Scholars' Mine

Doctoral Dissertations

Student Theses and Dissertations

1973

Tchebycheff approximation on a discrete point set: algorithms old and new

William Edward McBride

Follow this and additional works at: https://scholarsmine.mst.edu/doctoral_dissertations

Part of the Mathematics Commons

Department: Mathematics and Statistics

Recommended Citation

McBride, William Edward, "Tchebycheff approximation on a discrete point set: algorithms old and new" (1973). *Doctoral Dissertations*. 252.
https://scholarsmine.mst.edu/doctoral_dissertations/252

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

TCHEBYCHEFF APPROXIMATION ON A DISCRETE POINT SET: ALGORITHMS OLD AND NEW

by

WILLIAM EDWARD MCBRIDE, 1940-

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI AT ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

1973

T2989
99 pages
c.1

A K Rigler
Advisor

M. E. Engelhardt

Howard D. Pyron

A. J. Penico

[illegible]

D. Ray Edwards

ABSTRACT

Several linear and nonlinear algorithms for solving the discrete Tchebycheff problem are compared in this study. The Lawson algorithm is compared with two more well-known methods of linear Tchebycheff approximation. A new acceleration scheme for the Lawson algorithm is introduced and its performance is tested with an already existing acceleration technique. The new version is found to be better than the previous one but not as effective as the traditional Exchange method.

A nonlinear version of Lawson's algorithm is proposed for the solution of problems having approximating functions which are varisolvent. Some linear theorems of Lawson are extended to the nonlinear case. A modification of Osborne and Watson's nonlinear method is introduced and tested on five problems. This new technique improves the efficiency remarkably, particularly for larger problems.

ACKNOWLEDGEMENTS

The author wishes to express his deepest appreciation and thanks to Professor A. K. Rigler who was willing to "adopt" the author and guide him through the writing of this dissertation.

Thanks are also due to Professor Ralph Lee and his computer center staff who were always ready to help the author in his use of the computer facility at Rolla. A similar note of gratitude is expressed to the computer center staff at Western Illinois University.

Financial support from the National Science Foundation was received by the author during the 1971-72 school year through an N.S.F. Graduate Traineeship. The Department of Mathematics also helped out by granting assistantships during the 1969-70 and 1970-71 academic years.

The author wishes to thank Mrs. Linda Simpson, whose accuracy, speed and good judgment were so crucial in the preparation of this final copy. A special note of thanks to my wife, Janie, and the five little ones for their understanding and patience through these years of study.

TABLE OF CONTENTS

Table of Contents (continued) Page

Table of Contents (continued) Page

LIST OF TABLES

I. INTRODUCTION

A. The Minimax Problem

The main purpose of this study is to investigate various algorithms for the solution of the minimax problem on a discrete set of points. The investigation will be centered around the following topics: speed of convergence, the accuracy of results, the number of computations required, revision and improvement of some of the existing algorithms, and development of new algorithms. The minimax problem, or more formally, the Tchebycheff approximation problem on a finite point set X in [0,1] is stated as: given f(x) defined on X, determine $L(A^*,x), A^* \varepsilon P$, such that

$$\max_{x \varepsilon X} |L(A^*,x)-f(x)| \le \max_{x \varepsilon X} |L(A,x)-f(x)|$$

for all $A \varepsilon P$, where P is the parameter space. In the case of linear approximation, $L(A^*,x) = \sum_{i=0}^{n} a_i^* \emptyset_i(x)$.

B. A Brief History

Tchebycheff approximation makes use of the so-called uniform norm which was first proposed by Laplace in 1799. The first systematic study of uniform approximation is attributed to P. L. Tchebycheff and the resulting theory bears his name. His work was carried out in the second half of the nineteenth century and picked up by others in the early 1900's. Most of the basic results were established by 1915. These early investigations were primarily theoretical in nature and it wasn't until Remes' algorithms appeared in the 1930's that any workable tools were available. Tchebycheff approximation (hereafter called T-approximation) lagged behind least-squares approximation because it did not have such a simple characteristic property, computationally speaking. However, the characteristic property in T-approximation is still very important because it is the one thing which allows us to identify a solution. We'll see later in this study how the characterization theorems for T-approximation are put to good use in developing several algorithms.

It is only since the advent of high-speed computers after World War II that the uniform norm came into popular usage. Stiefel is perhaps the most important mathematician to be mentioned in connection with modern linear T-approximation. His exchange method turns out to be the most powerful algorithm for discrete linear T-approximation. Stiefel was one of the first to recognize the equivalence of linear programming and the exchange method. Although many authors have re-posed the problem using the linear programming technique, the exchange method has proved to be the more powerful one because it is computationally more efficient. In 1961 Lawson showed that T-approximation could actually be done in terms of weighted least-squares approximation. Since least-squares approximation has a desirable characteristic property in the linear case this result was very significant theoretically.

C. Review of the Literature

1. Introduction

This review is prefaced by a restatement and an alternate formulation of the linear problem. Let f(x) be a function given in a finite interval on the x-axis. We wish to approximate f(x) (which will be called the target function) by an expression

$$L(A,x)=a_0\emptyset_0(x)+a_1\emptyset_1(x)+\ldots+a_n\emptyset_n(x)$$

in such a way that the maximum of the absolute value of the error function $e(x)=L(A,x)-f(x)$ is as small as possible. The $a_0,a_1,\ldots,a_n$ are the unknowns of the problem and the $\emptyset_0,\emptyset_1,\ldots,\emptyset_n$ will be known as the base functions. We assume that only tabulated values $f_i=f(x_i)$ of the target function are known at distinct abscissas,

$$x_1<x_2<\ldots<x_m.$$

Historically this discrete minimax problem was posed in the following way. Find a solution to the inconsistent system of linear equations

$$\eta_j=\sum_{k=0}^{n} a_{jk}x_k+c_j=0,\ j=1,2,\ldots,m,m>n,$$

in such a way that the solution $\{x_k\}$ minimizes

$$\xi=\text{Max}|\eta_j|,j=1,2,\ldots,m.$$

2. Early Beginnings

The earliest discussion of the minimax problem for systems of linear equations is apparently due to de la Vallée Poussin [1]. Polya's algorithm [2], which involved the approximation of continuous functions by polynomials, could easily be adapted to the discrete problem. Although this algorithm had a recent rebirth due to Goldstein, Levine and Herreshoff [3], it turns out to be inefficient compared with newer methods. There was a definite lag in development until the work of Remes [4] and the appearance of his two algorithms.

Remes' name is so important because it was he who constructed the first really useful algorithm for T-approximation. The details of his algorithm may be found in Meinardus [5]. Suffice it to say that we'll be primarily interested in what is referred to in the literature as the first algorithm of Remes or the simplified method of Remes. More recently it has been referred to as the single exchange method or the "one-for-one" exchange method. This method is the basis for Stiefel's exchange algorithm and will be fully described in a later section in Chapter II. The general method of Remes, or more properly his second algorithm, involves simultaneous exchanges and has no direct bearing on this work.

3. Some Post-Computer Developments

It was not until after the development of the high-speed digital computer that Tchebycheff approximation came into its own and that methods were implemented which were truly useful for computation. The following quote, taken from the preface to Meinardus' book [5], the first German edition, attests to this fact. "It has only been in the past few years that those parts of approximation theory which can be applied to numerical problems have been strongly developed."

Two methods will now be mentioned briefly, not necessarily because of their usefulness, but rather because of their historic interest. Zuhovickii [6] was interested in solving the Tchebycheff problem as it applies to an inconsistent system of linear equations. He attacked the problem basically from a geometric point of view. Let the residuals be denoted by

$$R^i(x)=\langle A^i,x\rangle-b_i,\ 1\leq i\leq m,$$

and $F(x)=\max\limits_{1\leq i\leq m}|R^i(x)|$ be the deviation of the system, where $\langle\ \rangle$ denotes the inner product. The Tchebycheff problem is that of obtaining a point $\overline{x}$ in E_n which minimizes F. Here $\overline{x}$ is termed the minimax solution. Let $\xi=F(x)$. This equation may be thought of as defining a polyhedral surface in E_{n+1}, and the vector $\overline{x}$ is the "abscissa" of its lowest point. Zuhovickii's algorithm obtains $\overline{x}$ by proceeding from vertex to vertex on this surface. For details of this method see Cheney and Goldstein [7]. For the discrete T-approximation problem this algorithm does not appear to be competitive.

However, it is a forerunner of the linear programming method of solution and hence is essentially the exchange algorithm as we'll see shortly.

Cheney and Goldstein [8] published a rather complicated algorithm for solving the T-problem but again it does not appear to be competitive. The significant thing about their paper is that they appear to be among the first to recognize that the Tchebycheff problem is equivalent to a linear programming problem. Stiefel [9] also has shown that many of the algorithms for solving this Tchebycheff problem are closely related to the method of linear programming. According to him, "The exchange method is completely equivalent to the well-known simplex algorithm of G. B. Dantzig." However, the exchange method appears to be more economical than the simplex method.

Barrodale and Young [10] have popularized the use of linear programming in handling the Tchebycheff problem by utilizing a modified simplex algorithm. Their procedure will be described in depth in Chapter II. In his Ph.D. dissertation, C. L. Lawson [11] developed a method for solving the discrete Tchebycheff problem which had not appeared previously in the literature. Although at the time it was developed it had not been compared with the exchange or linear programming methods, it did provide a workable tool for T-approximation of vector-valued functions and functions of a complex variable, where none existed before.

4. The Nonlinear Problem

The type of T-approximation that has been treated up to now has been primarily of the linear variety. Nonlinear T-approximation is of relatively recent vintage. It has become popular only after the success that has been attained in the linear area through the application of high-speed computers. This success stimulated a rebirth of interest in both the theoretical and practical aspects of nonlinear T-approximation. Although several algorithms have been proposed for solving the nonlinear problem, each has its shortcomings. One major objective of this study was to try to push forward the state of the art and improve the applicability of a well-known algorithm.

Most of the literature on nonlinear T-approximation treats rational approximation. Hastings [12] and several of his associates at The Rand Corporation were early practitioners of the art of rational approximation. Loeb [13] and Maehly [14] are also given much of the credit for early investigations in this area. The methods for handling the nonlinear problem generally fall into two categories: (1) those that use a characteristic property of rational T-approximation and (2) those that use a linear programming approach on a sequence of linear problems. The algorithms of Remes and Maehly are typical of methods which utilize a characteristic theory. Osborne and Watson's method [15] and also the Differential Correction Algorithms of Cheney and Loeb as discussed by Lee and Roberts [16] are techniques which employ a linear programming formulation. The following conclusions were garnered from a paper by Lee and Roberts [16]. Remes' algorithm is usually the most rapid method to converge. The Differential Correction Algorithm III is rated slightly superior to Osborne and Watson's method. However, the Osborne-Watson technique has the advantage that its applicability need not be restricted to the rational problem.

D. Objectives of This Study

The primary objective of this study was to develop a new algorithm for nonlinear T-approximation. This new algorithm would be a nonlinear version of Lawson's algorithm. Another objective was to improve on an existing algorithm of Osborne and Watson. A detailed study was undertaken to compare the efficiency and effectiveness of these algorithms.

A secondary objective was to investigate the current state of the art in linear T-approximation. An acceleration scheme was devised which attempted to speed up the Lawson algorithm and hopefully do better than the acceleration method published by Rice and Usow [17]. A detailed study was made in an effort to determine the best linear method with respect to speed, accuracy and efficiency.

In Chapter II there appear the necessary definitions, theorems and background information needed in the later chapters. Details of various algorithms are also contained in Chapter II. Chapter III is comprised of the basic algorithms and theoretical results obtained in this study. The details of numerical experimentation are the subject matter of Chapter IV.

It should be noted here that when solving a nonlinear problem we will be assuming existence when convenient and that answers we obtain may not be unique.

II. ALGORITHMS FOR DISCRETE TCHEBYCHEFF APPROXIMATION

In this chapter, background material will be introduced and some basic methods of discrete Tchebycheff approximation will be described. We will start with a few basic definitions, theorems and notation.

A. Preliminaries

The following results are stated here for convenience as reference material for this chapter and later chapters. Most of the material involving the exchange method is taken from Stiefel [18]. The rest of the basic theory is from Rice's two volumes, [19] and [20].

Definition 2.1. The set $\{\emptyset_i(x)\}$ is said to form a Tchebycheff set in [0,1] if the difference

$$L(A_1,x)-L(A_2,x)$$

has at most n-1 zeros in [0,1] for $A_1 \neq A_2$.

Notation 2.1. L_∞ will represent Tchebycheff and L_2 will stand for least-squares.

Notation 2.2. $X_m=\{x_1,x_2,\ldots,x_m\}$ is the discrete set of points on which approximation takes place.

Definition 2.2. A reference is a set $\{x_\sigma\}$ of (n+2) distinct abscissas from the set X_m.

Definition 2.3. The functions $\emptyset_0, \emptyset_1, \ldots, \emptyset_n$ are called base functions.

We'll assume we wish to approximate f(x) on X_m by an expression:

(2.1) $$\emptyset(x)=a_0\emptyset_0(x)+a_1\emptyset_1(x)+\ldots+a_n\emptyset_n(x).$$

The values $\emptyset(x_\sigma)$ of any function $\emptyset(x)$ are related by a linear equation:

$$\lambda_1\emptyset(x_1)+\lambda_2\emptyset(x_2)+\ldots+\lambda_{n+2}\emptyset(x_{n+2})=0.$$

Admitting the existence and uniqueness of interpolation we have $\lambda_\sigma \neq 0$, $\sigma=1,2,\ldots,n+2$.

Definition 2.4. Let $\emptyset$ be any function of class (2.1) and let $e_\sigma=\emptyset(x_\sigma)-f(x_\sigma)$ be the errors at x_σ, the points of reference. $\emptyset(x)$ is called a reference function with respect to the reference $\{x_\sigma\}$ if sgn e_σ=sgn λ_σ or if sgn e_σ=-sgn λ_σ, where sgn denotes the signum function.

Definition 2.5. The levelled reference function with respect to a given reference $\{x_\sigma\}$ is that function characterized by the property that the errors e_σ have the same absolute value.

Definition 2.6. The common absolute value $|e|$ of the approximation errors e_σ is called the reference-deviation corresponding to the given reference.

<u>Theorem 2.1.</u> (Exchange Theorem, [18]) Let a reference $\{x_\sigma\}$ and a corresponding reference function $\emptyset(x)$ be given. Furthermore let x_i be any abscissa not coinciding with a reference point. Then there is an abscissa x_ρ out of $\{x_\sigma\}$ such that $\emptyset(x)$ is again a reference function with respect to the reference built by the remaining points $x_\sigma \neq x_\rho$ of $\{x_\sigma\}$ and x_i.

<u>Theorem 2.2.</u> [19] Let $L(A^*,x)$ be the best T-approximation to $f(x)$ on X_m where $A^*=(a_1{}^*,a_2{}^*,\ldots,a_n{}^*)$. Then there is a subset of $(n+1)$ points of X_m such that $L(A^*,x)$ is the best approximation to $f(x)$ on this subset. Furthermore, this subset is one which maximizes the deviation of the best T-approximation to $f(x)$ among all subsets of $(n+1)$ points.

<u>Theorem 2.3.</u> (Characterization, [19]) $L(A^*,x)$ is the best T-approximation to $f(x)$ on X_m if and only if there exists an alternating set for $f(x)-L(A^*,x)$ consisting of $(n+1)$ points.

<u>Theorem 2.4.</u> [21] Let $\{\emptyset_i(x)\}$ be a T-set and let $L(A,x)$ be defined in the usual way. Then, given $f(x)$ defined on X_m and $1<q<p\leq\infty$, we have the following sets identical:

$\{A|L(A,x)$ is a best weighted L_p approximation to $f(x)$ on $X_m\}$,

$\{A|L(A,x)$ is a best weighted L_q approximation to $f(x)$ on $X_m\}$.

The above results were concerned mainly with linear T-approximation. The next several concepts are more directly involved with nonlinear approximation. First of all we have the basic statement that the interpolation problem is uniquely solvable.

Definition 2.7. The approximation function $F(A,x)$ is said to be solvent (of degree n) if, given a set $\{x_i\}$ of n distinct points in $[0,1]$ and a set $\{y_i\}$ of arbitrary numbers, there is an $A\varepsilon P$ (P is the parameter space) such that

$$F(A,x_i)=y_i, \ i=1,2,\ldots,n.$$

The next definition we need is an abstraction of the original definition of a Tchebycheff-set, which was needed in the linear case.

Definition 2.8. An approximating function $F(A,x)$ is said to have Property Z of degree n in $[0,1]$ if $A_1,A_2\varepsilon P$, $A_1\neq A_2 \Rightarrow F(A_1,x)-F(A_2,x)$ has at most $(n-1)$ zeros in $[0,1]$.

These two ideas can be molded together to yield the concept of a unisolvent function.

Definition 2.9. The approximating function $F(A,x)$ is said to be a unisolvent function if (1) it is solvent of degree n, and (2) has Property Z of degree n.

Most of the "interesting" nonlinear approximating functions are not unisolvent and, in fact, require that the above definitions be modified to produce local properties. We need the following restricted idea of solvency.

Definition 2.10. $F(A,x)$ is locally solvent of degree m at $A^*\varepsilon P$ if given a set $\{x_j | 0 \leq x_1 < x_2 < \ldots < x_m \leq 1\}$ and $\varepsilon > 0$, then there exists a $\delta(A^*,\varepsilon,x_1,\ldots,x_m) > 0$ such that $|y_j - F(A^*,x_j)| < \delta \Rightarrow$ there exists a solution $A\varepsilon P$ to the system:

$$F(A,x_j) = y_j, \quad j=1,2,\ldots,m,$$

with

$$||F(A,x) - F(A^*,x)|| < \varepsilon.$$

Definition 2.11. A varisolvent function $F(A,x)$ is a function which has Property Z of degree m at A^* and is locally solvent of degree m at A^*. The degree of F at A^* is the common degree of Property Z and local solvence and is denoted by $m(A^*)$.

The following theorem, which comes directly from Rice [20] is a basic result needed in Chapter III.

Theorem 2.5. Let F be varisolvent of degree $m(A^*)$ at $A^*\varepsilon P$. Then $F(A^*,x)$ is a best approximation to $f(x)$ on X iff $f(x) - F(A^*,x)$ alternates at least $m(A^*)$ times on X.

This theorem implies that the set X_A corresponding to a best approximation $F(A^*,x)$ consists of at least $m(A^*)+1$ points. This leads us immediately to the fact that the set W (referred to in the Lawson algorithm) has at least $m(A^*)+1$ points. Since the Lawson algorithm must be used on a finite subset of X we must keep in mind the fact that we may be faced with nonexistence of a solution. However, we have a "subset theorem" which is similar to the one for the linear case.

Theorem 2.6. [20] Let $F(A,x)$ be a varisolvent function, and let $F(A^*,x)$ be the best approximation to $f(x)$ on X. Then there is a subset X_0 of $m(A^*)+1$ points of X such that $F(A^*,x)$ is the best approximation to $f(x)$ on X_0. Furthermore X_0 is a subset which maximizes the deviation of the best approximation to $f(x)$ on all subsets of $m(A^*)+1$ points.

In order to find the best approximation to $f(x)$ on a given subset X_0 of $m^*(A)+1$ points, it is sufficient to solve the system of nonlinear equations:

$$F(A^*,x_j)-f(x_j)=(-1)^j d, \quad j=1,2,\ldots,m(A^*)+1.$$

This is usually a difficult system to solve and, in fact, there may not be any "a priori" knowledge concerning the degree $m(A^*)$ of the best approximation. The method of Remes presupposes that the degree $m(A^*)$ is known before the problem is solved. We are interested in investigating procedures which do not require such "a priori" information. It will be shown in the next chapter that the Lawson algorithm may be extended to handle nonlinear approximation. Before the discussion of several algorithms in depth, we will state and prove a linear theorem which is a model for a nonlinear one. The nonlinear one, which is very important in the theory, will be proved in Chapter III.

Theorem 2.7. [22] Given f(x) is a discrete function defined on the point set $X_m=\{x_i|i=1,2,\ldots,m\}$ (the x_i distinct) and a weight function w(x) defined on X_m. If q_n^* is the least-squares approximation to f(x) out of T_n, where T_n is a T-set, then

$$\sum_{i=1}^{m} [f(x_i)-q_n^*(x_i)]t(x_i)w(x_i)=0 \text{ for every } t\varepsilon T_n.$$

Proof: Assume there exists a $\bar{t}\varepsilon T_n$ such that

$$\sum_{i=1}^{m} [f(x_i)-q_n^*(x_i)]\bar{t}(x_i)w(x_i)=a>0.$$

Then

$$h=\sum_{i=1}^{m} \bar{t}(x_i)^2 w(x_i)>0.$$

This is true because at least (n+1) of the weights must be nonvanishing; which follows because the error curve f-q* must alternate at least n times. Since $\bar{t}\varepsilon T_n$ (which is a T-set) $\bar{t}$ can vanish at most n times in X_m. Hence there must exist at least one term in h which does not vanish.

Let

$$\lambda=\frac{a}{h} \neq 0.$$

Then

$$\sum_{i=1}^{m} [f(x_i)-q_n^*(x_i)-\lambda\bar{t}(x_i)]^2 w(x_i) =$$

$$\sum_{i=1}^{m} [f(x_i)-q_n^*(x_i)]^2 w(x_i)-2\lambda a+\lambda^2 h=$$

$$\sum_{i=1}^{m} [f(x_i)-q_n^*(x_i)]^2 w(x_i)-\lambda^2 h.$$

However $\lambda^2 h>0$ implies

$$||f-(q_n^*+\lambda\bar{t})||_2<||f-q_n^*||_2$$

and then q_n^* is not the least-squares approximation to f. But this is a contradiction.

B. Some Well-known Algorithms for Tchebycheff Approximation

1. The Exchange Method of Stiefel

In this section we will give a general description of the exchange iterative routine after Stiefel [18] and then describe a routine for the discrete T-problem using polynomials as the base functions.

A reference $\{x_\sigma\}$ is selected and the corresponding levelled reference function $\emptyset(x)$ is constructed. Its errors e_i have the property

$$M=\max|e_i|\geq|e|, i=1,2,\ldots,m$$

where $|e|$ is the reference deviation of $\emptyset$. Hence either $M>|e|$ or $M=|e|$. In the latter case we stop the iteration because $\emptyset$ is already a function of best fit. However, if $M>|e|$ a point x_i is selected where the error assumes its maximum value M. Using Theorem 2.1, a new reference is selected including the point x_i and having the property that $\emptyset$ is again a reference function. Among the errors $e_\sigma{}^*$ of $\emptyset$ at the new reference points, (n+1) are equal to $|e|$ in absolute value and one is equal to M. Now construct the levelled reference function $\emptyset^*(x)$ with respect to the new reference $\{x_\sigma^*\}$. Let $|e^*|$ be its reference deviation. Now $|e^*|>|e|$. A new reference is constructed and we repeat the process.

After a finite number of steps the procedure must come to an end because there is only a finite number of references in the whole set of abscissas and because the same reference can never occur twice during the routine. This is true because the reference deviation is always raised monotonically. Now we'll describe how this procedure applies if polynomial approximation is used.

The minimax polynomial approximation $p_n^*(x)$ of degree n to a function f(x) defined by a table of values has associated with it an error $E_n^*(x)=p_n^*(x)-f(x)$ which has at least (n+2) extremes with an alternation of sign from one to the next. This follows from Theorem 2.2. Recall that a polynomial of degree n has (n+1) parameters associated with it. Now assume f(x) is defined for the set of m points $\{x_i\}$, $i=1,2,\ldots,m$. Corresponding to any subset of (n+2) points $x_{i_1}<x_{i_2}<x_{i_3}<\ldots<x_{i_{n+2}}$ a polynomial $p_n(x)$ and a number E can be found such that

$$(2.2) \qquad p_n(x_{i_k})-f(x_{i_k})=(-1)^k E, \; k=1,2,\ldots,n+2.$$

It has been shown by de la Vallee Poussin [1] that the minimax approximation $p_n^*(x)$ to f(x) on X_m is that obtained by using the subset of (n+2) points which provides the largest possible absolute value for the solution E of the system (2.2). This actually amounts to the above defined Theorem 2.2. We could, of course, just compute the best approximation to f(x) on all subsets of $\{x_i\}$ of (n+2) points and select the one with the largest deviation. Such a scheme is impractical even with large computing machines because generally $m>>n$. Thus we use the exchange method which allows us to proceed to the largest deviation in just a few steps.

We proceed as follows for the case of polynomial T-approximation. Choose a subset of (n+2) points $\{x_{i_k}\}$ from the m points $\{x_i\}$ and solve the system (2.2). Assume for the present that the points are equally spaced throughout the finite interval $[x_1,x_m]$. After solving (2.2) the residuals $r_i=p_n(x_i)-f(x_i)$ are evaluated for $i=1,2,\ldots,m$. If no residual is greater than $|E|$, the problem is finished. Otherwise at least one more cycle of the calculation is required. To start the next cycle the set $\{x_{i_k}\}$ is chosen so as to correspond to the (n+2) largest residuals, consistent with the requirement of alternation in sign.

In general, this will imply that if a local extreme of the residuals, r_i, is found at a point x_i which is not a member of the set $\{x_{i_k}\}$ used to solve (2.2), the point x_i is then made to replace the nearest x_{i_k} which provided a residual of the same sign as r_i. In the event that there is an extreme of the residuals to the right of $x_{i_{n+2}}$, of opposite sign to $r_{i_{n+2}}$, the corresponding point is included in the set $\{x_{i_k}\}$ and x_{i_1} deleted if the residual at the point is greater in magnitude than $|r_{i_1}|$; otherwise the point is not used. A comparable procedure is used if an extreme is located to the left of x_{i_1}. The first cycle is completed by formation of new residuals r_i and selection of the $\{x_i\}$ corresponding to their extremes, to be used as the set $\{x_{i_k}\}$ to begin cycle two. Again, the minimax solution will be found after a finite number of cycles.

The exchange routine has been programmed in FORTRAN and tested using a variety of problems. It has been found, as one would expect, that the speed of convergence is directly related to the "goodness" of starting values. The algorithm performs well if the starting values are equally spaced over the given interval. There will be some gain if the starting values are the "Tchebycheff abscissas". Several authors have suggested, and it has been verified in the course of this study, that a propitious set of starting abscissas are those corresponding to the peaks of the error curve of the least-squares solution. Therefore, it is recommended that the least-squares problem be solved first; then the peaks of the L_2 error curve be located; and finally the x-values which correspond to these peaks be used as starting values for the exchange method. This choice of starting values will usually give convergence in one or two iterations. Actual numerical experience is tabulated in Chapter IV.

2. The Linear Programming Method of Barrodale and Young

We noted in Chapter I that the discrete T-problem can be attacked from the vantage point of linear programming. In this section we'll describe a special method due to Barrodale and Young [10] which utilizes a modified simplex algorithm. According to them, their algorithm, due to the structure of the tableaux, requires a minimum of storage space. They were trying to improve on Stiefel's approach which doubled the number of constraints and required "tedious transformations" to reduce the constraints to the original number. A basic feature of their method is the use of a simple transformation which guarantees that the unknown variables in the simplex method remain non-negative.

In the formulation of the linear programming model we'll assume we have a polynomial approximating function of the form $p_n(x)=\sum_{i=0}^{n} a_i x^i$. In addition to the n+1 coefficients of $p_n(x)$ we'll introduce a new variable, ρ, as follows:

The condition

$$\max_{x\varepsilon X_m}|f(x)-p_n(x)|=\rho$$

can be stated

$$-\rho\leq f(x_j)-\sum_{i=0}^{n} a_i x_j^i\leq\rho, \quad j=1,2,\ldots,m.$$

The linear programming problem is:

$$\text{minimize } \rho$$

subject to the 2m linear constraints

$$\rho+\sum_{i=0}^{n} a_i x_j^i \geq f(x_j),\ j=1,\ldots,m$$

$$\rho-\sum_{i=0}^{n} a_i x_j^i \geq -f(x_j),\ j=1,\ldots,m.$$

Barrodale and Young's method proceeds as follows using the above notation. Set $\alpha_{n+1}=\max(0,-\min_j a_j)$ and $\alpha_j=a_j+\alpha_{n+1}$ for $0\leq j\leq n$. Then, for $1\leq i\leq m$, define

$$e_i=e(x_i)=p_n(x_i)-f(x_i)$$

$$=\sum_{j=0}^{n}\alpha_j\emptyset_j(x_i)-\alpha_{n+1}\sum_{j=0}^{n}\emptyset_j(x_i)-f(x_i)$$

$$=\alpha_0\emptyset_{0,i}+\alpha_1\emptyset_{1,i}+\ldots+\alpha_n\emptyset_{n,i}+\alpha_{n+1}\emptyset_{n+1,i}-f_i$$

where $\emptyset_i(x)=x^i$, $\emptyset_{n+1}(x_i)=-\sum_{j=0}^{n}\emptyset_j(x_i)$ for $1\leq i\leq m$, and we've used the notational conveniences $\emptyset_{j,i}=\emptyset_j(x_i)$ and $f_i=f(x_i)$. Finally, putting $e_i=u_i-v_i$ where $u_i\geq 0$ and $v_i\geq 0$, we have m constraints in the nonnegative variables,

$$f_i=\alpha_0\emptyset_{0,i}+\alpha_1\emptyset_{1,i}+\ldots+\alpha_{n+1}\emptyset_{n+1,i}-u_i+v_i \text{ for } 1\leq i\leq m.$$

The L_∞ approximation problem is to find $\{a_j\}$ such that $\max_{1<i<m}|e_i|$ is minimized. For any $A\varepsilon E_n$ we put $\rho=\max_{1\leq i\leq m}|e_i|$ and obtain the constraints

$$(2.3)\quad \begin{array}{l}\alpha_0\emptyset_{0,i}+\alpha_1\emptyset_{1,i}+\ldots+\alpha_{n+1}\emptyset_{n+1,i}+\rho\geq f_i\\ -\alpha_0\emptyset_{0,i}-\alpha_1\emptyset_{1,i}-\ldots-\alpha_{n+1},\emptyset_{n+1,i}+\rho\geq -f_i.\end{array}\quad 1\leq i\leq m$$

This yields the linear programming problem of minimizing ρ subject to (2.3). In actual practice, the dual problem is solved because it reduces the number of constraints from 2m to n+3. This is a drastic reduction in most problems we'll solve since m is usually much greater than n. In the dual problem, we find nonnegative values of s_i and t_i for $1 \leq i \leq m$ which maximize $\sum_{i=1}^{m} f_i(s_i - t_i)$ subject to the (n+3) constraints:

$$\sum_{i=1}^{m} \phi_{j,i}(s_i - t_i) \leq 0 \quad \text{for } 0 \leq j \leq n+1$$

and

$$\sum_{i=1}^{m} (s_i + t_i) \leq 1.$$

The constraints can be expressed as equalities using the variables α_j and ρ, the original variables of (2.3), as the slack variables.

This method has been programmed using the ordinary simplex and also the revised simplex method. The numerical experience will be discussed in Chapter IV. The linear programming method has a definite advantage for certain types of problems. In particular, it would be highly suitable for problems which have added linear constraints. However, this method is generally not as accurate nor as quick to converge as the exchange method. The accuracy problem can be circumvented by using orthogonal polynomials if the base set is the set of polynomials. If the method takes many iterations one can get into numerical difficulty due to the inevitable rounding errors. This problem can be alleviated by going to double precision but only at increased expense.

3. The Lawson Algorithm

The Lawson algorithm consists of solving the discrete L_∞ problem by means of weighted L_2 approximations. Lawson's original algorithm as published in his thesis [11] computed best Tchebycheff approximations as the limit of a special sequence of best weighted L_p approximations with p fixed. The interesting case is for p=2. The possibility that such an algorithm might exist follows from the work of Motzkin and Walsh [21] which resulted in Theorem 2.4. From Theorem 2.4 we see that it is indeed possible to compute a best T-approximation by computing a certain weighted least-squares approximation. This is desirable because the second computation involves solving a problem which has a more desirable characteristic property and hence a more stable solution. To be specific, the least-squares problem does not depend on an iterative scheme and hence results will not vary given a reliable least-squares routine. Lawson's algorithm computes the desired weight function.

In the Lawson algorithm, we define a sequence of weight functions $w^k(x_i)=w_i^k$ with $\sum_{i=1}^{m} w_i^k=1$ and corresponding approximations $L(A_k,x)$ as follows. Select $w_i^{(1)}>0$ arbitrarily. Then iterate on the following two statements.

(1) $L(A_k,x)$ is the best L_2 approximation to $f(x)$ on X_m with the weights w_i^k.

(2) $$w_i^{k+1} = \frac{w_i^k |f(x_i)-L(A_k,x_i)|}{\sum_{i=1}^{m} w_i^k |f(x_i)-L(A_k,x_i)|} .$$

Return to statement (1).

Obviously, we have defined an infinite iterative procedure and we must have some way of terminating the algorithm after a finite number of steps. We are guaranteed by the following theorem that the algorithm is convergent.

Theorem 2.8. [17] The sequence $L(A_k,x)$ converges to $L(A_0,x)$ which is the best L_∞ approximation to $f(x)$ on a set $X_2 \subset X_m$. The sequence $\{\sigma^k\}$,

$$\sigma^k = \left[\sum_{i=1}^{m} w_i^k [f(x_i)-L(A_k,x_i)]^2\right]^{1/2}$$

is monotonically increasing (strictly so unless convergence takes place in a finite number of steps), and

$$\lim_{k\to\infty} \sigma^k = \max_{x\in X_2} |f(x)-L(A,x)| = \sigma^* .$$

Thus a natural stopping criterion is to key on σ^k as we proceed from one step to another. There is the possibility that we might converge on a proper subset of X_m but if that happens Lawson has developed the following restart theorem which comes to our rescue.

Theorem 2.9. [17] If X_2 is a proper subset of X_m, then the algorithm may be restarted with

$$\bar{w}_i^1=(1-\lambda)\lim_{k\to\infty} w_i^k+\lambda u(x),\ 0\leq\lambda<1,$$

where $u(x)=0$ for $x\neq z$ and $u(z)=1$, where $z\varepsilon X_m-X_2$ and $|f(x)-L(A_o,z)|>\sigma^*$. For λ sufficiently small, $\bar{\sigma}^1>\sigma^*$, and after a finite number of restarts, we obtain the best L_∞ approximation to $f(x)$ on X_m.

In actual practice it is very rare that one must restart. Even though, in theory, the algorithm can interpolate at a critical point because of the inevitable rounding errors this will seldom occur.

Although theoretically pleasing (and also practically pleasing from the simplicity of implementation), the algorithm suffers from the handicap of very slow convergence. Rice and Usow [17] have attempted to accelerate the convergence by extending Lawson's original algorithm. We will briefly describe the acceleration scheme which they found useful.

(1) Do ℓ steps of the Lawson algorithm.

(2) Set $w_i^k=0$ if

$$|f(x_i)-L(A_k,x_i)|\leq\lambda_k\sigma^k \text{ where}$$

$$\lambda_k=\frac{\sigma^k}{\max_x|f(x)-L(A_k,x)|}\ .$$

(3) Go back to step 1.

In the algorithm one is interested in making $w^k(x)$ tend to zero as rapidly as possible except at the extremal points of the error curve of the best L_∞ approximation.

It is precisely to this task that the acceleration is addressing itself. According to Rice and Usow [17],

> For a typical problem involving $n=4$ parameters and $m=50$ points, the acceleration scheme reduced the number of iterations from over 250 to less than 15 using values of ℓ where $1\leq\ell\leq 4$. This is for convergence to 7 significant digits.

Although we found similar results holding true for problems involving relatively small values of n and m ($n\leq 6$ and $m\leq 51$), we discovered that this acceleration scheme failed quite often once the number of points was increased significantly ($m\doteq 100$). The reason for this can be traced to the fact that step (2) of the acceleration scheme only holds true in the limit as $k\to\infty$ and may not hold true early in the algorithm. Perhaps (2) should read

$$|f(x_i)-L(A_k,x_i)|<<\lambda_k\sigma^k.$$

4. Non-Lawson Nonlinear Approximation

There have been numerous papers written and methods proposed for solving the nonlinear problem via Remes-type algorithms. Thus, it will not be our concern to investigate such procedures here. Rather, we are interested in methods which handle a more general-type problem than the rational one, which is the principle one handled by the Remes algorithms. Such a method is the linear programming technique of Osborne and Watson [15].

The nonlinear L_∞ problem in the discrete case can be formulated in a manner analogous to the linear programming formulation of the linear L_∞ problem. The solution is obtained by minimizing h subject to the constraints:

$$|f_i - F_i(a)| \le h, \quad i=1,2,\ldots,m . \tag{2.6}$$

This problem is solved iteratively as follows:

(1) Calculate δa^j to minimize h^j subject to the constraints:

$$|f_i - F_i(a^j) - \langle \nabla F_i(a^j), \delta a^j \rangle| \le h^j, \quad i=1,2,\ldots,m . \tag{2.7}$$

This is a discrete L_∞ problem which can be solved by linear programming. Denote the minimum value of h^j by $\hat{h}^j$.

(2) Calculate γ^j to minimize the maximum value of

$$|f_i - F(a^j + \gamma^j \delta a^j)|, \quad i=1,2,\ldots,m .$$

Let this minimum value be denoted by $\bar{h}^{j+1}$.

(3) Set $a^{j+1} = a^j + \gamma^j \delta a^j$.

To get convergence we must assume the existence of at least one bounded minimum for each problem and that F is continuous as a function of x. In addition we need these assumptions:

(a) $F_i(a+\delta a) = F_i(a) + \nabla F_i \delta a + 0(||\delta a||^2)$, $i=1,2,\ldots,m$ where ∇F_i is the row vector with components $\frac{\delta F_i}{\delta a_j}$, $j=1,2,\ldots,n$.

This permits at least a local linearization of the non-linear problem.

(b) The rank of matrix M, $M=\nabla F$, is n.

This means the linearized problem can be solved via linear programming.

(c) The system of equations $f_i - F_i(a) = 0$, $i=1,2,\ldots,m$ is inconsistent.

Although Lee and Roberts [16] give this method a fairly good rating in their study, we are concerned with the method in a more general setting than they were. Experience in running this procedure indicated that the algorithm was often marking time and was perhaps much slower than it needed to be. Thus the procedure was modified and improvements were made which will be discussed in the next chapter.

III. REVISED AND NEW ALGORITHMS

The main purpose of this chapter is to describe two modified algorithms and some new methods for L_∞-approximation. In the case of the Lawson algorithm extended to handle the nonlinear problem, theoretical work will also be provided.

A. Some Modified and Improved Algorithms

1. The Lawson "Peaks" Acceleration

The Lawson algorithm needs to be accelerated in some manner if it is going to be competitive with the more popular procedures. Although Rice and Usow [17] put forward an acceleration scheme which was described in Chapter II, section B, their technique appeared to have some shortcomings. After much experimentation, a new algorithm which uses the Lawson algorithm as its base was developed. This new procedure was alluded to by Lawson in his thesis when he noticed some peculiarities in his numerical experimentation. This new algorithm capitalizes on the fact that Lawson's algorithm tends to "move" the peaks of the residual curve to the "reference set" or so-called "critical" set of points rather quickly. Once this "critical" point set is realized the problem is essentially solved since these are the points which should receive all the weight. Hence, you zero out the weights at non-critical points and the algorithm will converge immediately.

Using this new acceleration procedure, it became possible to reach convergence to six significant digits in only seven iterations where before the same problem took as many as 40 iterations for the same accuracy. Numerical results will be given in Chapter IV.

The following is a brief algorithmic description of the "peaks" acceleration method.

<u>Algorithm 3.1</u>

```
(1) Solve the weighted least-squares problem using
    Lawson's algorithm $\ell$ times ($\ell \geq 3$).
(2) Locate the "peaks" of the error curve.
(3) Do another Lawson iteration.
(4) Locate the "peaks" of the "new" error curve.
(5) Compare the "new peaks" with the "old peaks".
    (a) If they are equal go on to step (6).
    (b) If they are not equal go back to step (3).
(6) Zero out the weights at the non-critical points
    and continue with Lawson's algorithm.
```

Although this new algorithm appears to work well on a large class of problems, there do exist problems which give it difficulty. Suitable modifications can be made to this algorithm which enable it to handle these problems also. However, those modifications force the algorithm to do so many calculations and so much comparing that it is no longer competitive. It was decided that perhaps the weighted least-squares ideas of Lawson could be used in another context to develop a new algorithm. This algorithm will now be discussed.

2. L_∞ Approximation Via Unconstrained Least-Squares

The following is a description of how the discrete L_∞ problem was tackled by the method of unconstrained least-squares. A constrained least-squares problem is usually written:

$$\text{minimize:} \quad f(x)$$

$$\text{subject to:} \quad g_i(x) \geq 0, \; i=1,2,\ldots,m.$$

The unconstrained form of this is:

$$\text{Minimize:} \quad V(x,r_k)=f(x)+\frac{1}{r_k}\sum_{i=1}^{m}\{\min[0,g_i(x)]\}^2.$$

For the problem under consideration:

$f(a)=\sum_{i=1}^{n}(y_i-f_i)^2$ where y is typically in the form

$$y=a_0+a_1x+a_2x^2+\ldots\;.$$

$g_i(a)\geq 0$ takes the form:

$$|y_k-f_k|\leq\alpha \text{ or}$$

$$(y_k-f_k)^2\leq\alpha^2 \text{ or}$$

$$\alpha^2-(y_k-f_k)^2\geq 0 \text{ or}$$

$$g(a)=\alpha^2-(y_k-f_k)^2\geq 0.$$

Thus our problem takes the form:

$$\text{Min:}\quad V(a,r_k)=f(a)+\frac{1}{r_k}\sum_{i=1}^{m}\min[0,\alpha^2-(y_k-f_k)^2]^2$$

or

$$\text{Min:}\quad V(a,r_k)=f(a)-\frac{1}{r_k}\sum_{i=1}^{m}\min[0,\alpha^2-(y_k-f_k)^2].$$

For one constraint the problem is:

$$\text{Min:}\quad V(a,r_k)=(y_1-f_1)^2+\ldots+(y_n-f_n)^2-\frac{1}{r_k}[\alpha^2-(y_{k_1}-f_{k_1})^2].$$

Let's assume $k_1=2$; constraint number one is the place where the maximum error occurs in the least-square error curve. Then the problem is

$$\text{Min:}\quad V=(y_1-f_1)^2+(y_2-f_2)^2+\frac{1}{r_k}(y_2-f_2)^2+\ldots+(y_n-f_n)^2-\frac{\alpha 2}{r_k}$$

or

$$\text{Min:}\quad V=(y_1-f_1)^2+(1+\frac{1}{r_k})(y_2-f_2)^2+\ldots+(y_n-f_n)^2-\frac{\alpha^2}{r_k}\ .$$

This is a typical least-squares problem where all the weights are one except at the second point which has a weight of $(1+\frac{1}{r_k})$. The real problem is deciding how we should select r_k. We can choose the first r_k (call it $r_k^{(1)}$) experimentally. This will simply give the weight at the first "critical" point a disproportionate amount of the total.

We now check to see if:

$$\alpha^2-(y_k-f_k)^2\geq 0 \text{ for } k=k_1 \text{ (first critical point).}$$

If the answer is yes, the first stage is complete and we transfer immediately to the next paragraph below. If the answer is no, $r_k^{(1)}$ must be made smaller and we re-solve the weighted least-squares problem.

Next find another peak, excluding the first peak from consideration. We now have two constraints for our unconstrained least-squares problem. The problem now is

$$\text{Min:}\quad V=(y_1-f_1)^2+\ldots+(y_n-f_n)^2$$

$$\text{(2.4)}\qquad -\frac{1}{r_k^{(2)}}\,[\min\{0,[\alpha^2-(y_{k_1}-f_{k_1})^2]\}$$

$$+\min\{0,[\alpha^2-(y_{k_2}-f_{k_2})^2]\}]\ .$$

Now we desire to have different $r_k^{(2)}$ values for each constraint. That is $\frac{1}{r_k^{(2)}}=\left[\frac{1}{r_{k_1}^{(2)}}\ ,\ \frac{1}{r_{k_2}^{(2)}}\right] = R.$

The problem at (2.4) may now be written in the form:

$$V=\gamma_1(y_1-f_1)^2+\left[1+\frac{1}{r_k^{(1)}}+\varepsilon_1\right](y_2-f_2)^2$$

$$\text{(2.5)}\qquad +\gamma_1(y_3-f_3)^2+\ldots+\gamma_1(y_n-f_n)^2$$

$$-RB^T$$

where $B=\left[\min\{0,[\alpha^2-(y_{k_1}-f_{k_1})^2]\},\ \min\{0,[\alpha^2-(y_{k_2}-f_{k_2})^2]\}\right]$.

The big problem is how to select the vector entries in $R=\left[\frac{1}{r_{k_1}^{(2)}}\ ,\ \frac{1}{r_{k_2}^{(2)}}\right]$. If the first constraint is in bounds there's nothing to select; otherwise simply increase the weight experimentally. To have the second constraint be in bounds, $\frac{1}{r_{k_2}^{(2)}}$ was selected experimentally. If it did not do the job, we increased the weight at the second constraint further. If the first and second constraints are within bounds (i.e. the peaks are not out of range) then determine the third constraint in a similar manner.

The whole idea behind the method is to compute the weighted least-squares error curve and then check to see where it reaches its maximum value. At this point we should weight the curve down, forcing it to increase at other values. The procedure is based on a push-down, pop-up situation which we will know will occur because of the nature of the alternating error curve. It follows from the work of Motzkin and Walsh, to which we have made reference before, that L_∞ approximation is simply a weighted L_2 approximation. Hence, we are proposing an alternate method for finding the weights, or more importantly, the "critical" points. Numerical results will be given in Chapter IV.

3. Nonlinear L_∞ Approximation (Lawson)

Since the Lawson method is rather straightforward to program, depending on only an adequate least-squares solver as its base, it was decided to try and apply this procedure to the nonlinear problem. This is a rather natural extension and one suggested by Rice [20]. To quote from Rice,

> There are two directions for extending this algorithm which suggest themselves. The first is to approximation by varisolvent and other nonlinear approximating functions. This direction is of lesser interest because it is not clear at this time that it is easier to compute nonlinear L_2-approximations than it is to compute nonlinear Tchebycheff approximations. The other direction of extension is toward the computation of other L_p approximations for $p<\infty$.

This second direction of extension will not concern us here. Although it may not have been true when Rice was writing his text, it certainly appears to be true today that the nonlinear L_2 problem is easier to solve than the nonlinear L_∞ problem, provided a solution exists. Thus, it is natural to seek out an adequate nonlinear L_2 solver and build the nonlinear Lawson procedure around it. It was decided to use the Marquardt algorithm as the L_2 solver. This procedure was first developed by Levenberg [23] and later expanded on by Marquardt [24].

There is an inherent difficulty in attempting to solve the L_∞ problem in this manner. We will constantly be iterating within an iteration and therefore cannot hope for speedy results. However, we are interested in getting results where results have never been achieved before. Thus, the time of solution need only be a secondary consideration. We are more concerned with the problem of convergence. Results garnered from the theory, which appears later in this chapter, indicate that we do have a convergent algorithm for varisolvent functions. It may happen that we have an algorithm which works for other types of nonlinear functions as well.

The basic nonlinear algorithm, which follows from the linear model, will be stated here for the sake of completeness.

<u>Algorithm 3.2</u> Let $L(A,x)$ be a varisolvent approximating function having degree $m^*(A)$. We wish to approximate $f(x_i)=f_i$, $i=1,2,\ldots,m$ on a set $X_m=\{x_i \mid i=1,2,\ldots,m\}$. Define a sequence of weight functions $w^k(x)$ on X_m and a corresponding sequence $\{L(A_k,x)\}$ of best nonlinear L_2-approximations to $f(x)$ with weights $w^k(x)$. Select $w_i^{(1)}>0$ arbitrarily. Then iterate on the following two statements.

(1) $L(A_k,x)$ is the best nonlinear L_2-approximation to $f(x)$ on X_m with weights w_i^k.

(2) $$w_i^{k+1} = \frac{w_i^k |f(x_i)-L(A_k,x_i)|}{\sum_{i=1}^{m} w_i^k |f(x_i)-L(A_k,x_i)|} .$$

In addition to generating a sequence of weight functions as we iterate, we can also generate the following sequence

$$\sigma^k = \left[\sum_{i=1}^{m} w_i^k [f(x_i)-L(A_k,x_i)]^2\right]^{\frac{1}{2}} .$$

The significance of this sequence is that it converges to σ^*, the minimax error (in the limit). In section B we will prove a sequence of lemmas and theorems which give this algorithm its real power.

4. An Extension of Osborne and Watson's Algorithm

Since the method of linear programming as applied to the nonlinear T-problem by Osborne and Watson [15] seemed to converge quite slowly for many problems, it was decided to modify their method as follows. As we move from one outer iteration to the next we may change the A-matrix, actually ∇F, by a very small amount. But we are forced to re-solve the problem from the very beginning if we proceed as detailed by Osborne and Watson. Essentially, they do not make use of any previous information that was computed. Rather than going back and re-solving from an initial basis we decided to retain the last basis and restart using this basis as our new basis.

On several examples this technique seemed to work. However, if the initial guess was "bad" it turned out that restarting in this manner could lead to infeasibility. It was at this stage that the author was reminded of a result in Hadley [25] which was particularly appropriate for this occurrence. The procedure that was recommended was to use the dual simplex algorithm. This method should not be confused with the dual formulation of the primal problem. It is this dual formulation which was so useful in solving the linear L_∞ problem.

The dual simplex algorithm allows one to solve a linear programming problem by starting with an infeasible solution. However, it is necessary to be superoptimal or have $z_j-c_j \geq 0, \forall_j$, when we start this algorithm. After we restart this may not be the case. If $z_j-c_j<0$, for some j, we proceed on with the ordinary simplex until it has converged. This insures that $z_j-c_j \geq 0, \forall_j$, since this condition is required for convergence. Now we may still have infeasibilities. Here's where we check for these and pass into the dual simplex if necessary.

This dual simplex algorithm forces one to determine the vector to leave the basis first and then to choose a vector to enter. This is the reverse of what is done in the simplex method. The dual simplex method is applied directly to the primal problem. With the addition of the code for this procedure the modified algorithm was able to "restart" using the last basis and to "recover" if the resulting solution went infeasible. The installation of this routine into the old routine of Osborne and Watson can only make their algorithm more competitive. The use of this procedure was found to be extremely worthwhile as indicated by the numerical work in Chapter IV.

It's possible that we may run into numerical troubles when we restart with a solution which is infeasible but not superoptimal and return to the ordinary simplex. Since we have implemented the usual rule for determining a vector to enter the basis, we are forcing the negative z_j-c_j's out as fast as possible. However, we may decrease the objective function at any iteration by applying the usual rule for finding a vector to leave the basis. Thus there is the possibility that we might repeat an old basis and run into cycling problems. This will probably not happen due to the inevitable rounding errors. We will now indicate how the usual rule for determining a vector to leave the basis can be modified to alleviate this problem.

The usual rule, implementing Hadley's [25] notation is: compute

$$\frac{x_{B_r}}{y_{rk}} = \min_i \left[\frac{x_{B_i}}{y_{ik}} , y_{ik}>0 \right] ;$$

the vector in column r of the basis is removed and replaced by a_k. This rule naturally assumes feasibility or $x_{B_i} \geq 0, \forall i$. If we have at least one $y_{ik}>0$ and the corresponding $x_{B_i} \geq 0$ we can apply this rule. However, if not then we should use the following rule: compute

$$\frac{x_{B_r}}{y_{rk}} = \max_i \left[\frac{x_{B_i}}{y_{ik}} , y_{ik}<0 \right] ;$$

where the x_{B_i} we check are non-positive. This second rule guarantees that the objective function does not decrease. In actual test-case runs it was found that cycling did not occur when the usual rule was applied and that the modified rule only increased the number of iterations.

We will now make a few statements about starting values for any nonlinear L_∞ method. As a result of working with Lawson's nonlinear method it is conjectured that nonlinear L_∞-approximation is just weighted nonlinear L_2-approximation. Since an "initial guess" is needed to get the various L_∞ methods started it is conjectured that the best way to get a "good" initial guess is to first solve the L_2 problem (which is generally much easier to solve) and then to use the L_2 solution as a first guess at an L_∞ solution. The natural question which arises is "What happens if no solution exists for the L_2 problem?" It appears reasonable to conjecture that if we cannot solve this problem then the corresponding L_∞ problem cannot be solved either. Thus a logical route to follow on the way to the solution of the L_∞ problem is to proceed via the L_2 solution. We must watch for pitfalls, however, since a given L_2 algorithm may be very sensitive to certain types of problems and it may fail even when a solution exists.

B. Lawson Nonlinear - The Theory

Most of the theory treated in this section corresponds to similar results already proved by Lawson for the linear case. When a proof for the nonlinear case follows immediately from the linear one it will not be given here. The following theorem parallels one given in Chapter II and is crucial for the proof of Lemma 3.2.

Theorem 3.1. Given $f(x)$ is a discrete function defined on the point set $X_m=\{x_i|i=1,2,...,m\}$ (the x_i distinct) and a weight function defined on the set X_m. Assume that $L(A,x)$, the set of approximating functions, is varisolvent. If q^* is the least-squares approximation to f out of $L(A,x)$, then

$$\sum_{i=1}^{m}[f(x_i)-q^*(x_i)]\bar{L}(x_i)w(x_i)=0$$

for every $\bar{L}\varepsilon L(A,x)$.

Proof: First, recall that if $L(A,x)$ is varisolvent then there will be at least $d^*(A)+1$ nonvanishing weights where $d^*(A)$ is the degree of the approximating function. Now assume there exists an $\hat{L}\varepsilon L(A,x)$ such that

$$\sum_{i=1}^{m}[f(x_i)-q^*(x_i)]\hat{L}(x_i)w(x_i)=a>0.$$

Then

$$h=\sum_{i=1}^{m}\hat{L}(x_i)^2w(x_i)>0.$$

This is true because of the varisolvent property of $L(A,x)$. The varisolvence of $\hat{L}$ implies that $\hat{L}$ has at most $d^*(A)$ zeros on X_m, where $d^*(A)$ is the degree of varisolvency. However, from above, the weights cannot vanish at $d^*(A)+1$ points. Thus all the terms in the sum at h cannot be zero and in fact one must be greater than zero.

Let $$\lambda = \frac{a}{h} \neq 0.$$

Then $$\sum_{i=1}^{m}[f(x_i)-q^*(x_i)-\lambda\hat{L}(x_i)]^2w(x_i)=$$

$$\sum_{i=1}^{m}[f(x_i)-q^*(x_i)]^2-2\lambda a+\lambda^2h \quad =$$

$$\sum_{i=1}^{m}[f(x_i)-q^*(x_i)]^2-\lambda^2h.$$

However $\lambda^2 h$ positive implies

$$||f-(q^*+\lambda\hat{L})||_2 \le ||f-g^*||_2 \ .$$

But this is a contradiction.

Next we wish to prove a sequence of lemmas and theorems which give the nonlinear Lawson algorithm its real power.

<u>Lemma 3.1</u> If $\sigma^1>0$, then $\sigma^k>0$, for all k.

Proof: same as in the linear case.

In several of the following lemmas we will use the inner product notation:

$$\left\langle f,g\right\rangle_w = \sum_{i=1}^{m} w(x_i)f(x_i)g(x_i).$$

We will also let $W_k=\{x_i|w_i^k>0\}$. All summations are over the set X_m unless otherwise indicated.

<u>Lemma 3.2</u> If $w_i^{k+1}=w_i^k, \forall_i$, then $\sigma^{k+1}=\sigma^k$; otherwise $\sigma^{k+1}>\sigma^k$.

Proof: The first assertion is clear; therefore we assume $w^{k+1}(x)\neq w^k(x)$.

Since $\sum w_i^{k+1} e_i^{k+1} L(A_{k+1},x)=0$, (this follows from Theorem 3.1) we have

$$(\sigma^{k+1})^2 = \sum w_i^{k+1} [f(x_i) - L(A_{k+1}, x)]^2$$

$$= \sum w_i^{k+1} [e_i^{k+1}][f(x_i) - L(A_{k+1}, x_i)]$$

or

$$\sigma^{k+1} = \frac{\sum w_i^{k+1} e_i^{k+1} f(x_i) - \sum w_i^{k+1} e_i^{k+1} L(A_{k+1}, x)}{\sigma^{k+1}}$$

$$= \left\langle f_i, \frac{e_i^{k+1}}{\sigma^{k+1}} \right\rangle_{w_i^{k+1}}$$

or

(3.1) $$\sigma^{k+1} = \frac{\sum f_i e_i^{k+1} w_i^{k+1}}{\left[\sum (e_i^{k+1})^2 w_i^{k+1}\right]^{1/2}}$$

Consider $$g_j = \frac{e_j^{k+1}}{\left[\sum (e_i^{k+1})^2 w_i^{k+1}\right]^{1/2}}$$

and recall that it is a property of least-squares approximation that:

(1) $\langle g, g \rangle_{w^{k+1}} = 1$.

(2) $g \perp L(A_{k+1}, x)$ in the L_2 norm with weights w^{k+1}.

(3) g maximizes $\sum f_i g_i w_i^{k+1}$ over all g satisfying (1) and (2).

Since $\sum L(A_k, x_i) e_i^k w_i^k = 0$ we have

$$\sum L(A_k, x_i) \left[\frac{e_i^k w_i^k}{w_i^{k+1}} \right] w_i^{k+1} = 0, \text{ for } w_i^{k+1} > 0.$$

Now

$$0 = \sum L(A_k, x_i) e_i^k w_i^k = \sum_{W_k} L(A_k, x_i) e_i^k w_i^k = \sum_{W_{k+1}} L(A_k, x_i) e_i^k w_i^k.$$

Let

$$\hat{g}_j = \begin{cases} (e_j^k w_j^k / w_j^{k+1}) / \left[\sum (e_i^k w_i^k)^2 / w_i^{k+1}\right]^{1/2} & \text{for } w_i^{k+1} > 0 \\ 0, \text{ otherwise.} \end{cases}$$

$\hat{g}_j$ satisfies (1) and (2) above. Thus replacing g by $\hat{g}$ in (3.1) does not increase the left hand side.

Hence we have

$$(3.2) \quad \sigma^{k+1} \geq \frac{\sum f_i e_i^k w_i^k}{\left[\sum (e_i^k w_i^k)^2 / w_i^{k+1}\right]^{1/2}} = \frac{\sum f_i e_i^k w_i^k}{\sum w_i^k |e_i^k|}$$

The equality in (3.2) follows by writing the denominator as:

$$\left[\sum (e_i^k w_i^k)^2 / w_i^{k+1}\right]^{1/2} = \left[\frac{\sum |e_i^k|^2 (w_i^k)^2}{\frac{w_i^k |e_i^k|}{\sum w_i^k |e_i^k|}}\right]^{1/2}$$

$$= \left[(\sum |e_i^k| w_i^k)(\sum |e_i^k| w_i^k)\right]^{1/2} = \sum w_i^k |e_i^k| \ .$$

Now compare $\sum w_i^k |e_i^k|$ (which is the denominator on the right hand side of (3.2)) with $\left[\sum w_i^k (e_i^k)^2\right]^{1/2}$. It's certainly true that:

$$\left[\sum |e_i^k| (w_i^k)\right]^2 \leq \sum (e_i^k)^2 w_i^k \ .$$

But this implies $\sum |e_i^k| w_i^k \leq \left[\sum (e_i^k)^2 w_i^k\right]^{1/2}$.

It follows that $\dfrac{1}{\sum|e_i^k|w_i^k} \geq \dfrac{1}{[\sum(e_i^k)^2w_i^k]^{1/2}}$.

Using this fact in (3.2) we get the result

$$\sigma^{k+1} > \frac{\sum f_i e_i^k w_i^k}{\sum w_i^k|e_i^k|} \geq \frac{\sum f_i e_i^k w_i^k}{[\sum(e_i^k)^2 w_i^k]^{1/2}} = \sigma^k.$$

<u>Lemma 3.3</u> Let L(A*,x) be the best L_∞ approximation to f(x) on X. Then

$$\sigma^k \leq \xi^* = \max_{x\varepsilon X}|f(x)-L(A^*,x)|.$$

Proof: This follows as in the linear case.

Lemmas 3.4 through 3.6 are leading up to a very important result, Theorem 3.2. It is this convergence theorem which gives the new Lawson algorithm its real power. All summations are still over the whole set X_m. Let $L(A_u,x)$ and $w^u(x)$ be subsequences which converge to the limits L(A',x) and w'(x) respectively, where L(A',x) is a weighted L_2 approximation. Let

$$W'=\{x|w'(x)>0\}.$$

<u>Lemma 3.4</u> L(A',x) is the best T-approximation to f(x) on W'.

Proof: We'll first show that L(A',x) is a T-approximation. W' is not empty since $\sum w^k(x)=1$ and $\sigma^* = \lim_{u\to\infty}\sigma^u = \lim_{u\to\infty}\langle e^u,e^u\rangle_{w^u}^{1/2} \neq 0$.

Since $L(A',x)$ is a weighted L_2 approximation it must alternate at least $(n+1)$ times, where n is the degree of varisolvency. Therefore $(n+1)$ of the e^u must not vanish. But if these errors are to make their contribution to the least-squares error then the corresponding weights must not vanish. Hence W' must contain at least $(n+1)$ points.

Now w^{u+1} and σ^u are continuous functions of w^u. Let us start the algorithm for this new sequence with

$$w^{(1)}(x_i)=w'.$$

Now either $w^{(2)}(x_i)=w^{(1)}(x_i)$ or $\sigma^2>\sigma^*$. We know that $\lim_{k\to\infty} w^k(x)=w^{(1)}=w'$; also $\lim_{k\to\infty} \sigma^k=\sigma^*$ and $\sigma^{k+1}(w^{(k)})$ is a continuous function of $w_i^{(k)}$. Hence $\sigma^2(w^{(1)})=\sigma^*$, for otherwise σ^k does not converge to σ^*. Thus $\sigma^2=\sigma^*=(\sigma^{(1)})$. So $|e^{(1)}(x)|=|f(x)-L(A',x)|$ is constant on W'. Therefore $L(A',x)$ is a T-approximation. But is it a best approximation? Assume there exists a better T-approximation, call it $L(A'',x)$. Then

$$|f(x)-L(A'',x)|<|f(x)-L(A',x)|, \quad x\varepsilon W'.$$

But this contradicts the fact that $L(A',x)$ is a best weighted L_2 approximation to $f(x)$ on W'.

<u>Lemma 3.5</u> $\lim_{k\to\infty}\{w_i^{k+1}(x)-w_i^k(x)\}=0.$

Proof: Assume the contrary. Then there is a subsequence denoted by $\{\hat{w}_i^{k+1}-\hat{w}_i^k\}$ which converges to a nonzero limit ℓ.

Let $\{w_i^\ell\}$ be a subsequence of $\{\hat{w}_i^k\}$ which converges to $w^{(0)}(x)$. We know that if the algorithm is started with $w_i^{(1)}=w_i^{(0)}$ then $\sigma^2=\sigma^0$ and $w_i^{(2)}=w_i^{(0)}$.

$$\text{Therefore } \lim_{\ell\to\infty} w_i^{\ell+1} = \lim_{\ell\to\infty} \frac{w_i^\ell|e_i^\ell|}{\sum w_i^\ell|e_i^\ell|}$$

$$= \frac{w_i^{(0)}|e_i^{(0)}|}{\sum w_i^{(0)}e_i^{(0)}} = w_i^{(0)} = \lim_{\ell\to\infty} w_i^\ell$$

This implies that $\lim_{\ell\to\infty} \{w_i^{\ell+1}-w_i^\ell\}=0$.
Therefore for any convergent subsequence of $\hat{w}_i^k$ we have $\{(\hat{w}_i^{k+1}-\hat{w}_i^k)\}$ converges to zero, which then must be true for the whole sequence. So $\lim_{k\to\infty} (w_i^{k+1}-w_i^k)=0$; but this is a contradiction.
Let W be the limit points of $w^k(x)$. It is obvious that W is non-empty, closed and bounded. Also by Lemma 3.5 we know that it is connected.

<u>Lemma 3.6</u> Every $w(x)\varepsilon W$ gives the same L_2 approximation to f(x).

Proof: We can decompose the set W into equivalence classes by saying two weight functions are equivalent if they give rise to the same approximation. If L(A,x) is a best L_2 approximation to f(x) with weights w(x), then it is the unique best L_∞ approximation to f(x) on W'. This follows from Lemma 3.4. However, the set X is finite so there is at most a finite number of equivalence classes, each of which is compact and distinct. But the connectedness of W implies there is at most one equivalence class. Therefore every $w \varepsilon W$ yields the same L_2-approximation.

Combining these results we finally have the following theorem.

<u>Theorem 3.2.</u> The sequence $L(A_k,x)$ converges to $L(A_0,x)$ which is a best T-approximation to $f(x)$ on X_1.

Proof: We really only need to show that $\{L(A_k,x)\}$ converges. This sequence is obviously bounded and hence contains convergent subsequences. If there exist two subsequences with different limits, consider the corresponding weight functions. These sequences have convergent subsequences which lead to the same approximation by the previous lemma. Hence there are not two different limits but only one which we have called $L(A',x)$ in Lemma 3.4. Identifying $L(A',x)$ with $L(A_0,x)$ in this theorem gives us our desired result.

There is the distinct possibility that we might converge on a subset X_1 of X. If this happens we have not solved our original problem but need to restart our algorithm and try again. The following theorem does allow us to restart.

<u>Theorem 3.3.</u> If X_1 is a proper subset of X, then the algorithm may be restarted with

$$w_i^\lambda=(1-\lambda)w^{(0)}(x)+\lambda u(x),\ 0\leq\lambda<1,$$

where $u(x)=0$ for $x\neq z$ and $u(z)=1$, for $z\varepsilon X-X_1$ and $L(A_0,z)-f(z)>\sigma^*$. For λ sufficiently small, we have

$$\sigma^1>\sigma^*$$

and after a finite number of restarts we obtain the best L_∞ approximation $L(A^*,x)$ to $f(x)$ on X.

Proof: Denote by $L(A_\lambda,x)$ the best L_2 approximation to $f(x)$ on X (also on $X_1\cup\{z\}$) with weights w_i^λ.
Set $e_i^\lambda=(f(x_i)-L(A_\lambda,x_i))$ and denote the corresponding σ value by

$$[\sigma(\lambda)]^2=\sum w_i^\lambda|e_i^\lambda|^2 .$$

Now $$[\sigma(\lambda)]^2=\lambda|e_i^\lambda(z)|^2+(1-\lambda)\sum_{X_1} w_i^{(0)}|e_i^\lambda|^2 .$$

For λ sufficiently small, say $0<\lambda\leq\lambda_0<1$, we have that $L(A_\lambda,x)$ and $L(A_0,x)$ are arbitrarily close, and hence $|e^\lambda(z)|>\sigma^*$. Furthermore, we have

$$\sum_{X_1} w_i^{(0)}|e_i^\lambda|^2\geq\sum_{X_1} w_i^{(0)}|e_i^{(0)}|^2$$

since $L(A_0,x)$ minimizes $\sum_{X_1} w_i^{(0)}|e_i^{(0)}|^2$ among all $L(A_k,x)$.

After manipulating

$$[\sigma(\lambda)]^2\geq(1-\lambda)\sum_{X_1} w_i^{(0)}|e_i^{(0)}|+\lambda|e_i^\lambda(z)|^2$$

$$>(1-\lambda)(\sigma^*)^2+\lambda(\sigma^*)^2$$

$$=(\sigma^*)^2$$

Thus $\sigma(\lambda)>\sigma^*$. For any choice of λ in the range $(0,\lambda_0]$ we have $\sigma^1=\sigma(\lambda)$ and hence $\sigma^1>\sigma^*$.

Thus the second start of Lawson's algorithm yields another approximation, a corresponding σ_1^*, and W_1 where $\sigma_1^*>\sigma^*$. Since X is finite, there are only a finite number of restarts possible and we'll converge eventually to $L(A_0,x)$ on X.

IV. COMPUTATIONAL EXPERIENCE

A. Linear Algorithms

In this section we'll report the results of numerical experience with the various linear algorithms which were previously discussed. We will always be trying to find the best Tchebycheff approximation to a set of discrete data.

1. The Problems to be Solved

(a) Find the best approximating function of the form $F=a_0+a_1x$ to the function defined by the following table.

x	0	1	2	3	4	5
f(x)	1.520	1.025	0.475	0.010	-0.475	-1.005

This problem is taken from Barrodale and Young [10].

(b) Find the best approximating function of the form $F=\sum_{i=0}^{3} a_i x^i$ to $\sqrt{x}$ by sixteen points equally spaced in the interval [0,3].

(c) Find the best approximating function of the form $F=\sum_{i=0}^{5} a_i x^i$ to the function y=tan x by 51 equally spaced points in the interval $[0,\pi/4]$.

(d) Find the best approximating function of the form $F=\sum_{i=0}^{4} a_i x^i$ to x^5 on 129 equally spaced points in [-1,1]. This problem is taken from Lawson's thesis [11].

(e) Find the best approximating function of the form $F=\sum_{i=0}^{5} a_i x^i$ to the function defined at 101 points in the interval $[-\pi,\pi]$ in the following way. The basic function used to generate y_i values was $y_i=\sin x_i$; however, if a value of y_i was created such that $|y_i|\geq 0.70$, then y_i was set equal to 0.70. This function will subsequently be referred to as the "clipped sine" problem for obvious reasons. This non-smooth function was purposely designed as a function that might give the Lawson algorithm and other algorithms a real test.

(f) Find the best approximating function of the form $F=\sum_{i=0}^{2} a_i x^i$ to $\sqrt[3]{x}$ by 31 equally spaced points in the interval $[0,3]$. This problem was selected because it was one which gave the acceleration scheme of Rice and Usow some troubles.

2. Numerical Results

All of the algorithms were run on an IBM 360/50 using single precision arithmetic. Specific routines that were used in various algorithms are discussed in Appendix A. The details of how the operation counts were computed are given in Appendix B. The following notation is used to denote the errors for the various methods:

E_L: the Lawson error

E_X: the Exchange error

E_{LP}: the Linear-programming error

E_{LA}: the Lawson error (as algorithm was accelerated by Rice and Usow)

E_{LM}: the Lawson error (as algorithm was accelerated by the author)

The various error entries are reported to six significant digits.

The following abbreviations are used for the algorithms:

LAWS: for the ordinary unaccelerated Lawson algorithm

EXCH: for the Exchange algorithm

LP: for the linear programming method

LAWRU: for the Lawson algorithm as accelerated by Rice and Usow

LAWM: for the Lawson algorithm as accelerated by the author

The errors are only reported at the so-called "critical points" since it is the errors at these points which characterize the solution. The relative position of the critical points will also be given. For example, in problem (a) there are six data pairs and the critical points occur at the second, third and fifth points. Hence, 2, 3, and 5 will be listed as critical points.

The weights for the various Lawson algorithms are also reported only at critical points. A notation for the weights is used which corresponds to the notation used for errors. For example, W_L represents a weight for the Lawson algorithm. CP is a shortcut for critical point.

Table 4.1

Minimax Errors and Iteration Counts

Problem	(a)		(b)		(c)	
Method	Error	Iterations	Error	Iterations	Error	Iterations
LAWS	0.0249813	20	0.0744499	39	0.460E-4	40
EXCH	0.0250000	2	0.0745029	4	0.461E-4	4
LP	0.0249999	5	0.0745028	8	0.461E-4	14
LAWRU	0.0250004	9	0.0745029	12	0.454E-4	9
LAWM	0.0250000	7	0.0745029	7	0.461E-4	9

Problem	(d)		(e)		(f)	
LAWS	0.0619006	40	0.672281	40	0.238005	39
EXCH	0.0624848	3	0.678718	4	0.23803 4	3
LP	0.0624849	12	0.678709	9	0.23803 5	7
LAWRU	0.0621998	18	0.678670	30	0.238027	9
LAWM	0.0624846	8	0.671473	7	0.23803 4	7

Table 4.2

Errors and Weights

Problem (a)

CP	E_L	E_X	E_{LP}	E_{LA}	E_{LM}	W_L	W_{LA}	W_{LM}
2	0.0250609	0.0250002	0.0249998	0.0249996	0.0249996	0.327849	0.333349	0.333348
3	-0.0249795	-0.0249998	-0.0249998	-0.0250010	-0.0250005	0.499987	0.499980	0.499978
5	0.0249390	0.0249991	0.0249998	0.0250000	0.0249991	0.168046	0.166670	0.166674

Problem (b)

1	-0.0744536	-0.0745029	-0.0745042	-0.0745029	-0.0745034	0.279791	0.280000	0.279999
2	0.0744568	0.0745035	0.0745028	0.0745030	0.0745035	0.408730	0.409092	0.409090
6	-0.0746031	-0.0745010	-0.0745028	-0.0745029	-0.0745020	0.177046	0.179996	0.179998
13	0.0746689	0.0745039	0.0745028	0.0745029	0.0745029	0.085734	0.090913	0.090913
16	-0.0744896	-0.0745010	-0.0745023	-0.0745020	-0.0744991	0.039414	0.039999	0.040000

Table 4.2 (continued)

Problem (c)

CP	E_L	E_X	E_{LP}	E_{LA}	E_{LM}	W_L	W_{LA}	W_{LM}
1	0.459E-4	0.461E-4	0.461E-4	0.452E-4	0.461E-4	0.058803	0.064390	0.055079
5	-0.463E-4	-0.460E-4	-0.461E-4	-0.460E-4	-0.458E-4	0.107677	0.078045	0.120701
15	0.462E-4	0.467E-4	0.461E-4	0.466E-4	0.460E-4	0.074483	0.047162	0.147357
27	-0.466E-4	-0.455E-4	-0.461E-4	-0.460E-4	-0.463E-4	0.077801	0.052604	0.159976
40	0.464E-4	0.466E-4	0.461E-4	0.474E-4	0.460E-4	0.110434	0.059756	0.190483
48	-0.459E-4	-0.449E-4	-0.461E-4	-0.451E-4	-0.463E-4	0.206735	0.200263	0.212061
51	0.461E-4	0.471E-4	0.461E-4	0.455E-4	0.460E-4	0.110185	0.106599	0.114345

Problem (d)

1	-0.0619283	-0.0624857	-0.0624847	-0.0621868	-0.0624828	0.0988396	0.101174	0.101885
13	0.0626341	0.0624841	0.0624849	0.0625322	0.0624849	0.0660253	0.073743	0.201145
45	-0.0626195	-0.0624847	-0.0624847	-0.0625898	-0.0624845	0.0422826	0.044292	0.196971
85	0.0626196	0.0624848	0.0624849	0.0625898	0.0624844	0.0422825	0.044292	0.196971
117	-0.0626340	-0.0624841	-0.0624848	-0.0625322	-0.0624855	0.0660254	0.073744	0.201145
129	0.0619283	0.0624857	0.0624849	0.0626868	0.0624848	0.0988398	0.101176	0.101883

Table 4.2 (continued)

Problem (e)

CP	E_L	E_X	E_{LP}	E_{LA}	E_{LM}	W_L	W_{LA}	W_{LM}
1	0.665329	0.678716	0.678709	0.678688	0.671463	0.009023	0.003793	0.015002
13	-0.669158	-0.678721	-0.678700	-0.678685	-0.671474	0.174559	0.013408	0.466792
38	0.682889	0.678717	0.678709	0.678670	0.686733	0.231004	0.474621	0.000000
39	-0.675324	-0.678717	-0.678734	-0.678669	-0.671471	0.313563	0.480086	0.026879
63	0.673019	0.678710	0.678709	0.680201	0.671438	0.009948	0.011461	0.014263
90	-0.673857	-0.678721	-0.678716	-0.681405	-0.671554	0.001324	0.001651	0.000000
101	0.666252	0.678715	0.678709	0.678676	0.671388	0.002602	0.002249	0.001878
				Problem (f)				
2	0.238019	0.238034	0.238035	0.238039	0.238034	0.039500	0.039493	0.039683
10	-0.238008	-0.238034	-0.238032	-0.238028	-0.238034	0.492968	0.493001	0.493196
11	0.238046	0.238034	0.238035	0.238026	0.238034	0.460004	0.460508	0.460319
31	-0.239172	-0.238030	-0.238061	-0.238943	-0.238035	0.005027	0.004694	0.006803

Table 4.3

Operation Counts

Problem	(a)			(b)			(c)		
Method	Adds	Mults	Compares	Adds	Mults	Compares	Adds	Mults	Compares
LAWS	2,140	2,020	--	22,230	21,918	---	120,840	123,120	---
EXCH	106	88	24	688	677	128	3,318	3,306	408
LP	560	560	--	2,688	2,688	---	15,008	15,008	---
LAWRU	963	909	36	6,840	6,744	128	27,189	27,702	306
LAWM	749	707	24	3,990	3,934	64	27,189	27,702	408

Problem	(d)			(e)			(f)		
LAWS	221,720	234,200	---	232,840	241,120	---	26,520	28,626	---
EXCH	5,455	5,422	774	6,018	6,006	808	670	615	186
LP	24,024	24,024	---	16,848	16,848	---	2,870	2,870	---
LAWRU	99,774	105,390	1,548	174,630	180,840	2,020	6,120	6,606	186
LAWM	44,344	46,840	774	40,747	42,196	404	4,760	5,138	124

Table 4.4

Coefficients

Problem	(a)		(b)			
Method	a_0	a_1	a_0	a_1	a_2	a_3
LAWS	1.499900	-0.499959	0.074454	1.643110	-0.786808	0.143854
EXCH	1.499990	-0.499999	0.074503	1.642520	-0.786253	0.143732
LP	1.499990	-0.499999	0.074501	1.642520	-0.786253	0.143732
LAWRU	1.500001	-0.500000	0.074503	1.642522	-0.786253	0.143732
LAWM	1.500000	-0.499999	0.074503	1.642516	-0.786247	0.143730

(c)

Method	a_0	a_1	a_2	a_3	a_4	a_5
LAWS	-0.000046	1.003820	-0.050708	0.572651	-0.477312	0.492112
EXCH	-0.000046	1.003820	-0.050673	0.572453	-0.476946	0.491895
LP	-0.000046	1.003810	-0.050616	0.572270	-0.476610	0.491778
LAWRU	-0.000045	1.003800	-0.050570	0.572338	-0.477047	0.492055
LAWM	-0.000046	1.003810	-0.050576	0.572136	-0.476531	0.491706

Table 4.4 (continued)

Problem	(d)				
Method	a_0	a_1	a_2	a_3	a_4
LAWS	0.000000	-0.313023	-0.000000	1.251096	0.000000
EXCH	0.000000	-0.312485	-0.000000	1.250000	0.000000
LP	0.000000	-0.312483	0.000000	1.249998	-0.000000
LAWRU	0.000000	-0.312890	-0.000000	1.250704	0.000000
LAWM	0.000000	-0.312484	0.000000	1.250000	0.000000

(e)

Method	a_0	a_1	a_2	a_3	a_4	a_5
LAWS	-0.228910	-0.058617	0.432129	0.134740	-0.048269	-0.013052
EXCH	-0.233274	-0.065464	0.438184	0.136231	-0.048970	-0.013131
LP	-0.233274	-0.065481	0.438170	0.136235	-0.048968	-0.013131
LAWRU	-0.234518	-0.066737	0.439060	0.136814	-0.049046	-0.013177
LAWM	-0.230778	-0.054127	0.433519	0.133111	-0.048448	-0.012931

Table 4.4 (continued)

Problem		(f)	
Method	a_0	a_1	a_2
LAWS	-0.123955	1.027966	-0.142058
EXCH	-0.123994	1.028227	-0.142267
LP	-0.123994	1.028215	-0.142261
LAWRU	-0.123985	1.028075	-0.142116
LAWM	-0.123994	1.028228	-0.142267

3. The Unconstrained Least-Squares Procedure

The results for this method cannot be compared directly with the other methods. This is simply because the primary purpose of this method is to locate the critical points of the error curve and, once this had been done, to relay this information to the Lawson algorithm. If the Lawson algorithm has this data it can give all the weight to these critical points and converge immediately.

This algorithm was designed with the expressed purpose of trying to do better than LAWM on problems which were not so smooth and well-behaved. Some measure of success was attained as will now be illustrated.

The LAWM procedure had much difficulty with problem (e), the "clipped sine" problem, because the error peaks would not settle down. However the unconstrained technique was able to find the peaks or critical points of the error curve in thirteen iterations. This procedure would seem to be better than the Lawson algorithm on some problems; however, it certainly cannot compete with the Exchange algorithm. This unconstrained method takes about as long on a smooth problem as on a nonsmooth one. For example, it located the critical points of the error curve correctly for problem (b) but it took 15 iterations. The inherent difficulty with the method is that it may, indeed, converge to the wrong set of critical points and hence it suffers the same fate that plagues both accelerated versions of Lawson's algorithm.

4. Summary of Numerical Results

The ordinary Lawson algorithm consistently gave much poorer results than the other methods. It can be considered out of the running on slowness of convergence alone. This algorithm simply cannot compete with the others unless it is accelerated in some manner. This is obvious if we compare it with EXCH on any problem. Forty iterations were set as a maximum number of iterations for LAWS. In almost every case it failed to converge before reaching this cut-off point.

LAWRU is certainly an improvement over LAWS but it can also yield unsatisfactory answers. In particular on problems (c), (d) and (e) it gave minimax errors which were not very accurate. This algorithm apparently cannot locate non-critical points very accurately. For example, on problem (c) there were still many non-zero weights at non-critical points when LAWRU had converged. For all of the problems an acceleration parameter of $\ell=3$ was used. It was discovered, while experimenting with ℓ on problem (f), that an $\ell=2$ gave very bad results. In fact, the weight vanished at one critical point. This special problem will be handled in Appendix C. It was because of results like this that another acceleration scheme was attempted.

The LAWM algorithm generally performed much better than LAWRU. In most cases it converged faster and took far less arithmetic. It performed well on all problems except for problem (e), the "clipped sine" one. LAWM "thought" it had converged in seven iterations whereas it had selected the wrong set of points for the critical point set. On all of the other problems LAWM proved to be satisfactory if we only look at numerical results. It always took much longer than EXCH to reach convergence.

The LP method was programmed using the revised simplex algorithm and the base set of polynomials used was the set of Tchebycheff polynomials. This method performed competitively on all problems, and indeed, often gave the best error results. Although it is not nearly as efficient as EXCH, it can yield better answers because of the use of orthogonal polynomials. It should be pointed out that using the set $\{1,x,\ldots,x^n\}$ as the base set of polynomials can lead to disastrous results.

EXCH was consistently the best method on all types of problems. This procedure was not only the fastest but also gave good error results. In addition, it is far and away the most efficient algorithm from a computational point of view. Its nearest competitor, LP, takes anywhere from three to five times as much arithmetic to solve the same problem.

B. Nonlinear Algorithms

The three nonlinear algorithms discussed in Chapter II were programmed and the results of the numerical experiments will now be reported. The methods were tried on five different types of nonlinear problems.

1. The Problems to be Solved

(a) Find the best approximating function of the form $F=\frac{a_0}{a_1+a_2x}$ to the Gamma function using 21 points uniformly spaced in the interval [2,3]. This problem is taken from Rice [20].

(b) Find the best approximating function of the form $F=a_0e^{a_1x}\cos(a_2x+a_3)$ to the discrete function defined by a table of values. One hundred one values were selected in the interval $[0,5\pi/2]$ and a corresponding set of y values were generated. The exact way the y values were obtained is contained in Appendix D.

(c) Find the best approximating function of the form $F=a_0x^{a_1}$ to the discrete function defined by the following table.

x	0.10000	0.20000	0.30000	0.40000	0.50000
y	0.00008	0.00150	0.00800	0.02500	0.06200

0.60000	0.70000	0.80000	0.90000	1.00000
0.13000	0.24000	0.40000	0.65000	0.73000

This problem was chosen by the author to illustrate how the algorithms might perform using another type of non-rational approximating function.

(d) Find the best approximating function of the form $F=\sum_{i=1}^{3} a_{i,1}e^{a_{i,2}x}\sin(a_{i,3}x+a_{i,4})$ to the discrete function defined by a table of values. One hundred twenty one values were selected in the interval $[0,4\pi]$ and a corresponding set of y values were generated. The exact way that the y values were obtained is contained in Appendix D.

(e) Find the best approximating function of the form $F=a_1e^{a_2x}+a_3e^{a_4x}$ to the function $y=x^2+4$ using 51 points uniformly spaced in the interval $[-1,1]$. This problem was selected because the approximating function is known to be a varisolvent one.

2. Numerical Results

These algorithms were run using the same hardware and precision as were used for the linear procedures with the following exception. Parts of the revised simplex were done in double precision. Specific routines which were used will be discussed in Appendix E. Details of operation counts will be given in Appendix F.

The following notation will be used for errors:

E_L: the Lawson error

E_{LP}: the linear programming error (as the algorithm was devised by Osborne and Watson)

E_{LPM}: the linear programming error (as the algorithm was modified by the author)

The following abbreviations will be used for the algorithms:

LAWNON: for the Lawson nonlinear algorithm

LPOW: for the linear programming method of Osborne and Watson

LPMA: for the linear programming method as modified by the author

Table 4.5

Nonlinear Minimax Errors and Iteration Counts

Problems	(a)		(b)		(c)	
Method	Error	Iterations	Error	Iterations	Error	Iterations
LAWNON	0.007276	31	0.036450	29	0.061586	38
LPOW	0.007457	11	0.037696	120	0.060530	19
LPMA	0.007457	7	0.037696	43	0.060530	13

Problem	(d)		(e)	
LAWNON	———	——	0.004743	28
LPOW	0.999844E-4	406	0.004914	42
LPMA	0.999556E-4	266	0.004914	13

Table 4.6

Nonlinear Errors

Problem		(a)				(b)	
CP	E_L	E_{LP}	E_{LPM}	CP	E_L	E_{LP}	E_{LPM}
7	-0.006624	-0.007458	-0.007458	1	-0.006643	-0.037698	-0.037698
17	0.007978	0.007455	0.007456	3	0.004004	0.037691	0.037693
21	-0.007377	-0.007458	-0.007458	9	-0.004512	-0.037697	-0.037696
				18	-0.004224	0.037696	0.037697
				50	-0.038387	-0.037696	-0.037696
Problem		(c)				(e)	
6	-0.048075	-0.060530	-0.060530	1	-0.005278	-0.004914	-0.004914
9	0.069033	0.060530	0.060530	8	0.004903	0.004915	0.004915
10	-0.059946	-0.060530	-0.060530	26	-0.004579	-0.004913	-0.004913
				44	0.004901	0.004917	0.004916
				51	-0.005278	-0.004912	-0.004912

Table 4.7

Operation Counts and Function Evaluations

Problem	(a)			(b)		
Method	Adds	Mults	F.E.	Adds	Mults	F.E.
LAWNON	9,665	11,561	2,604	64,452	75,261	14,645
LPOW	3,410	3,410	231	162,720	162,720	7,474
LPMA	2,890	2,506	252	60,324	62,592	3,535
Problem	(c)			(e)		
LAWNON	3,214	4,254	1,140	31,814	35,827	7,140
LPOW	2,736	2,736	230	31,752	31,752	1,326
LPMA	2,502	3,237	230	11,340	13,041	1,326

Table 4.8

Starting Values of Coefficients

Problem	a_0	a_1	a_2	a_3	a_4	a_5
(a)	0.69570	1.40785	-0.35400			
(b)	4.80000	-1.50000	3.40000	1.30000		
(c)	0.60000	5.00000				
(e)	1.00000	0.50000	1.00000	-0.50000		
(d)	3.20000	-0.90000	2.10000	1.70000	3.85000	-2.10000
	a_6	a_7	a_8	a_9	a_{10}	a_{11}
	1.10000	3.30000	1.85000	-1.10000	3.20000	-0.95000

Table 4.9

Nonlinear Coefficients

Problem	(a)			
Method	a_0	a_1	a_2	a_3
LAWNON	0.699025	1.409073	-0.353615	
LPOW	0.700942	1.411168	-0.354000	
LPMA	0.700942	1.411168	-0.354000	
	(b)			
LAWNON	5.06954	-2.02469	2.98710	1.56948
LPOW	5.40648	-2.17781	2.95882	1.56382
LPMA	5.40649	-2.17782	2.95882	1.56382

Problem	(c)		(e)			
Method	a_0	a_1	a_0	a_1	a_2	a_3
LAWNON	0.78995	2.91637	2.002291	0.692998	2.002289	-0.692999
LPOW	0.79053	2.78548	2.002450	0.692740	2.002464	-0.692736
LPMA	0.79053	2.78548	2.002460	0.692737	2.002455	-0.692739

Table 4.9 (continued)

Problem	(d)					
Method	a_0	a_1	a_2	a_3	a_4	a_5
LPOW	3.084579	-1.007960	2.006388	1.494844	3.985788	-2.025707
LPMA	2.999864	-0.999986	2.000152	1.498976	4.034380	-2.005369
	a_6	a_7	a_8	a_9	a_{10}	a_{11}
LPOW	0.970969	3.021482	1.976996	-1.208012	3.011764	-1.030861
LPMA	0.995009	3.002073	1.997907	-1.199813	2.999616	-0.998126

3. Summary of Numerical Results

Before summarizing the performance of the nonlinear algorithms a few comments are in order with regard to problem (d) and some missing entries in Tables 4.5, 4.6 and 4.7. This problem appears to be too unwieldy for LAWNON to handle. The Levenberg-Marquardt algorithm ran into numerical difficulties (rank problems) on the third iteration and could not recover. Thus numerical results could not be reported for LAWNON on problem (d). Likewise LPOW experienced difficulties and did not actually converge in 406 iterations but this was the last iteration for which numerical results could be reported. It was still not converging in 1000 plus total simplex iterations. Since it was impossible to determine where the critical points were in problem (d), there are no errors reported at the critical points.

Although LAWNON appears to be converging for most of the problems, it's obvious that this algorithm is plagued by the same difficulties that plagued its counterpart in linear approximation. This procedure converges in too slow a manner to be competitive. There does not appear to be any way to accelerate this algorithm as was done in the linear case because we do not know that all the weights vanish at non-critical points or even how many critical points there are for a problem "a priori". To keep this algorithm from taking too much computer time it was always shut down after approximately thirty iterations. Although the operation counts for LAWNON look favorable, the corresponding coefficient and error results were not very good and would not improve much if we let the algorithm run twice as long.

In general LPOW appears to be a reliable method. However, there do exist problems which can give it difficulties. One should probably use starting values which are very "good" if one is to hope for convergence with this method. It is recommended that starting values from a nonlinear least-squares solution be passed to this method for "good" guesses at the parameters. LPOW gave essentially the same error and coefficient results as LPMA but as the problem got larger the efficiency of LPMA stood out.

LPMA seems to be a worthwhile modification of the method of Osborne and Watson. On small scale problems it is just as good as the original and on larger problems it seems to be much better than the original. In fact it has been demonstrated with problems (b), (d) and (e) that it can cut the work in more than half. This modification was accomplished using a relatively small amount of code which is more than offset by the speed gained in solving a problem. Indeed, one is able to solve a problem like (d) which could not even be handled by the original algorithm.

V. CONCLUSIONS

One purpose of this study was to compare the Lawson algorithm with the more popular methods of linear L_∞ approximation on a discrete set. In this light, the Lawson algorithm was accelerated by the author and another accelerated version was also tested. The resounding conclusion is that the Exchange method is the method of choice. The only real competition was given by the linear programming technique. It should be mentioned that the original Lawson algorithm does have one major advantage over all of the other methods and that is in the ease of programming it. However, this advantage is more than offset by its slowness in converging.

This paper shows that it is possible, through an acceleration procedure, to make the Lawson algorithm somewhat competitive but it is not as reliable as the EXCH or LP methods. Any accelerated version of Lawson will probably run into some kind of difficulty sooner or later. For example, LAWRU had difficulties with some rather simple problems and LAWM ran into troubles on a non-smooth problem. Although the author's acceleration can fail, there are modifications which could be made to the method which would circumvent this failure. It's doubtful whether such modifications would be worthwhile since EXCH would still be unbeatable. In a similar vein, the unconstrained least-squares technique is just too sensitive with respect to certain parameters to be of value in a general setting.

A significant contribution has been made by showing that the Lawson algorithm does generalize to the nonlinear case for certain types of approximating functions. Namely, the LAWNON algorithm has been established for varisolvent functions but it may hold true for other types of approximating functions as well. For example, problem (b) is not known to be of varisolvent type and yet the algorithm appears to be converging. On the other hand, problem (d) is not known to be varisolvent either and the method has failed. Although the numerical results for LAWNON might not be as good as we would hope for, it appears that convergence is taking place for varisolvent types of approximating functions.

Many of the popular nonlinear L_∞ algorithms are of the Remes-type and hence also limit one to varisolvent type functions. A more general approach like that of Osborne and Watson allows one more freedom in the choice of approximating functions. Their method is not dependent on any alternating error property or other "a priori" information. Perhaps the most significant contribution made in this study was the modification of Osborne and Watson's method. Using the author's modification, which is detailed at the end of Chapter III, it appears as if the bigger the nonlinear problem to be solved, the bigger should be the net gain in using the new procedure.

It appears that other modifications of Osborne and Watson's method would be desirable also. In particular, a more elaborate search procedure would be appropriate. When there are many parameters involved in a problem, such as in nonlinear problem (d), the current use of γ as a scalar does not yield the best results. Perhaps a parameter vector could be selected for γ, although then one could get hung up for a long time in the search. The question of "How good must your starting values be to guarantee convergence?" is also an appropriate subject for further study.

BIBLIOGRAPHY

1. Poussin, Ch. de la Vallée. "Sur la méthode de l' approximation minimum," Ann. Soc. Sci., Bruxelles, Vol. 35, 1911, pp. 1-16. English translation by H. E. Salzer.

2. Polya, G. "Sur un algorithm toujours convergent pour obtenir les polynomes de meilleure approximation de Tchebychef pour une fonction continue quelconque," C.R. Acad. Sci., Paris, Vol. 157, 1913, pp. 840-843.

3. Goldstein, A. A.; Levine, N.; and Hereshoff, J. B. "On the Best and Least Qth Approximation of an Overdetermined System of Linear Equations," Journal of the Association for Computing Machinery, Vol. 4, 1957, pp. 341-347.

4. Remes, E. Ya. "Sur un procédé çonvergent d' approximations successives pour determiner les polynomes d' approximation," C.R. Acad. Sci., Paris, Vol. 198, 1934, pp. 2063-2065.

5. Meinardus, Günter. Approximation of Functions: Theory and Numerical Methods. Berlin: Springer-Verlag, 1967.

6. Zuhovickii, S. I. "An Algorithm for the Solution of the Tchebycheff Approximation Problem in the Case of a Finite System of Incompatible Linear Equations," Doklady Akademii Nauk SSSR, Vol. 79, 1951, pp. 561-564.

7. Cheney, E. W. and Goldstein, A. A. "Note on a Paper by Zuhovickii Concerning the Tchebycheff Problem for Linear Equations," Journal of the Society for Industrial and Applied Mathematics, Vol. 6, 1958, pp. 233-239.

8. Cheney, E. W. and Goldstein, A. A. "A Finite Algorithm for the Solution of Consistent Linear Equations and Inequalities and for the Tchebycheff Approximation of Inconsistent Linear Equations," Pacific Journal of Mathematics, Vol. 8, 1958, pp. 415-427.

9. Stiefel, E. L. "Note on Jordan Elimination, Linear Programming and Tchebycheff Approximation," Numerische Mathematik, Vol. 2, 1960, pp. 1-17.

10. Barrodale, I. and Young, A. "Algorithms for Best L_1 and L_∞ Linear Approximations on a Discrete Set," Numerische Mathematik, Vol. 8, 1966, pp. 295-306.

11. Lawson, C. L. "Contributions to the Theory of Linear Least Maximum Approximation," Ph.D. Thesis, U.C.L.A., 1961.

12. Hastings, Cecil. Approximations for Digital Computers. Princeton, New Jersey: Princeton University Press, 1955.

13. Loeb, Henry L. "Algorithms for Chebyshev Approximation Using the Ratio of Linear Forms," Journal of the Society for Industrial and Applied Mathematics, Vol. 8, 1960, pp. 458-465.

14. Maehly, Hans J. "Methods for Fitting Rational Approximations, Part I: Telescoping Procedures for Continued Fractions," Journal of the Association for Computing Machinery, Vol. 7, 1960, pp. 150-162.

15. Osborne, M. R. and Watson, G. A. "An Algorithm for Minimax Approximation in the Nonlinear Case," The Computer Journal, Vol. 12, 1969-70, pp. 63-68.

16. Lee, C. M. and Roberts, F. D. K. "A Comparison of Algorithms for Rational L_∞ Approximation," Mathematics of Computation, Vol. 27, 1973, pp. 111-121.

17. Rice, J. R. and Usow, K. H. "The Lawson Algorithm and Extensions," Mathematics of Computation, Vol. 22, 1968, pp. 118-127.

18. Stiefel, E. L. "Numerical Methods of Tchebycheff Approximation," On Numerical Approximation, ed. by R. E. Langer. Madison, Wisconsin: University of Wisconsin Press, 1959, pp. 217-232.

19. Rice. J. R. The Approximation of Functions, Vol. 1, Linear Theory. Reading, Massachusetts: Addison-Wesley, 1964.

20. Rice, J. R. The Approximation of Functions, Vol. 2, Nonlinear and Multivariate Theory. Reading, Massachusetts: Addison-Wesley, 1968.

21. Motzkin, T. S. and Walsh, J. L. "Polynomials of Best Approximation on an Interval," Proceedings of the National Academy of Science. U.S.A., Vol. 45, 1959, pp. 1523-1528.

22. Rivlin, T. J. *An Introduction to the Approximation of Functions*. Waltham, Massachusetts: Blaisdell Publishing Company, 1969.

23. Levenberg, K. "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Quarterly of Applied Mathematics*, Vol. 2, 1944, pp. 164-168.

24. Marquardt, D. W. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 11, 1963, pp. 431-441.

25. Hadley, G. *Linear Programming*. Reading, Massachusetts: Addison-Wesley, 1962.

26. Bauer, F. L. "Elimination with Weighted Row Combinations for Solving Linear Equations and Least-Squares Problems," *Numerische Mathematik*, Vol. 7, 1965, pp. 338-352.

27. Wagner, H. M. "A Comparison of the Original and Revised Simplex Methods," *Operations Research*, Vol. 5, 1957, pp. 361-369.

VITA

William E. McBride was born on August 23, 1940 in Albert Lea, Minnesota. He attended secondary school at Pacelli High School in Austin, Minnesota, graduating in May of 1958. In September of 1958 he enrolled at St. John's University, Collegeville, Minnesota. He received a Bachelor of Arts degree from St. John's with a major in mathematics. From September of 1962 to May of 1964 he was a graduate student at the University of North Dakota, holding a teaching assistantship in the Department of Mathematics.

Upon completion of his Master of Arts degree with a major in mathematics, he went to work for Control Data Corporation as an associate programmer analyst. He was with Control Data until August of 1966 when he accepted a position as Instructor of Mathematics at Bemidji State College, Bemidji, Minnesota. During the summers of 1968 and 1969 he was granted N.S.F. fellowships to attend a Computer Science Institute at the University of Missouri at Rolla.

In September of 1969 he was granted a leave of absence and returned to Rolla after receiving a Title III Faculty Improvement Grant. He was a graduate teaching assistant at the University of Missouri at Rolla for two years and held an N.S.F. Traineeship during the 1971-72 school year. Since September of 1972 he has been employed as an Assistant Professor of Mathematics at Western Illinois University in Macomb, Illinois.

On December 28, 1963 he was married to the former Jane S. Goihl of Lake City, Minnesota. Their marriage has been blessed with five children: Michael, Patricia, Daniel, Colleen and Erin.

APPENDIX A

Routines Used for Linear Algorithms

1. Exchange Method

The Gaussian elimination method was used to solve the system of equations. To determine starting values the least-squares problem was solved first using Bauer's method which is mentioned in (3) below.

2. Linear Programming Method

The revised simplex procedure was used to solve the linear program. Tchebycheff polynomials were the orthogonal polynomials used. The program APMM in the IBM Scientific Subroutine Package was used as the basic program.

3. The Lawson Algorithms

All three of the Lawson procedures have at their heart the solution of a weighted least-squares problem. In order to solve this problem as accurately as possible Bauer's "Ortholin 2" procedure was used. The details of this algorithm may be found in [26].

APPENDIX B

Operation Counts for Linear Algorithms

1. Exchange Method

Assume there are M data pairs and N is the degree of the approximating polynomial. Using Gaussian elimination to solve a system of (N+2) linear equations in (N+2) unknowns requires the following work:

(a) Number of additions:

$$\frac{(N+2)^3}{3} + \frac{(N+2)^2}{2} - \frac{5(N+2)}{6} \; .$$

(b) Number of multiplications:

$$\frac{(N+2)(N+3)}{2} + \frac{(N+2)^2}{3} + \frac{(N+2)^2}{2} - \frac{5(N+2)}{6} \; .$$

This much effort is needed for each iteration. In addition there are 2M comparisons at each pass in order to locate the current "critical points". There is also the following effort needed to get the starting values via "Ortholin 2":

$$(M+15)(N+1)+\frac{(2M+5)(N)(N+1)}{2} \text{ additions,}$$

$(4M+4)(N+1)+\frac{(2M+3)(N)(N+1)}{2}$ multiplications and 2M comparisons.

2. LP Algorithm

We'll assume that there are N data pairs and M parameters and the revised simplex procedure was used. The

amount of work needed for each revised simplex iteration, according to Wagner [27], is:
$3m^2+mn$ multiplications and $3m^2+mn$ additions, where n is the number of unknowns and m is the number of equations. For our problem n=2N+M+2 and m=M+2.

3. The Ordinary Lawson Algorithm

In the Lawson algorithm proper there are 2N additions and 5N multiplications where N is the number of data pairs. M will represent the number of parameters. Most of the computational effort is expended in the call to "Ortholin 2" which requires $(4N+15)(M)+\frac{(2N+5)(M)(M-1)}{2}$ additions and $(4N+4)(M)+\frac{(2N+3)(M)(M-1)}{2}$ multiplications. Totaling these results the following number of operations are needed per iteration:

$$(4N+15)(M)+\frac{(2N+5)(M)(M-1)}{2}+2N \text{ additions,}$$

$$(4N+4)(M)+\frac{(2N+3)(M)(M+1)}{2}+2N \text{ multiplications.}$$

4. The LAWRU Algorithm

All the computations needed for the ordinary Lawson method are needed here. In addition every third iteration there are 2N compares. The "third" here is based on the fact that the acceleration parameter is set equal to three.

5. The LAWM Algorithm

Everything needed in the ordinary Lawson algorithm is needed here also. Additionally, from the third iteration on there are 2N compares. However, for the last three iterations there are no compares.

APPENDIX C

LAWRU on Problem (f)

This example illustrates how the acceleration of Lawson's algorithm due to Rice and Usow may fail, depending on the choice of the acceleration parameter. In Chapter IV an acceleration parameter of $\ell=3$ was used and the method converged. Here an acceleration parameter of $\ell=2$ is used. The results are given below, with the correct results in parentheses.

Minimax error = 0.2059688 (0.238035)

Coefficients:

$a_0=0.0212516$ (-.123994)

$a_1=0.076780$ (1.128215)

$a_2=0.6960068$ (-.142261)

Weights at critical points:

CP	Weights	
2	0.000569	(.039683)
10	0.412358	(.493196)
11	0.481380	(.460319)
31	0.000000	(.006803)

The method failed because the weight at "critical point" 31 was accidentally set to zero.

APPENDIX D

Generation of Data for Nonlinear Problems

1. For Problem (b)

The data was essentially generated from the function:

$$f(x)=5e^{-2x}\cos(3x+\pi/2)$$

with the following perturbations. If x was greater than 3.925 then 0.001 times x was added to f. If x was less than or equal to 3.925 then 0.01 times x was subtracted from f.

2. For Problem (d)

The data was essentially generated from the function:

$$f(x)=3e^{-x}\sin(2x+1.5)+4e^{-2x}\sin(x+3.0)$$
$$+2e^{-1.2x}\sin(3x-1.0)$$

with the following perturbations. If x was greater than 6.28 then 0.0001 was added to f. If x was less than or equal to 6.28 then 0.0001 was subtracted from f.

APPENDIX E

Routines Used for Nonlinear Algorithms

1. Lawson Nonlinear Algorithm

The heart of this method is the Levenberg-Marquardt algorithm which is used to solve the nonlinear L_2-problem.

2. The LP Method of Osborne and Watson

This method relies on the solution of a linear program. Hence the core of this method is the revised simplex procedure for solving the linear L_∞ problem. This method also uses a search procedure to locate the proper gamma multiplier.

3. The Modified LP Method

This method requires the same routines as the method of Osborne and Watson with one addition. Depending on the outcome of the restart procedure the dual simplex may be called into use.

APPENDIX F

Operation Counts for Nonlinear Algorithms

For all the methods we'll assume there are N data pairs and M parameters.

1. LPOW Method

(a) Each LP iteration takes as much work as one revised simplex iteration. Recall, from Appendix B, that for each iteration there are $3m^2+mn$ multiplications and $3m^2+mn$ additions, where n is the number of unknowns and m is the number of equations. For our problem $n=2N+M+2$ and $m=M+2$.

(b) Each outer iteration takes a certain number of function evaluations (F.E.). There are N F.E. in the main program plus N times the number of passes through the search routine FIND for this outer iteration. Each outer iteration makes a call to DELF (to evaluate the partials). Such an evaluation will be equated with an F.E. There are M times N such evaluations per call to DELF.

2. LPMA Method

There is about the same amount of work done here as in LPOW with the following exceptions.

(a) Let MM=M+3.

Each time we restart with the old basis we need to do:

$$(MM)[2(MM+1)^2+3(MM+1)+1] \text{ multiplications}$$

and

$$(MM[(MM+1)^2+(MM+1)] \text{ additions.}$$

(b) If no infeasible solutions exist after the simplex has converged we continue on as usual. If there are infeasible solutions we enter the dual simplex, which takes the same amount of work as a usual simplex iteration.

3. LAWNON Method

(a) In the main program there are 2N additions and 5N multiplications needed for each outer iteration.

(b) In subprogram MARQ (the Levenberg-Marquardt algorithm) and related subprograms the following amount of work is required per step:

$$\frac{M^3}{3} + (N+5)(M^2) + \frac{8M}{3} + (M+2)(N) \text{ multiplications;}$$

$$\frac{M^3}{3} + (N+\frac{9}{2})(M^2) + \frac{7M}{6} + (M)(N) \text{ additions;}$$

(M+1)(N) F.E. (this includes derivative evaluations).

243105