the standard free energy of formation of a compound

$\Delta G'$, is defined as the difference in free energy between one mole of the compound at one atmosphere pressure at a particular temperature and the total free energies of its elements in their standard states at the same temperature.

Suppose we have a system composed of four species reacting according to:

$$aA + bB \underset{K_F}{\overset{K_B}{\rightleftharpoons}} cC + dD$$

The steady state free energy, ΔG , change is given by

$$1) \qquad \Delta G = \Delta G' - RT \ln \frac{[A]^a[B]^b}{[C]^c[D]^d} ,$$

where $\Delta G'$ is the standard free energy, R is the universal gas constant, T is the temperature in degrees Kelvin, [A], [B], [C], [D], are the concentrations of the species.

When the concentrations of these four compounds are at equilibrium the steady state free energy ΔG is equal to zero. Thus

When $\Delta G = 0$

Equation 1) becomes:

$$\Delta G' = - RT \ln \frac{[C]^c[D]^d}{[A]^a[B]^b}$$

since $\frac{[C]^c[D]^d}{[A]^a[B]^b}$ represents the proportion between the concentrations at equilibrium, this expression has a constant value known as the equilibrium constant, k, of the reaction; i.e.

$$2) \qquad \Delta G' = -RT \ln K .$$

The equilibrium constant K represents the net value of the forward reaction $\xrightarrow{K_F}$ and the backward reaction $\xleftarrow{K_B}$; i.e. $K = \frac{K_F}{K_B}$. These forward and backward reactions are the parameters of the dynamical system.

Therefore, if we know $\Delta G'$ then we can determine a linear relation between K_F and K_B; from 2) we get

$$3) \qquad \text{Exp}\left[-\frac{\Delta G'}{RT}\right] K_B = K_F .$$

The values of $\Delta G'$ were obtained by [5] and are recorded in Table [2.0].

Note: The standard free energy $\Delta G'$ is a constant which does not change under different experimental conditions or in different systems. Therefore, its value is the same inside of the reconsituted system as well as inside of a system with whole leaves or chloroplasts.

Table 2.0

Standard Free Energy of the Calvin Cycle

Reaction	ΔG'	Reaction	ΔG'
A	-8.4 R	H	-3.4 R
B	+4.3 U	I	+0.1 U
C	-1.8 U	J	+0.5 U
D	-5.2 U	K	+0.2 U
E	-3.4 R	L	-5.2 R^1
F	+1.5 U	M	-0.5 U
G	-5.6 U		

Abbreviations A through M: See Section [2.1.7]

R, probable sites of metabolic regulation

R^1, possible sites of " "

U, unregulated and freely "reversible" reactions

The following are the linear relation, between the forward and backward reactions using the parameters in Section

$(.701 \times 10^{-3})$ $K_3 = K_2$

$(.209 \times 10^{2})$ $K_4 = K_5$

$(.651 \times 10^{4})$ $K_6 = K_7$

$(.794 \times 10^{-1})$ $K_{10} = K_9$

$(.128 \quad 10^{5})$ $K_{11} = K_{12}$

$(.844)$ $K_{15} = K_{14}$

$(.429)$ $K_{17} = K_{16}$

$(.713)$ $K_{18} = K_{19}$

$(.232 \times 10^{1})$ $K_{22} = K_{21}$

2.3 NUMERICAL INTEGRATION OF STIFF SYSTEMS

A system of non-linear differential equations is called "stiff" if it is a system whose linearization has eigenvalues of very different magnitudes. For example consider the equation

$$\frac{dy}{dt} = \lambda y \qquad y(0) = 0$$

with solution $y = ce^{\lambda t}$.

For values of $\lambda < 0$, y decays exponentially, i.e. by a factor e^{-1} in time $-1/\lambda$, the last expression is called the time constant. The "stiffness" of the system is due to the existence of greatly differing time constants.

In a system $\bar{y} = F[\bar{y}]$ different components decay at different rates (especially in physical and biological problems). These decay rates are related locally to the partial derivatives $\frac{\partial F}{\partial y}$; if some reactions are slow and others are fast, the fast ones will control the stability of the method of integration, and the slow ones will determine the magnitude of the truncation error.

The mathematical model of the Calvin Cycle is a stiff system of differential equations. Therefore, the integration of our system will be a major step in the process of obtaining the kinetic parameters.

After each successful iterative step of our optimization we obtain a new set of parameters. Using these parameters, we integrate our system; at this point we may

find that with the new set of parameters our system became more stiff. Thus, a good and reliable method of integration is required to obtain the desired result. To this end, initially we tried a 4^{th}-order Runge Kutta Method; but the method is not suitable for our problem because of its computational instability as well as inaccuracy for a reasonably small step size h.

Finally we settled on a combination of two methods: the Adams-Moulton prediction corrector method and Gear's method up to 6^{th} order. This is a package developed and implemented by Clifford Risk at the University of California Berkeley Computer Center.

2.3.1 The Adams Moulton Predictor Corrector Method

For the purposes of clarity consider a single differential equation. These results can be easily extended to systems. Suppose we have the initial value problem

$$\dot{y} = f(x,y),\ y(0) = c \qquad [2.3]$$

where $f(x,y)$ is defined and continuous in the strip $a \leq x \leq b \quad -\infty < y < \infty$, a, b are finite; and the equation satisfies a Lipshitz condition.

This equation satisfies the identity

$$y(x+d) - y(x) = \int_{x}^{x+d} f(t,y(t))dt \qquad [2.4]$$

For any two points x and $x+d$ in $[a,b]$.

The Adams-Moulton method is based on replacing the function $f(x,y(x))$, which is unknown, by an interpolating polynomial, $P(x)$, having the values $f_n = f(x_n,y_n)$ on a set of points x_n where y_n has already been computed.

Let us assume that $x_p = x_0 + ph$, where h is a constant and p is an integer. The first backward difference of a function $f(x)$ at the point $x = x_p$ is defined to be:

$$\nabla f_p = f_p - f_{p-1} \text{ , where } f_p = f(x_p)$$

A q-order backward difference is obtained from:

$$\nabla^q f_p = \nabla(\nabla^{q-1} f_p)$$

also by definition $\nabla^0 f_p = f_p$.

Now, by induction we can obtain the following representation for $\nabla^q f_p$ in terms of the function values f.

$$\nabla^q f_p = \sum_{m=0}^{q} (-1)^m \binom{q}{m} f_{p-m} \quad , \quad q = 0,1,2,3,.....$$

where

$\binom{q}{m}$ is the binomial coefficient and

$$\binom{q}{0} = 1; \quad \binom{q}{m} = \frac{q(q-1)(q-2)\ldots q-m+1}{1,2,m} \qquad m = 1,2,3....$$

If we require to find a representation for the values of f' in terms of differences, $\nabla^q f_p$, we use mathematical induction to prove that

$$f^{(q)}_{p-q} = \sum_{m=0}^{q} (-1)^m \binom{q}{m} \nabla^m f_p \qquad q = 0,1,2,3,\ldots \qquad [2.5]$$

Now, using [2.5], and assuming we have the interpolating points, $x_p, x_{p-1}, \ldots, x_{p-q}$, we can form the interpolating polynomial $P(x)$:

$$P(x) = \sum_{m=0}^{q} (-1)^m \binom{-s}{m} \nabla^m f_p \qquad [2.6]$$

where $s = \dfrac{x-x_p}{h}$, $0 \leq q \leq 6$

Then Equation [2.4] becomes

$$y_p - y_{p-1} = \int_{x_{p-1}}^{p} P(x)dx \qquad [2.7]$$

$$= h \sum_{m=0}^{q} \gamma_m^* \nabla^m f_p$$

where h is a steplength chosen and

$$\gamma_m^* = (-1)^m h^{-1} \int_{x_{p-1}}^{x_p} \binom{-s}{m} dx \qquad [2.8]$$

$$= (-1)^m \int_0^1 \binom{-s}{m} ds$$

Numerical values of γ_m^* are given in Table 2.17.

Table 2.1

m	0	1	2	3	4	5	6
γ_m^*	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{160}$	$-\frac{863}{60480}$

Now assuming that we know an approximation $y_p^{(0)}$ (the subscript ( ) denotes the approximation) of a solution of Equation [2.7] , we calculate $f_p^{(0)} = f(x_p, y_p^{(0)})$ and form the differences

$$\nabla^1 f_p^{(0)} = f_p^{(0)} - f_{p-1} \; ; \quad \nabla^2 f_p^{(0)} = \nabla f_p^{(0)} - \nabla f_{p-1}, \; \ldots.$$

Then, a better approximation $y_p^{(1)}$ is obtained from

$$y_p^{(1)} = y_{p-1} + h \sum_{m=0}^{q} \gamma_m^* \nabla^m f_p^{(0)} \qquad [2.9]$$

Following the same procedure, we can improve the value of $y_p^{(1)}$. In general a sequence $y_p^{(v)}$, v = 0, 1, 2, ... of approximations is obtained recursively from the relation

$$y_p^{(v+1)} = y_{p-1} + h \sum_{m=0}^{q} \gamma_m^* \nabla^m f_p^{(v)} \qquad [2.10]$$

where $f_p^{(v)} = f(x_p, y_p^{(v)})$.

These sequences of numbers $y_p^{(0)}$, $y_p^{(1)}$, $y_p^{(2)}$ ..... will converge to y_p for sufficiently small values of h and this solution is unique (Henrici 1962, pp. 216). In actual numerical computation, there is no need to evaluate [2.7] at each iteration step. We can subtract two consecutive expressions and obtain

$$y_p^{(v+1)} - y_p^{v} = h \sum_{m=j}^{q} \gamma_m^* (\nabla^m f^{(v)} - \nabla^m f_p^{(v-1)} \qquad [2.11]$$

and since $(\nabla^m f_p^v - \nabla^m f_p^{(v-1)}) = f_p^{(v)} - f_p^{(v-1)}$ Equation [2.11] becomes

$$y_p^{(v+1)} - y_p^{v} = h \sum_{m=0}^{q} \gamma_m^* (f_p^{(v)} - f_p^{(v-1)}) \qquad [2.1]$$

and if set $\gamma_0^* + \gamma_1^* + \gamma_2^* + \ldots + \gamma_n^* = \beta_m$ m = 0, 1, 2, ...

Then

$$y_p^{(v+1)} - y_p^{v} = h\,\beta_q\,(f_p^{(v)} - f_p^{(v-1)})$$

The solution y_p of [ ] is now obtained by summing up the terms of the series

$$y_p = y_0^{(1)} + [y_p^{(2)} - y_p^{(1)}] + [y_p^{(3)} - y_p^{(2)}] + \ldots$$

Practically, the computation is stopped when

$$|f^{(v)} - f^{(v-1)}| \leq \varepsilon \qquad \varepsilon \geq 0$$

where ε is a preselected small value. Since in this procedure there is no need to carry differences explicitly using [2.7] we can express differences in terms of function values. Thus, Equation [2.7] becomes:

$$y_p - y_{p-1} = h \sum_{\rho=0}^{q} \beta^*_{q\rho}\, f_{p-\rho}$$

where

$$\beta^*_{q\rho} = (-1)^{\rho} \left\{ \binom{\rho}{\rho} \gamma_\rho + \binom{\rho+1}{\rho} \gamma^*_{\rho+1} + \ldots + \binom{q}{\rho} \gamma^*_q \right\}$$

$q = 0, 1, 2, 3, \ldots, \quad \rho = 0, 1, \ldots, q$.

Some numerical values for $\beta^*_{q\rho}$ are recorded in Table 2.2

Table 2.2 Coefficients of Adams-Moulton Method

ρ	0	1	2	3	4	5
$2\beta^*_{1\rho}$	1	1				
$12\beta^*_{2\rho}$	5	81	-1			
$24\beta^*_{3\rho}$	9	19	-5	1		
$720\beta^*_{4\rho}$	251	646	-264	106	-19	
$1440\beta^*_{5\rho}$	475	1427	-798	482	-173	27

Note: A formula which provides a first approximation for $y^{(0)}$, is called a predictor formula and a formula like [2.10] is called a corrector formula. Furthermore since the new value y_p is obtained by solving a non-linear equation that involves the function f the method is called an implicit method.

2.3.2 Gears Method

In this section we shall discuss some characteristics of Gear's method. Further details can be found in [40]

Let h be a fixed positive real number and λ complex number with negative real part. Then:

Definition: A multistep method of integration is absolutely stable for those values of $h\lambda$ such that roots of the characteristic equation

$$P(\zeta) + h\lambda\, Q(\zeta) = 0 \qquad [2.13]$$

are ≤ 1 in absolute value. Where

$$P(\zeta) = \sum_{i=0}^{k} \alpha_i\, \zeta^{k-i} \text{ and } Q(\zeta) = \sum_{i=0}^{k} \beta_i\, \zeta^{k-i} \qquad [2.14]$$

Definition (Gear 1964), a method is stiffly stable if in the region $R_1 = \{h\lambda |\ Re(h\lambda) \leq D < 0\}$ it is absolutely stable, and in the region $R_2 = \{h\lambda | D < Re(h\lambda) < \alpha,\ \alpha > 0$, and $|1M(h\lambda)| < \theta\}$ it is accurate.

Gears method has all the characteristic roots, ζ, equal to zero at $\lambda h = \infty$ only. The k-step methods of order k with $Q(\zeta) = \zeta^k$ are shown to be stiffly stable for $k \leq$, for some D, α and θ. To obtain this result it is required to compute $P(\zeta)$ from $Q(\zeta)$ so as to get an order k-method

For a stiffly stable method, $Q(\zeta)$ has to be a polynomial of degree at least equal to the degree of $P(\zeta)$. Otherwise, one root at $h\lambda = \infty$ is ∞. This property implies that stiffly methods are implicit, therefore it is necessary to solve a corrector equation.

Gears method calculates the solution of a stiff system according to the following corrector equation

$$y_p = \sum_{i=1}^{k} \alpha_i \, y_{p-i} + h\beta_0 f_p \qquad [2.15]$$

The values of α_i and β_0 for the method are given in Table 2.3.

If we consider the characteristic equation $P(\zeta) + \lambda h \, Q(\zeta) = 0$, the locus λh with roots $\zeta = e^{i\theta}$ of modulus 1 is given by a continuous closed curve:

$$\lambda h = \frac{P(\zeta)}{Q(\zeta)} = \frac{P(e^{i\theta})}{Q(e^{i\theta})} \qquad 0 \leq \theta \leq 2\pi \qquad [2.16]$$

These curves will describe the region of absolute stability of the method.

Figure [ ] Region of absolute stability for stiffly stable methods of order one through six.

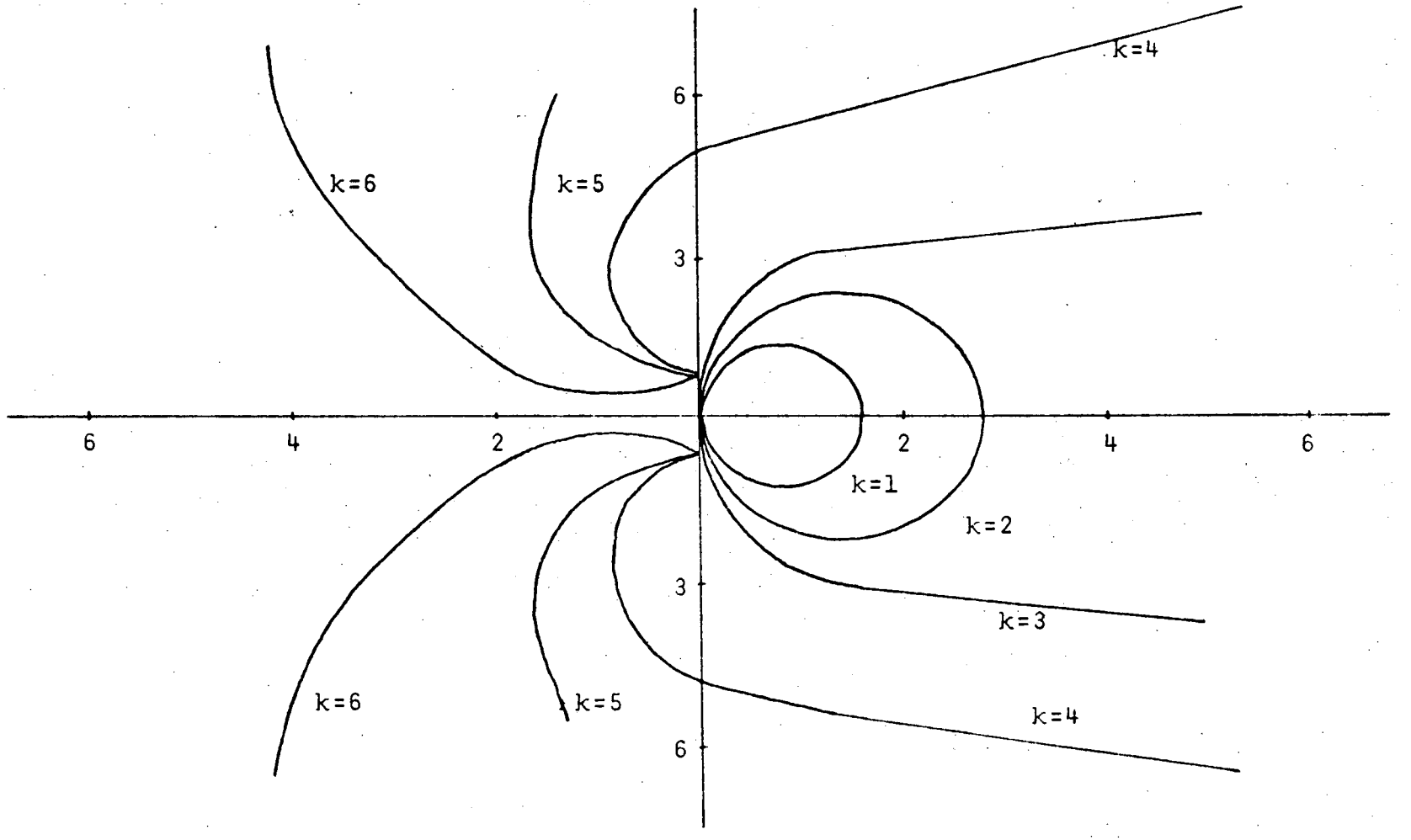


Table 2.3

Coefficients of Gear's Stiffly Stable Method

k	1	2	3	4	5	6
β_0	*	2/3	6/11	12/25	60/137	60/147
α_1	*	4/3	18/11	48/25	300/137	360/147
α_2	*	-1/3	-9/11	-36/25	-300/137	-450/147
α_3	*		2/11	16/25	200/137	400/147
α_4	*			-3/25	-75/137	-225/147
α_5	*				12/137	72/147
α_6	*					-10/147

*For $k = 1$ we obtain the backward Euler method i.e.

$$y_p = y_{p-1} + hf(y_p) + O(h^2)$$

2.4 ERROR ANALYSIS OF THE KINETIC PARAMETERS

As earlier studies have shown, J. Swartz [83] it is necessary to estimate the expected error of the kinetic parameters from the measurement errors in the data. To this end we will discuss the error analysis technique, a la J. Swartz, which is based upon Storey and Rosenbrook [75],

Note: Although we followed the main ideas of Storey and Rosenbrook, gaps in their presentation made it necessary to give an independent derivation of the error analysis.

Error Analysis

The purpose of the error analysis is to verify the accuracy of the parameters found after applying the optimization and integration procedure. By having a measure of the error in the parameters we shall obtain an insight into the following points:

1) Consistency of the model and the data
2) The estimated error of the parameters due to errors in the data.

To this end let us consider a dynamical system of the form:

$$\dot{\bar{x}} = f(\bar{x},\bar{k},t) \quad x(0) = c \qquad [2.17]$$

The following assumptions will be used:

a) The parameters values, $\overset{*}{\bar{k}}$, obtained at the end of the optimization procedure are in reasonably good agreement with actual values, $\overset{=}{k}$ of $\bar{k}$.

b) Let $\bar{x} = \bar{x}(t,\hat{\bar{k}})$ denote a solution of [2.17] , where $\hat{\bar{k}}$ is a particular value of $\bar{k}$. Denote by $\bar{z}(\alpha)$ a solution of [2.17] depending on $\alpha \in \mathbb{R}^p$ such that $\bar{z}(\alpha) = \bar{x}(t,\hat{\bar{k}}+\alpha)$. Denote by $\varepsilon(\alpha)$ the error in $\bar{x}$ which results from an error, α , in $\bar{k}$, i.e.

$$\begin{aligned}\varepsilon(\alpha) &= \bar{z} - \bar{x} \\ &= \bar{x}(t,\hat{\bar{k}}+\alpha) - \bar{x}(t,\hat{\bar{k}})\end{aligned}$$

c) A first order Taylor expansion of $f = f(\bar{x},\bar{k},t)$ around the point $(\hat{\bar{x}},\hat{\bar{k}},t)$ is

$$f(\hat{\bar{x}}+\bar{\varepsilon},\ \hat{\bar{k}}+\bar{\alpha},\ t+\Delta t) = f(\hat{\bar{x}},\hat{\bar{k}},t) + f_{\bar{x}}\,\varepsilon + f_{\bar{k}}\,\alpha + f_t\Delta t + 0\ ((\varepsilon+\alpha+\Delta t)^2)$$

similarly for $\bar{x} = \bar{x}(\bar{k},t)$ at $(\hat{\bar{k}},t)$

$$\bar{x}(\hat{\bar{k}}+\alpha,t+\Delta t) = \bar{x}(\hat{\bar{k}},t) + \bar{x}_{\bar{k}}\,\alpha + \bar{x}_t\Delta t + 0\ ((\alpha+\Delta t)^2)$$

where

$$f_{\bar{x}} = \frac{\partial f_i}{\partial x_j}\ ;\ f_{\bar{k}} = \frac{\partial f_i}{\partial \bar{k}_j}\ ;\ f_t = \frac{\partial f_i}{\partial t_j}\ ;\ \bar{x}_{\bar{k}} = \frac{\partial x_i}{\partial \bar{k}_j}\ ;\ \bar{x}_t = \frac{\partial x_i}{\partial t_j}$$

In all that follows we are only interested in the case when $\Delta t = 0$ and only considering the first order terms in ε and α .

$\|\bar{\bar{k}} - \overset{*}{\bar{k}}\| \leq \alpha$ for some sufficiently small $\alpha \in \mathbb{R}^+$

Now, consider a system on non-linear ordinary differential equations

$$\dot{\bar{x}} = f(\bar{x},\bar{k},t) \ , \quad \bar{x}(0) = c$$

Let $\bar{x} = \bar{x}(t,\bar{k})$ be a general solution of this system, then varying $\bar{k}$ slightly would only change a particular solution slightly; i.e. we will obtain a solution

$$\bar{x} + \bar{\varepsilon} = \bar{x}(t,\bar{k}+\alpha)$$

where $\bar{\varepsilon}$ is the perturbation of $\bar{x}$ due to small changes α in $\bar{k}$. This is equivalent to saying that the general solution of [2.17] is a continuous function of the parameter $\bar{k}$. This is certainly true locally since

Theorem. Let Equations [2.17] be given where $f(x,k,t)$ and $\frac{f_i}{x_i}$ are defined and continuous in domain, $B \in \mathbb{R}^{n_t p+1}$. If $(\bar{x}_o,t_0,\vec{k})$ belongs to B ($\vec{k}$ is a value of $\bar{k}$), then there exist positive numbers r and q such that:

1) Given $\vec{k}$ such that $\|\vec{k} - \vec{k}\| < q$, there exist a unique solution $x = x(t,\vec{k})$ of [2.17] defined for $|t - t_0| \leq r$ and satisfying $x(t_0,\vec{k}) = x_0$
2) The solution is a continuous function of t and $\bar{k}$.

A proof of this theorem can be found in [77].

Now that we established that small perturbations in $\bar{k}$ will generate small errors in $\bar{x}$ we are interested in finding:

a) A time relation describing the dependence of $\bar{x}$ upon variations α of $\bar{k}$ around $\bar{\bar{k}}$, and

b) a relation describing the dependence of α upon variations η around $\bar{x}(t)$.

Therefore, given a solution $\bar{x}(t,\bar{k}+\alpha)$ a first order approximation is given by:

$$\bar{x}(t,\bar{k}+\alpha) = \bar{x}(t,\bar{k}) + \alpha \frac{\partial \bar{\bar{x}}}{\partial k}(t,\bar{k}) \qquad [2.18]$$

set $\bar{x}(t,\bar{k}+\,) - \bar{x}(t,\bar{k}) = \varepsilon(t)$, $\quad \frac{\partial \bar{x}}{\partial \bar{k}}(t,\bar{k}) = D(t)$ and

$$\alpha \frac{\partial \bar{x}}{\partial \bar{k}}(t,\bar{k}) = \varepsilon(t)$$

$$\alpha \frac{\partial \dot{\bar{x}}}{\partial k}(t,\bar{k}) = \dot{\varepsilon}(t)$$

$$\mathbb{D}\,\dot{\bar{x}}(t,\bar{k}) = \dot{\bar{\varepsilon}}(t) + \gamma \frac{\partial \dot{\bar{x}}}{\partial t}, \quad \text{for some } \gamma \in \mathbb{R}^1, \qquad [2.19]$$

where $\mathbb{D}$ denotes the total derivatives. Then Equation [2.18] becomes:

$$\varepsilon(t) = D(t)\alpha \qquad [2.20]$$

for $t = t_r$ $\quad \varepsilon(t_r) = D_r\alpha$, where $D_r = D(t_r) = \frac{\partial \bar{x}}{\partial \bar{k}}(t_r,\bar{k})$

where D_r is a $n \times p$ matrix which gives the dependence of $\bar{x}(t,\bar{k})$ upon variations α of $\bar{k}$ around $\bar{\bar{k}}$. Since we wish to find the changes of D with time let us introduce the errors ε and α into Equation [2.17] and obtain:

$$\dot{\bar{x}} + \dot{\bar{\varepsilon}} = f(t,\bar{x}+\varepsilon\ ,\ \bar{k}+\alpha) \qquad \begin{matrix} \bar{x}(0) = c \\ \varepsilon(0) = 0 \end{matrix} \qquad [2.21]$$

By keeping t fixed and expanding this expression in a Taylor series using only first terms and considering [2.19] we get

$$\dot{\bar{x}} + \dot{\bar{\varepsilon}} = f(t_0,\bar{x},\bar{k}) + A\varepsilon + B\alpha \qquad [2.22]$$

and

$$\dot{\bar{\varepsilon}} = A\varepsilon + B\alpha \qquad [2.23]$$

where

$$A = (A_{ij}) = \left| \frac{\partial f_i}{\partial \bar{x}_j} (\bar{x},\hat{\bar{k}},t_0) \right|$$

$$B = (B_{ij}) = \left| \frac{\partial f_i}{\partial \bar{k}_j} (\bar{x},\hat{\bar{k}},t_0) \right|$$

Both matrices are functions of t but not of ε or α, also A is $n \times n$ and B is $n \times p$. Since the above calculations do not depend on t_0, differentiate Equation [2.20] with respect to time and substitute the result into Equation [2.23], then:

$$\dot{D}\alpha = AD\alpha + B\alpha \qquad [2.24]$$

since α is a scalar, cancelling it we obtain:

$$\dot{D}(t) = A(t)\cdot D(t) + B(t) \qquad [2.25]$$
$$D(0) = 0$$

The $n\times p$ matrix $D(t)$ will contain all the information pertaining to the dependence of the solution $\bar{x}(t)$ on errors in the parameters.

Equation [2.25] is called "the variational equation" of the system [2.17] , for further properties of this equation, see [44], [50], [77].

At this point we are ready to determine the relation between the error η in the observation (data) and α in the parameters. Consider the function F described in Section [2.2] . By minimizing F over the space of parameters $\bar{k}$ we will obtain a set of parameters which is an estimate of the value $\bar{\bar{k}}$ of $\bar{k}$. Let $\bar{\bar{k}}+\alpha$ be such a set for which $F[\bar{k}+\alpha]$ is a minimum. Then Equation [2.2] becomes

$$F[\bar{\bar{k}}+\alpha] = \sum_{r=1}^{M} \left\{ \bar{y}^T(t_r) - \bar{x}^T(\bar{\bar{k}}+\alpha,t_r) \right\} W_r \left\{ \bar{y}(t_r) - \bar{x}(k+\alpha,t_r) \right\} \qquad [2.26]$$

and at the minimum $\bar{\bar{k}}+\alpha$

$$\frac{\partial F}{\partial \alpha}[\bar{\bar{k}}+\alpha] = 0$$

or

$$\frac{\partial}{\partial \alpha_i} \sum_{r=1}^{M} \left\{ \bar{y}^T(t_r) - \bar{x}^T(\bar{\bar{k}}+\alpha, t_r) \right\} \; W_r \left\{ \bar{y}(t_r) - \bar{x}(\bar{\bar{k}}+\alpha, t_r) \right\} = 0$$

or

$$\frac{\partial}{\partial \alpha_i} \sum_{r=1}^{M} \left\{ \bar{x}^T(t_r) + \eta_r^T - (\bar{x}^T(t_r)+\epsilon_r^T) \right\} W_r \left\{ \bar{x}^T(t_r + \eta - (\bar{x}^T(t_r)+\epsilon) \right\} = 0$$

or

$$\frac{\partial}{\partial \alpha_i} \sum_{r=1}^{M} \left\{ \eta_r^T - \alpha^T D_r^T \right\} \; W_r \; \left\{ \eta_r - D_r \alpha \right\} = 0$$

$$= \frac{\partial}{\partial \alpha_i} \sum_{r=1}^{M} \left\{ \eta_r^T W_r \eta_r - \eta_r^T W_r D_r \alpha - \alpha^T D_r^T W_r \eta_r + \alpha^T D_r^T W_r D_r \alpha \right\}$$

This is a scalar equation. Hence when taking the partial derivatives with respect to the i^{th} component of the vector we get

$$\sum_{r=1}^{M} - \left[\eta^T W_r D_r \right]_i - \left[D_r^T W_r \eta_r \right]_i + \left[\alpha^T D_r^T W_r D_r \right]_i + \left[D_r W_r D_r \alpha \right]_i = 0$$

Since the i^{th} component of a vector and transpose are the same we get that

$$\sum_{r=1}^{M} - 2\left[D_r^T W_r \eta_r\right]_i + 2\left[D_r^T W_r \alpha\right]_i = 0$$

or

$$\sum_{r=1}^{M} \left[D_r^T W_r \eta_r\right] = \sum_{r=1}^{M} \left[D_r^T W_r D_r \alpha\right]$$

Let $H = \sum_{r=1}^{M} \left[D_r^T W_r D_r\right]$, then

$$H\alpha = \sum_{r=1}^{M} \left[D_r^T W_r \eta_r\right] \qquad [2.27]$$

When the symmetric matrix H is non-singular, Equation [2.27] gives the relation between η and α sought.

At this point we are ready to calculate the expected error α due to error η in the observations (data).

It is reasonable to assume that the error vectors $\{\eta_i\}$ corresponding to different sets of measurements will be statistically independent, i.e.

$$E\left[\eta_i \eta_j^T\right] = 0 \qquad i \neq j \qquad [2.28]$$

where E denotes the "expectation values" of η_i , and is the operation of taking the average over a large number

of similar experiments. In contrast the individual components of each vector η are not assumed to be independent. Then the expectation values are represented by a matrix called the covariance matrix i.e.

$$E\left[\eta_i \eta_i^T\right] = M_i \qquad [2.29]$$

where M is the covariance matrix.

The expectation value of α is zero, (since its components are independent) therefore we are interested in finding the covariance matrix of the components of α, that is, the expected value, p, of α. Assuming that the expectation values are known, it follows from Equation [2.28] that:

$$E\left[H\alpha\alpha^T H\right] = E\left[\sum_{r=1}^{M} \sum_{r=1}^{M} D_r^T W_r \eta_r \eta_r^T W_r D_r\right] \qquad [2.30]$$

Let P be the covariance matrix representing the expected values of α i.e.

$$E\left[\alpha_i \alpha_i^T\right] = P_{ii} \quad \text{and} \quad E\left[\alpha_i \alpha_j^T\right] = 0 \text{ , } P_{ij}=0, i \neq j \qquad i \neq 1$$

Then since the expectation of each component in Equation [2.30] is well defined we obtain:

$$HPH = \sum_{r=1}^{M} D_r^T W_r M_r W_r D_r \qquad [2.31]$$

If we choose $W_r = (M_r)^{-1}$ to be the weighting matrix, for each r, W_r has the following representation

$$W_r = \begin{bmatrix} \frac{1}{\sigma_{1r}^2} & 0 & \cdots & 0 \\ \vdots & & & \\ 0 & \cdots & \cdots & \frac{1}{\sigma_{n_r}^2} \end{bmatrix}$$

where σ_i^2 = the variance of $\alpha_i(t)$

and Equation [2.31] becomes

$$HPH = \sum_{r=1}^{M} D_r^T W_r D_r \qquad [2.32]$$

The right side of this equation is defined equal to H thus

$$HPH = H \qquad [2.33]$$

If H is non-singular then

$$P = H^{-1} \qquad [2.34]$$

which determines P.

From Equation [2.27] we observe that α is a linear function of η_r. Its probability density will be Gaussian [66], and is given by

$$\frac{1}{(2\pi)^{p/2}\sqrt{|P|}} \exp\left|-\frac{1}{2}\alpha^T P^{-1}\alpha\right| \qquad [2.35]$$

and, the expected variance σ_ℓ^2 for the parameters in the direction b is given by

$$\sigma_\ell^2 = \bar{b}^T P \bar{b} \qquad [2.36]$$

where $\bar{b}$ is a unit p-vector in the parameters space. By By choosing $\bar{b}$ parallel to the direction of one of the parameters, i.e. $b_i = 1$, $b_j = 0$, $i \neq j$ we obtain the variance of this parameter, i.e.

$$\sigma_i^2 = P_{ii} \qquad [2.37]$$

This result is of central importance because it determines the relationship between a measurement of experimental error in the data and a measurement of error in the calculated parameters. Furthermore it indicates how well the data fits the model.

2.4.1 Implementation of the Error Analysis Technique on the Dynamical Equation of the Calvin Cycle

The main difficulty in implementing the error analysis is the calculation of Equation [2.25] , i.e.

$$\dot{D} = AD + B$$

in spite of the nice analytical represetnation, this calculation can run into a major difficulty, which is, the calculation of the matrices

$$A = \frac{\partial f_i}{\partial \bar{x}_j}(\bar{x}_i, \bar{k}, t) \qquad B = \frac{\partial f_i}{\partial \bar{k}_j}(\bar{x}, \bar{k}, t)$$

If we try to obtain the partial derivatives by formally differentiating, the task is almost humanly impossible, since the dimension of A is 17×17 and B is 17×22 the chances for human mistakes while taking derivatives is very high. This is especially true since each entry of these matrices has many terms. Furthermore forming the product, A·B , requires over 7000 multiplications of the entries. Since all entries have five terms on the average, the 35000 multiplication required makes the existence of a mistake a virtual certainty.

This computational problem is simply a result of the size of the system of equations. For a smaller problem analytic differentiation can be performed by hand and checked efficiently.

We have overcomed this problem by employing the computer itself to perform analytical differentiation using a new experimental computer language called ALTRAN (Algebra Translator).

ALTRAN is a language and system for performing symbolic computations on algebraic data. The basic capability of the language is to perform operations on rational expressions in one or more indeterminants. The system is designed to handle very large problems involving such data, with considerable efficiency.

Operations on integers rational and real (floating point) data are also included together with computer procedures for dealing with truncated power series and systems of linear equations.

The ALTRAN language is still an experimental language in its early development and only few institutions have it in operational form in their computer installations.

The usefullness of the ALTRAN language can be stressed by considering the following aspect in the calculation of the matrix $\dot{D}$. The matrix D is equal to:

$$\dot{D}(t) = A(t)\, D(t) + B(t)$$

where $A = \frac{\partial f_i}{\partial x_j}$; $B = \frac{\partial f_i}{\partial k_j}$, $D = \frac{\partial x_i}{\partial k_j}$

A is (17×17), B is (17×22) , and D is (17×22) .

Now let us consider what is the probability of making no mistakes in the actual computation of these partial derivatives, when it is not done by a computer.

Let p be the probability of making a mistake when taking the partial derivatives of a term in an entry of matrix B, then $1-p$ is the probability of not making a mistake. Since each time we take a derivative we can make a mistake, we can consider that each time is an independent event.

Assume that each entry of the matrix B has an average of five terms then we get that the probability of making no mistakes is:

$$(1-p)^{(17\times 22)\times 5} \simeq (1-p)^{2000}$$

if $p = \frac{1}{100}$ i.e. one out of 100, then

$$\left|(1-p)^{\frac{1}{p}}\right|^{20} \approx e^{-2000p}$$

which implies that it is almost impossible to compute it by hand.without having a mistake unless $p < 10^{-4}$. By a similar computation and assumptions for the matrix A we get that

$$(1-p)^{(17\times 17)\times 5} = (1-p)^{1445} \approx (1-p)^{1500}$$

or

$$\left|(1-p)^{\frac{1}{p}}\right| \approx e^{-1500p}$$

If we note that our matrix $\dot{D}$ is the result of the computation of matrix A and B then we can conclude that if we have a large system and we try to compute the matrix $\dot{D}$ by hand, then we almost certainly will make a mistake in the computation since for most people the error rate is much larger than 10^{-4} [46], [87].

To avoid an almost certain error, we obtained the matrix $\dot{D}$, by writing and implementing a computer program written in the ALTRAN language. The matrix $\dot{D}$ so obtained has 82 nonzero entries. Each entry contains on the average, about 15 algebraic terms.

From our experience in deriving the matrix $\dot{D}$, we can conclude that the error analysis could not be implemented unless we used the ALTRAN system. It is interesting to note that in our initial attempts in trying to obtain the matrix $\dot{D}$, by performing formal differentiation by hand, we found that we always generated mistakes. Thus the reader should be aware that the use of ALTRAN as a new tool is of utmost importance.

2.5 ILL-CONDITIONED SYSTEM OF EQUATIONS

In our previous section we discussed the theoretical aspects of an error analysis. Our main result was a measure of the expected error in the kinetic parameters due to error in the data. The variance, σ^2, was represented by a matrix P, the matrix equation $\dot{D} = AD + B$ required the calculation of the matrix A and B. The nature of these matrices was

never discussed. Thus it is of interest to determine how errors in the data affect such matrices. To this end we shall carefully analyse the behavior of the solution $\bar{x}$, of a typical linear system $M\bar{x} = \bar{b}$, M is an invertible $n \times n$ matrix, when variations are made in the value of M and $\bar{b}$. When large changes in $\bar{x}$ result from small changes in M and $\bar{b}$ we speak of the matrix M as "ill-conditioned" or the system as "ill conditioned". In this section we present standard results on this topic. We also give a geometric interpretation of the results by applying them to the matrix P.

Sensitivity of a Linear System to Perturbations

Let $M\bar{x} = b$ be a linear system, where M is a nonsingular matrix of order n, $\bar{b}$ is an n-dimensional vector and $\bar{x}$ is the n-dimensional solution of the system.

We are interested in finding some kind of measure of the error affecting the solution $\bar{x}$ resulting from errors in the vector $\bar{b}$, and the dependence of errors in $\bar{x}$ on errors in M.

The "norm" of a matrix M; $\|M\|$, is a number which measures the magnitude of a matrix. The norm satisfies the following properties:

1) $\|M\| \geq 0$, $\|M\| = 0$ iff $A = 0$

2) $\|cM\| = |c| \, \|M\|$ where $c \in \mathbb{R}^1$

3) $\|M+Q\| \leq \|M\| + \|Q\|$

4) $\|MQ\| \leq \|M\| \cdot \|Q\|$

There exist different ways of defining M which satisfy the above conditions. In this work we will mention two kinds of norms:

1) $\|M\| = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} m_{ij}^2 \right)^{1/2}$ where m_{ij} are the entries of m. This norm is called the Euclidean norm.

2) $\|M\| = \max_i [\lambda_i(MM^T)]^{1/2}$ where $\lambda_i(MM^T)$ denotes an eigenvalue of MM^T, and M^T is the transpose of M. This norm is called the spectral norm.

These two norms are defined for any $m \times n$ matrix.

Definition. Let $\|M\|$, $\|M^{-1}\|$ denote the spectral norm of M and M^{-1} respectively. The condition number of a matrix M is the scalar K such that $K = \|M\| \cdot \|M^{-1}\|$. We shall denote the condition number of M by $K(M)$.

Note. The norm of a vector is defined in the Euclidean sense as $\|\bar{x}\| = (x^T x)^{1/2} = |\bar{x}|$

Proposition. Let $\bar{x}$ represent a small error in $\bar{x}$ due to a small error $\bar{b}$ then:

$$\frac{\|\Delta\bar{x}\|}{\|\bar{x}\|} \leq K(M) \frac{\|\Delta\bar{b}\|}{\|\bar{b}\|} , \qquad [2.37a]$$

where $\| \|$ is the spectral norm.

Proof. Since $M(\bar{x}+\Delta\bar{x}) = (\bar{b}+\Delta\bar{b})$, and $M\bar{x} = \bar{b}$ by definition then

$$\Delta\bar{x} = M^{-1}\Delta\bar{b} \qquad [2.38]$$

taking norms on both sides

$$\|\Delta\bar{x}\| \leq \|M^{-1}\| \, \|\bar{b}\| \qquad [2.39]$$

we also have

$$\|\bar{b}\| \leq \|M\| \, \|\bar{x}\|$$

Now multiplying Expression [2.38] by Expression [2.39] we get

$$\|\Delta\bar{x}\| \cdot \|\bar{b}\| \leq \|M^{-1}\| \cdot \|M\| \cdot \|\Delta\bar{b}\| \cdot \|\bar{x}\|$$

and dividing by $\|b\| \cdot \|x\|$ we obtain our result

$$\frac{\|\Delta\bar{x}\|}{\|x\|} \leq \|M^{-1}\| \cdot \|M\| \cdot \frac{\|\Delta\bar{b}\|}{\|\bar{b}\|}$$

Q.E.D.

Notice that this result depends on the choice of the norm. For the spectral norm the condition number k(M) of a matrix M turns out to be given by:

$$\|M\| \cdot \|M^{-1}\| = \sqrt{\frac{\lambda_{max}AA^T}{\lambda_{min}AA^T}}$$

The condition number $k(M)$ is a measure of the maximum distortion which the linear transformation M makes on the unit sphere. The expression $\frac{\|\Delta\bar{b}\|}{\|\bar{b}\|}$ represents the relative error in the vector $\bar{b}$, and $\frac{\|\Delta\bar{x}\|}{\|\bar{x}\|}$ is the relative error in the vector solution $\bar{x}$ due to error in the vector $\bar{b}$.

So far we considered the matrix M as if it were known precisely. We now look at the system $M\bar{x} = \bar{b}$ when an error ΔM in M and b is also known precisely. Again as in Proposition 1 we are interested in determining how ΔM affects the solution vector $\bar{x}$.

Let $(\bar{x}+\Delta\bar{x}) = (M+\Delta M)^{-1}\bar{b}$ be given

Proposition. If M is the matrix representing the error in the matrix M and $\bar{x}$ results from this error then

$$\frac{\|\Delta\bar{x}\|}{\|\bar{x}+\Delta\bar{x}\|} \leq K(M)\frac{\|\Delta M\|}{\|M\|}$$

Proof. Clearly $\Delta\bar{x} = (M + \Delta M)^{-1}\bar{b} - M^{-1}\bar{b}$

or $\Delta\bar{x} = [(M + \Delta M)^{-1} - M^{-1}]\bar{b}$

This equality can be rewritten as

$$\Delta\bar{x} = \{M^{-1}[M-(M+\Delta M)]\ (M+\Delta M)^{-1}\}\ \bar{b}$$

$$= M^{-1}\Delta M(M+\Delta M)^{-1}\ \bar{b} \qquad [2.40]$$

Now since $(\bar{x}+\Delta\bar{x}) = (M+\Delta M)^{-1}\,\bar{b}$ Equation [2.40] becomes

$$\Delta x = - M^{-1}\Delta M(\bar{x}+\Delta\bar{x})$$

Taking norms on both sides we get

$$\|\Delta\bar{x}\| \leq \|M^{-1}\| \cdot \|\Delta M\| \cdot \|\bar{x}+\Delta\bar{x}\|$$

or

$$\frac{\|\Delta\bar{x}\|}{\|\bar{x}+\Delta\bar{x}\|} \leq \|M\|\ \|M^{-1}\|\ \frac{\|\Delta M\|}{\|M\|} \qquad [2.41]$$

which is the desired result.

Q.E.D.

Expression [2-41] means that the relative error in the vector solution $\bar{x}$ is bounded by the relative error in the matrix M times the condition number of $K(M)$.

2.5.1 Ill-Condition: An Example

After discussing sensitivity properties of a matrix M consider an example of an ill-conditioned linear system: M is a real symmetric matrix such that:

$$M = \begin{bmatrix} 1 & .99 \\ .99 & .98 \end{bmatrix}$$

The eigenvalues λ_1 , λ_2 , of M are found by taking the determinant of M and solving a quadratic equation, i.e.

$$\begin{bmatrix} 1-\lambda_1 & .99 \\ .99 & .98-\lambda_2 \end{bmatrix} = (1-\lambda_1)(.98-\lambda_2) - .99^2 = 0$$

We find that $\lambda_1 = -.00005$ and $\lambda_2 = 1.98005$. The condition number of M is given by

$$K(M) = \sqrt{\frac{\lambda_{max}MM^T}{\lambda_{min}MM^T}}$$

$$= \sqrt{\frac{(1.9005)^2}{(-.00005)^2}} = 39.600$$

So K(M) is a very large number. Now consider the problem of finding the point of intersection of two lines:

$$x_1 + .99x_2 = 1.99$$

$$.99x_1 + .98x_2 = 1.97$$

The solution to this system is $x_1 = 1$ $x_2 = 1$ but for $x_1 = 3$ and $x_2 = -1.0203$ we obtain

$$x_1 + .99x_2 = 1.989903$$

$$.99x_1 + .98x_2 = 1.970106$$

Therefore a change $\Delta\bar{b} = \begin{bmatrix} -.000097 \\ \\ +.000106 \end{bmatrix}$

can give $\bar{x} = \begin{bmatrix} +2.0000 \\ \\ -2.0203 \end{bmatrix}$

Now, we can calculate $\|\Delta\bar{x}\|$ and obtain $\|\Delta\bar{x}\| \approx 2\sqrt{2}$

Similarly $\|\bar{x}\| = \sqrt{2}$, thus $\frac{\|\Delta\bar{x}\|}{\|\bar{x}\|} \approx 2$

since $\|\Delta\bar{b}\| \approx 10^{-4}\sqrt{2}$ and $\|\bar{b}\| \approx 2\sqrt{2}$, then $\frac{\|\Delta\bar{b}\|}{\|\bar{b}\|} \approx \frac{10^{-4}}{2}$

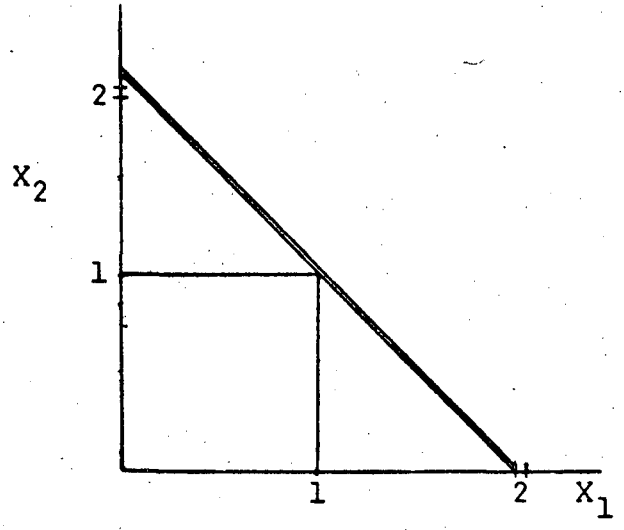
Hence, using

$$\frac{\|\Delta\bar{x}\|}{\|\bar{x}\|} \leq K(M) \frac{\|\Delta\bar{b}\|}{\|\bar{b}\|} \quad \text{we see that}$$

$$2 \leq K(M) \frac{10^{-4}}{2}.$$

For this to be true $K(M)$ has to be at least 40.000 since we found that $K(M)$ 39,600, we found approximately the worst case. Therefore if $\bar{b}$ is known to have an error of above .00001, then the vector x can only be known to within two units of each of its components. This system is an example of a very ill-conditioned problem. Geometrically

this means that the point of intersection of these two lines is hard to determine, i.e. the lines are practically coincident.



2.5.2 Geometric Interpretation of the Condition Number of a n×n matrix A

The condition number is a measure of the maximum distortion which the linear transformation with matrix A makes on the unit sphere.

Using the spectral norm, assume that A is a real symmetric matrix and the condition number K(A) is one. This implies that we have an undistorted sphere, and A and A^{-1} stretch all directions by the same factor. Since the eigenvalues, λ_i of A are all equal in magnitude, i.e. $|\lambda_1| = |\lambda_2| = \ldots = |\lambda_n|$ it will be impossible for A^{-1} to stretch $\Delta\bar{b}$ more than $\bar{b}$ itself, in other words, the relative uncertainty of the vector solution $\bar{x}$ is the same as that of $\bar{b}$. However, when K(A) is 10^{10} A^{-1} stretches one direction 10^{10} times as much as another

direction, in this case if $\bar{b}$ gets the short stretch, $\Delta\bar{b}$ may get the long one, and $\frac{\|\Delta x\|}{\|x\|}$ would be 10^{10} times $\frac{\|\Delta b\|}{\|b\|}$ and our original unit sphere becomes distorted into a hyperellipsoid. When the deformed sphere is very elongated the implication is that our matrix is nearly singular. The closer the ellipsoid is to a sphere the farther it is from singular and the eigenvalues (associated with the coordinates of this ellipsoid) would be similar in magnitude.

In our discussion of the error analysis our main result,

$$\sigma^2_{\bar{b}} = \bar{b}^T P \bar{b} \qquad [2.42]$$

represents the variance of the parameters K_i in the $\bar{b}$ direction. P is a real symmetric matrix, hence its eigenvalues are real, and its eigenvectors are real and orthogonal.

Now consider the hyperellipsoid Z having the eigenvectors of P as the directions of the principal axes and the length of each principal semidiameter is equal to the square-root of the corresponding eigenvalues.

In any direction $\bar{b}$ the distance from the origin to the hyperellipsoid is $\bar{b}^T P \bar{b} = \sigma_b$. If P is already a diagonalized matrix, the diagonal entries are the eigenvalues, λ_i, and the axes of the ellipsoid Z have length

equal to $\sqrt{\lambda_i}$ $i = 1, \ldots, n$.

The geometric interpretation of [2.42] i.e. for the matrix P , implies

$$|\lambda_{min}(P)| \leq \sigma^2 \leq |\lambda_{max}(P)| \qquad [2.43]$$

Thus the expected error of the parameters is between λ_{max} and λ_{min} . If the ratio between the larger to the smaller eigenvalue is very large our ellipsoid would be very elongated. This will imply that some parameters are poorly determined and they might be very sensitive to the noise in the data, or equivalently the expected error in some parameters is very large.

Now, suppose we have a set of parameters found by our technique using one set of initial values, and suppose that after the error analysis some parameters are poorly determined, i.e. $K(P)$ is very large. Then the following question arises: What can be done to make P better conditioned? Or equivalently, how can we better determine our poorly determined parameters?

To consider this question let us reconsider the badly ill-conditioned problem discussed in [2.5.1]:

"Find the point of intersection of the lines

$$x_1 + .99x_2 = 1.99$$
$$.99x_1 + .98x_2 = 1.97 \quad \text{represented by } M\bar{x} = \bar{b}"$$

If instead of having only one piece of data i.e.

$$\text{say } b_1 = \begin{pmatrix} 1.989903 \\ 1.970106 \end{pmatrix}$$

which we have shown to satisfy

$$Mx = b \quad \text{for} \quad x = \begin{pmatrix} 3.0000 \\ -1.0203 \end{pmatrix}$$

Suppose we have two data points, that is, suppose we have

b_1 as above and $b_2 = \begin{pmatrix} 1.989999 \\ 1.970050 \end{pmatrix}$

If we use both data points the problem is no longer linear but it is better conditioned in the following sense. Our motive is to obtain the best value of x from the data given. One optimization scheme is described by:

1) Find the vector $\hat{x} \in R^2$ which minimizes

$$\|Mx - b_1\|^2 + \|Mx - b_2\|^2 \qquad [2.44]$$

The question of ill-condition revolves around the concept: what accuracy in the data $\frac{\|\Delta b\|}{\|b_i\|}$ (where Δb is the error in b) is required to insure a desired accuracy in the answer, $\frac{\|\Delta x\|}{\|x\|}$ (where Δx is the error in x)?

The ratio of the mean $\bar{b} = \frac{1}{2}(b_1 + b_2)$ to the square root of the variance (standard deviation), $\sqrt{(b_1-\bar{b})^2 + (b_2-\bar{b})^2} = \sigma$ is defined to be the signal to noise ratio of the data.

Thus the data really contains information corresponding to $\frac{\|\Delta b\|}{\|b\|}$ in the statisticaly form: $\frac{\sigma}{\|\bar{b}\|}$.

The previously mentioned optimization scheme utilizes this information. Therefore, we can infer that statistical knowledge of $\frac{\|\Delta x\|}{\|x\|}$ is better when two data points are utilized than only one is used. In other words: the signal-to-noise ratio or accuracy of x can be predicted via [2.37a] i.e.

$$\frac{\|\Delta x\|}{\|x\|} \leq K(A) \frac{\sigma}{\|\bar{b}\|} \qquad [2.45]$$

using b_1 and b_2 we obtain $\sigma^2 = 6.176 \quad 10^{-9}$

$$\frac{\|\Delta x\|}{\|x\|} \leq K(A) \frac{\sigma}{\|\bar{b}\|} \approx (3.6\times10^4) \frac{\left(\sqrt{6.176\times10^{-9}}\right)}{2\sqrt{2}}$$

$$\approx \frac{3.6\times.785}{2\sqrt{2}}$$

$$\approx .9$$

which is an improvement over the expected accuracy, 1.8, when only one piece of data is used.

Using this idea we were able to "better condition" the parameter identification problem. Consider Step g Section 2.2. Now for each $x(0)$ (initial value) we can get a set of parameters for our dynamical system. This set would contain some poorly determined parameters. This means that for each set of parameters K^i, σ_b is very large and more information is needed to determine these parameters. Clearly the union of the sets K^i contains more information about our system than each individual set. Therefore it is natural to consider the sum F and minimize over the space of parameters to get a set of parameters which best satisfies our system under several initial conditions.

In our dynamical model for the Calvin cycle, we were unable to recover our exact parameters, with reasonable error, with one or three different initial values (though there was a big improvement from one initial value to three). But we were able to recover all of the parameters with some reasonable error bounds with six initial conditions [2.6].

When calculating the variance of the kinetic parameters of the Calvin cycle due to six percent error in the data, it was found that, when using only one initial value the

parameters were poorly determined and their variance range from 10^{-4} to 10^{9}. While when using six initial values the parameters found were well determined, and their variance range from 10^{-5} to 10^{4} -- see Table 2.7.

Clearly the matrix P, obtained by using the best parameters found is still ill condition, but the ratio between the maximal to the minimal variance in the parameters has imporved by six order of magnitude. This improvement indicates that we were able to better determine the values of some parameters and therefore reduce the value of the expected errors significantly enough so as to improve the condition number of P.

2.6 TEST OF THE NUMERICAL MACHINERY USED IN THE PARAMETER IDENTIFICATION PROBLEM

We will now perform a test of our numerical machinery in order to double check that it really works and in order to test our error estimates. The computation of the error estimates involves linearization and is valid strictly only for small errors in the kinetic parameters.

The idea of this test is as follows:

We do not know the dynamics of the actual system, however, if our method really works, then our equations with the parameters values that we have developed should be a

good approximation of the real system. If this is the case we can generate simulated data (synthetic data, this will be defined in Section 2.61) from our equations. We then feed this data to our numerical machinery and compute parameter values from the simulated data. To make this process realistic we should also simulate experimental error. This can easily be done by adding noise (from a random number generator) to the simulated data. Then we should perform the error analysis and see whether the computed parameter values are within the error bounds. If this is not the case then our numerical machinery cannot be trusted. If repeated tests (with different simulated experimental error) give the expected result it still does not absolutely prove that our computed values are the true rate constants of the Calvin cycle, but it is a strong indication that the values are reliable. The ultimate test must be prediction of experimental results.

2.6.1 Implementation of the Numerical Machinery

Definition: The set of integrated values of a dynamical system, having the exact parameters will be called "synthetic data", "generated data" or "perfect data".

In this Section we will discuss how all the parts previously considered, performed under simulated conditions.

The main procedure involves the optimization and integration routines. We first started with the assumption

that our technique should be able to perform well on a small linear system, before considering the larger non-linear one. To that end we chose the following test problem

$$\frac{dy_1}{dt} = y_1 - 4y_2 \qquad y_1(0) = 1$$

$$\frac{dy_2}{dt} = -y_1 + y_2 \qquad y_2(0) = 0 \qquad [2.46]$$

with closed form solution

$$y_1 = \frac{e^{-t} + e^{3t}}{2} \qquad y_2 = \frac{e^{-t} - e^{3t}}{4}$$

This solution is the source for our synthetic data. The technique was implemented as follows: Using the set of parameters $k_1 = 1$, $k_2 = 4$, $k_3 = 1$, $k_4 = 1$ we integrated the system using our integration routines. We took four intermediate points on the trajectory (solution curve). These points will be considered "perfect data". We perturbed the data uniformly by 10% noise using the following formula

$$\bar{x} = \bar{x}^0 \quad 1 + (R-.5)\frac{2P}{100}$$

where $\bar{x}^0$ is the exact data point; $\bar{x}$ is the perturbed data point; R is a random number uniformly distributed between 0 and 1 generated by the random number generator

on the CDC 6400 computer at the University of California, Berkeley; P is the maximum percentage noise in the data (in our case 10%).

In order to begin optimization we chose our first iterate by perturbing our original set of k_i $i = 1,2,3,4$ uniformly by 100%.

The task is to recover the original parameters, i.e. $k_1 = 1$, $k_2 = 4$, $k_3 = 1$, $k_4 = 1$. We started with one initial condition $y_1(0) = 1$, $y_2(0) = 0$ after 1000 iterations. Using Bremermann's optimizer we were able to recover only two of the parameters with 10^{-2} accuracy. We stopped the procedure due to the amount of time consumed and the little progress toward the recovery of <u>all</u> the parameters.

Normally, the set of parameters found after such a computation is used as a starting guess for our multi-initial value procedure. But in this case we chose to start with our original guess and use multi-initial values.

We chose two different initial values such that the two sets are not a scalar multiple of each other. The sets chosen were

$$\begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \end{cases} \qquad \begin{cases} y_1'(0) = 1 \\ y_2(0) = 1 \end{cases}.$$

Again, performing our technique, but with two simultaneous initial values, we succeeded in recovering three of the four parameters within 10^{-3} accuracy. This was achieved after 300 iterations using Bremermann's optimizer. Consequently

we tried the technique with 3-initial values

$$\begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \end{cases} \qquad \begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \end{cases} \qquad \begin{cases} y_1(0) = \frac{1}{2} \\ y_2(0) = 2 \end{cases}$$

After 150 iterations we succeeded to recover all the parameters sought within 10^{-6} accuracy. These results are recorded in Table 2.4.

At this stage we were ready to perform the technique on our dynamical model describing the Calvin cycle. This system is large, non-linear, and stiff. For this purpose we generated synthetic data by integrating the system for a specified initial condition and a set of parameters chosen arbitrarily as "perfect parameters".

We followed the same steps tried in the 2×2 system but this time we started with one initial condition, continued with three and six initial conditons.

With six initial conditions we recovered all the 22 parameters after 72 iterations and with accuracy up to 1-10% error of the initial parameters, clearly we could have recovered all of them with better accuracy if we would have run our program for longer time. The results are recorded in Table

Table 2.4: Results on the Determination of the Parameters of System [2.46] Using Synthetic Data with up to 10% Noise.

Perfect parameters value	1.0	4.0	1.0	1.0
Initial parameters values	.23	7.19	1.47	1.71
After 150 iteractions Calculated parameters values	.984	3.967	.998	1.02
After 300 iterations Calculated parameters values	1.000002	3.999999	.999999	.99999

Final function value (after 300 interations) was *F = $.702\ 10^{-9}$

Total C.P.U. time (in a C.D.C. 6400) 320 seconds

*Here $F = \sum_{j=1}^{3} \sum_{i=1}^{4} \left(\frac{\bar{y}_i - \hat{y}_i}{\bar{y}_i} \right)^2$

3 is the number of distinct initial values

4 is the number of time intervals

$\bar{y}_i$ is the synthetic data

$\hat{y}_i$ the integrated data

(It is interesting to compare these results with the results obtained using perfect data without noise and using the initial guess, perturbation of 100% in the exact parameters.)

Table 2.5

Perfect parameters values	1.0	4.0	1.0	1.0
Initial parameters values	.13	3.189	1.811	.356
After 150 iterations Calculated parameters values	.6159	4.112	1.042	1.022
After 300 iterations Calculated parameters values	.9986	4.0009	1.00005	.99913

Final function value (after 300 iterations) was *F = .803 10^{-6}

Total C.P.U. time (in a C.D.C. 6400) 340 seconds

*Here $$F = \sum_{j=1}^{3} \sum_{i=1}^{4} \left(\frac{\bar{y}_i - \hat{y}_i}{\bar{y}_u} \right)^2$$

3 is the number of distinct initial values

4 is the number of time intervals

$\bar{y}_i$ is the synthetic data

$\hat{y}_i$ the integrated data

Table 2.6: Parameters Obtained After Testing The Numerical Machinery Using Synthetic Data With 3% Noise

Parameter*	Exact Values	Initial Values	Values Found	Function Value**	(CDC 7600) C.P.U.Time	Number of Iterations
K(1)	.68	.28	.68	Initial F =		
K(2)	1.99	1.79	1.97	284.87		
K(5)	1.38	2.29	1.48			
K(7)	1.83	1.04	1.81			
K(8)	.096	.12	.099			
K(9)	21.27	29.28	22.01			
K(12)	22.85	38.96	24.59			
K(13)	.12	.11	.12			
K(14)	4.46	8.63	3.99			
K(17)	11.52	17.14	11.71			
K(19)	6.91	6.5	7.00			
K(20)	100.58	130.18	101.21			
K(21)	.83	.68	.84	Final F =		
				1.66	400	120

*These thirteen parameters were obtained by our procedure. The other nine are obtained by using the linear relations in Section [2.2.2].

$$F = \sum_{j=1}^{6} \sum_{i=1}^{14} \left(\frac{\bar{y}_i - \hat{y}_i}{\bar{y}_i} \right)^2 ,$$ using 6 different initial values and 14 times intervals.

$\bar{y}_i$ is the perturbed data (i.e. 3% noise) $\hat{y}$ is the integrated values.

Table 2.7 Error Analysis with 3% Noise in the Data and Using Six Sets of Initial Values

Parameter	Exact Value	Value Found When F = 1.66	Actual Error	Expected Error	Percentage of Actual Error (WFT) Exact Value
1	.68	.68	0	$1.\times10^{-6}$	0%
2	1.98	1.97	.02	$8.\times10^{-5}$	1%
3	2835.85	2807.34	28.51	$2.\times10^{2}$	1%
4	.066	.070	.004	$1.\times10^{-5}$	6%
5	1.38	1.48	.1	$1.\times10^{-2}$	7.2%
6	.000280	.000277	.000003	$1.\times10^{-5}$	1%
7	1.83	1.81	.02	$3.\times10^{-2}$	1%
8	.096	.099	.003	$8.\times10^{-6}$	3.1%
9	21.27	22.01	.74	.1	3.4%
10	267.78	277.10	9.32	$2.\times10^{2}$	3.4%
11	.0017	.0019	.0002	$4.\times10^{-7}$	11.7%
12	22.85	24.59	1.74	.25	7.6%
13	.12	.12	0	$1.\times10^{-7}$	0%
14	4.46	3.99	.047	.28	1%
15	5.26	4.71	.55	$1.\times10^{3}$	10.45%
16	26.72	27.16	.44	5.	1.6%
17	11.52	11.71	.19	8.	1.6%
18	9.67	9.8	.13	6.	1.3%
19	6.91	7.00	.09	.65	1.3%
20	100.58	101.21	.63	.36	.6%
21	.83	.84	.01	$3.\times10^{-4}$	1.2%
22	.34	.35	.01	$6.\times10^{-5}$	1.3%

Table 2.7: continued

Error Analysis with 6% Noise in the Data

Parameter	Exact Value	Value Found When F = 4.1 *	Actual Error	Expected Error with one set of Initial Values	Expected error with six Initial Values	Percentage of absolute error **
1	.68	.68	0	3×10^{-4}	$.9\times10^{-5}$	--
2	1.99	1.52	.47	$.32\times10^{-3}$	$.7\times10^{-4}$	$\approx$ 24%
3	2835.85	2166.0	$.667\ 10^{3}$	$.13\times10^{4}$	$.32\times10^{3}$	$\approx$ 23%
4	.066	.064	.002	$.5\times10^{-3}$	$.25\times10^{-5}$	$\approx$ 3%
5	1.38	1.34	.04	.36	$.17\times10^{-2}$	$\approx$ 2.5%
6	.00028	.00025	.00003	$.94\times10^{-4}$	$.18\times10^{-5}$	$\approx$ 10%
7	1.83	1.6	.23	$.5\times10^{-1}$	$.85\times10^{-3}$	$\approx$ 12%
8	.096	.09	.006	$.5\times10^{-3}$	$.28\times10^{-5}$	$\approx$ 6%
9	21.27	19.5	1.77	$.2\times10^{2}$	.16	$\approx$ 8%
10	267.78	245.70	22.08	1×10^{7}	$.34\times10^{4}$	$\approx$ 8%
11	.0017	.0015	.0015	$.25\times10^{-3}$	$.29\times10^{-6}$	$\approx$ 11%
12	22.85	21.2	1.65	1×10^{1}	.2	$\approx$ 7%
13	.12	.12	0	$.4\times10^{-5}$	$.52\times10^{-6}$	--
14	4.46	5.5	1.04	$.17\times10^{2}$	$.86\times10^{-1}$	$\approx$ 26%
15	5.26	6.4	1.14	$.31\times10^{9}$	$.59\times10^{4}$	$\approx$ 21%
16	26.72	26.9	.18	$.22\times10^{3}$	6.	$\approx$ .9%
17	11.52	11.7	.18	$.28\times10^{2}$	.93	$\approx$ 1.7%
18	9.67	9.5	.17	$.55\times10^{3}$	.16	$\approx$ 1.3%
19	6.91	6.84	.07	$.23\times10^{2}$	$.34\times10^{-1}$	$\approx$ 1%
20	100.58	105.4	5.18	$.75\times10^{1}$	$.10\times10^{1}$	$\approx$ 5.2%
21	.83	.64	.19	$.69\times10^{-2}$	$.58\times10^{-3}$	$\approx$ 22%
22	.34	.27	.07	$.14\times10^{-2}$	$.12\times10^{-3}$	$\approx$ 20%

*The mean of the error is 9.7% .

2.6.2 Graphical Display of Results

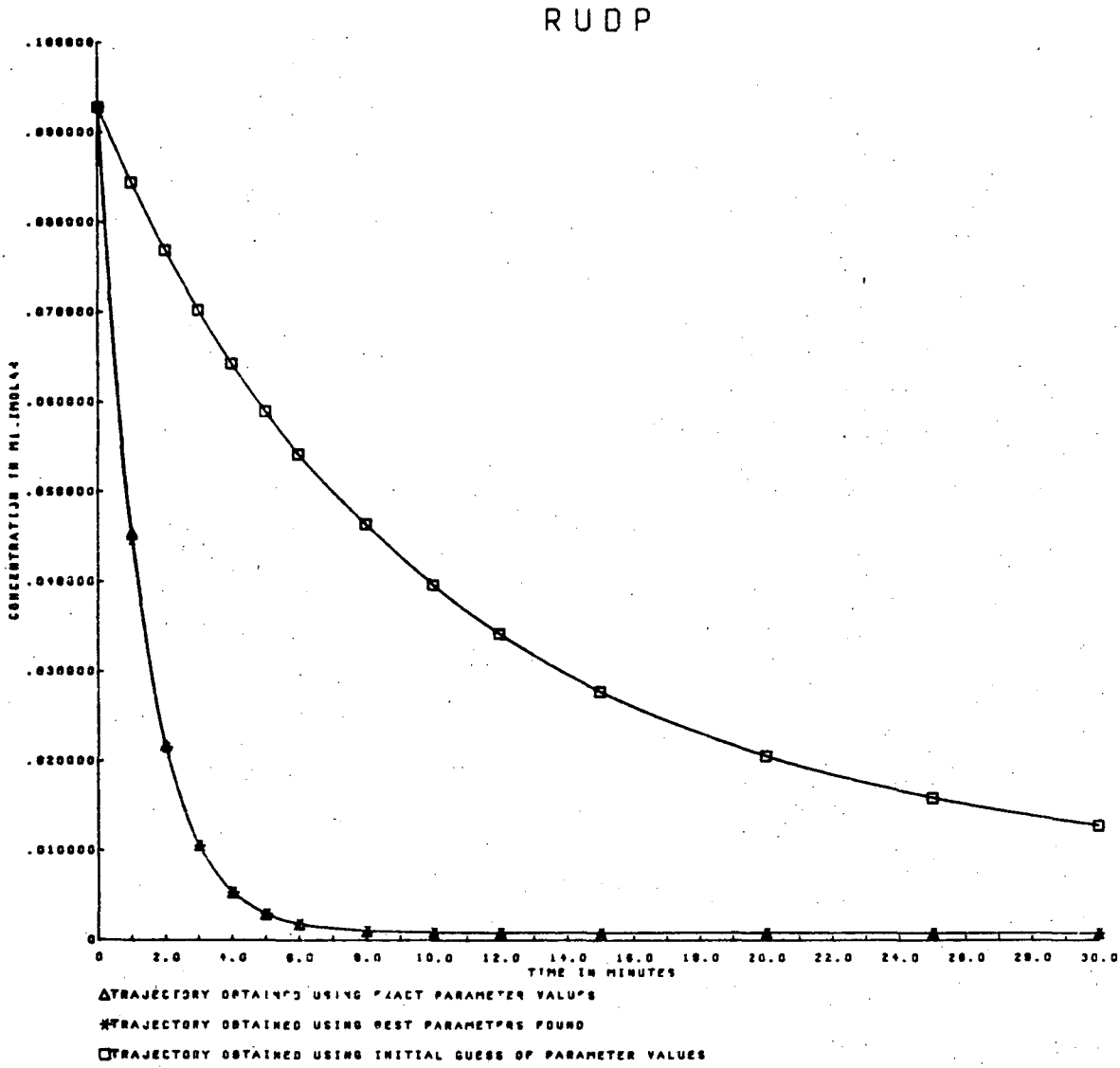
In this Section we will present the graphs of each intermediate of the Calvin cycle. These graphs were obtained using the best set of parameters found by using our numerical machinery (Table 2.7) . For graphical representation we have written a computer subroutine which implements the task of graphing any set of data with great accuracy.

Each graph will represent:

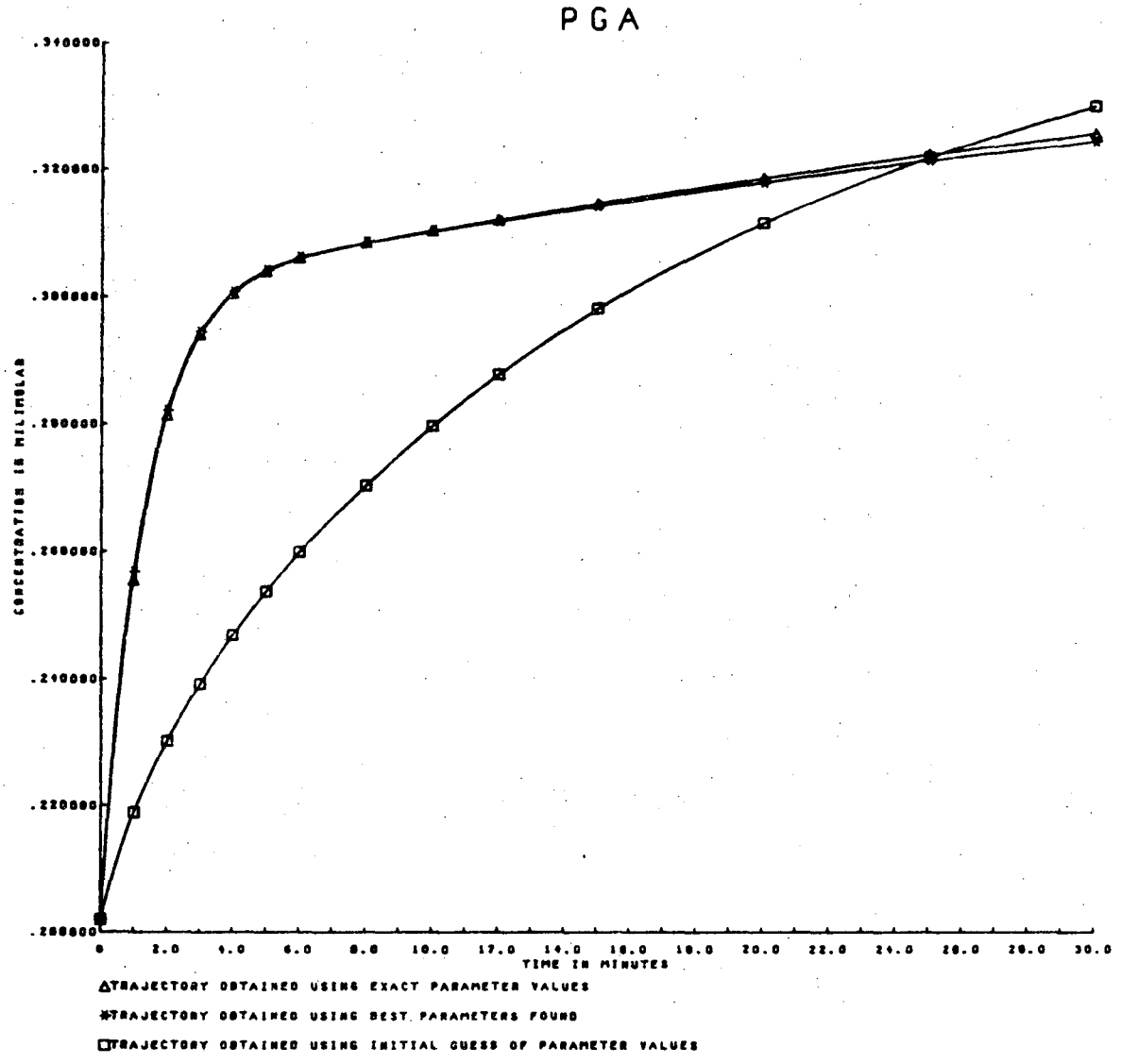
a) The trajectory of an intermediate using an initial guess of the parameters.

b) The trajectory of an intermediate using the exact parameter values. This is a set chosen arbitrarily and considered to be the "exact" parameter.

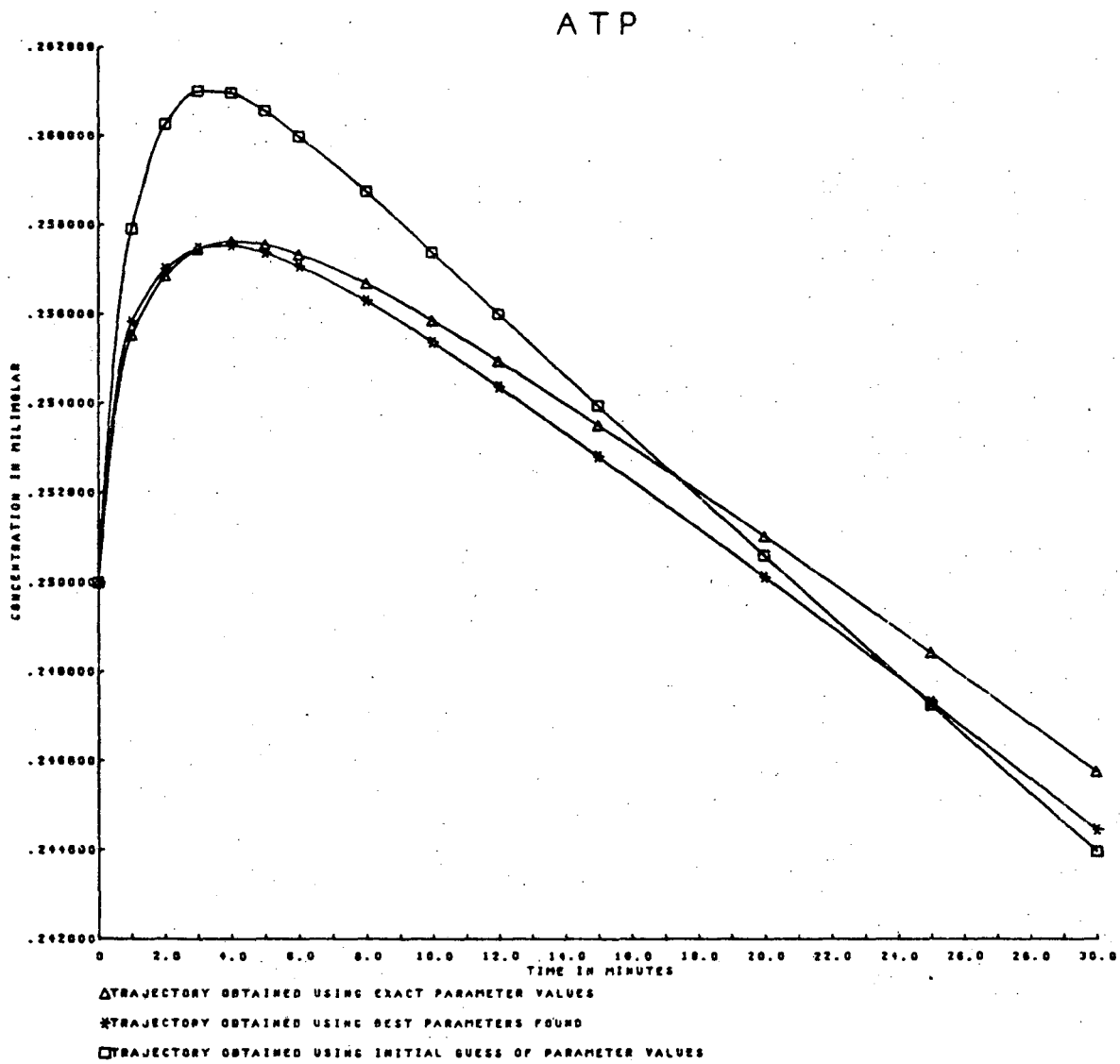
c) The trajectory of an intermediate using the best parameters found after applying our numerical machinery.


RUDP
CONCENTRATION IN MI.MOLES
.100000
.090000
.080000
.070000
.060000
.050000
.040000
.030000
.020000
.010000
0
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
ΔTRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

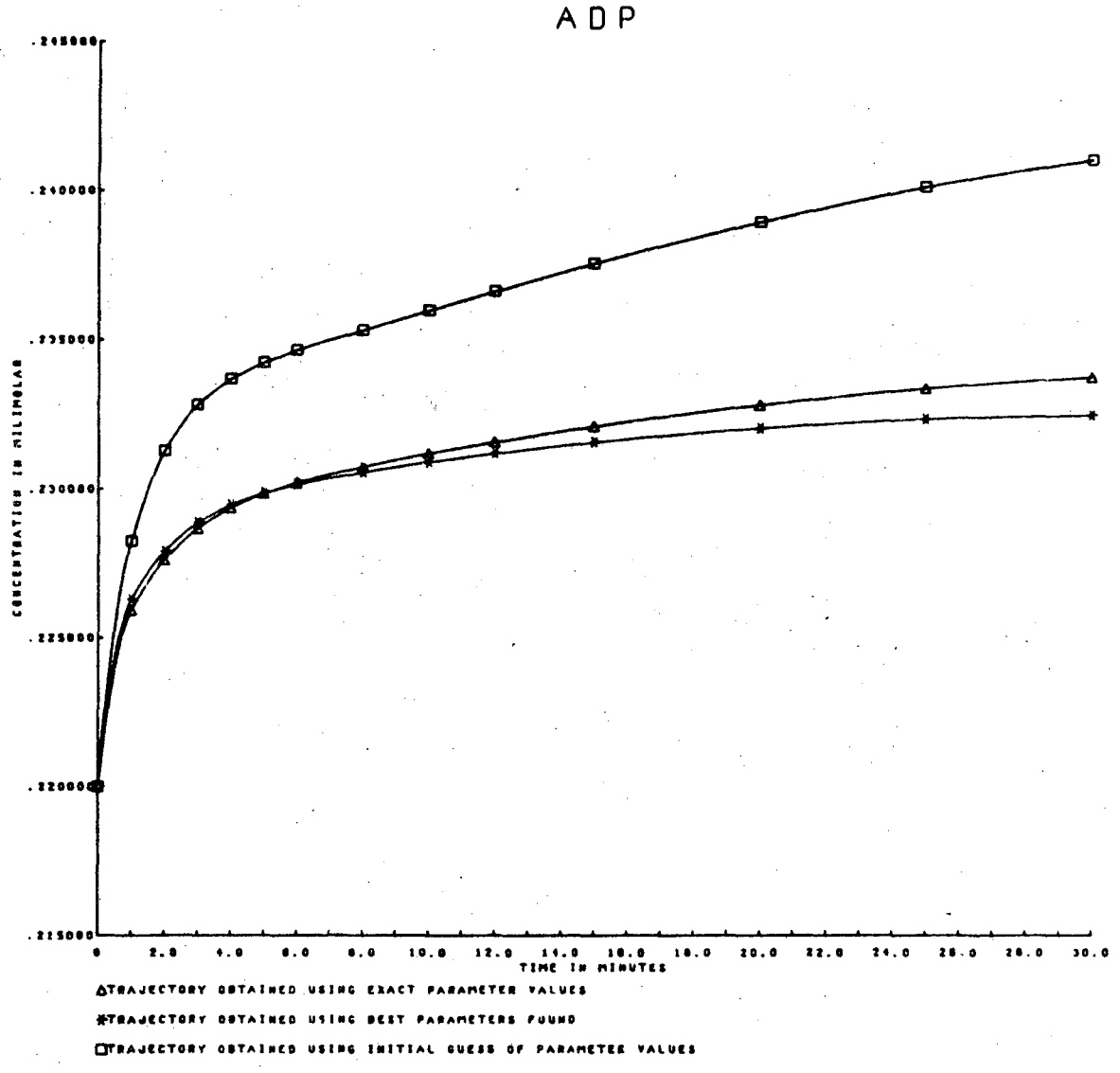
XBL 759-8012



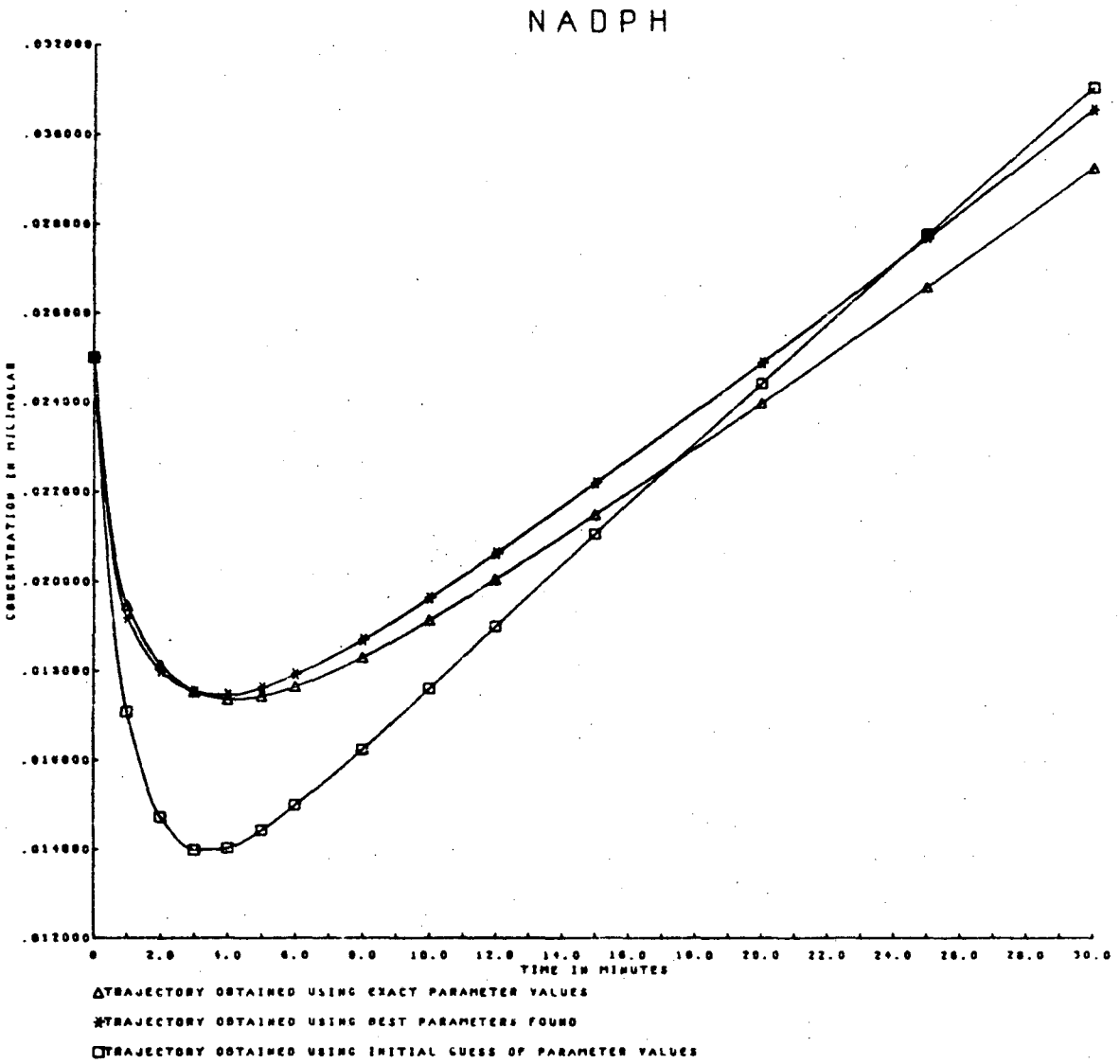
XBL 759-8011


ATP
.262000
.260000
.258000
.256000
.254000
.252000
.250000
.248000
.246000
.244000
.242000
CONCENTRATION IN MILLIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
△TRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

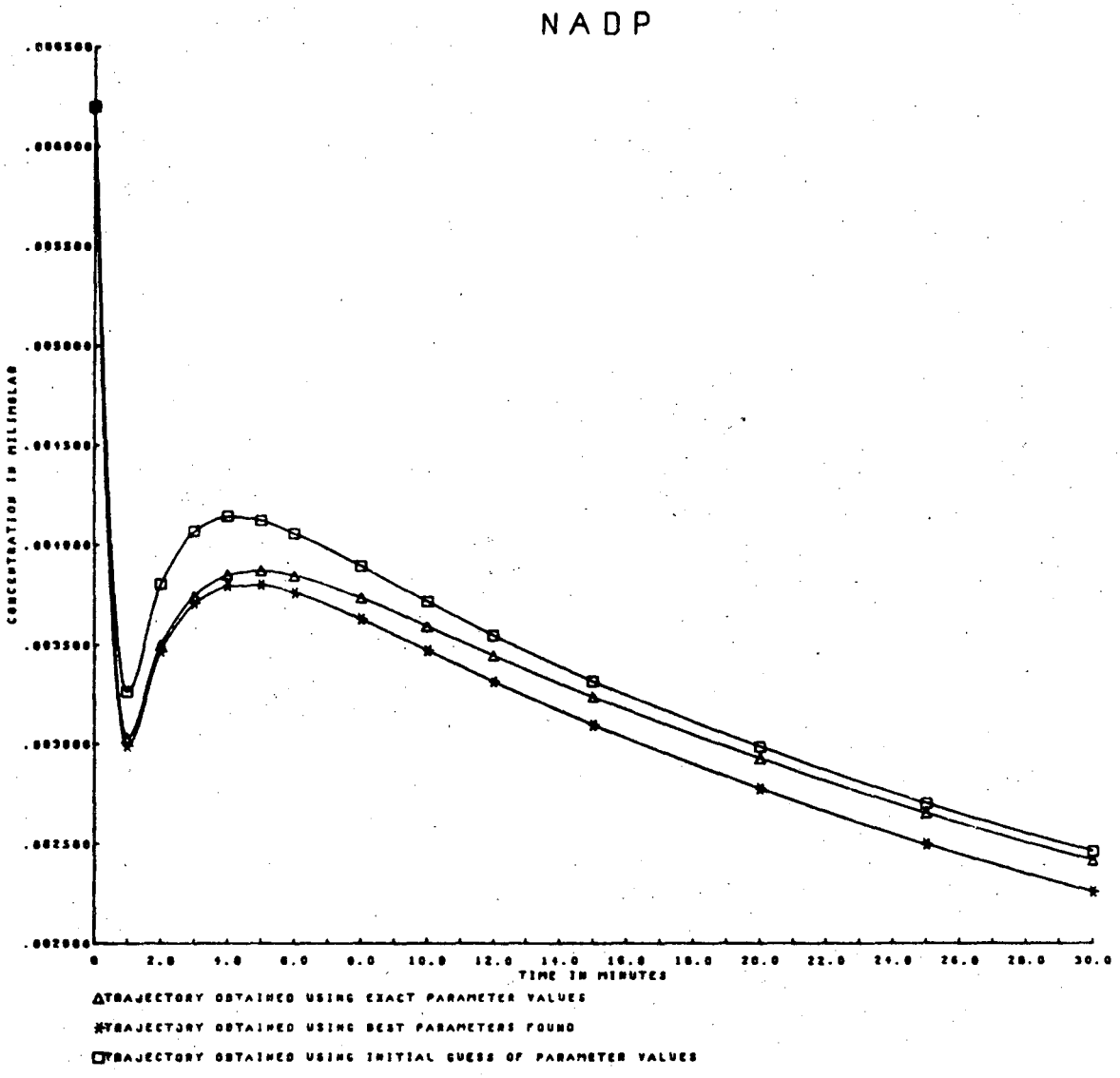
XBL 759-8010



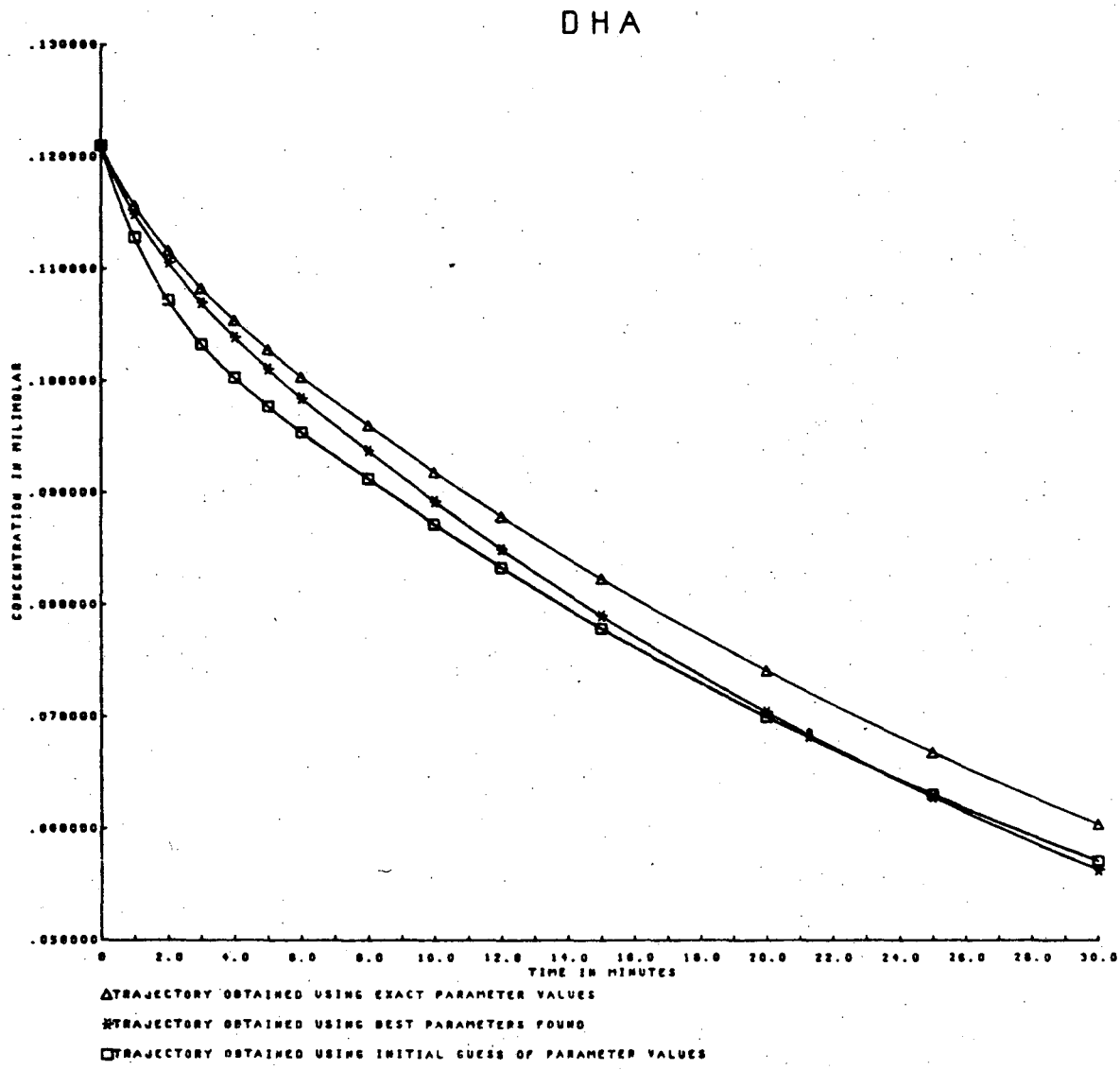
XBL 759-8009


NADPH
.032000
.030000
.028000
.026000
.024000
.022000
.020000
.018000
.016000
.014000
.012000
CONCENTRATION IN MILLIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
△TRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

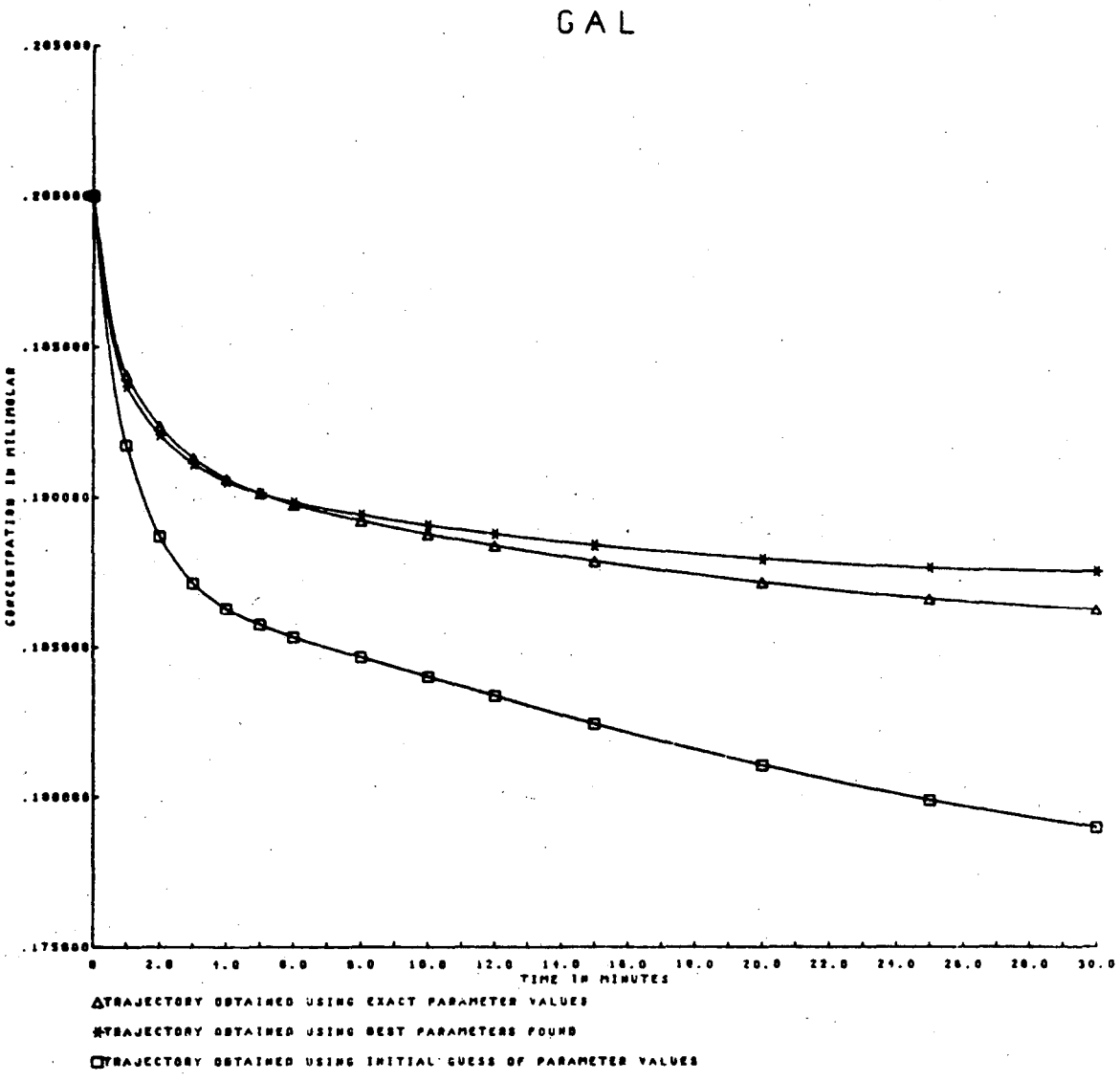
XBL 759-8008



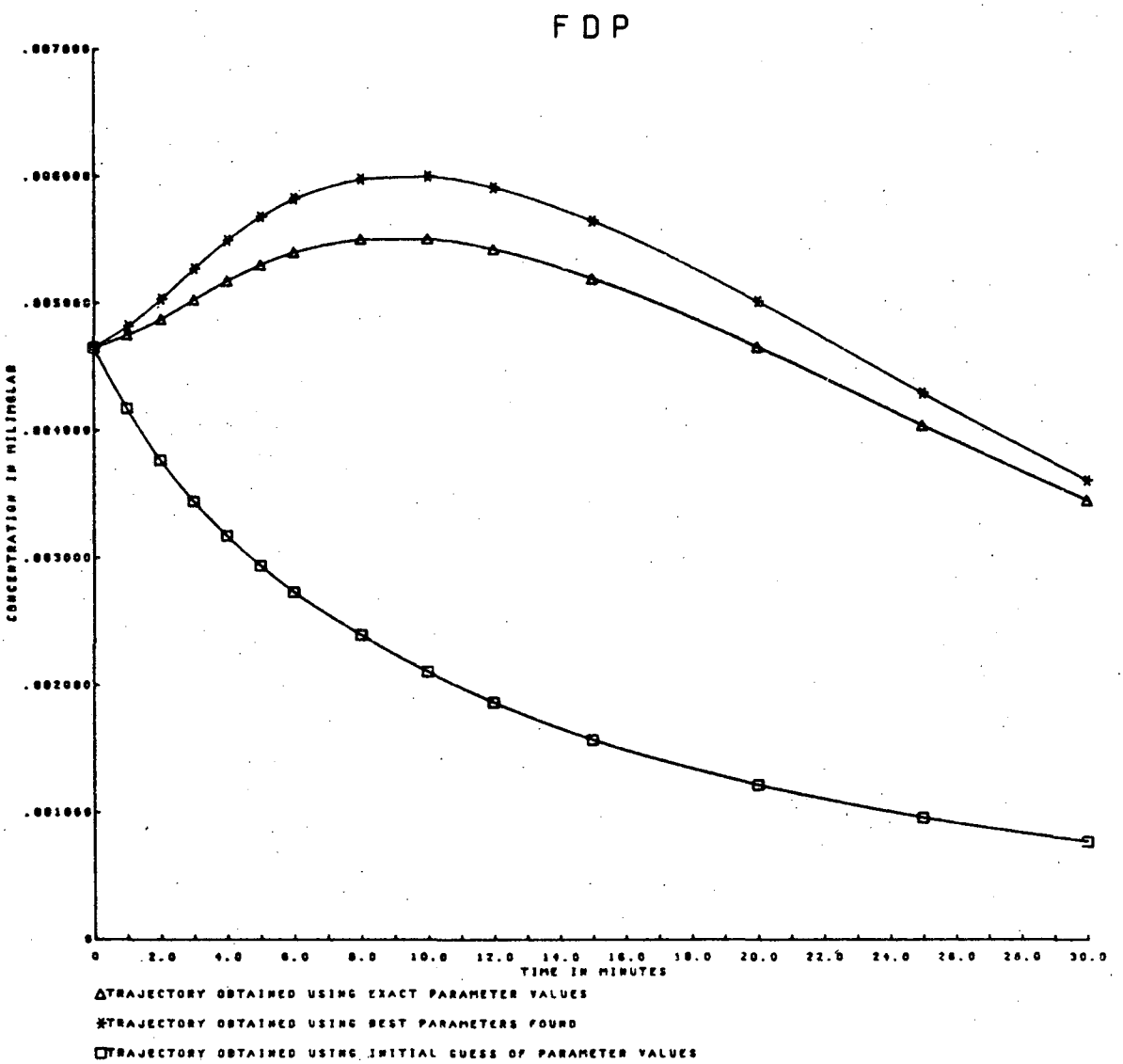
XBL 759-8007


DHA
.130000
.120000
.110000
.100000
.090000
.080000
.070000
.060000
.050000
CONCENTRATION IN MILIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
△TRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

XBL 759-8006

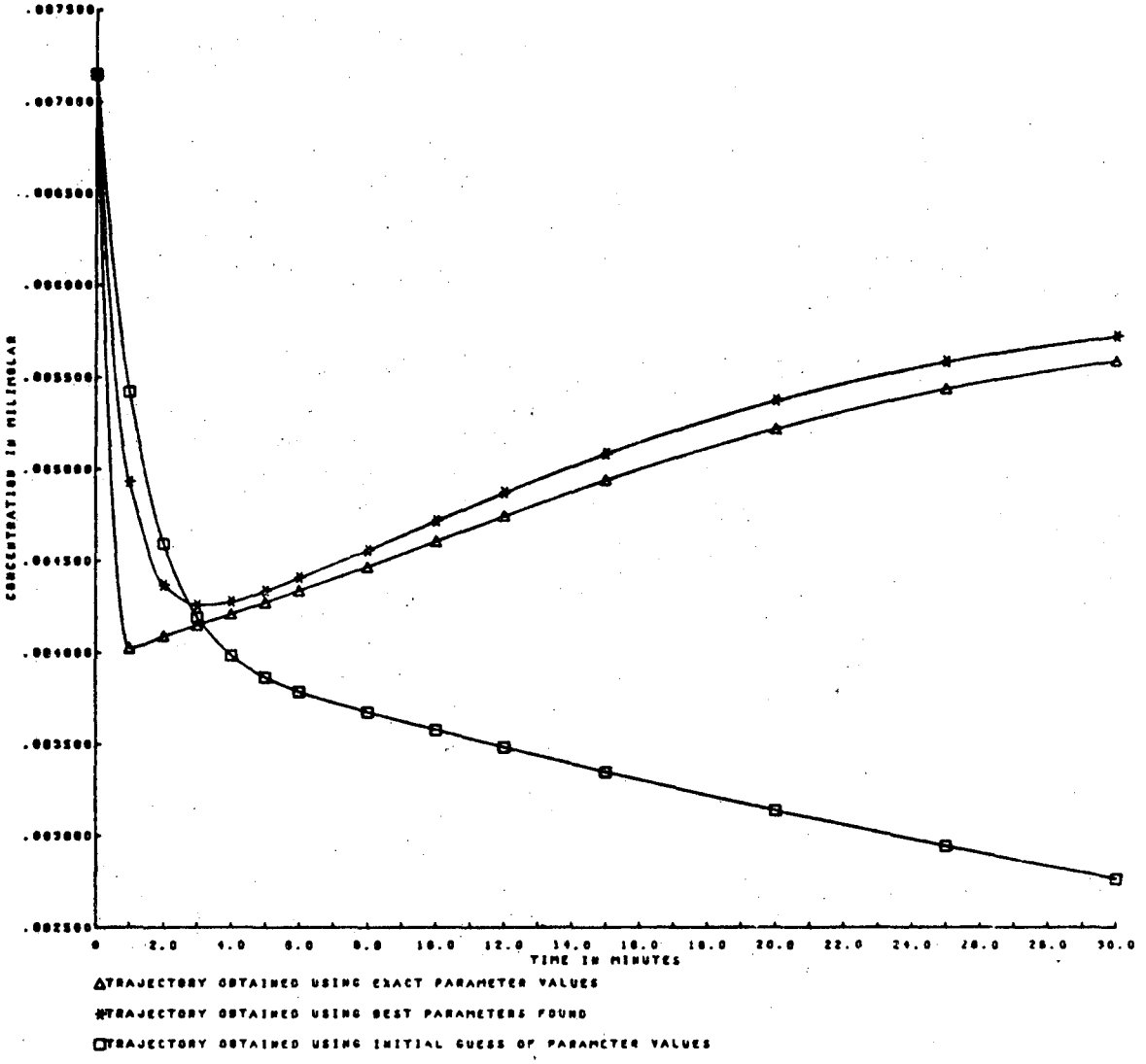


XBL 759-8005

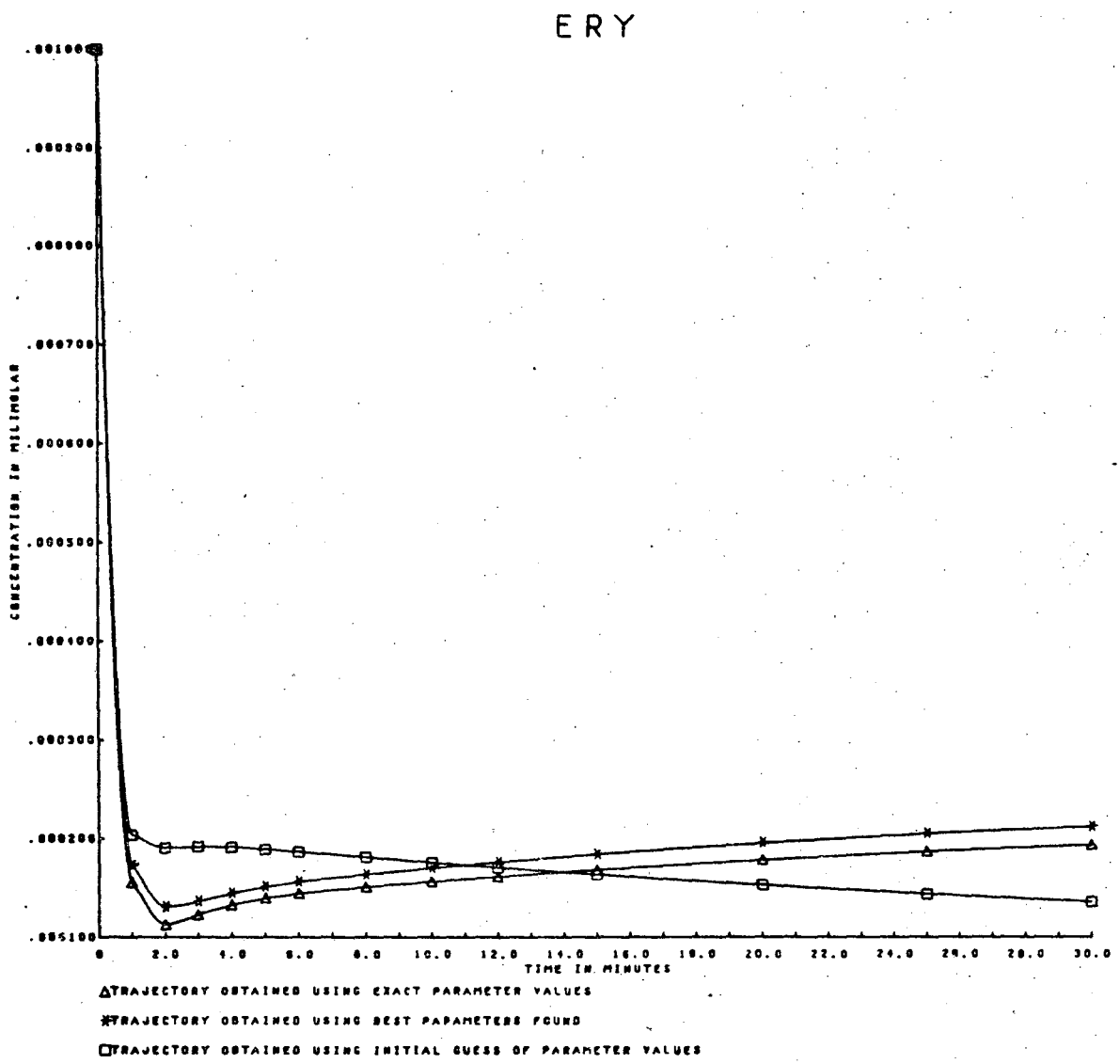

FDP
.007000
.006000
.005000
.004000
.003000
.002000
.001000
0
CONCENTRATION IN MILLIMLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
ΔTRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

XBL 759-8004

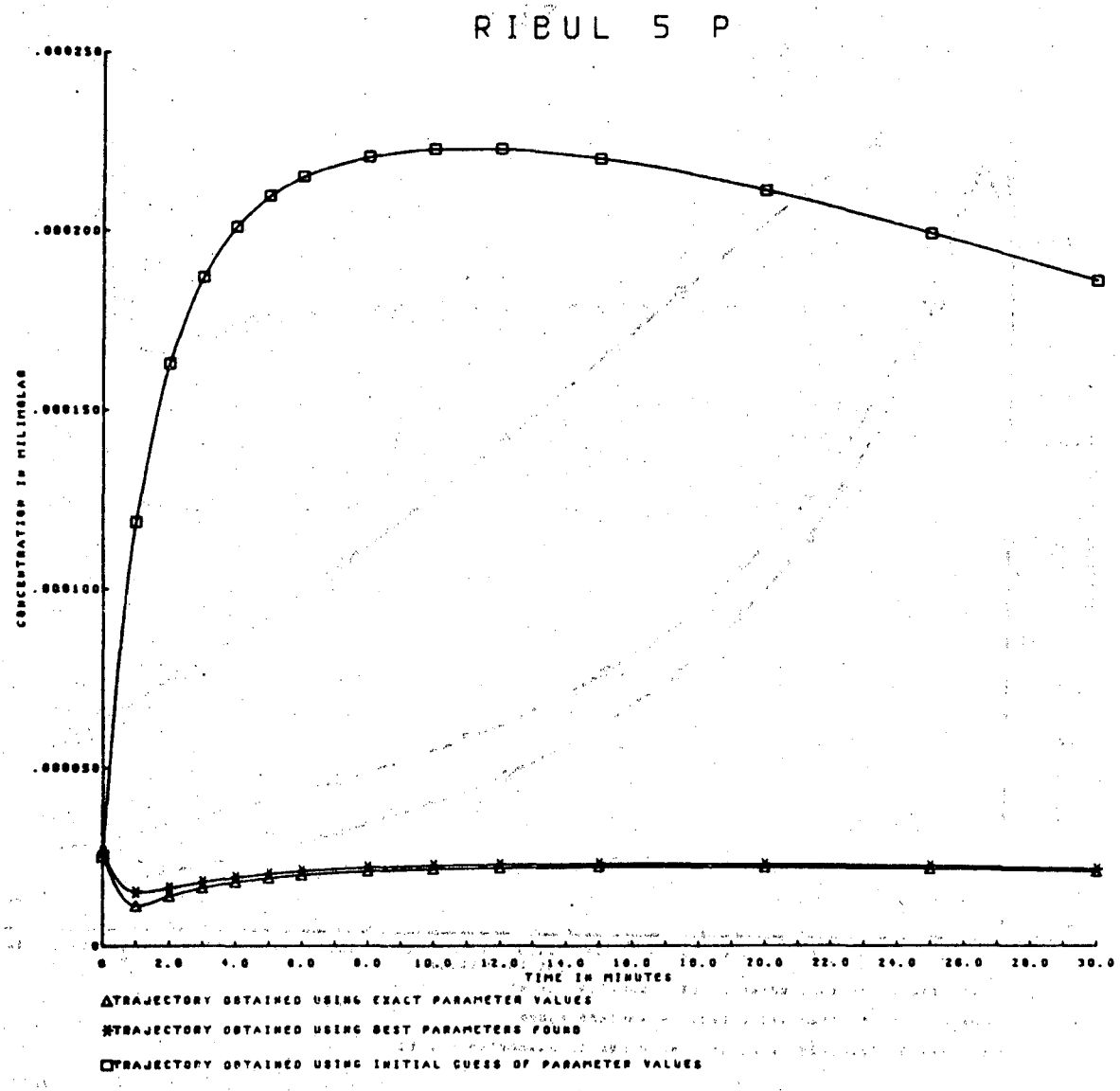
F 6 P



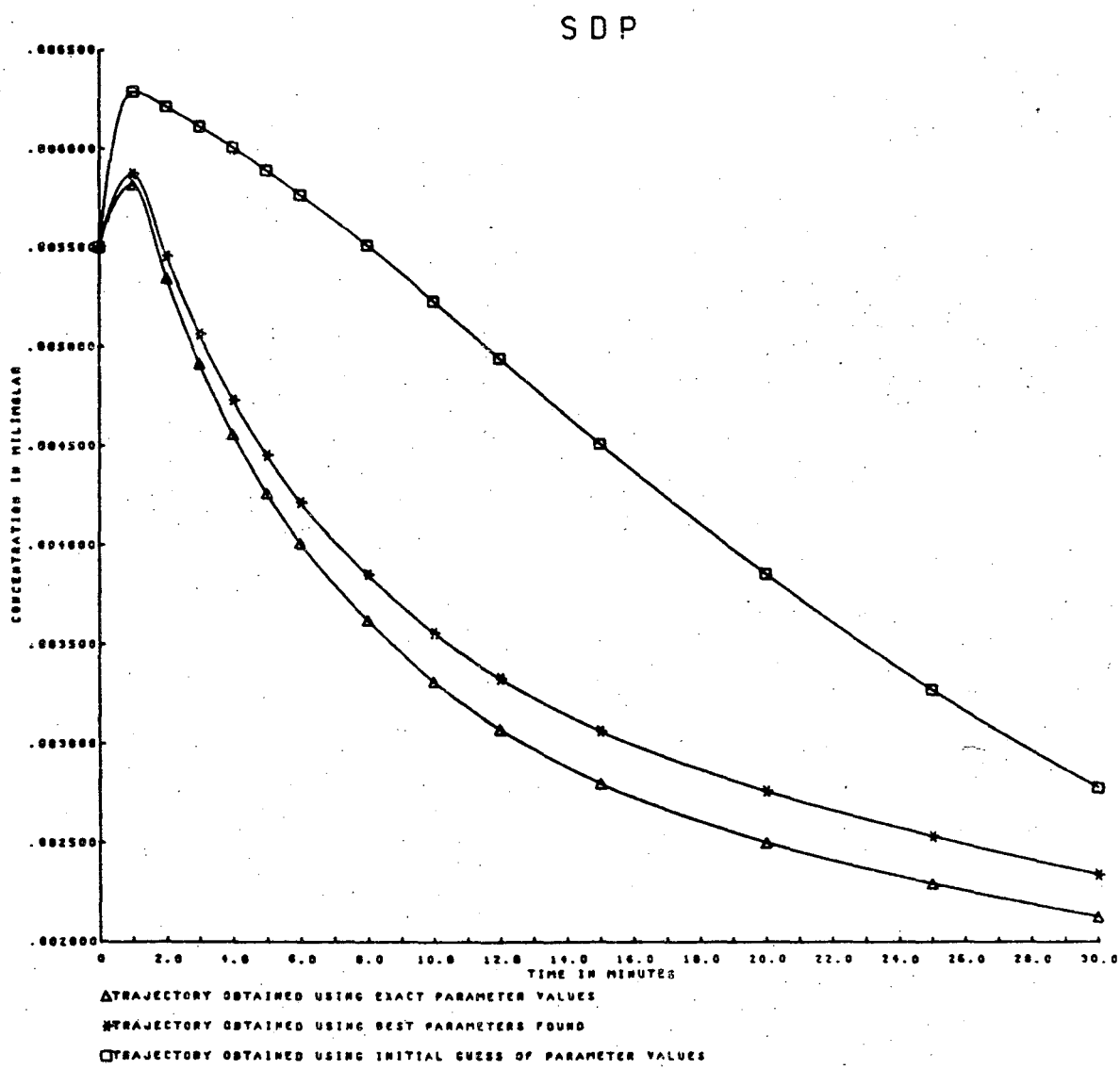
XBL 759-8003


E R Y
CONCENTRATION IN MILLIMOLAR
.001000
.000900
.000800
.000700
.000600
.000500
.000400
.000300
.000200
.000100
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
ΔTRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

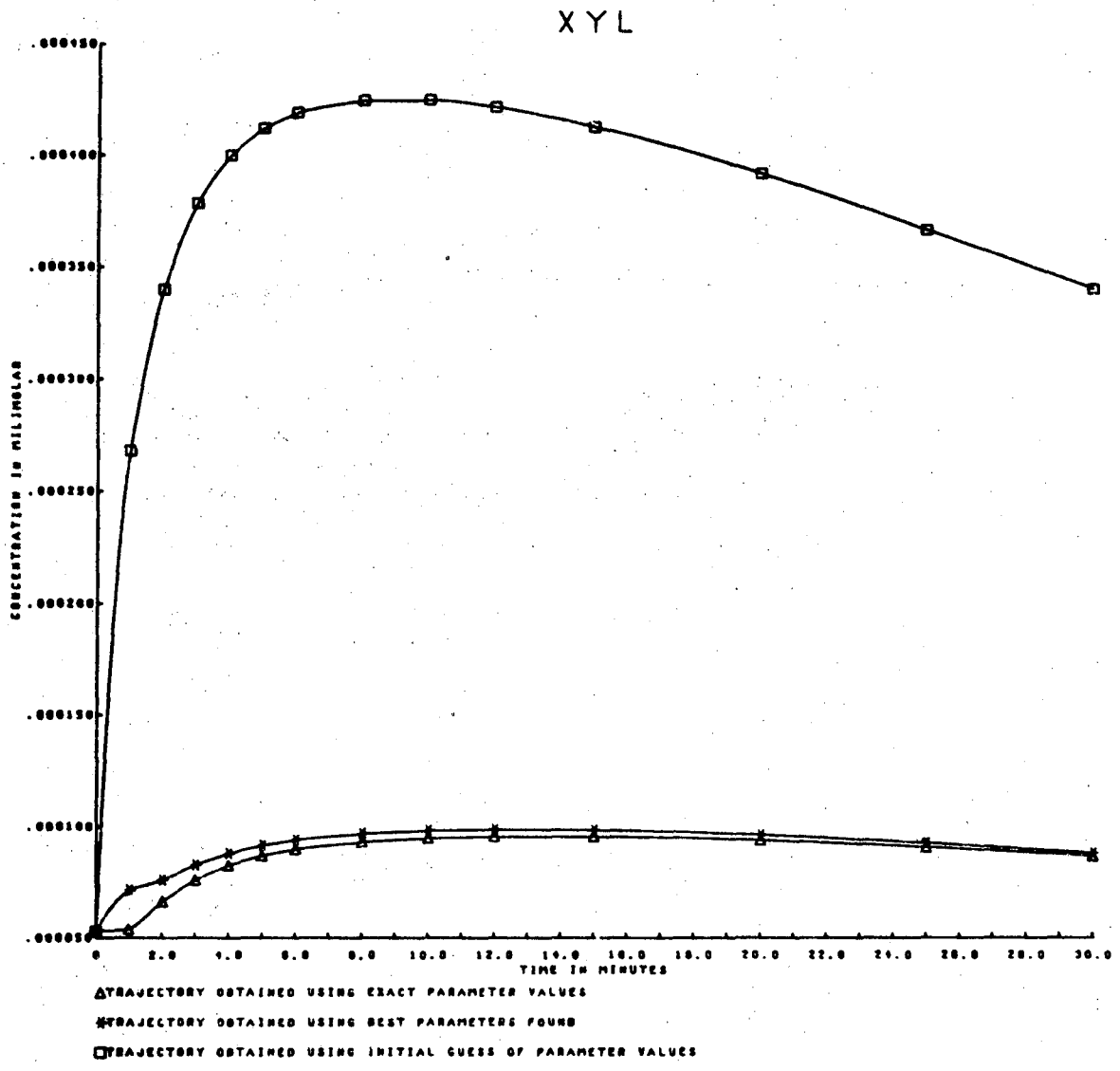
XBL 759-8002



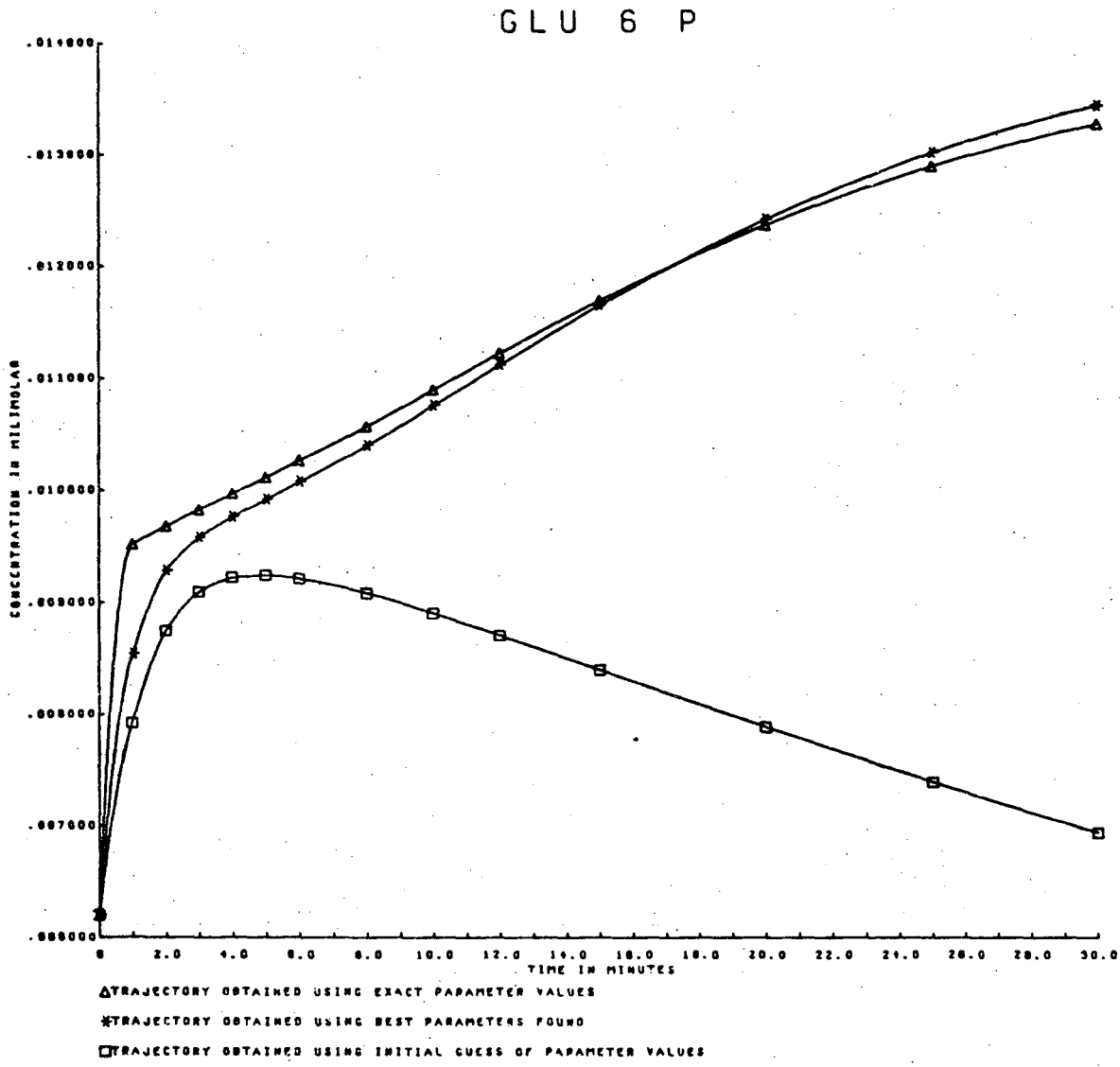
XBL 739-7998


SDP
.006500
.006000
.005500
.005000
.004500
.004000
.003500
.003000
.002500
.002000
CONCENTRATION IN MILLIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
ΔTRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

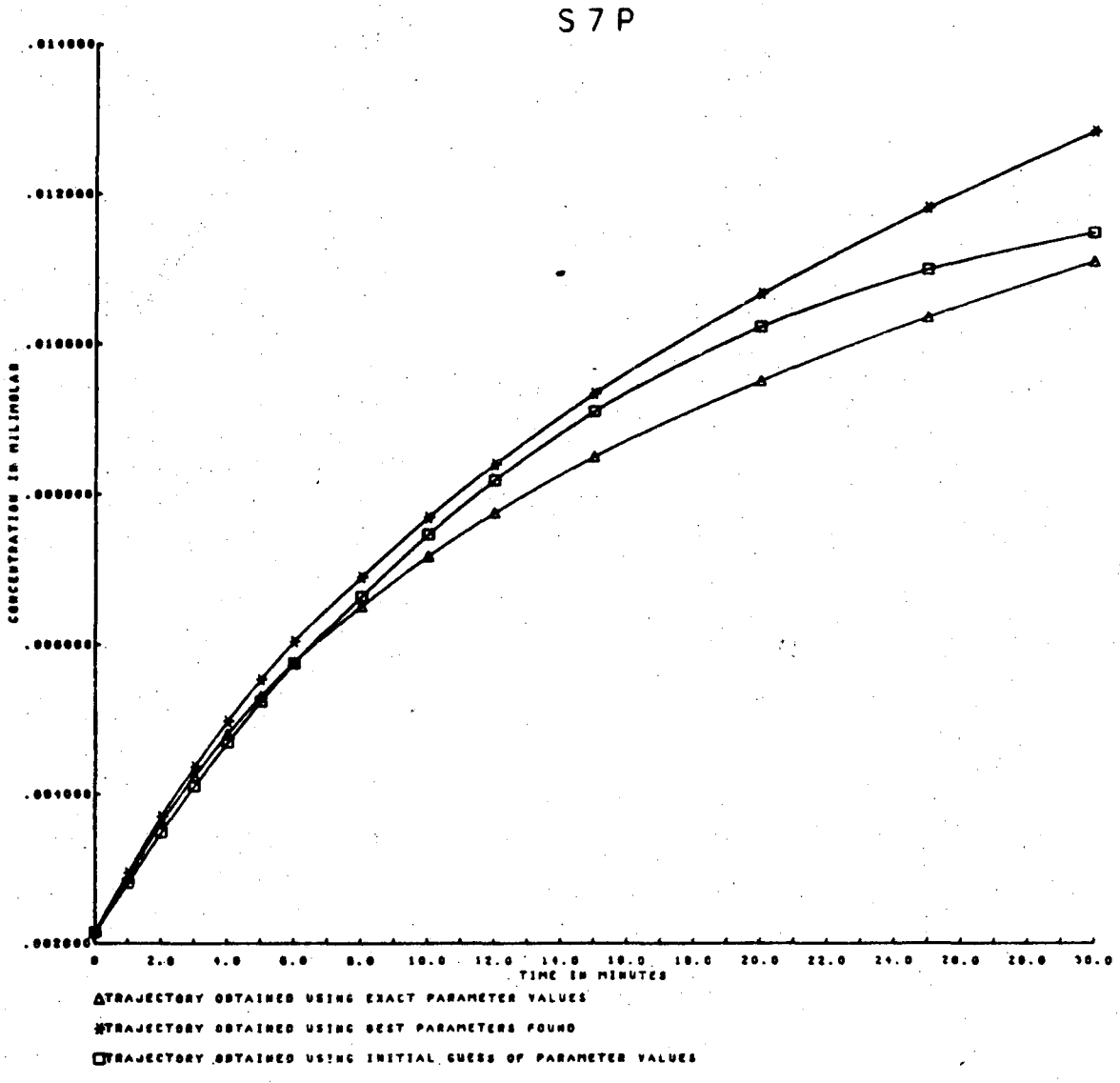
XBL 759-8001



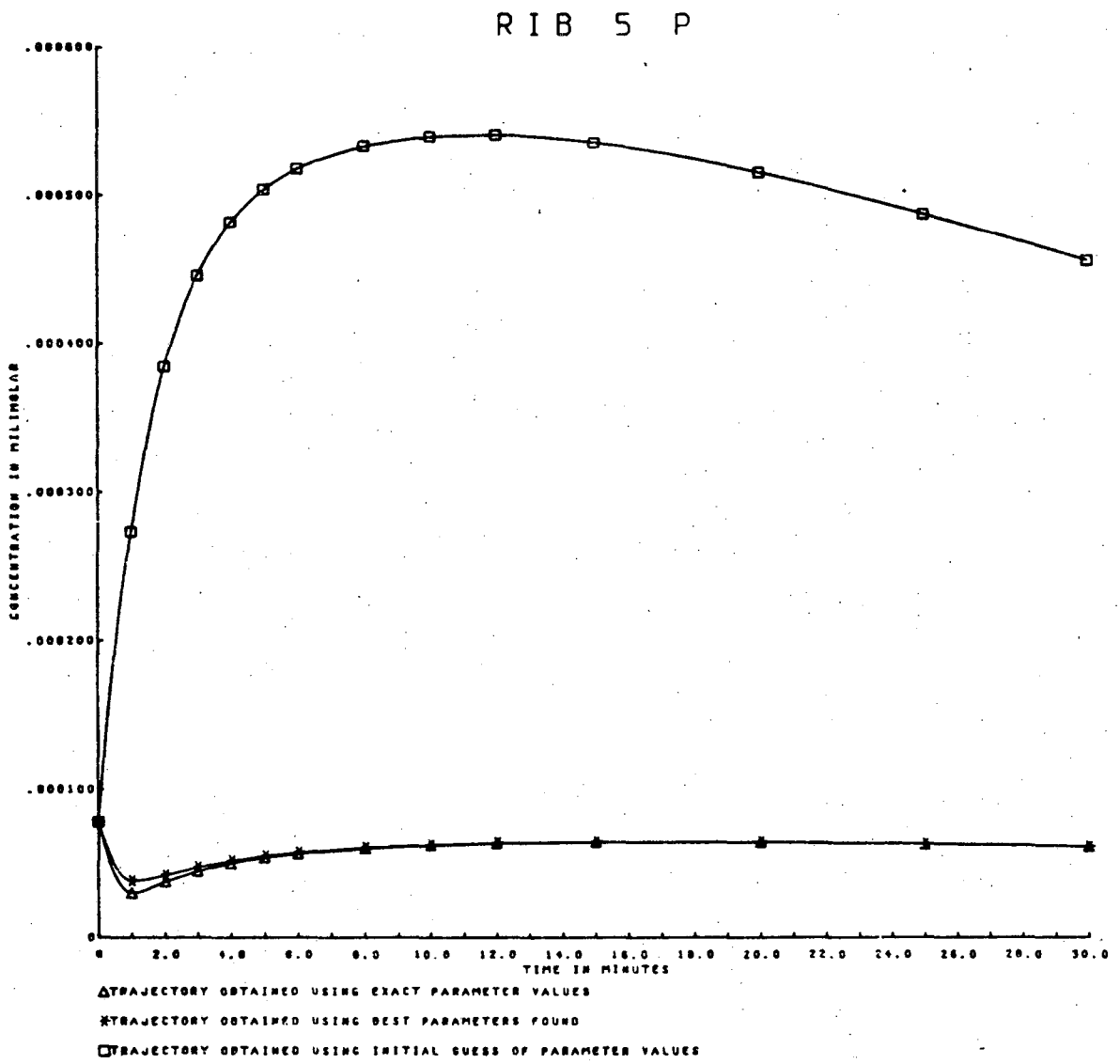
XBL 759-8000


GLU 6 P
.014000
.013000
.012000
.011000
.010000
.009000
.008000
.007000
.006000
CONCENTRATION IN MILIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
△TRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

XBL 759-7999



XBL 759-8013


RIB 5 P
.000600
.000500
.000400
.000300
.000200
.000100
0
CONCENTRATION IN MILLIMOLAR
0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
26.0
28.0
30.0
TIME IN MINUTES
ΔTRAJECTORY OBTAINED USING EXACT PARAMETER VALUES
*TRAJECTORY OBTAINED USING BEST PARAMETERS FOUND
□TRAJECTORY OBTAINED USING INITIAL GUESS OF PARAMETER VALUES

XBL 759-7997

2.6.3 Using Real Experimental Data

As we mentioned earlier in Section 2.1, the process by which experimental data is obtained is very laborious and costly. After the initial success of our methodology in January of 1975, A. Bassham's laboratory did various experiments which, if successful would have given us data for an accurate determination of the parameters. Unfortunately these did not give useful data because either some impurities were present in the reconstituted system or good steady-state concentration levels of the intermediates could not be obtained. The reader should be aware that usually when an experiment fails to obtain its objective, results are discarded and many days (even months) of research by several scientists are wasted.

Since an experiment requires much time (including fracturing the chloroplast, paper chromatography, autoradiography, computer outputs, graphs preparation, etc) data is not yet available for use in this problem. We expect that the needed data will soon be available and that implementation will then be complete.

2.7 SUMMARY

By implementing our technique using synthetic data, our purpose was

1) How well can we determine our set of parameters assuming we had good experimental data (for this purpose we generated synthetic data).
2) How could we make our ill conditioned problems "better conditioned".
3) Can we recover the original parameters.

After applying our numerical machinery we can certainly state that given a good set of data (about 10% noise) with several initial conditions, our technique will make the problem a "better conditioned" and by doing so we can recover the set of parameters that govern the dynamic system with better accuracy.

3.1 CONCLUSION

Our general task was the identification of the kinetic parameters of the Calvin cycle. To this end we used synthetic data and a set of parameters (arbitrarily chosen) to check the performance of our numerical machinery. The results indicate that our technique is a reliable tool for parameter identification and in particular for the determination of the parameters of the Calvin cycle. The numerical routines, though elaborate, can be adapted fairly easy to systems of a different nature than the Calvin cycle. An extension

of our methodology to other biochemical and metabolic systems should be fairly straightforward and of direct biological importance.

In the process of developing our method we have put together the pieces of a very complex system using information from different fields of science; such as non-linear optimization, numerical integration of stiff systems, algebraic manipulation with computers, error analysis of the results, biochemical kinetics, thermodynamics properties of the system and the chemical pathway of the Calvin cycle.

For the first time we have obtained an approximation of the parameters value which determine the kinetics of the Calvin cycle.

The results justify our encouragement for further investigation in the following areas:

1) Determination of more accurate parameters value from better data.
2) The incorporation of the enzyme kinetic of the Calvin cycle to our mathematical model.
3) The analytical investigation of the multi-initial value problem.
4) The introduction of controls into the system.

Once the kinetic parameters are known we can introduce new control parameters that would determine the maximum output desired by "steering" the system toward such an output.

```
      PROGRAM MATH(INPUT,OUTPUT,PUNCH,TAPE6=OUTPUT)
C*****  PROGRAM MATH IS THE MAIN PROGRAM, IT CONTAINS EREMERMAN*S
C*****  ALGORITHM. MATH CALLS FUNCTION F(Z). THIS FUNCTION IS THE
C*****  FUNCTION DESCRIBED IN SECTION 2.2.
C*****  FUNCTION F(Z) CALLS THE INTEGRATION ROUTINE #DRIVES# WHICH
C*****  IS THE IMPLEMENTATION OF GEARS METHOD DISCUSSED IN SEC. 2.3.2.
C*****  YNI= INITIAL CONDITIONS FOR THE DYNAMICAL SYSTEMS .
C*****  E = ARRAY CONTAINING THE DATA POINTS (SYNTHETIC,OR EXPERIMENTAL)
C*****  DR = ARRAY CONTAINIG THE INTEGRATED VALUES OBTAINED BY USING
C*****  GEARS METHOD.
C*****  K3 = NUMBER OF SETS EACH OF #HICH CONTAINS 17 INITIAL VALUES.
C*****  K2 = NUMBER OF DATA POINTS (=13)
C*****  K1 = NUMBER OF STATE VARIABLES (THE 17 INTERMIDIATES OF THE
C*****  CALVIN CYCLE )
C      IT = NUMBER OF ITERATIONS
C      N = NUMBER OF VARIABLES
C      X = ARRAY OF VARIABLES
C--RUDP ----F6P----DHAP----FGA----ADP----ATP---FDP-----SDP----G6#----S7P
C---XYL--R5P----IU5P-----GALPH------NERY-------NADP----- --NADPH--------
      COMMON CZ(13)
      COMMON/JAIME/F(17,13,6),K1,K2,K3
      COMMON /EIN/DR(17,13),X(13),N
      COMMON /DOG/YNI(17,1,6)
      COMMON/FISH/PI(13),IEQ
      DIMENSION XX(13)
      DIMENSION VAL(13)
      DIMENSION HHH(96)
      DIMENSION R(200)
      LOGICAL WARN
      DATA K1,K2,K3/17,13,1/
      DATA HHH/24(.001,.002,.01,.1)/
C
      READ 300,N
      READ 300,IT
  300 FORMAT(I6)
      READ2,(((YNI(I,1,L),I=1,K1),L=1,K3)
2     FORMAT(8F10.9,/,8F10.9,/,F10.9)
      READ 17,X
   17 FORMAT(1(8F10.8/),5F10.8)
      PRINT 324,(X(I), I=1,N)
  324 FORMAT (13X, * CZ PARAMETERS VALUES *///(1X,6(E20.10, 1X)//))
      READ 501,(((F(I,J,K),J=1,K2),I=1,K1),K=1,K3)
  501 FORMAT(8F10.9,/,5F10.9)
      PRINT 502,(((F(I,J,K),J=1,K2),I=1,K1),K=1,K3)
  502 FORMAT( * DATA POINTS *//(1X,  8(F10.6,5X)//,1H ,5(F10.6,5X)///))
      PUNCH 896,(((F(I,J,K),J=1,K2),I=1,K1),K=1,K3)
      DO 23 I=1,N
      X(I)=ALOG(X(I))
      XX(I)=X(I)
   23 CONTINUE
      F0=F(Z)
      PRINT 311, F0
  311 FORMAT ( * INITIAL FUNCTION VALUE * , E20.10)
```

```
C
C       BEGINNING OF THE OPTIMIZATION
        KG = 0
        KSA=0
        DO 10 J = 1, IT
C       RANDOM DIRECTION GENERATOR
C       RANF(0)    RETURNS A RANDOM NUMBER UNIFORMLY DISTRIBUTED BETWEEN
C                  ZERO AND ONE AT EACH CALL
C       R(I)    HAS A GAUSSIAN DISTRIBUTION
C
        K6=0
        KG = KG + 1
        DO 40 I = 1, N
   39   XY = -ALOG(RANF(0))
        YY = -ALOG(RANF(0))
        IF (YY.LT.0.5*(XY-1.0)**2)GO TO 39
        R(I)  =SIGN(XY,RANF(0)  -.5)
        R(I)=R(I)*XX(I)
   40   CONTINUE
C
C       FORMATION OF FOURTH ORDER POLYNOMIAL
C
C       H  = DISTANCE BETWEEN THE POINTS OF LAGRANGIAN INTERPOLATION
        KSA=KSA+1
        IF(KSA.GT.40)KSA=1
        H=HHH(KSA)
   35   DO 41 I = 1, N
   41   X(I) = XX(I) + 2. * R(I) * H
        F2 = F(Z)
        DO 42 I = 1, N
   42   X(I) = XX(I) + R(I) * H
        F1 = F(Z)
        DO 43 I = 1, N
   43   X(I) = XX(I) - R(I) * H
        FM1 = F(Z)
        DO 44 I = 1, N
   44   X(I) = XX(I) - 2. * R(I) * H
        FM2 = F(Z)
        A = (F2+FM2)/6.+F0-2.*(F1+FM1)/3.
        B = (F2-FM2)/4.-(F1-FM1)/2.
        C = 4.*(F1+FM1)/3.-5.*F0/2.-(F2+FM2)/12.
        D = 2.*(F1-FM1)/3.-(F2-FM2)/12.
        IF(ABS(A).GE.1.E-20) GOTO 50
C
C       LINEAR SOLUTION
C
        Y1 = - D / C
        GO TO 51
C
C       CUBIC SOLUTION
C
C          ONE ROOT
C
   50   P = C/A-B**2/(3.*A**2)
```

```
      Q = D/A-B*C/(3.*A**2)+2.*B**3/(27.*A**3)
      DELTA = 4.*P**3 +27. *Q**2
      IF (DELTA.LE.0.) GO TO 60
      ROOT =SQRT(DELTA/108.)
      AA =-Q/2. +ROOT
      BA=-Q/2.-ROOT
      Y1=SIGN(ABS(AA)**(1./3.),AA)+SIGN(ABS(BA)**(1./3.),BA)-B/(3.*A)
      IF(ABS(Y1).GT.500. ) GO TO  83
   51 DO  52  I= 1,N
   52 X(I) = XX(I) + Y1 * R(I) * H
      FUNC = F(Z)
      GO TO 70
C
C        THREE ROOTS
C
   60 PHI = ASIN((SQRT(27.)*Q)/(2.*P*SQRT(-P))) /3.
      U = SQRT(-P/3.) * 2.
      V = - B /(3. * A)
      S = SIN(PHI)
      CS = COS(PHI) * SQRT(3.) / 2.
      Y1 = U * (CS + 0.5 * S) + V
      Y2 = -U * S + V
      Y3 = -U * (CS -00.5 * S) + V
      IF(ABS(Y1).GT.500. ) GO TO 84
      IF(ABS(Y2).GT.500.) GO TO 84
      IF(ABS(Y3).GT.500.) GO TO 84
C
C     MINIMUM OF THE POLYNOMIAL
C
      FF1 = A/4. * Y1**4 + B/3. * Y1**3 + C/2. * Y1**2 + D * Y1 + F0
      FF2 = A/4. * Y2**4 + B/3. * Y2**3 + C/2. * Y2**2 + D * Y2 + F0
      FF3 = A/4. * Y3**4 + B/3. * Y3**3 + C/2. * Y3**2 + D * Y3 + F0
      IF ((FF1.LE.FF2).AND.(FF1.LE.FF3)) GO TO 65
      IF ((FF2.LE.FF1).AND.(FF2.LE.FF3)) GO TO 64
      DO 61 I = 1, N
   61 X(I) = XX(I) + Y3 * R(I) * H
      FUNC = F(Z)
      GO TO 70
   64 DO 62 I = 1, N
   62 X(I) = XX(I) + Y2 * R(I) * H
      FUNC = F(Z)
      GO TO 70
   65 DO 63 I = 1, N
   63 X(I) = XX(I) + Y1 * R(I) * H
      FUNC = F(Z)
   70 CONTINUE
C
C     FUNC    IS THE FUNCTION VALUE AT THE END OF EACH ITERATION
C
      IF (FUNC - F0) 80, 81, 81
   80 F0 = FUNC
      DO 82 I=1,N
   82 XX(I) = X(I)
      GO TO 72
```

```
   94  CONTINUE
   84 CONTINUE
      GO TO 72
   81  PRINT 323
  323  FORMAT(  *THIS FUNCTION IS  LCCC *   )
   72  IF(J.EC.IT) GO TO 71
       IF(KG.NE.3) GO TO 10
C
C      PRINT OUTPUT
C
   71  PRINT 322,J,FC
  322  FORMAT(* J IS*, I5, 10X, * FUNC IS *, E20.10)
       KG = 0
       IF(MOD(J,3).EQ.0) GO TO 16
       IF(J.NE.IT) GO TO 10
   16 CONTINUE
       DO 600 I=1,N
  600  VAL(I)= EXP(XX(I))
       PRINT 324, (VAL(I),I=1,N)
   10 CONTINUE
       PRINT 322,((DP(I,J),J=1,K2),I=1,K1)
       PUNCH 896,((DP(I,J),J=1,K2),I=1,K1)
  896  FORMAT(10X,7E10.6,/,6E10.6)
       PUNCH 17,VAL
       STOP
       END
```

```
      FUNCTION F(Z)
C*****  F = THE FUNCTION TO BE OPTIMIZED.
C*****  CZ = ARRAY CONTAINING 22 KINETIC PARAMETERS,THEIR DETERMINATION
C*****  IS OUR MAIN OBJECTIVE.
      COMMON CZ(13)
      COMMON/JAIME/E(17,13,6),K1,K2,K3
      COMMON /FIN/DR(17,13),X(13),N
      COMMON /DOG/YNI(17,1,6)
C*****  T = ARRAY CONTAINIG THE 13 TIME INTERVALS.
      DIMENSION T(13)
C*****  NO,TO,TLAST,Y,HO,EPS,MF,KFLAG,YMAX,ERROR,PW,FSAVE ARE
C*****  NO = NO. OF STATE VARIABLES.
C*****  TO = INITIAL TIME
C*****  TLAST-TO = INTERVAL OF INTEGRATION.
C*****  MF REFERS TO THE METHOD OF INTEGRATION.
C*****  MF = 10 IS ADAMS-MOULTON , MF = 22 IS GEARS METHOD.
C*****  EPS IS LOCAL ACCURACY OF INTEGRATION.
C*****  KFLAG,ERROR,FSAVE,PW ARE ARRAYS DESCRIBING SOME INTERNAL
C*****  PARAMETERS OF THE INTEGRATING ROUTINES.
C*****  YMAX = UPPER BOUND ON THE STATE VARIABLES.
      REAL Y(17,13)
      REAL YMAX(17),ERROR(17),PW(306),FSAVE(34)
      COMMON/FISH/PI(13),IEC
      DATA T/5.,8.,9.,10.,12.,16.,21.,24.,27.,31.,35.,38.,41./
      DO 108 J=1,N
      CZ(J)= EXP(X(J))
 108  CONTINUE
      F=0.
      DO 30 L=1,K3
      DO 2 K=1,K1
      Y(K,1)=YNI(K,1,L)
      YMAX(K)=.6
    2 CONTINUE
      MF=22
      NO=K1
      HO=.000000000005
      EPS=.1
      TO=0.
      DO 30 M=1,K2
      IEC=M
      TLAST=T(M)
C*****  ROUTINE DRIVES INTEGRATES THE DIFFERENTIAL EQUATIONS .
      CALL DRIVES(NO,TO,TLAST,Y,HO,EPS,MF,KFLAG,YMAX,ERROR,PW,FSAVE)
      TO=TLAST
      DO 30 K=1,K1
      DR(K,M)=Y(K,1)
      C=DR(K,M)/E(K,M,L)
      F=F+(1.-C)**2
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIFFUN(N,T,Y,YDOT)
C*****  SUBROUTINE DIFFUN CONTAINS EQUATIONS TO BE INTEGRATED WITH
C*****  CURRENT PARAMETER ESTIMATES. IT ALSO CONTAINS THE LINEAR
C*****  RELATIONS BETWEEN THE PARAMETERS. THESE RELATIONS ARE
C*****  OBTAINED USING THE THERMODYNAMIC DATA DISCUSSED IN SEC. 2.2.2.
      COMMON CC(13)
C*****  CZ = ARRAY CONTAINING 22 KINETIC PARAMETERS,THEIR DETERMINATION
C*****  IS OUR MAIN OBJECTIVE.
      DIMENSION CZ(22)
C*****  YDOT(1) TO YDOT(17) ARE THE DERIVATIVES OF THE STATE VARIABLES.
      REAL YDOT(17),Y(17,13)
      COMMON/FISH/PI(13),IEC
      CZ(1)=CC(1)
      CZ(3)=CC(2)
      CZ(4)=CC(3)
      CZ(6)=CC(4)
      CZ(8)=CC(5)
      CZ(10)=CC(6)
      CZ(11)=CC(7)
      CZ(13)=CC(8)
      CZ(15)=CC(9)
      CZ(17)=CC(10)
      CZ(18)=CC(11)
      CZ(20)=CC(12)
      CZ(22)=CC(13)
      CZ(2)=CZ(3)*.0007
      CZ(5)=CZ(4)*20.9
      CZ(7)=CZ(6)*6510.
      CZ(9)=CZ(10)*.0794
      CZ(12)=CZ(11)*12800.
      CZ(14)=CZ(15)*.844
      CZ(16)=CZ(17)*.429
      CZ(19)=CZ(18)*.713
      CZ(21)=CZ(22)*2.32
      YDOT(1)= -CZ(1)*Y(1)  +CZ(20)*Y(3)*Y(16)
      YDOT(4)=-CZ(2)*Y(3)*Y(4)*Y(2)+CZ(3)*Y(6)*Y(7)*Y(5)
      YDOT(2)= YDOT(4)  + CZ(1)*Y(1)
      YDOT(3)=YDOT(4)-CZ(20)*Y(16)*Y(3)
      YDOT(5)=-YDOT(4)+CZ(20)*Y(16)*Y(3)
      YDOT(7)=-YDOT(4)
      YDOT(6)=-CZ(3)*Y(6)*Y(7)*Y(5)-CZ(5)*Y(6)+CZ(4)*Y(8)-CZ(7)*Y(6)*Y(8
     1)+CZ(6)*Y(9) +CZ(2)*Y(3)*Y(4)*Y(2)-CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*Y
     2(12)+CZ(15)*Y(15)*Y(12)-CZ(14)*Y(14)*Y(6)
      YDOT(8)=-CZ(4)*Y(8)+CZ(5)*Y(6)+CZ(6)*Y(9) -CZ(7)*Y(6)*Y(8)+CZ(11)*
     1Y(13)-CZ(12)*Y(11)*Y(8)
      YDOT(9)= -CZ(6)*Y(9) +CZ(7)*Y(6)*Y(8)-CZ(8)*Y(9)
      YDOT(10)=-CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*Y(12)+CZ(8)*Y(9) -CZ(21)*Y
     1(10)+CZ(22)*Y(17)
      YDOT(11)=-CZ(10)*Y(11)*Y(12)+CZ(9)*Y(10)*Y(6)+CZ(11)*Y(13)-CZ(12)*
     1Y(11)*Y(8)
      YDOT(12)=CZ(9)*Y(10)*Y(6)-CZ(10)*Y(11)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(
     115)*Y(15)*Y(12)+CZ(18)*Y(16)-CZ(19)*Y(12)
      YDOT(13)=-CZ(11)*Y(13)+CZ(12)*Y(11)*Y(8)-CZ(13)*Y(13)
```

```
      YDOT(14)=-CZ(14)*Y(14)*Y(6)+CZ(15)*Y(15)*Y(12)+CZ(13)*Y(13)
      YDOT(15)=-CZ(15)*Y(15)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(17)*Y(15)+CZ(16)
     1*Y(16)
      YDOT(16)=+CZ(17)*Y(15)-CZ(16)*Y(16)+CZ(19)*Y(12)-CZ(18)*Y(16)-CZ(2
     10)*Y(16)*Y(3)
      YDOT(17)=-CZ(22)*Y(17)+CZ(21)*Y(10)
      IF(T.EQ.41)GO TO 13
      GO TO 14
13    PRINT 12,(CZ(I),I=1,22)
12    FORMAT (13X, * CZ PARAMETERS VALUES *///(1X,6(E20.10, 1X)//))
   14 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CAT(X)
C*****  THIS SUBROUTINE WILL PERTURB THE EXACT DATA OBTAINED USING
C*****  AN EXACT SET OF PARAMETERS , BY A PREDETERMINED PERCENTAGE.
      COMMON/JAIME/E(17,13,6),K1,K2,K3
      PER=.06
      DO 1 K=1,K3
      DO 2 I=1,K1
      DO 3 J=2,K2
      SIR=RANF(0)-.5
      E(I,J,K)= E(I,J,K)*(1. + SIR*PER)
    3 CONTINUE
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NOISE(X)
C*****  THIS SUBROUTINE WILL PERTURB AN EXACT SET OF PARAMETERS
C*****  (ARBITRARLY CHOSEN) BY A PREDETERMINED PERCENTAGE ACCORDING TO
C*****  THE FORMULA DISCUSSED IN SECTION 2.6.1.
      DIMENSION X(13)
      PER=2.
      DO 31 I=1,13
      SIR=RANF(0)-.5
      X(I)= X(I)*(1. + SIR*PER)
 31   CONTINUE
      RETURN
      END
```

```
      PROGRAM ALFF(INPUT,OUTPUT,PUNCH,TAPE7=PUNCH,TAPE8,TAPE9)
C     THIS PROGRAM USES THE ALTRAN LANGUAGE DISCUSSED IN SECTION  2.4.3.
C
C     IT CALCULATES THE MATRIX D-DOT- DESCRIBED IN SECTION  2.4.1.
C
C     THE PROGRAM PERFORMS FORMAL DIFFERENTIATION OF ALGEBRAIC DATA.
C
C     DIM1= THE NUMBER OF EQUATIONS USED AS ALGEBRAIC DATA.
C
C     DIM2= THE NUMBER OF PARAMETERS IN THE ALGEBRAIC DATA.
C
C     DIM3= THE MAXIMUM NUMBER OF NONZERO ENTRIES OF D-DOT-.
C
C     CZ= NOTATION FOR THE PARAMETERS.
C
C     DY= EQUATIONS OF THE DYNAMICAL SYSTEM(ALGEBRAIC DATA).
C
C
C
C
C
CALTRAN
PROCEDURE MAIN
INTEGER DIM1=17,DIM2=22,DIM3=90
LONG ALGEBRAIC (CZ(1:DIM2):1, Y(1:DIM1):1) DY,A,B,ADD
ALTRAN LONG ALGEBRAIC DERD=$DERD
ARRAY(DIM1) DY
INTEGER ARRAY(DIM1,DIM2)  D
INTEGER I,J, INDEX=DIM1,K
REAL T1,T2,T3,T4,T5,T6,T7,T8
DO I=1,DIM1
READ DY(I)
DOEND
V COMPUTE THE MATRIX B BY CALCULATING THE PARTIAL DERIVATIVES W.R.T. K#S
T2=TIME()
V ONLY THE NONZERO ELEMENTS OF B ARE COMPUTED.  THEY ARE STORED BY ROWS
V ON TAPE 8
V ONLY THE INDEX OF THE INDETERMINATE IS STORE IN D.
WRITE *THE FOLLOWING LIST REPRESENTS THE NONZERO ELEMENTS OF B AND  D*
INDEX=DIM1
DO I=1,DIM1
DO J=1,DIM2
B=DERD(DY(I),CZ(J))
D(I,J)=0
IF(B.EQ.0) GO TO SMOKE
INDEX=INDEX+1
D(I,J)=INDEX
WRITE(8,D(I,J))
WRITE(8) B
SMOKE*A
DOEND
DOEND
T4=TIME(T2)
WRITE *T4 IS THE TIME REQUIRED TO COMPUTE MATRIX B AND DZ,T4
ENDFIL(8)
B=.NULL.
```

```
V  COMPUTE THE MATRIX A BY CALCULATING THE PARTIAL DERIVATIVES OF DY
V ALL THE ELEMENTS OF A ARE STORE BY ROWS ON TAPE 9
T1=TIME()
WRITE *THE FOLLOWING LIST REPRESENTS THE NONZERO ELEMENTS OF A*
DO I=1,DIM1
 DO J=1,DIM1
A=DERD(DY(I),Y(J))
WRITE(9) A
IF(A.EQ.0) GO TO BAT
WRITE I,J,A
BATA
DOEND
DOEND
T2=TIME(T1)
WRITE *T2 IS THE TIME REQUIRED TO COMPUTE MATRIX A *,T2
A=.NULL.
ENDFIL(9)
V PUNCH ELEMENTS OF DY INCOMPATIBLE FORMAT
DO INDEX= 1,DIM1
WRITE INDEX,DY(INDEX)
WRITE(7) INDEX,DY(INDEX)
DOEND
WRITE *ELEMENTS OF MATRIX A*D+B *
REWIND(8)
REWIND(9)
INDEX=DIM1
T7=TIME()
DO I=1,DIM1
V DY IS USED TO STORE A ROW OF  A
DO J=1,DIM1
READ(9) DY(J)
DOEND
DO J=1,DIM2
IF(D(I,J).EQ.0) GO TO BZERO
INDEX=INDEX+1
V PASS NEXT NONZERO ELEMENT OF B
READ(8) ADB
DO K=1,DIM1
IF(D(K,J).EQ.0.OR.DY(K).EQ.0)  GO TO CRY
ADB=ADB+DY(K)*Y(D(K,J))
CRYA
DOEND
WRITE I,J,INDEX,ADB
WRITE(7) INDEX,ADB
BZEROA
DOEND
DOEND
T8=TIME(T7)
WRITE *T8 IS THE TIME REQUIRED TO COMPUTE MATRIX A*D+B AND OUTPUT *,T8
END
.EOF
C      ALGEBRAIC DATA
                -CZ(1)*Y(1)  +CZ(20)*Y(3)*Y(16)
               -CZ(2)*Y(3)*Y(4)*Y(2)+CZ(3)*Y(6)*Y(7)*Y(5)+CZ(1)*Y(1)
               -CZ(2)*Y(3)*Y(4)*Y(2)+CZ(3)*Y(6)*Y(7)*Y(5)-CZ(20)*Y(16)*Y(
3)
```

```
                -CZ(2)*Y(3)*Y(4)*Y(2)+CZ(3)*Y(6)*Y(7)*Y(5)
                 CZ(2)*Y(3)*Y(4)*Y(2)-CZ(3)*Y(6)*Y(7)*Y(5)+CZ(20)*Y(16)*Y(
3)
                -CZ(3)*Y(6)*Y(7)*Y(5)-CZ(5)*Y(6)+CZ(4)*Y(9)-CZ(7)*Y(6)*Y(
8)      +CZ(6)*Y(9) +CZ(2)*Y(3)*Y(4)*Y(2)-CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*
Y(12)      +CZ(15)*Y(15)*Y(12)-CZ(14)*Y(14)*Y(6)
                 CZ(2)*Y(3)*Y(4)*Y(2)-CZ(3)*Y(6)*Y(7)*Y(5)
                -CZ(4)*Y(8)+CZ(5)*Y(6)+CZ(6)*Y(9) -CZ(7)*Y(6)*Y(8)+CZ(11)*
        Y(13)-CZ(12)*Y(11)*Y(8)
                -CZ(6)*Y(9) +CZ(7)*Y(6)*Y(9)-CZ(8)*Y(9)
                -CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*Y(12)+CZ(9)*Y(9) -CZ(21)*
Y(10)      +CZ(22)*Y(17)
                -CZ(10)*Y(11)*Y(12)+CZ(9)*Y(10)*Y(6)+CZ(11)*Y(13)-CZ(12)*
        Y(11)*Y(8)
                 CZ(9)*Y(10)*Y(6)-CZ(10)*Y(11)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(
14)       *Y(15)*Y(13)+CZ(18)*Y(16)-CZ(19)*Y(12)
                -CZ(11)*Y(13)+CZ(12)*Y(11)*Y(8)-CZ(13)*Y(13)
                -CZ(14)*Y(14)*Y(6)+CZ(15)*Y(15)*Y(12)+CZ(13)*Y(13)
                -CZ(15)*Y(15)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(17)*Y(15)+CZ(
16)*Y(16)
                 CZ(17)*Y(15)-CZ(16)*Y(16)+CZ(19)*Y(12)-CZ(18)*Y(16)-CZ(
20)      *Y(16)*Y(3)
                -CZ(22)*Y(17)+CZ(21)*Y(10)
.EOF
```

```
      PROGRAM PLOT(INPUT,OUTPUT,TAPE99)
C     THIS IS  A  PLOTTING ROUTINE WHICH CAN BE IMPLEMENTED AT THE COMPU
C
C     TER CENTER OF THE UNIVERSITY OF CALIFORNIA BERKELEY.
C
C     SPECS,ARE PARAMETERS OF THE GRAPHICAL DISPLAY SYSTEM,THEY SPECIFY
C
C     THE LENGTH OF A COORDINATE SYSTEM,NUMBER OF SUBDIVISIONS,TYPE OF C
C
C     ORDINATES (POLAR,RECTANGULAR),AND ETC.
C
C     THE PROGRAM REQUIRES THAT THE TRAJECTORIES OF THE SOLUTION AND EXP
C
C     ERIMENTAL DATA BE GIVEN.
C
C     Z,Z1,Z2 ARE THE POINTS REPRESENTING 1) AN EXPERIMENTAL DATA POINT
C
C     2) AN INTEGRATED DATA POINT, 3) A POINT ON THE TRAJECTORY WHEN USI
C
C     NG AN INITIAL GUESS OF PARAMETERS VALUES.
C
C     M= NUMBER OF INTERMEDIATES OF THE CALVIN CYCLE.
C
C     N= NUMBER OF DATA POINTS(OR TIME INTERVALS).
C
C
C
C
C
C
C
      DIMENSION SPECS(30),Z(14),Z1(14),T(14),C(17)
      DIMENSION Z2(14)
      DIMENSION E(17,14),DR(17,14),TR(17,14)
      DIMENSION BUFX(500),BUFY(500)
      DIMENSION LINE(17)
      DIMENSION LABC(2)
      DIMENSION GIVEN(3)
      DATA T/0.,5.,8.,9.,10.,12.,16.,21.,24.,27.,31.,35.,38.,41./
      M=17
      N=14
      LABC(2)=0
      READ 3,(LINE(I),I=1,M)
    3 FORMAT(2(8A10/),1A10)
      READ 1,((E(I,J),J=1,N ),I=1,M)
      READ 1,((DR(I,J),J=1,N),I=1,M)
      READ 1,((TR(I,J),J=1,N),I=1,M)
    1 FORMAT(8F10.8,/,6F10.8)
    2 FORMAT (2(8F10.6/),1F10.6)
      SPECS(1)=1.5
      SPECS(2)=1.5
      SPECS(3)=41.
      SPECS(4)=0.
```

```
      SPECS(6)=0.
      SPECS(7)=10.
      SPECS(8)=9.
      SPECS(9)=30.
      SPECS(10)= 10.
      SPECS(12)=99
      SPECS(13)=14
      SPECS(14)=1.
      SPECS(15)=1.
      SPECS(19)=0.
      SPECS(20)=0.
      SPECS(21)=1.
     M=3
      DO 20 J=1,M
      SPECS(17)=.1
      SPECS(18)=.1
      SPECS(25)=0.0
      D=1.
      C=0.
      DO 21 I=1,N
      Z1(I)=E(J,I)
      Z(I)=DR(J,I)
      Z2(I)=TR(J,I)
      IF(Z1(I).GT.C)C=Z1(I)
      IF(Z(I).GT.C)C=Z(I)
      IF(Z2(I).GT.C)C=Z2(I)
      IF(Z1(I).LT.D)D=Z1(I)
      IF(Z(I).LT.D)D=Z(I)
      IF(Z2(I).LT.D)D=Z2(I)
   21 CONTINUE
      SPECS(24)=0.1
      SPECS(26)=0.
      SPECS(16)=1.
      SPECS(11)=1.
      GIVEN(1)=C
      GIVEN(2)=D
      GIVEN(3)=10.
      CALL FABLIY(GIVEN,SPECS)
      CALL AXLILI(SPECS)
      SPECS(28)=1.
      CALL NODLIB(SPECS)
      CALL TITLEE(15HTIME IN MINUTES,SPECS)
      SPECS(28)=6.
      CALL NODLIL(SPECS)
      CALL TITLEL(27HCONCENTRATION IN MILLIMOLAR,SPECS)
      SPECS(17)=.3
      SPECS(18)=.3
      LABC(1)=LINE(J)
      CALL TITLET(LABC,SPECS)
      SPECS(17)=.1
      SPECS(18)=.1
      CALL PSLILI(T,Z,SPECS)
      CALL PELILI(T,Z,BUFX,BUFY,SPECS)
      SPECS(22)=1.5
```

```
      SPECS(23)=.9
      RULE=1.
      CALL SYMKEY(RULE,48HTRAJECTORY OBTAINED USING EXACT PARAMETER VALU
     1ES,SPECS)
      SPECS(16)=13
      CALL PSLILI(T,Z1,SPECS)
      CALL PFLILI(T,Z1,BUFX,BUFY,SPECS)
      SPECS(23)=.6
      CALL SYMKEY(RULE,47HTRAJECTORY OBTAINED USING BEST PARAMETERS FOUN
     1D,SPECS)
      SPECS(16)=5
      CALL PSLILI(T,Z2,SPECS)
      CALL PFLILI(T,Z2,BUFX,BUFY,SPECS)
      SPECS(23)=.3
      CALL SYMKEY(RULE,59HTRAJECTORY OBTAINED USING INITIAL GUESS OF PAR
     1AMETER VALUES,SPECS)
      IF(J.EQ.M)GO TO 20
      CALL NXTERM(SPECS)
   20 CONTINUE
      CALL GDSEND(SPECS)
      STOP
      END
```

```
      PROGRAM ERROR(INPUT,OUTPUT,PUNCH,TAPE6=OUTPUT)
C*****  THIS PROGRAM GIVES AN ESTIMATE OF THE EXPECTED ERROR IN THE
C*****  PARAMETERS DUE TO ERROR IN THE DATA.
C*****  IBOT IS THE FIRST DATA POINT.
C*****  ITOP IS THE LAST  DATA POINT.
      COMMON CZ(13)
      COMMON B,DE
      DIMENSION YNI(17,1,6)
      DIMENSION B(22,22),DE(22,22),E(17,14,6)
      DIMENSION H(22,22),P(22),PI(22),EIVU(22),PP(22)
      DIMENSION F(22,22),T(22,22),S(22,22)
      DIMENSION AINV(22,22),A(22,22)
      DIMENSION WKAREA(572)
      DIMENSION DUMMY(22,22)
      DATA K/22/,M/17/,IBOT/1/,ITOP/13/,K1/6/
      READ 17, (((YNI(I,1,L),I=1,M),L=1,K1)
   17 FORMAT (2(8F10.6/),F10.6)
      READ 16,(CZ(I),I=1,13)
   16 FORMAT(8F10.6,/,5F10.6)
      READ 501,(((E(I,J,K),J=1,ITOP),I=1,M),K=1,K1)
  501 FORMAT(5F10.14,/,5F10.13,/,3F10.13)
      DO 750 I=1,K
      DO 750 J=1,K
      H(I,J)=0
  750 CONTINUE
      DO 21 KK=1,K1
      DO 1 I=1,K
      DO 1 J=1,K
      H(I,J)=0
      DE(I,J)=0
    1 CONTINUE
      DO 2 L=IBOT,ITOP
C*****  INTGRL = A SUBROUTINE TO BE USED TO FIND THE VARIANCE OF THE
C*****  PARAMETERS.
      CALL INTGRL(YNI,L,E,IBOT,K,M,KK)
      DO 51 I=1,4
C*****  CXX=THE DATA ASSUMING 06 PERCENT NOISE IN IT .
      CXX=1./(E(I,L,KK)*E(I,L,KK))
      CXX=CXX*10000./36.
      DO 51 J=1,K
      S(I,J)=CXX*DE(I,J)
      F(J,I)=DE(I,J)
   51 CONTINUE
      IA=K
      IB=M
      IC=K
C*****  SUBROUTINE MAMUL PERFORMS MATRIX MULTIPLICATIONS.
      CALL MAMUL(B,S,IA,IB,IC,T)
      DO 41 I=1,K
      DO 41 J=1,K
C*****  CALCULATION OF THE MATRIX H (DISCUSSED IN SEC. 2.4.1. )
      H(I,J)= H(I,J) + T(I,J)
      A(I,J)=H(I,J)
      P(I,J)=H(I,J)/6.
   41 CONTINUE
    2 CONTINUE
```

```
 21   CONTINUE
      DO 100 J=1,K
      DO 100 I=1,K
      PRINT 11,J,I,H(J,I)
 11   FORMAT(1H ,I5,I5,E16.8)
100   CONTINUE
      NDIM=22
      N=22
      NOYES=0
C*****  HOGR= ROUTINE TO FIND EIGENVALUES OF MATRIX P.
      CALL HOGR(H,NDIM,N,EIVU,NOYES,DUMMY)
      DO 304 I=1,K
      EIVU(I)= 1.0/EIVU(I)
 304  CONTINUE
      PRINT 302,(EIVU(I),I=1,K)
 302  FORMAT(6X, * EIGENVALUES OF P * ///(1X,6E20.10,1X)//)
      N=K
      IA=K
      IDGT=0
C*****  LINV2F = ROUTINE TO FIND THE INVERSE OF H (WHICH IS THE MATRIX
C*****  P DISCUSSED IN SECTION 2.4.1.)
      CALL LINV2F(A,N,IA,AINV,IDGT,WKAREA,IER)
      DO 99 I=1,K
      P(I)=AINV(I,I)
      DO 99 J=1,K
      PRINT 11,I,J,AINV(I,J)
 99   CONTINUE
      DO  97 I=1,K
      PI(I)=P(I)
 97   CONTINUE
      PRINT 306, (PI(I),I=1,K)
 306  FORMAT(6X, * VARIANCE WITH SIX INITIAL VALUES *///(1X,6E20.10,1X)/
     1/)
      N=K
      IA=K
      IDGT=0
      CALL LINV2F(R,N,IA,AINV,IDGT,WKAREA,IER)
      DO 101 I=1,K
      PP(I)=AINV(I,I)
      DO 101 J=1,K
      PRINT 11,I,J,AINV(I,J)
 101  CONTINUE
C*****  PRINT OUT OF THE VARIANCE OF P. (THE EXPECTED ERROR IN THE
C*****  PARAMETERS.)
      PRINT 305,(PP(I),I=1,K)
 305  FORMAT(6X, * VARIANCE OF PARAMETERS *///(1X,6E20.10,1X)//)
      STOP
      END
      SUBROUTINE MAMUL(B,S,IA,IB,IC,T)
      DIMENSION B(22,22),T(22,22),S(22,22)
      DO 1 I=1,IA
      DO 1 J=1,IC
      T(I,J)=0
   1  CONTINUE
      DO 2 I=1,IA
      DO 2 J=1,IC
```

```
      DO 2 LK=1,IB
      T(I,J)= T(I,J) + B(I,LK)*S(LK,J)
    2 CONTINUE
      RETURN
      END
      SUBROUTINE YOUT(NO,Y,TLAST,T,N,MF)
      COMMON/STCOM2/M,NQ,H
      REAL Y(NO,13)
      RETURN
      END
      SUBROUTINE INTGRL(YNI,L,E,IBOT,K,M,KK)
      COMMON CZ(13)
      COMMON B,DE
      DIMENSION B(22,22),DE(22,22)
      DIMENSION YNI(17,1,6),E(17,14,6)
      DIMENSION T(13)
C*****  YDOT(18) TO YDOT(82) ARE THE NON ZERO ELEMENTS OF THE ENTERIES
C*****  OF THE D-MATRIX DISCUSSED IN SECTION 2.4.1.
      REAL Y(82,13)
      REAL YMAX(82),ERROR(82),PW(6,6),FSAVE(154)
      DATA T/1.,2.,3.,4.,5.,6.,8.,10.,12.,15.,20.,25.,30./
      MOM=82
      DO 19 I=1,MOM
      YMAX(I)=1.
   19 CONTINUE
      IF(L-IBOT)1,1,2
    1 CONTINUE
      DO 3 I=18,MOM
      Y(I,1)=0.
    3 CONTINUE
      DO 21 I=1,M
      Y(I,1)=YNI(I,1,KK)
   21 CONTINUE
      DO 4 I=1,K
      DO 4 J=1,K
      DE(I,J)=0
    4 CONTINUE
      NO=MOM
      HO=0.001
      EPS=0.001
      TO=0.
      MF=22
    2 CONTINUE
      TLAST=T(L)
      CALL DRIVES(NO,TO,TLAST,Y,HO,EPS,MF,KFLAG,YMAX,ERROR,PW,FSAVE)
      DO 30 I=1,4
      E(I,L,KK)=Y(I,1)
   30 CONTINUE
      DE(1,1)=Y(18)
      DE(1,2)=Y(19)
      DE(2,1)=Y(20)
      DE(2,2)=Y(21)
      DE(2,3)=Y(22)
      DE(3,2)=Y(23)
      DE(3,3)=Y(24)
      DE(3,4)=Y(25)
```

```
DE(4,2)=Y(26)
DE(4,3)=Y(27)
DE(5,2)=Y(28)
DE(5,3)=Y(29)
DE(5,20)=Y(30)
DE(6,2)=Y(31)
DE(6,3)=Y(32)
DE(6,4)=Y(33)
DE(6,5)=Y(34)
DE(6,6)=Y(35)
DE(6,7)=Y(36)
DE(6,9)=Y(37)
DE(6,10)=Y(38)
DE(6,14)=Y(39)
DE(6,15)=Y(40)
DE(7,2)=Y(41)
DE(7,3)=Y(42)
DE(8,4)=Y(43)
DE(8,5)=Y(44)
DE(8,6)=Y(45)
DE(8,7)=Y(46)
DE(8,11)=Y(47)
DE(8,12)=Y(48)
DE(9,6)=Y(49)
DE(9,7)=Y(50)
DE(9,8)=Y(51)
DE(10,8)=Y(52)
DE(10,9)=Y(53)
DE(10,10)=Y(54)
DE(10,21)=Y(55)
DE(10,22)=Y(56)
DE(11,9)=Y(57)
DE(11,10)=Y(58)
DE(11,11)=Y(59)
DE(11,12)=Y(60)
DE(12,9)=Y(61)
DE(12,10)=Y(62)
DE(12,14)=Y(63)
DE(12,18)=Y(64)
DE(12,19)=Y(65)
DE(13,11)=Y(66)
DE(13,12)=Y(67)
DE(13,13)=Y(68)
DE(14,13)=Y(69)
DE(14,14)=Y(70)
DE(14,15)=Y(71)
DE(15,14)=Y(72)
DE(15,15)=Y(73)
DE(15,16)=Y(74)
DE(15,17)=Y(75)
DE(16,16)=Y(76)
DE(15,17)=Y(77)
DE(16,18)=Y(78)
DE(16,19)=Y(79)
DE(16,20)=Y(80)
DE(17,21)=Y(81)
```

```
      DE(17,22)=Y(82)
      RETURN
      END
      SUBROUTINE DIFFUN(N,T,Y,YDOT)
      COMMON CC(13)
      DIMENSION CZ(22)
      REAL YDOT(82),Y(82,13)
      CZ(1)=CC(1)
      CZ(2)=CC(2)
      CZ(5)=CC(3)
      CZ(7)=CC(4)
      CZ(8)=CC(5)
      CZ(9)=CC(6)
      CZ(12)=CC(7)
      CZ(13)=CC(8)
      CZ(14)=CC(9)
      CZ(17)=CC(10)
      CZ(19)=CC(11)
      CZ(20)=CC(12)
      CZ(21)=CC(13)
      CZ(3)= CZ(2)*1425.05
      CZ(4)=CZ(5)*.04783
      CZ(6)=CZ(7)*.000153
      CZ(10)=CZ(9) *12.59
      CZ(11)=CZ(12)*.000078
      CZ(15)=CZ(14)*1.18
      CZ(16)=CZ(17)*2.32
      CZ(18)=CZ(19)*1.4
      CZ(22)=CZ(21)*.42
C*****  YDOT= DERIVATIVES OF THE STATE VARIABLES.
      YDOT(1)= -CZ(1)*Y(1)  +CZ(20)*Y(3)*Y(16)
      YDOT(4)=-CZ(2)*Y(3)*Y(4)*Y(2)+CZ(3)*Y(6)*Y(7)*Y(5)
      YDOT(2)= YDOT(4)  + CZ(1)*Y(1)
      YDOT(3)=YDOT(4)-CZ(20)*Y(16)*Y(3)
      YDOT(5)=-YDOT(4)+CZ(20)*Y(16)*Y(3)
      YDOT(7)=-YDOT(4)
      YDOT(6)=-CZ(3)*Y(6)*Y(7)*Y(5)-CZ(5)*Y(6)+CZ(4)*Y(8)-CZ(7)*Y(6)*Y(8
     1)+CZ(6)*Y(9) +CZ(2)*Y(3)*Y(4)*Y(2)-CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*Y
     2(12)+CZ(15)*Y(15)*Y(12)-CZ(14)*Y(14)*Y(6)
      YDOT(8)=-CZ(4)*Y(8)+CZ(5)*Y(6)+CZ(6)*Y(5) -CZ(7)*Y(6)*Y(8)+CZ(11)*
     1Y(13)-CZ(12)*Y(11)*Y(8)
      YDOT(9)= -CZ(6)*Y(9) +CZ(7)*Y(6)*Y(8)-CZ(8)*Y(9)
      YDOT(10)=-CZ(9)*Y(10)*Y(6)+CZ(10)*Y(11)*Y(12)+CZ(8)*Y(9) -CZ(21)*Y
     1(10)+CZ(22)*Y(17)
      YDOT(11)=-CZ(10)*Y(11)*Y(12)+CZ(9)*Y(10)*Y(6)+CZ(11)*Y(13)-CZ(12)*
     1Y(11)*Y(8)
      YDOT(12)=CZ(9)*Y(10)*Y(6)-CZ(10)*Y(11)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(
     114)*Y(15)*Y(13)+CZ(18)*Y(16)-CZ(19)*Y(12)
      YDOT(13)=-CZ(11)*Y(13)+CZ(12)*Y(11)*Y(8)-CZ(13)*Y(13)
      YDOT(14)=-CZ(14)*Y(14)*Y(6)+CZ(15)*Y(15)*Y(12)+CZ(13)*Y(13)
      YDOT(15)=-CZ(15)*Y(15)*Y(12)+CZ(14)*Y(14)*Y(6)-CZ(17)*Y(15)+CZ(16)
     1*Y(16)
      YDOT(16)=+CZ(17)*Y(15)-CZ(16)*Y(16)+CZ(19)*Y(12)-CZ(18)*Y(16)-CZ(2
     10)*Y(16)*Y(3)
      YDOT(17)=-CZ(22)*Y(17)+CZ(21)*Y(10)
      YDOT(18)=
```

```
C*****  EACH OF THE FOLLOWING EQUATIONS IS A NON-ZERO ENTRY OF THE
C*****  MATRIX D.
      1 - CZ(1)*Y(19) - Y(1)
       YDOT(19)=
      1 - CZ(1)*Y(19) + CZ(20)*Y(3)*Y(30) + CZ(20)*Y(16)*Y(25) + Y(3)*Y(1
      16)
       YDOT(20)=
      1 CZ(1)*Y(18) - CZ(2)*Y(3)*Y(4)*Y(20) + Y(1)
       YDOT(21)=
      1 - CZ(2)*Y(2)*Y(3)*Y(26) - CZ(2)*Y(2)*Y(4)*Y(23) - CZ(2)*Y(3)*Y(4)
      1*Y(21) + CZ(3)*Y(5)*Y(6)*Y(41) + CZ(3)*Y(5)*Y(7)*Y(31)
      1         - Y(2)*Y(3)*Y(4)+CZ(3)*Y(6)*Y(7)*Y(28)
       YDOT(22)=
      1 - CZ(2)*Y(2)*Y(3)*Y(27) - CZ(2)*Y(2)*Y(4)*Y(24) - CZ(2)*Y(3)*Y(4)
      1*Y(22) + CZ(3)*Y(5)*Y(6)*Y(42) + CZ(3)*Y(5)*Y(7)*Y(32)
      1         + Y(5)*Y(6)*Y(7)+CZ(3)*Y(6)*Y(7)*Y(29)
       YDOT(23)=
      1 - CZ(2)*Y(2)*Y(3)*Y(26) - CZ(2)*Y(2)*Y(4)*Y(23) - CZ(2)*Y(3)*Y(4)
      1*Y(21) + CZ(3)*Y(5)*Y(6)*Y(41) + CZ(3)*Y(5)*Y(7)*Y(31)
      1         - CZ(20)*Y(16)*Y(23) - Y(2)*Y(3)*Y(4)+CZ(3)*Y(6)*Y(7)*Y(28)
       YDOT(24)=
      1 - CZ(2)*Y(2)*Y(3)*Y(27) - CZ(2)*Y(2)*Y(4)*Y(24) - CZ(2)*Y(3)*Y(4)
      1*Y(22) + CZ(3)*Y(5)*Y(6)*Y(42) + CZ(3)*Y(5)*Y(7)*Y(32)
      1         - CZ(20)*Y(16)*Y(24) + Y(5)*Y(6)*Y(7)+CZ(3)*Y(6)*Y(7)*Y(29)
       YDOT(25)=
      1 - CZ(2)*Y(2)*Y(4)*Y(25) + CZ(3)*Y(6)*Y(7)*Y(30)
      1-CZ(20)*Y(16)*Y(25) - Y(3)*Y(16)-CZ(20)*Y(3)*Y(30)
       YDOT(26)=
      1 - CZ(2)*Y(2)*Y(3)*Y(26) - CZ(2)*Y(2)*Y(4)*Y(23) - CZ(2)*Y(3)*Y(4)
      1*Y(21) + CZ(3)*Y(5)*Y(6)*Y(41) + CZ(3)*Y(5)*Y(7)*Y(31)
      1         - Y(2)*Y(3)*Y(4) +CZ(3)*Y(6)*Y(7)*Y(28)
       YDOT(27)=
      1 - CZ(2)*Y(2)*Y(3)*Y(27) - CZ(2)*Y(2)*Y(4)*Y(24) - CZ(2)*Y(3)*Y(4)
      1*Y(22) + CZ(3)*Y(5)*Y(6)*Y(42) + CZ(3)*Y(5)*Y(7)*Y(32)
      1         + Y(5)*Y(6)*Y(7)+CZ(3)*Y(6)*Y(7)*Y(29)
       YDOT(28)=
      1 CZ(2)*Y(2)*Y(3)*Y(26) + CZ(2)*Y(2)*Y(4)*Y(23)
      1-CZ(3)*Y(5)*Y(6)*Y(41) - CZ(3)*Y(5)*Y(7)*Y(31)
      1+CZ(20)*Y(16)*Y(23) + Y(2)*Y(3)*Y(4)+CZ(2)*Y(3)*Y(4)*Y(21)-
      1CZ(3)*Y(3)*Y(4)*Y(21)
       YDOT(29)=
      1 CZ(2)*Y(2)*Y(3)*Y(27) + CZ(2)*Y(2)*Y(4)*Y(24)
      1-CZ(3)*Y(5)*Y(6)*Y(42) - CZ(3)*Y(5)*Y(7)*Y(32)
      1+CZ(20)*Y(16)*Y(24) - Y(5)*Y(6)*Y(7)+CZ(2)*Y(3)*Y(4)*Y(22)-
      1CZ(3)*Y(6)*Y(7)*Y(29)
       YDOT(30)=
      1 CZ(2)*Y(2)*Y(4)*Y(25) - CZ(3)*Y(6)*Y(7)*Y(30) + CZ(20)*Y(3)*Y(30)
      1+CZ(20)*Y(16)*Y(25) + Y(3)*Y(16)
       YDOT(31)=
      1 CZ(2)*Y(2)*Y(3)*Y(26) + CZ(2)*Y(2)*Y(4)*Y(23)
      1+ CZ(2)*Y(3)*Y(4)*Y(21) -CZ(3)*Y(6)*Y(7)*Y(28)
      1-CZ(3)*Y(5)*Y(6)*Y(41) - CZ(3)*Y(5)*Y(7)*Y(31)
      1-CZ(5)*Y(31) - CZ(7)*Y(8)*Y(31) - CZ(9)*Y(10)*Y(31) - CZ(14)*Y(14)
      1*Y(31) + Y(2)*Y(3)*Y(4)
       YDOT(32)=
      1 CZ(2)*Y(2)*Y(3)*Y(27) + CZ(2)*Y(2)*Y(4)*Y(24)
```

```
1-CZ(3)*Y(5)*Y(6)*Y(42) - CZ(3)*Y(5)*Y(7)*Y(32)
1+CZ(2)*Y(3)*Y(4)*Y(22)-CZ(3)*Y(6)*Y(7)*Y(22)
1-CZ(5)*Y(32) - CZ(7)*Y(8)*Y(32) - CZ(9)*Y(10)*Y(32) - CZ(14)*Y(14)
1*Y(32) - Y(5)*Y(6)*Y(7)
 YDOT(33)=
1 - CZ(3)*Y(5)*Y(7)*Y(33) + CZ(4)*Y(43) - CZ(5)*Y(33) -
1 CZ(7)*Y(8)*Y(33) - CZ(9)*Y(10)*Y(33) - CZ(14)*Y(14)*Y(33) + Y(8)
1-CZ(7)*Y(6)*Y(43)
 YDOT(34)=
1 - CZ(3)*Y(5)*Y(7)*Y(34) + CZ(4)*Y(44) - CZ(5)*Y(34) -
1 CZ(7)*Y(8)*Y(34) - CZ(9)*Y(10)*Y(34) - CZ(14)*Y(14)*Y(34) - Y(6)
1-CZ(7)*Y(6)*Y(44)
 YDOT(35)=
1 - CZ(3)*Y(5)*Y(7)*Y(35) + CZ(4)*Y(45) - CZ(5)*Y(35) + CZ(6)*Y(49)
1-CZ(7)*Y(6)*Y(45) - CZ(7)*Y(8)*Y(35) - CZ(9)*Y(10)*Y(35) -
1 CZ(14)*Y(14)*Y(35)+Y(9)
 YDOT(36)=
1 - CZ(3)*Y(5)*Y(7)*Y(36) + CZ(4)*Y(46) - CZ(5)*Y(36) + CZ(6)*Y(50)
1-CZ(7)*Y(6)*Y(46) - CZ(7)*Y(8)*Y(36) - CZ(9)*Y(10)*Y(36) -
1CZ(14)*Y(14)*Y(36)-Y(6)*Y(8)
 YDOT(37)=
1 - CZ(3)*Y(5)*Y(7)*Y(37) - CZ(5)*Y(37) - CZ(7)*Y(8)*Y(37)
1       - CZ(9)*Y(10)*Y(37) + CZ(10)*Y(11)*Y(61) -
1 CZ(14)*Y(14)*Y(37) + CZ(15)*Y(15)*Y(61) - Y(6)*Y(10)
1-CZ(9)*Y(6)*Y(53)+CZ(10)*Y(12)*Y(57)
 YDOT(38)=
1 - CZ(3)*Y(5)*Y(7)*Y(38) - CZ(5)*Y(38) - CZ(7)*Y(8)*Y(38)
1       - CZ(9)*Y(10)*Y(38) + CZ(10)*Y(11)*Y(62)
1-CZ(14)*Y(14)*Y(38) + CZ(15)*Y(15)*Y(62) + Y(11)*Y(12)
1 -CZ(9)*Y(6)*Y(54) +CZ(10)*Y(12)*Y(58)
 YDOT(39)=
1 - CZ(3)*Y(5)*Y(7)*Y(39) - CZ(5)*Y(39) - CZ(7)*Y(8)*Y(39)
1       + CZ(10)*Y(11)*Y(63) - CZ(14)*Y(6)*Y(70)
1+CZ(15)*Y(12)*Y(72) + CZ(15)*Y(15)*Y(63) - Y(6)*Y(14)
1 -CZ(9)*Y(10)*Y(39) - CZ(14)*Y(14)*Y(39)
 YDOT(40)=
1 - CZ(3)*Y(5)*Y(7)*Y(40) - CZ(5)*Y(40) - CZ(7)*Y(8)*Y(40)
1       - CZ(14)*Y(6)*Y(71) - CZ(14)*Y(14)*Y(40)
1 -CZ(9)*Y(10)*Y(40) + CZ(15)*Y(12)*Y(73)+Y(12)*Y(15)
 YDOT(41)=
1 CZ(2)*Y(2)*Y(3)*Y(26) + CZ(2)*Y(2)*Y(4)*Y(23) -
1 CZ(3)*Y(5)*Y(6)*Y(41) - CZ(3)*Y(5)*Y(7)*Y(31) +
1CZ(2)*Y(3)*Y(4)*Y(21) - CZ(3)*Y(6)*Y(7)*Y(28)+Y(2)*Y(3)*Y(4)
 YDOT(42)=
1 CZ(2)*Y(2)*Y(3)*Y(27) + CZ(2)*Y(2)*Y(4)*Y(24)
1-CZ(3)*Y(5)*Y(6)*Y(42) - CZ(3)*Y(5)*Y(7)*Y(32)
1 +CZ(2)*Y(3)*Y(4)*Y(22) - CZ(3)*Y(6)*Y(7)*Y(29) - Y(5)*Y(6)*Y(7)
 YDOT(43)=
1 - CZ(4)*Y(43) + CZ(5)*Y(33) - CZ(7)*Y(6)*Y(43) - CZ(7)*Y(8)*Y(33)
1-CZ(12)*Y(11)*Y(43) - Y(8)
 YDOT(44)=
1 - CZ(4)*Y(44) + CZ(5)*Y(34) - CZ(7)*Y(6)*Y(44) - CZ(7)*Y(8)*Y(34)
1-CZ(12)*Y(11)*Y(44) + Y(6)
 YDOT(45)=
1 - CZ(4)*Y(45) + CZ(5)*Y(35) + CZ(6)*Y(49) - CZ(7)*Y(6)*Y(45) -
1 CZ(7)*Y(8)*Y(35) - CZ(12)*Y(11)*Y(45) + Y(9)
```

```
      YDOT(46)=
     1  - CZ(4)*Y(46) + CZ(5)*Y(36) + CZ(6)*Y(50) - CZ(7)*Y(6)*Y(46) -
     1  CZ(7)*Y(8)*Y(36) - CZ(12)*Y(11)*Y(46) - Y(6)*Y(8)
      YDOT(47)=
     1  - CZ(4)*Y(47) - CZ(7)*Y(6)*Y(47) + CZ(11)*Y(66) -
     1  CZ(12)*Y(11)*Y(47) + Y(13) -CZ(12)*Y(8)*Y(59)
      YDOT(48)=
     1  - CZ(4)*Y(48) - CZ(7)*Y(6)*Y(43) + CZ(11)*Y(67) -
     1  CZ(12)*Y(11)*Y(48) - Y(8)*Y(11)-CZ(12)*Y(8)*Y(60)
      YDOT(49)=
     1  - CZ(6)*Y(49) + CZ(7)*Y(6)*Y(45) + CZ(7)*Y(8)*Y(35) - CZ(8)*Y(49)
     1  -Y(9)
      YDOT(50)=
     1  - CZ(6)*Y(50) + CZ(7)*Y(6)*Y(46) + CZ(7)*Y(8)*Y(36) - CZ(8)*Y(50)
     1 +Y(6)*Y(8)
      YDOT(51)=
     1  - CZ(6)*Y(51) - CZ(8)*Y(51) - Y(9)
      YDOT(52)=
     1  CZ(8)*Y(51) - CZ(9)*Y(6)*Y(52) - CZ(21)*Y(52) + Y(9)
      YDOT(53)=
     1  - CZ(9)*Y(6)*Y(53) - CZ(9)*Y(10)*Y(37) + CZ(10)*Y(11)*Y(61) +
     1  CZ(10)*Y(12)*Y(57) - CZ(21)*Y(53) - Y(6)*Y(10)
      YDOT(54)=
     1  - CZ(9)*Y(6)*Y(54) - CZ(9)*Y(10)*Y(38) + CZ(10)*Y(11)*Y(62) +
     1  CZ(10)*Y(12)*Y(58) - CZ(21)*Y(54) + Y(11)*Y(12)
      YDOT(55)=
     1  - CZ(9)*Y(6)*Y(55) - CZ(21)*Y(55) + CZ(22)*Y(81) - Y(10)
      YDOT(56)=
     1  - CZ(9)*Y(6)*Y(56) - CZ(21)*Y(56) + CZ(22)*Y(82) + Y(17)
      YDOT(57)=
     1  CZ(9)*Y(6)*Y(53) + CZ(9)*Y(10)*Y(37) - CZ(10)*Y(11)*Y(61) -
     1  CZ(10)*Y(12)*Y(57) - CZ(12)*Y(8)*Y(57) + Y(6)*Y(10)
      YDOT(58)=
     1  CZ(9)*Y(6)*Y(54) + CZ(9)*Y(10)*Y(38) - CZ(10)*Y(11)*Y(62) -
     1  CZ(10)*Y(12)*Y(58) - CZ(12)*Y(8)*Y(58) - Y(11)*Y(12)
      YDOT(59)=
     1  - CZ(10)*Y(12)*Y(59) + CZ(11)*Y(66) - CZ(12)*Y(8)*Y(59) - CZ(12)*
     1Y(11)*Y(47) + Y(13)
      YDOT(60)=
     1  - CZ(10)*Y(12)*Y(60) + CZ(11)*Y(67) - CZ(12)*Y(8)*Y(60) - CZ(12)*
     1Y(11)*Y(48) - Y(8)*Y(11)
      YDOT(61)=
     1  CZ(9)*Y(6)*Y(53) + CZ(9)*Y(10)*Y(37) - CZ(10)*Y(11)*Y(61) -
     1  CZ(10)*Y(12)*Y(57) + CZ(14)*Y(14)*Y(37) - CZ(19)*Y(61) + Y(6)*Y(1
     10)
      YDOT(62)=
     1  CZ(9)*Y(6)*Y(54) + CZ(9)*Y(10)*Y(38) - CZ(10)*Y(11)*Y(62) -
     1  CZ(10)*Y(12)*Y(58) + CZ(14)*Y(14)*Y(38) - CZ(19)*Y(62) - Y(11)*Y(
     112)
      YDOT(63)=
     1  CZ(9)*Y(10)*Y(39) - CZ(10)*Y(11)*Y(63) + CZ(14)*Y(5)*Y(70) -
     1CZ(14)*Y(13)*Y(72) + CZ(14)*Y(14)*Y(39) - CZ(19)*Y(63)+Y(6)*Y(14)
     1-Y(13)*Y(15)
      YDOT(64)=
     1  - CZ(10)*Y(11)*Y(64) + CZ(15)*Y(73) - CZ(19)*Y(64) + Y(15)
      YDOT(65)=
```

```
     1 - CZ(10)*Y(11)*Y(65) + CZ(18)*Y(79) - CZ(19)*Y(65) - Y(12)
      YDOT(66)=
     1 - CZ(11)*Y(66) + CZ(12)*Y(8)*Y(59) + CZ(12)*Y(11)*Y(47) - CZ(13)*
     1Y(66) - Y(13)
      YDOT(67)=
     1 - CZ(11)*Y(67) + CZ(12)*Y(8)*Y(60) + CZ(12)*Y(11)*Y(48) - CZ(13)*
     1Y(67) + Y(8)*Y(11)
      YDOT(68)=
     1 - CZ(11)*Y(68) - CZ(13)*Y(68) - Y(13)
      YDOT(69)=
     1 CZ(13)*Y(68) - CZ(14)*Y(6)*Y(69) + Y(13)
      YDOT(70)=
     1 - CZ(14)*Y(6)*Y(70) - CZ(14)*Y(14)*Y(39) + CZ(15)*Y(12)*Y(72) +
     1 CZ(15)*Y(15)*Y(63) - Y(6)*Y(14)
      YDOT(71)=
     1 - CZ(14)*Y(6)*Y(71) - CZ(14)*Y(14)*Y(40) + CZ(15)*Y(12)*Y(73) +
     1 Y(12)*Y(15)
      YDOT(72)=
     1 CZ(14)*Y(6)*Y(70) + CZ(14)*Y(14)*Y(39) - CZ(15)*Y(12)*Y(72) -
     1 CZ(15)*Y(15)*Y(63) - CZ(17)*Y(72) + Y(6)*Y(14)
      YDOT(73)=
     1 CZ(14)*Y(6)*Y(71) + CZ(14)*Y(14)*Y(40) - CZ(15)*Y(12)*Y(73) -
     1 CZ(17)*Y(73) - Y(12)*Y(15)
      YDOT(74)=
     1 - CZ(15)*Y(12)*Y(74) + CZ(16)*Y(76) - CZ(17)*Y(74) + Y(16)
      YDOT(75)=
     1 - CZ(15)*Y(12)*Y(75) + CZ(16)*Y(77) - CZ(17)*Y(75) - Y(15)
      YDOT(76)=
     1 - CZ(16)*Y(76) + CZ(17)*Y(74) - CZ(18)*Y(76) - CZ(20)*Y(3)*Y(76)
     1 -Y(16)
      YDOT(77)=
     1 - CZ(16)*Y(77) + CZ(17)*Y(75) - CZ(18)*Y(77) - CZ(20)*Y(3)*Y(77)
     1+Y(15)
      YDOT(78)=
     1 - CZ(16)*Y(78) - CZ(18)*Y(78) + CZ(19)*Y(64) - CZ(20)*Y(3)*Y(78)
     1-Y(15)
      YDOT(79)=
     1 - CZ(16)*Y(79) - CZ(18)*Y(79) + CZ(19)*Y(65) - CZ(20)*Y(3)*Y(79)
     1+Y(12)
      YDOT(80)=
     1 - CZ(16)*Y(80) - CZ(18)*Y(80) - CZ(20)*Y(3)*Y(80) - CZ(20)*Y(16)*
     1Y(25) - Y(3)*Y(16)
      YDOT(81)=
     1 CZ(21)*Y(55) - CZ(22)*Y(81) + Y(10)
      YDOT(82)=
     1 CZ(21)*Y(56) - CZ(22)*Y(82) - Y(17)
      RETURN
      END
```

```
      PROGRAM MODMAK (INPUT,OUTPUT)
C     THIS PROGRAM WILL GENERATE THE DYNAMIC EQUATIONS USING THE RELATIO
CCC   NS DISCUSSED IN SECTION 2.1.4.
CCC   THE PROGRAM IS SET UP  TO GENERATE  UP TO 100 DIFFERENT INTERACTIO          N
CCC   THE PROGRAM IS WRITTEN IN FORTRAN AND COMPASS(MACHINE LANGUAGE).
CCC   TO GENERATE THE EQUATIONS THE FOLLOWING DATA IS REQUIRED:
CCC   THE NAMES OF THE INTERACTING SPECIES(DENOTED AS THREE ALPHANUMERIC
CCC   CHARACTERS).
CCC   THE SPECIES INTERACTION EQUATIONS.THIS EQUATIONS FOLLOW THE LOGICA
CCC   L STRUCTURE OF THE SCHEMATIC REPRESENTATION OF SECTION 2.1.4.
C
      REAL K(100),X(20),DIFFX(20)
      INTEGER CHARAC(80),LOCATE,NTERMS,NERRORS,SPETAB2(20)
      COMMON /COM2/LOCATE,NTERMS /COM3/SPETAB2
 1000 FORMAT(1H1,*MATHEMATICAL MODEL*//)
 1001 FORMAT(8A10)
 1002 FORMAT(80R1)
 1003 FORMAT(/1X,8A10)
 1004 FORMAT(1X,80R1)
 1005 FORMAT(/1X,*SPECIES LIST OVERFLOW*/)
 1006 FORMAT(/1X,*INTERACTION LIST OVERFLOW*/)
 1007 FORMAT(1X,*UNDEFINED SPECIES*/)
 1008 FORMAT(/1X,*CONTROL CARD ERROR, JOB TERMINATED*)
 1009 FORMAT(1X,*SYNTAX ERROR*/)
 1010 FORMAT(/1X,*DIFFERENTIAL EQUATION TOO LARGE, JOB TERMINATED*)
 1011 FORMAT(1X,*NUMBER OF ERRORS IS *,I4,* JOB TERMINATED*)
 1012 FORMAT(/1X,*D(*,A3,*)/DT = *,5(R1,*K(*,I3,*)*,A10,2X),(/14X,5(R1,*
     1K(*,I3,*)*,A10,2X)))
 1013 FORMAT(/1X,*PROGRAM OVERFLOW, JOB TERMINATED*)
 1014 FORMAT(E12.5)
 1015 FORMAT(/(1X,*RATE CONSTANT K(*,I3,*) = *,E12.5))
 1016 FORMAT(/(1X,A3,* = *,E12.5))
 1017 FORMAT(/(1X,*D(*,A3,*)/DT = *,E12.5))
C
      DO  900 I=1,2
      PRINT 1000
      NERRORS=0
      READ 1001,(CHARAC(I), I=1,8)
      IF(CHARAC(1) .EQ. 10HSPECIES LI .AND. CHARAC(2) .EQ. 10HST
     1) GO TO 1
      PRINT 1008
      STOP
    1 PRINT 1003,(CHARAC(I), I=1,8)
      READ 1002,CHARAC
```

```
      PRINT 1004,CHARAC
      CALL MOD1(CHARAC)
    2 IF(CHARAC(3) .EQ. 0) GO TO 3
      IF(CHARAC(1) .EQ. 0) GO TO 5
      IF(CHARAC(2) .EQ. 0) GO TO 4
      READ 1002,CHARAC
      PRINT 1004,CHARAC
      CALL ALT1(CHARAC)
      GO TO 2
    3 PRINT 1005
      NERRORS=NERRORS+1
      GO TO 5
    4 PRINT 1009
      NERRORS=NERRORS+1
    5 READ 1001,(CHARAC(I), I=1,8)
      IF(CHARAC(1) .EQ. 10HSPECIES IN .AND. CHARAC(2) .EQ. 10HTERACTION
     1 .AND. CHARAC(3) .EQ. 10HLIST      ) GO TO 6
      PRINT 1008
      STOP
    6 PRINT 1003,(CHARAC(I), I=1,8)
      READ 1002,CHARAC
      PRINT 1004,CHARAC
      CALL MOD2(CHARAC)
    7 IF(CHARAC(1) .NE. 0) GO TO 8
      IF(CHARAC(6) .EQ. 0) GO TO 12
      IF(LOCATE .EQ. 1001) GO TO 14
      IF(CHARAC(2) .NE. 0) GO TO 9
      PRINT 1009
      NERRORS=NERRORS+1
    9 IF(CHARAC(3) .NE. 0) GO TO 10
      PRINT 1007
      NERRORS=NERRORS+1
   10 IF(CHARAC(4) .NE. 0) GO TO 11
      PRINT 1006
      NERRORS=NERRORS+1
   11 IF(CHARAC(5) .NE. 0) GO TO 8
      PRINT 1013
    8 READ 1002,CHARAC
      PRINT 1004,CHARAC
      CALL ALT2(CHARAC)
      GO TO 7
   12 IF(NERRORS .GT. 0) GO TO 16
      N=1
      CALL MOD3(CHARAC,N)
   13 PRINT 1012,(CHARAC(I), I=1,NTERMS)
      IF(LOCATE .EQ. 999) GO TO 15
      N=N+1
      CALL ALT3(CHARAC,N)
      GO TO 13
   14 PRINT 1010
      STOP
      STOP
   15 GO TO (91,92,93,94,900,900), I
   91 M=3
```

```
      N=8
      GO TO 99
   92 M=2
      N=6
      GO TO 99
   93 M=3
      N=8
      GO TO 99
   94 M=3
      N=8
   99 READ 1014,(K(J), J=1,N),(X(J), J=1,M)
      PRINT 1015,(J,K(J), J=1,N)
      PRINT 1016,(SPETAB2(J),X(J), J=1,M)
      CALL MODELA(K,X,DIFFX)
      PRINT 1017,(SPETAB2(J),DIFFX(J), J=1,M)
  900 CONTINUE
      STOP
   16 PRINT 1011,NERRORS
      STOP
      END
```

```
          BINARY CONTROL CARDS.

          IDENT  MOD1
          END

BLOCKS    TYPE       ADDRESS     LENGTH

PROGRAM*  LOCAL            0         61
COM1      COMMON           0       1130
COM3      COMMON           0         24

ENTRY POINTS.

MOD1      -        0  ALT1      -        7

EXTERNAL SYMBOLS.

DUMPREG
```

```
          IDENT  MOD1
          ENTRY    MOD1,ALT1
          USE      /COM1/
SPETAB1   BSS      96
INTAB     BSS      401
AVAIL     BSS      100
NUMSF     BSS      1
NUMIN     BSS      1
TOF       BSS      1
          USE      *
          USE      /COM3/
SPETAB2   BSS      20
          USE      *
          MACRO    HASH,NAME,XREG1,XREG2
NAME      BX3      XREG1
          B*XREG2  X5*X3
          AX3      12
          I*XREG2  X3+XREG2
          B*XREG2  XREG2*X5
          ENDM
```

```
*
          MACRO     SEARCH,NAMA,REGX,REGB,P1,P2
NAMA      SA1       REGB+SPETAB1
          ZR        X1,P1
          BX3       X1-REGX
          AX3       36
          ZR        X3,P2
          SX3       REGB+5
          BX4       X3*X5
          S+REGB    X4
          JP        NAMA
          ENDM
*
*REGISTER B1 HOLDS THE ADDRESS OF CHARAC(I)
*REGISTER B2 COUNTS THE NUMBER OF SPECIES
*REGISTER B3 HOLDS THE NUMBER 20
*REGISTER B4 SERVES VARIOUS PURPOSES
*REGISTER B5 SERVES AS A FLAG
*REGISTER B6 IS AN INDEX I
*REGISTER B7 HOLDS THE NUMBER 80
*REGISTER X0 SERVES VARIOUS PURPOSES
*REGISTERS X0,X2,X3,X4,6 SERVE VARIOUS PURPOSES
*REGISTER X1 HOLDS CHARAC(I)
*REGISTER X5 HOLDS A 5-BIT, RIGHT JUSTIFIED MASK
*REGISTER X7 HOLDS THE NUMBER 55B
*
*THE A-REGISTERS SERVE ONLY TO PUT THE PROPER VALUES INTO THE X-REGISTERS
*
MOD1      BSSZ      1
*FIRST CLEAR THE ARRAY SPETAB1
          SB2       32
          SX6       0
ALPHA     SB2       B2-1
          SA6       B2+SPETAB1
          GT        B2,B0,ALPHA
          SA6       NUMSP
          SA1       MOD1
          BX6       X1
          SA6       ALT1
*         JP        *+2
ALT1      BSSZ      1
          SA5       NUMSP
          SB2       X5
          SB3       20
          SB5       0                 SET FLAG DOWN
          SB6       -1
          SB7       80
          SX5       37B               SET 5-BIT, RIGHT-JUSTIFIED MASK IN X5
          SX7       55B               55B DISPLAY CODE FOR BLANK
BETA      SB6       B6+1
          EQ        B6,B7,NU          TEST FOR END OF CARD
          SA1       B1+B6             LOAD CHARAC(I)
*DETERMINE WHAT TYPE OF SYMBOL CHARAC(I) IS
```

```
            SX0       X1-45B
            NG        X0,GAMMA
            SX0       X1-57B
            ZR        X0,PERIOD
            PL        X0,ERROR1          ILLEGAL CHARACTER CHECK
            SX0       X1-55B
            ZR        X0,BETA
            NG        X0,ERROR1          ILLEGAL CHARACTER CHECK
            JP        COMMA
*FIRST CHARACTER IN SPECIES NAME MUST BE A LETTER
GAMMA       NE        B0,B5,DELTA
            SX0       X1-33B
            PL        X0,ERROR1
            SB4       3
            SB5       1                  SET FLAG UP
            BX2       X1
            JP        BETA
*A SPECIES NAME CAN HAVE A MAXIMUM OF ONLY 3 CHARACTERS
*FORMATION OF A SPECIES NAME
DELTA       SB4       B4-1
            LX2       6
            BX2       X1+X2
            GT        B4,B0,BETA
            RJ        =XDUMPREG
ERROR1      SX6       0                  ERROR IN SYNTAX
            RJ        =XDUMPREG
            SA6       B1+1
            JP        ALT1
COMMA       EQ        B0,B5,BETA
            SB5       0                  SET FLAG DOWN
*SPECIES LIST CAN HAVE A MAXIMUM OF 20 NAMES
ZETA        EQ        B2,B3,ERROR2
LP1         SB4       B4-1
            EQ        B4,B0,THETA
            LX2       6
            BX2       X2+X7
            JP        LP1
*
THETA       HASH      X2,X4
            SB4       X4                 STORE HASH VALUE IN B4
            LX2       42
*HASH SEARCH
ICTA        SEARCH    X2,B4,KAPPA,BETA
*PLACE SPECIES NAME , WITH POINTERS, INTO SPECIES TABLES
*FIRST POINTER IS ITS INDEX NUMBER IN SPETAB2
KAPPA       BX6       X2
            SA6       B2+SPETAB2
            SX3       B2
            BX4       X3
            LX4       18
            BX6       X2+X3
            BX6       X6+X4
            SA6       B4+SPETAB1
```

```
            SB2     B2+1
*
*COMMA,PERIOD,END-OF-CARD CHECK
            EQ      B5,B0,BETA
            LT      B5,B0,MU
LAMDA       SX6     0
            SA6     B1
MU          SX6     B2
            SA6     NUMSP
            JP      ALT1
PERIOD      EQ      B0,B5,LAMDA
            SB5     1
            JP      ZETA
MU          EQ      B0,B5,MU
            SB5     -1
            JP      ZETA
ERRCP2      SX6     0                   SPECIES LIST OVERFLOW
            SA6     B1+2
            JP      ALT1
            END
```

```
                BINARY CONTROL CARDS.

                IDENT   MOD2
                END

BLOCKS          TYPE            ADDRESS         LENGTH

PROGRAM*        LOCAL                 0            165
LITERALS*       LOCAL               165              2
COM1            COMMON                0           1130
COM3            COMMON                0             24

ENTRY POINTS.

MOD2        -           1   ALT2        -          12
```

```
IDENT   MOD2
        ENTRY   MOD2,ALT2
*
        USE     /CCM1/
SPETAB1 BSS     96
INTAB   BSS     401
AVAIL   BSS     100
NUMSP   BSS     1
NUMIN   BSS     1
TOP     BSS     1
        USE     *
        USE     /CCM3/
SPETAB2 BSS     20
        USE     *
SINHOLD BSSZ    1
*
        MACRO   HASH,NAME,XREG1,XREG2
NAME    BX3     XREG1
        B*XREG2 X5*X3
        AX3     12
        I*XREG2 X3+XREG2
        B*XREG2 XREG2*X5
        ENDM
*
        MACRO   SEARCH,NAMA,REGX,REGB,P1,P2
NAMA    SA1     REGB+SPETAB1
        ZR      X1,P1
        BX3     X1-REGX
        AX3     36
        ZR      X3,P2
        SX3     REGB+5
        BX4     X3*X5
        S*REGB  X4
        JP      NAMA
        ENDM
*
MOD2    BSSZ    1
```

```
            SA1     MOD2
            BX6     X1
            SA6     ALT2
*SET AVAILABLE SPACE STACK INITIALLY TO ZERO
            SB6     100
            SX6     0
ONE         SB6     B6-1
            SA6     B6+AVAIL
            GT      B6,B0,ONE
            SA6     NUMIN
            SA1     NUMSP
            BX6     X1
            SA6     TOP
+           JP      *+2
ALT2        BSSZ    1
            SA1     NUMIN
            SB2     X1                  RETRIEVE NUMBER OF INTERACTIONS
            SB6     -1                  INITIALIZE I
            SB7     80
            SX0     55B                 55B DISPLAY CODE FOR BLANK
            SX5     37B                 SET 5-BIT, RIGHT-JUSTIFIED MASK IN X5
            SB5     0                   SET FLAG DOWN
*CHECK FOR END CARD
            SA1     B1
            SX4     X1-53B              53B DISPLAY CODE FOR $
            NZ      X4,TWO
            SX6     0
            SA6     B1
            SA6     B1+5
            JP      ALT2
TWO         SB6     B6+1                I=I+1
            EQ      B6,B7,ERRORA        TEST FOR END OF CARD
            SA1     B1+B6               LOAD CHARAC(I)
            SX4     X1-55B              55B DISPLAY CODE FOR BLANK
            ZR      X4,TWO
            SX4     X1-33B
            NG      X4,THREE
ERRORA      SX6     0                   SYNTAX ERROR DETECTED
            SA6     B1
            SA6     B1+1
            JP      ALT2
THREE       BX2     X1
            SB5     B5+2
            SB4     3
LP1         SB6     B6+1                I=I+1
            EQ      B6,B7,ERRORA
            SA1     B1+B6               LOAD CHARAC(I)
            SX4     X1-47B              47B DISPLAY CODE FOR *
            ZR      X4,FOUR
            SX4     X1-60B              60B DISPLAY CODE FOR =
            ZR      X4,FIVE
            SX4     X1-55B              55B DISPLAY CODE FOR BLANK
            ZR      X4,LP1
```

```
          SX4       X1-45B          IS CHARAC(I) A LETTER OR A NUMBER
          PL        X4,ERRORA
          SB4       B4-1
*SPECIES NAME CAN HAVE A MAXIMUM OF ONLY 3 CHARACTERS
          LX2       6
          BX2       X2+X1
          JP        LP1
FOUR      SX1       B5-2
          NZ        X1,ERRORA
          SB5       1
LP2       SB4       B4-1
          EQ        B4,B0,SIX
          LX2       6
          BX2       X2+X0
          JP        LP2
*LP2 FILLS OUT THE SPECIES NAME WITH BLANKS
FIVE      SX1       B5-1
          ZR        X1,ERRORA
          JP        LP2
SIX       HASH      X2,X4
          SB4       X4
          LX2       42
*HASH SEARCH
SEVEN     SEARCH    X2,B4,ERRORB,EIGHT
EIGHT     SX3       B5-2
          PL        X3,NINE
          BX7       X1
          JP        TWO
NINE      SB3       100
          EQ        B2,B3,ERRORC    INTERACTION LIST OVERFLOW
          ZR        X3,TEN
*CONSTRUCTION OF BINARY INTERACTION LABEL OF FORM (XXX)(XXX)
          MX2       18
          BX3       X7*X2           MASK INTO X3 1ST SPECIES NAME
          LX3       54
          BX4       X1*X2           MASK INTO X4 2ND SPECIES NAME
          LX4       24
          BX6       X4+X3
          SA3       =-267777772526777777258         (   )(   )
          BX6       X6+X3           PLACE PARENTHESES AROUND LABEL
          SB3       B2+101
          SA6       B3+INTAB        STORE LABEL INTO INTAB
*CONSTRUCTION  OF BINARY INTERACTION CODE WORD
          LX2       36
          BX3       X7*X2
          BX4       X1*X2
          AX4       18
          BX6       X3+X4           CODE WORD FORMED IN X6
          SB3       B2+1
          SA6       B3+INTAB        STORE CODE WORD INTO INTAB
          JP        ELEVEN
*CONSTRUCTION OF UNARY INTERACTION LABEL OF FORM (XXX)
TEN       MX2       18
```

```
                BX3     X1*X2
                LX3     54
                SA4     =-2677777725222222228          (   )
                BX6     X3+X4                  PLACE PARENTHESES AROUND LABEL
                SB3     B2+101
                SA6     B3+INTAB               STORE LABEL INTO INTAB
*CONSTRUCTION OF UNARY INTERACTION CODE WORD
                LX2     36
                BX3     X1*X2
                SX4     777758
                BX6     X3+X4                  CODE WORD FORMED INTO INTAB
                SB3     B2+1
                SA6     B3+INTAB
ELEVEN          SB6     B6+1                   I=I+1
                EQ      B6,B7,OUT              TEST FOR END OF CARD
TWELVE          SA1     B1+B6                  LOAD CHARAC(I)
                SX7     X1-55B                 55B DISPLAY CODE FOR BLANK
                ZR      X7,ELEVEN
                SX7     X1-57B                 DISPLAY CODE FOR .
                ZR      X7,OUT
                SX7     X1-46B                 46B DISPLAY CODE FOR -
                ZR      X7,THTEEN
                SX7     X1-45B                 45B DISPLAY CODE FOR +
                NZ      X7,ERRORA
                SX6     0
                SA6     SINHOLD
                SB5     0
                JP      FORTEEN
THTEEN          SX6     10C0B
                SA6     SINHOLD
                SB5     0
FORTEEN         SB6     B6+1                   I=I+1
                EQ      B6,B7,SIXTEEN          TEST FOR END OF CARD
                SA1     B1+B6                  LOAD CHARAC(I)
                SX7     X1-55B                 55B DISPLAY CODE FOR BLANK
                ZR      X7,FORTEEN
                SX7     X1-57B                 57B DISPLAY CODE FOR .
                ZR      X7,SIXTEEN
                SX7     X1-46B                 46B DISPLAY CODE FOR -
                ZR      X7,SIXTEEN
                SX7     X1-45B                 45B DISPLAY CODE FOR +
                ZR      X7,SIXTEEN
                PL      X7,ERRORA
                GT      B5,B0,FIFTEEN
                SX7     X1-33B                 IS THE CHARACTER A LETTER
                PL      X7,ERRORA
                SB5     1
                SB4     3
                BX2     X1
                JP      FORTEEN
*FORMATION OF SPECIES NAME
FIFTEEN         SB4     B4-1
                EQ      B4,B0,ERRORA
```

```
            LX2     6
            BX2     X2+X1
            JP      FORTEEN
SIXTEEN     ZR      B5,ERRORA
            SB4     B4-1
            EQ      B4,B0,SEVTEEN
            LX2     6
            BX2     X2+X0           FILL SPECIES WITH BLANKS
            JP      SIXTEEN
SEVTEEN     MASH    X2,X4
            SB4     X4
            LX2     42
EITTEEN     SEARCH  X2,B4,ERRORB,NINTEEN
NINTEEN     SX2     177B            FORM 7-BIT , RIGHT-JUSTIFIED MASK IN X2
            BX3     X1*X2           MASK LOCATION POINTER INTO X3
            SA4     X3+AVAIL
            LX2     11
            BX7     X1*X2           MASK POSITION POINTER INTO X7
            AX7     15
            SX7     X7+1
            SB5     X7
            SA2     SINHOLD
            BX4     X4+X2
            SX2     B3
            BX4     X4+X2           PACK SIGN AND POINTER INTO BEAD FIELD
            LX4     10
            SB5     B5-5
            EQ      B5,B0,TWENTY
            LX7     15
            BX6     X7+X3           PACK POINTERS INTO ONE WORD
            MX2     42
            BX2     X2*X1
            BX6     X6+X2           REPACK SPETABI WORD
            SA6     B4+SPETABI
            BX6     X4
            SA6     X3+AVAIL
            LT      B6,B7,TWELVE
            JP      OUT
*PLACE MODIFIED LIST BEAD BACK INTO AVAIL
TWENTY      SA2     TOP
            BX7     X2
            SB5     X7-99
            GT      B5,B0,ERRORD    ERROR IF AVAIL OVERFLOWS
            BX6     X7+X4
            SA6     X3+AVAIL
            MX2     42
            BX6     X1*X2
            BX6     X6+X7
            SA6     B4+SPETABI
            SX7     X7+1            INCREMENT TOP
            SA7     TOP
            JP      TWELVE
            LT      B6,B7,TWELVE
```

```
OUT         SX6     B2+1
            SA6     NUMIN
            JP      ALT2
ERRORB      SX6     0               UNDEFINED SPECIES
            SA6     B1
            SA6     B1+2
            JP      ALT2
ERRORC      SX6     0               INTERACTION LIST OVERFLOW
            SA6     B1
            SA6     B1+3
            JP      ALT2
ERRORD      SX6     0               PROGRAM OVERFLOW
            SA6     B1
            SA6     B1+4
            JP      ALT2
            END

STORAGE NEEDED          265 STATEMENTS          0 SYMBOLS
6400 ECS ASSEMBLY     1.038 SECONDS             0 REFERENCES        5 INTERNAL TABLE MOVES

        1 ERROR IN  *******
```

```
          BINARY CONTROL CARDS.

          IDENT     MOD3
          END

BLOCKS    TYPE        ADDRESS     LENGTH

PROGRAM*  LOCAL             0        117
CODE      COMMON            0       4304
COM1      COMMON            0       1130
COM3      COMMON            0         24
COM2      COMMON            0          2

ENTRY POINTS.

MOD3      -       13  ALT3      -       51
```

```
          IDENT     MOD3
          ENTRY     MOD3,ALT3
          USE       /CODE/
STORE.    BSSZ      301
MODEL     BSSZ      1943
          USE       *
LOADX     MACRO     INSTR,XREG,N
          SA4       INSTR
          BX6       XREG+X4
          DUP       N,1
          LX6       30
          SA6       B3+MODEL
          SB3       B3+1                INCREMENT LOCATION COUNTER
          ENDM
*
          USE       /COM1/
SPETAB1   BSS       96
INTAB     BSS       401
AVAIL     BSS       100
NUMSP     BSS       1
NUMIN     BSS       1
TOP       BSS       1
          USE       *
```

```
                USE     /CCM3/
SPETAB2         BSS     20
                USE     *
                USE     /CCM2/
LOCATE          BSSZ    1
NTERMS          BSSZ    1
                USE     *
TYPE1           VFD     12/6140B,18/STORE,30/46C0C460000B

TYPE2A          VFD     30/2415146000B,30/5111000C00B
TYPE2B          VFD     30/2425246CC0B,30/5122CC0C00B
TYPE2C          VFD     30/2435406120B,30/5132C000000B
TYPE2D          VFD     30/4600040663B,30/5164000000B
*
TYPE3B          VFD     30/2425240612B,30/5122CC0CC0B
TYPE3C          VFD     30/460CC460C0B,30/5164CC0CC0B
*
TYPE4           VFD     30/7160002000B,30/2066C46C00B
TYPE5           VFD     30/3066124656B,30/5114CC0CC0B
TYPE6           VFD     30/4600C46000B,30/5163C00CC0B
TYPE7           VFD     12/C2C0B,18/MODEL,30/0B

*
MCD3            BSSZ    1
                SA1     TYPE1
                BX6     X1
                SB3     1
                SA6     B3+MCDEL
                SB3     2                       B3 IS A LOCATION COUNTER
                MX0     18
                LX0     18
                SA1     NUMIN
                SB7     X1                      B7 HOLDS NUMBER OF INTERACTIONS
                SB6     1
LP1             SA1     B6+INTAB
                BX2     X1*XC
                AX1     18
                SX3     B6-1
                SX7     77775B
                IX7     X2-X7
                ZR      X7,ABB
*LOAD CODE FOR COMPUTING  RATE CONSTANT*BINARY INTERACTION FACTOR
                LOADX   TYPE2A,X3,1
                LOADX   TYPE2B,X1,1
                LOADX   TYPE2C,X2,1
                LOADX   TYPE2D,X3,0
                SB6     B6+1
                LE      B6,B7,LP1
                JP      ABES
*LOAD CODE FOR COMPUTING RATE CONSTANT*UNARY INTERACTION FACTOR
ABB             LOADX   TYPE2A,X3,1
                LOADX   TYPE3B,X1,1
                LOADX   TYPE3C,X3,0
```

```
          SB6     B6+1
          LE      B6,B7,LP1
ABES      SX6     B3
          SA6     LOCATE
          SA1     MCC3
          BX6     X1
          SA6     ALT3
+         JP      *+2
*
ALT3      BSSZ    1
          SA1     LOCATE
          SB3     X1
          SA2     B2
          SX0     X2-1
          SX5     7778          SET 9-BIT, RIGHT-JUSTIFIED MASK IN X5
          SB7     76
          SB6     1
          SA1     TYPE4
          BX6     X1
          SA6     B3+MODEL
          SB3     B3+1
          SA1     X0+SPETAB2    IDENTIFY DIFFERENTIAL EQUATION IN OUTPUT
          BX6     X1
          SA6     B1
ABLE      SA1     X0+AVAIL
          SB5     5
LP2       LX1     50
          BX2     X1*X5
          ZR      X2,FOX
          EQ      B6,B7,ERRORF
          SX4     10CCB         MASK USED TO DETECT PREDENCE OF -SIGN
          BX0     X1*X4
          NZ      X0,BAKER
          SX6     45B           45B DISPLAY CODE FOR +
          SA6     B1+B6
          JP      CHARLIE
BAKER     SX6     46B           46B  DISPLAY CODE FOR -
          SA6     B1+B6
CHARLIE   SB6     B6+1
          BX6     X2
          SA6     B1+B6         IDENTIFY RATE CONSTANT IN OUTPUT
          SB6     B6+1
          SB4     X2+100
          SA3     B4+INTAB
          BX6     X3
          SA6     B1+B6         SET UP INTERACTION LABEL FOR OUTPUT
          SB6     B6+1
          SX2     X2-1
          NZ      X0,DOG
          LOADX   TYPES,X2,1
          JP      EARNEST
DOG       SA4     TYPES
          BX4     X2+X4
```

```
            LX4       15
            BX6       X0+X4
*LOAD CODE FOR COMPUTING VALUE OF ITH DIFFERENTIAL EQUATION
            LX6       15
            SA6       B3+MODEL
            SB3       B3+1                INCREMENT LOCATION COUNTER
EARNEST     SB5       B5-1
            GT        B5,B0,LP2
            LX1       50
            BX0       X1*X5
            JP        ABLE
FOX         SX6       B6
            SA6       NTERMS
            SA2       B2
            SX2       X2-1
            LOADX     TYPE6,X2,1
            SX6       B3
            SA6       LOCATE
            SA1       NUMSP
            SX2       X2+1
            IX3       X1-X2
            NZ        X3,ALT3
            SA1       TYPE7
            BX6       X1
            SA6       B3+MODEL            PLACE END JUMP INSTRUCTION IN MODEL ARRAY
            SX6       999
            SA6       LOCATE
            JP        ALT3
ERRORF      SX6       1001                DIFFERENTIAL EQUATION TOO LONG
            SA6       LOCATE
            JP        ALT3
            END
```

```
BINARY CONTROL CARDS.

IDENT    MODELA
END

BLOCKS      TYPE        ADDRESS     LENGTH

PROGRAM*    LOCAL             0          3
CODE        COMMON            0       2024

ENTRY POINTS.

MODELA   -        0
```

```
            IDENT    MODELA
            ENTRY    MODELA
            USE      /CODE/
STORE       BSSZ     101
MODEL       BSSZ     943
            USE      *
MODELA      BSSZ     1
            RJ       MODEL
            JP       MODELA
            END
```

REFERENCES AND BIBLIOGRAPHY

1. Adachi, N., "On variable metric algorithms". Journal of Optimization Theory and Applications, vol. 7, 391 (1971).

2. Balakrishnan, A. V., Techniques of Optimization. Academic Press, New York and London (1972).

3. Bassham, J. A., "Control of photosynthetic carbon metabolism". Science, New York, 172, 526-534.

4. _____, Gerri Levine and John Forder III, "Photosynthesis in vitro. I. Achievements of High Rates". Plant Sci. Ltrs. 2, 15-21 (1974).

5. _____ and G. H. Krause, "Free energy changes and metabolic regulation in steady state photosynthetic carbon reduction". Biochem. Biophys. Acta 189, 207-221 (1969).

6. _____ and M. Kirk, "Dynamics of the photosynthesis of carbon compounds. I. Carboxylation reactions". Biochem. Biophys. Acta 43, 447-464 (1960).

7. Bellman, R. and R. Kalaba, Quasilinearization and nonlinear boundary-value problems. American Elsevier, New York (1965).

8. _____, J. Jacquez, R. Kalaba and S. Schwimmer, "Quasilinearization and the estimation of chemical-rate constants from raw kinetic data". Math. Biosci. 1, 71-76 (1967).

9. Bremermann, H. J., "Computation of equilibria and kinetics of chemical systems with many species". Quantitative Biology of Metabolism, 3rd International Symposium, Biologische Anstalt Helgoland, Sept. 26-29, 1967, Springer-Verlag (1967).

10. _____, "Identification of rate constants in chemical cycles of known structure". IEEE Symposium on Adaptive Processes, Decision and Control, pp. xxiii. 2.1 - xxlll. 23 (1970).

11. _____, "A method of unconstrained global optimization". Math. Biosci. 9, 1-15 (1970).

12. _____ and L. S. Lam, "Analysis of spectra with nonlinear superposition". Math. Biosci. 8, 449-460 (1970).

13. Blum, E. K., Numerical Analysis and Computation Theory and Practice. Addison-Wesley (1972).

14. Box, M. J., D. Davies and W. H. Swann, "Nonlinear Optimization Techniques". ICI Ltd., Monograph No. 5, Oliver and Boyd, Edinburgh (1969).

15. Brent, R. P., "On Maximizing the Efficiency of Algorithms for Solving Systems of Nonlinear Equations". Report RC 3725, IBM, Yorktown Heights (1971).

16. _____, "An algorithm with guaranteed convergence for finding a zero of a function". Comp. J. 14, 422-425 (1971).

17. _____, Algorithms for Minimization without Derivatives. Prentice Hall (1973).

18. Brown, K. M. and J. E. Dennis, "On Newton-like iteration functions: general convergence theorems and a specific algorithm". Numer. Math. 12, 186-191 (1968).

19. _____ and W. B. Gearhart, "Deflation techniques for the calculation of further solutions of a nonlinear system". Numerische Math. 16, 334-342, Springer-Verlag (1971).

20. _____, "A quadratically convergent Newton-like method based upon Gaussian eliminations". SIAM J. Numer. Anal. 6, 560-569 (1959).

21. Brown, W. S., Altran User's Manual. Bell Laboratories (1973).

22. Broyden, C. G., "Quasi-Newton methods and their applications to function minimization". Math. of Computation, Vol. 21, 368 (1967).

23. _____, "The convergence of a class of double-rank minimization algorithms: 1. General considerations, and 2. The new algorithm". J. Inst. Math. Appls., vol. 6, 76 and 222 (1970).

24. Colville, A. R., "A comparative study on nonlinear programming codes". Report No. 320-2949, IBM, Yorktown Heights (1968).

25. Conn, Eric E. and P. K. Tumpf, Outlines of Biochemistry. John Wiley and Sons, Inc. (1972).

26. Coppell, W. A., "Stability and asymptotic behavior of differential equations". Heath Mathematical Monographs (1965).

27. Cragg, E. E. and A. V. Levy, "Study on a supermemory gradient method for the minimization of functions". J. of Optimization Theory and Applications, vol. 4, 191 (1969).

28. Davidson, W. C., "Variable metric method for minimization". Report ANL-5990, AEC Research and Development (1959).

29. Eason, E. D. and R. G. Fenton, "A comparison of numerical optimization methods for engineering design". Paper No. 73-DET-17, ASME Publication (1973).

30. Eigen, M. and L. deMaeyer, "Theoretical basis of relaxation spectrometry". In Techniques of Chemistry, vol. 6, part 2, ed. by A. Weissberg and Gordon Hammes, John Wiley and Sons, Inc., New York (1973).

31. Engvall, J. L., "Numerical algorithm for solving overdetermined systems of nonlinear equations". NASA Document N70-35600 (1966).

32. Fiacco, A. V. and G. P. McCormick, "Computational aspects of unconstrained minimization algorithms". Chapter 8 in Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, Inc., New York (1968).

33. Fletcher, R., "Function minimization without evaluating derivatives -- a review". Computer Journal, vol. 8, 33 (1965).

34. _____, "A new approach to metric variable algorithms". Computer Journal 13, 317-322.

35. _____ and M. J. D. Powell, "A rapidly convergent descent method for minimization". Computer Journal, vol. 6, 163 (1963).

36. _____ and C. M. Reeves, "Function minimization by conjugate gradients". Computer Journal, vol. 7, 149 (1964).

37. Forsythe, George and Cleve B. Moler, Computer Solution of Linear Algebraic Systems. Prentice Hall, Inc. (1967).

38. _____ and T. S. Motzkin, "Asymptotic properties of the optimum gradient method (abstract)". American Mathematical Society Bulletin, vol. 57, 183 (1951).

39. Garfinkel, D., "Simulation of glycolitic systems". Concepts and Models of Biomathematics, F. Heinmetz, ed., Marcel Dekker, New York, 1-74 (1969).

40. Gear, C. William, Numerical Initial Value Problems In Ordinary Differential Equations. Prentice Hall, Inc. (1971).

41. Goldfarb, D., "Sufficient conditions for the convergence of a variable metric algorithm". In Optimization, R. Fletcher, ed., Academic Press, Inc., New York, 273 (1969).

42. Goldstein, A. A. and J. F. Price, "An effective algorithm for minimization". Numerische Mathematik, vol. 10, 184 (1967).

43. Greenstadt, J., "On the relative efficiencies of gradient methods". Math. of Computation, vol. 21, 360 (1967).

44. Hartman, Philip, Ordinary Differential Equations. John Wiley and Sons, Inc. (1973).

45. Heinmetz, F., "Analog computer analysis of a model system for the induced enzyme synthesis". J. Theoret. Biol. 6, 60-75 (1964).

46. Henneman, R. and L. S. Reid, "Memorandum on the conditions determining complex task proficiency". Psychological Laboratory, University of Virginia (1952).

47. Henrici, Peter, Discrete Variable Methods in Ordinary Differential Equations. John Wiley and Sons, Inc. (1962).

48. Himmelblau, D. M., Applied Nonlinear Programming. McGraw-Hill, New York (1972).

49. _____, "Determination of rate constants for complex kinetic models". Ind. Eng. Chem. Fund. 6, 539 (1967).

50. Hirsch, Morris W. and Stephen Smale, Differential Equations, Dynamical Systems and Linear Algebra. Academic Press, New York (1974).

51. Hooker, R. and T. A. Jeeves, "Direct search solution of numerical and statistical problems". Journal of the ACM, vol. 8, 212 (1961).

52. Huang, H. Y. and A. V. Levy, "Numerical experiments on quadratically convergent algorithms for function minimization". Journal of Optimization Theory and Applications, vol. 6, 269 (1970).

53. Jacoby, S. C. S., J. S. Kowalik and J. T. Pizzo, Iterative Methods for Nonlinear Optimization Problems. Prentice Hall (1972).

54. Jensen, R. G. and J. A. Bassham, Biochem. Biophys. Acta 153, 227 (1968).

55. Kowalk, J. and M. R. Osborne, Methods for Unconstrained Optimization Problems. American Elsevier, New York, (1968).

56. _____, "Descent Methods". Chapter 3, 29, in Methods for Unconstrained Optimization Problems, American Elsevier, New York (1968).

57. Lanczos, C., Applied Analysis. Prentice Hall (1956).

58. Leon, A., "A comparison among eight known optimizing procedures". In Recent Advances in Optimization Techniques, ed. by A. Lavi and T. P. Vogl, John Wiley and Sons, Inc., New York (1966).

59. Lootsma, F. A., Numerical Methods for Nonlinear Optimization. Academic Press (1973).

60. Marquardt, D. W., "An algorithm for least squares estimation of nonlinear parameters". SIAM Journal, vol. 11, 431 (1963).

61. Miele, A. and J. W. Cantrell, "Study on a memory gradient method for the minimization of functions". Journal of Optimization Theory and Applications, vol. 3, 459 (1969).

62. Murray, W., Numerical Methods for Unconstrained Optimization. Academic Press (1972).

63. Murtach, B. A. and R. W. H. Sargent, "Computational experience with quadratically convergent minimization methods". Computer Journal, vol. 13, 185 (1970).

64. Myers, G. E., "Properties of the conjugate gradient and Davidon methods". Journal of Optimization Theory and Applications, vol. 2, 209 (1968).

65. Nelder, J. A. and R. Mead, "A simplex method for function minimization". Computer Journal, vol. 7, 308 (1965).

66. Noel, Paul, Introduction To Mathematical Statistics. John Wiley and Sons, Inc. (1963).

67. Paviani, D. A. and D. M. Himmelblau, "Constrained non-linear optimization by heuristic programming". Journal of ORSA, vol. 17, 872 (1969).

68. Pearson, J. D., "Variable metric methods of minimization". Computer Journal, vol. 12, 171 (1969).

69. Pedersen, T. A., Martha Kirk and J. A. Bassham, "Light-dark transients in levels of intermediate compounds during photosynthesis in air-adapted Chlorella". Physiol. Plantarum 19, 219-231 (1966).

70. Polack, E., "A modified second method for unconstrained minimization". (To be published.)

71. Powell, M. J. D., "An efficient method for finding the minimum of a function of several variables without calculating derivatives". Computer Journal, vol. 7, 155 (1964).

72. Quisthoudt, Godelieve, De Optmering van Thermodynamische en Kinetische Parameters bij Chemische Relaxtiemetingen. Thesis, Catholic University, Leuven, Belgium (1973).

73. Ralston, A., A First Course In Numerical Analysis. McGraw-Hill (1965).

74. Rosennbrock, H. H., "An automatic method for finding the greatest or least value of a function". Computer Journal, vol. 3, 175 (1960).

75. _____ and C. Storey, Computational Techniques for Chemical Engineers. Pergamon Press, Oxford (1966).

76. Roth, R. S. and Micheline M. Roth, "Data unscrambling and the analysis of inducible enzyme synthesis". Math. Biosci. 5, 57-92 (1969).

77. Sanchez, David A., Ordinary Differential Equations and Stability Theory. W. H. Freeman and Co. (1968).

78. Shah, B. V., R. J. Buehler and O. Kempthorne, "Some algorithms for minimizing a function of several variables". SIAM Journal, vol. 12, 74 (1964).

79. Smale, Stephen, "A global Newton Rapshom algorithm". (Personal communication.)

80. Sorenson, H. W., "Comparison of some conjugate direction procedures for function minimization". Journal of the Franklin Institute, vol. 288, 421 (1969).

81. Spanier, E., Algebraic Topology. McGraw-Hill, New York (1966).

82. Stewart, G. W. III, "A modification of Davidson's minimization method to accept difference approximations of derivatives". Journal of the ACM, vol. 14, 72 (1967).

83. Strehlow, H. and J. Jen, "On the accuracy of chemical relaxation measurements". Chem. Instr. 3, 47-69 (1971).

84. Swartz, J., Parameter Estimation In Biological Systems. Thesis, University of California, Berkeley (1973).

85. _____ and H. J. Bremermann, "Discussion of parameter estimation in biological modelling: algorithms for estimation and evaluation of the estimates". J. Math. Biol. (1975).

86. Van Zeggeren, F. and S. H. Storey, The Computation of Chemical Equilibria. Cambridge University Press, Cambridge (1970).

87. Wallace, Simaiko H., Selected Papers on Human Factors in the Design and Use of Control Systems. Dover Publications (1961).

88. Wilkinson, J. H., Rounding Errors in Algebraic Processes. Prentice Hall, Inc. (1963).

89. Zangwill, W. I., "Minimizing a function without calculating derivatives". Computer Journal, vol. 10, 293 (1967).

90. _____, "Unconstrained problems". Chapter 5 in Nonlinear Programming, A Unified Approach, Prentice-Hall, Inc. (1969).

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720