SIAM J. NUMER. ANAL.
Vol. 16, No. 1, February 1979

AN ALGORITHM FOR l_1-NORM MINIMIZATION WITH APPLICATION TO NONLINEAR l_1-APPROXIMATION*

R. A. EL-ATTAR,† M. VIDYASAGAR† AND S. R. K. DUTTA‡

Abstract. Necessary and sufficient conditions for minimizing an l_1-norm type of objective function are derived using the nonlinear programming (NLP) approach. Broader sufficient conditions are made possible by using directional derivatives. It is shown that an algorithm previously proposed by Osborne and Watson (1971) for nonlinear l_1-approximation falls under a prototype steepest descent algorithm. The l_1-problem is converted to a sequence of problems, each of which involves the minimization of a continuously differentiable function. Based on this conversion and on the optimality conditions obtained, an algorithm that solves the l_1-minimization problem is proposed. An extrapolation technique due to Fiacco and McCormick (1966) and (1968, p. 188) is used to accelerate the convergence of the algorithm and to improve its numerical stability. To illustrate some of the theoretical ideas and to give numerical evidence, several examples are solved. The algorithm is then used to solve some nonlinear l_1-approximation problems. A comparison between the Osborne–Watson algorithm and the proposed one is also presented.

1. Introduction. In this paper, we consider the problem of minimizing an l_1-norm type of objective function. This problem is stated as:

Problem I. Given continuously differentiable functions $f_i: R^n \to R$, $i = 1, \cdots, m$, minimize

$$F(\mathbf{x}) = \|\mathbf{f}(\mathbf{x})\| = \sum_{i=1}^{m} |f_i(\mathbf{x})|, \tag{1.1}$$

where $\|\cdot\|$ denotes the l_1-norm on R^n.

This type of problem arises in a variety of areas [4], [5], [6] ranging from digital communication to approximation techniques. It is well known that best l_1-approximations are often superior to best l_p-approximations with $p > 1$, when dealing with raw data that contain wild points [5], [7]. We focus attention here on the problem of nonlinear l_1-approximation, which can be stated as follows.

Problem II. Given a real-valued function $f(x)$ defined on a discrete set $X = \{x_1, \cdots, x_m\}$, define an approximating function $K(A, x)$ for any set $A = \{a_1, \cdots, a_n\}$ of real numbers. Find A^* which minimizes

$$F(A) = \sum_{i=1}^{m} |f(x_i) - K(A, x_i)|. \tag{1.2}$$

If $K(A, \cdot)$ is a linear function in A, the situation is then a linear l_1-approximation problem, which is solved satisfactorily by Barrodale and Roberts [8]. So, our concern is only with $K(A, \cdot)$ being a nonlinear function in A. This problem has not yet been solved efficiently. It is obvious, however, that Problem II is a special case of Problem I.

In § 2, we present some necessary and sufficient conditions for optimality for Problem I. First, we transform Problem I into a nonlinear programming (NLP) problem, and thus derive the conditions for optimality. Next, we derive essentially equivalent conditions using a directional derivative approach.

In § 3, we show that a previously proposed algorithm for nonlinear l_1-approximation due to Osborne and Watson [1] is a steepest descent type technique.

Examining Problem I, we see that although the functions $f_i(\cdot)$ are continuously differentiable, $F(\cdot)$ is not necessarily differentiable at all points. Therefore, the

* Received by the editors February 1, 1977, and in revised form September 20, 1977. This research was supported by the National Research Council of Canada, under Grant No. A-7790 and by Alcan Smelters and Chemicals, Limited, Montreal, Canada.

† Department of Electrical Engineering, Concordia University, Montreal, Canada H3G 1M8.

‡ Alcan Smelters and Chemicals, Limited, Montreal Canada.

existing powerful gradient methods cannot be used directly to solve Problem I. In § 4, we convert Problem I to a sequence of problems, each involving the minimization of a continuously differentiable function; we show that, under suitable conditions, the sequence of solutions thus generated converges to a solution of Problem I.

In § 5, we propose an algorithm for l_1-norm minimization based on the conversion presented in § 4. The convergence of the algorithm is accelerated by employing an extrapolation technique proposed by Fiacco and McCormick [2], [3, p. 188]. Several numerical examples are solved using the proposed algorithm to illustrate some of the theoretical ideas.

In § 6, the algorithm is used to solve some nonlinear l_1-approximation problems. Finally, § 7 contains the conclusion.

2. Necessary and sufficient conditions for optimality.

2.1. Optimality conditions via an NLP approach. Problem I can be transformed to the following equivalent nonlinear programming (NLP) problem:

Problem III. Minimize $g: R^{n+m} \to R$ where

$$g = \sum_{i=1}^{m} \phi_i, \tag{2.1}$$

subject to

$$-\phi_i + f_i(\mathbf{x}) \leqq 0 \quad \forall i, \tag{2.2}$$

$$-\phi_i - f_i(\mathbf{x}) \leqq 0 \quad \forall i, \tag{2.3}$$

where $f_i(\cdot)$ are as given in Problem I.

The gradient of the objective function is

$$\nabla g = \left[\frac{\mathbf{u}}{\mathbf{0}}\right], \tag{2.4}$$

where $\mathbf{u} = [1 \quad 1 \cdots 1]^T$ is an m-vector representing the gradient with respect to ϕ_i, $i = 1, \cdots, m$, T denotes transposition, and $\mathbf{0}$ is the n-vector with zero elements representing the gradient with respect to $\mathbf{x}$.

We show now that this problem satisfies the constraint qualification [9, p. 7] and then apply the Kuhn–Tucker conditions to obtain a necessary condition for optimality. Suppose $(\boldsymbol{\phi}^*, \mathbf{x}^*)$ is a solution for Problem III; then clearly $\phi_i^* = |f_i(\mathbf{x}^*)|$, $\forall i$. Define the sets

$$I(\mathbf{x}^*) = \{i : f_i(\mathbf{x}^*) > 0\}, \tag{2.5}$$

$$J(\mathbf{x}^*) = \{i : f_i(\mathbf{x}^*) < 0\}, \tag{2.6}$$

$$K(\mathbf{x}^*) = \{i : f_i(\mathbf{x}^*) = 0\}. \tag{2.7}$$

Then the gradients of the active constraints are given by

$$\left[\frac{-\mathbf{e}_i}{\nabla f_i(\mathbf{x}^*)}\right] \quad \forall i \in I(\mathbf{x}^*), \tag{2.8a}$$

$$\left[\frac{-\mathbf{e}_i}{-\nabla f_i(\mathbf{x}^*)}\right] \quad \forall i \in J(\mathbf{x}^*), \tag{2.8b}$$

and

$$\left[\frac{-\mathbf{e}_i}{\nabla f_i(\mathbf{x}^*)}\right], \quad \left[\frac{-\mathbf{e}_i}{-\nabla f_i(\mathbf{x}^*)}\right] \quad \forall i \in K(\mathbf{x}^*), \tag{2.8c}$$

where $\mathbf{e}_i$ is an m-vector with a 1 in the ith position and zeros elsewhere. Pick an $(m+n)$-vector $\mathbf{h} = [-\mathbf{u}|\mathbf{0}]^T$; note that $\mathbf{h}$ makes an acute angle with each of the gradient vectors of the active constraints, so that the constraint qualification [9, p. 7] is satisfied; then, by the Kuhn–Tucker conditions, we get the following necessary condition for optimality

$$\begin{aligned}(2.9)\qquad &\left[\frac{\mathbf{u}}{\mathbf{0}}\right] + \sum_{i\in I(\mathbf{x}^*)} \lambda_i\left[\frac{-\mathbf{e}_i}{\nabla f_i(\mathbf{x}^*)}\right] + \sum_{i\in J(\mathbf{x}^*)} \lambda_i\left[\frac{-\mathbf{e}_i}{-\nabla f_i(\mathbf{x}^*)}\right] \\ &+ \sum_{i\in K(\mathbf{x}^*)} \left\{\lambda_i\left[\frac{-\mathbf{e}_i}{\nabla f_i(\mathbf{x}^*)}\right] + \gamma_i\left[\frac{-\mathbf{e}_i}{-\nabla f_i(\mathbf{x}^*)}\right]\right\} = \left[\frac{\mathbf{0}}{\mathbf{0}}\right],\end{aligned}$$

where $\lambda_i \geqq 0$ and $\gamma_i \geqq 0$ are the corresponding Lagrange multipliers. Splitting equation (2.9), we see the necessary conditions for an l_1-norm optimum are

$$(2.9\text{a})\qquad \sum_{i\in I(\mathbf{x}^*)\cup J(\mathbf{x}^*)} \lambda_i \cdot \nabla f_i(\mathbf{x}^*) + \sum_{i\in K(\mathbf{x}^*)} (\lambda_i - \gamma_i)\cdot \nabla f_i(\mathbf{x}^*) = \mathbf{0},$$

$$(2.9\text{b})\qquad \lambda_i = 1, \qquad i \in I(\mathbf{x}^*),$$

$$(2.9\text{c})\qquad \lambda_i = 1, \qquad i \in J(\mathbf{x}^*),$$

$$(2.9\text{d})\qquad \lambda_i + \gamma_i = 1, \qquad \lambda_i, \gamma_i \geqq 0, \quad i \in K(\mathbf{x}^*).$$

This result is summarized in the following lemma.

LEMMA 2.1. *A necessary condition for* $\mathbf{x}^*$ *to solve Problem* I *is that there exist constants* $\alpha_i \in [-1, 1]\ \forall i \in K(\mathbf{x}^*)$ *such that*

$$(2.10)\qquad \sum_{i\notin K(\mathbf{x}^*)} [\operatorname{sign} f_i(\mathbf{x}^*)]\cdot \nabla f_i(\mathbf{x}^*) + \sum_{i\in K(\mathbf{x}^*)} \alpha_i \cdot \nabla f_i(\mathbf{x}^*) = \mathbf{0}.$$

Note that if the functions f_i, $i = 1, 2, \cdots, m$ are linear, then the constraints (2.2), (2.3) are all convex, so that (2.10) is also sufficient for $\mathbf{x}^*$ to be a minimum.

2.2. Directional derivative approach.

DEFINITION 2.1. Suppose $f: R^n \to R$, and let $\mathbf{x}, \mathbf{h} \in R^n$; then the quantity

$$(2.11)\qquad Df(\mathbf{x}; \mathbf{h}) = \lim_{\beta\to 0^+} \frac{f(\mathbf{x}+\beta\mathbf{h}) - f(\mathbf{x})}{\beta},$$

if it exists, is known as the *directional derivative* of f at $\mathbf{x}$ in the direction of $\mathbf{h}$.

Although the function $F(\cdot)$ defined in (1.1) may not be differentiable at all $\mathbf{x} \in R^n$, it is still directionally differentiable at all points and in each direction.

LEMMA 2.2. *Let* $F(\mathbf{x})$ *be as in* (1.1); *then* $F(\mathbf{x})$ *is directionally differentiable at each point* $\mathbf{x} \in R^n$ *in each direction* $\mathbf{h} \in R^n$; *in particular,*

$$(2.12)\qquad DF(\mathbf{x}; \mathbf{h}) = \sum_{i\notin K(\mathbf{x})} \mathbf{h}^T \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i\in K(\mathbf{x})} |\mathbf{h}^T \nabla f_i(\mathbf{x})|,$$

where

$$(2.13)\qquad K(\mathbf{x}) = \{i : f_i(\mathbf{x}) = 0\}.$$

Proof. For sufficiently small β, expand $f_i(\mathbf{x}+\beta\mathbf{h})$ using the Taylor series around $\mathbf{x} \in R^n$, to obtain

$$(2.14)\qquad f_i(\mathbf{x}+\beta\mathbf{h}) = f_i(\mathbf{x}) + \beta\mathbf{h}^T \nabla f_i(\mathbf{x}) + o(\beta),$$

where $o(\beta)$ denotes a higher-order term in β, namely,

$$(2.15)\qquad o(\beta)/\beta \to 0 \quad \text{as} \quad \beta \to 0^+.$$

Then for $i \in K(\mathbf{x})$ we have

$$\lim_{\beta \to 0^+} \frac{|f_i(\mathbf{x}+\beta \mathbf{h})| - |f_i(\mathbf{x})|}{\beta} = |\mathbf{h}^T \nabla f_i(\mathbf{x})|, \tag{2.16}$$

and for $i \notin K(\mathbf{x})$,

$$|f_i(\mathbf{x}+\beta \mathbf{h})| = |f_i(\mathbf{x})| \cdot \left|1 + \beta \mathbf{h}^T \cdot \frac{\nabla f_i(\mathbf{x})}{f_i(\mathbf{x})} + \frac{o(\beta)}{f_i(\mathbf{x})}\right|. \tag{2.17}$$

For sufficiently small β, the term $(1+\beta \mathbf{h}^T \cdot [\nabla f_i(\mathbf{x})]/[f_i(\mathbf{x})] + [o(\beta)]/[f_i(\mathbf{x})])$ is always positive, so that

$$|f_i(\mathbf{x}+\beta \mathbf{h})| - |f_i(\mathbf{x})| = \beta \mathbf{h}^T \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + o(\beta), \tag{2.18}$$

and in the limit

$$\lim_{\beta \to 0^+} \frac{|f_i(\mathbf{x}+\beta \mathbf{h})| - |f_i(\mathbf{x})|}{\beta} = \mathbf{h}^T \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})]. \tag{2.19}$$

Combining (2.16) and (2.19) and summing over appropriate sets we obtain

$$DF(\mathbf{x}; \mathbf{h}) = \sum_{i \notin K(\mathbf{x})} \mathbf{h}^T \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i \in K(\mathbf{x})} |\mathbf{h}^T \nabla f_i(\mathbf{x})|, \quad \forall \mathbf{h} \in R^n. \;\Box \tag{2.20}$$

LEMMA 2.3. *A necessary condition for* $\mathbf{x}^* \in R^n$ *to solve Problem* I *is that*

$$DF(\mathbf{x}^*; \mathbf{h}) \geqq 0 \quad \forall \mathbf{h} \in R^n. \tag{2.21}$$

Moreover if the functions $|f_i|$ *are all convex, the latter condition is also sufficient.*

Proof. The proof is obvious.

Remark. The conditions obtained from the nonlinear programming approach are sufficient if the functions f_i are linear, whereas the conditions obtained by Lemma 2.3 allow the functions $|f_i|$ to be convex, so obviously the latter conditions are broader than the former.

In the next lemma we prove that both conditions (2.10) and (2.21) are equivalent; for this we need the following separation theorem.

THEOREM 2.1 [10, p. 49]. *If the set* $S \subset R^n$ *is convex, closed, and nonempty, and* $\mathbf{v} \notin S$, *then there exists a hyperplane* P *separating* $\mathbf{v}$ *and* S; *in other words, there exists a vector* $\mathbf{h} \in R^n$ *and a scalar* $b \in R$ *such that*

$$\mathbf{v}^T \mathbf{h} < b, \qquad \mathbf{u}^T \mathbf{h} \geqq b \quad \forall \mathbf{u} \in S. \tag{2.22}$$

LEMMA 2.4. *Let* $K(\mathbf{x}) = \{i : f_i(\mathbf{x}) = 0\}$; *then the following two statements are equivalent*:

(S1) $\sum_{i \notin K(\mathbf{x})} \mathbf{h}^T \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i \in K(\mathbf{x})} |\mathbf{h}^T \nabla f_i(\mathbf{x})| \geqq 0 \quad \forall \mathbf{h} \in R^n$.

(S2) *There exist constants* $\alpha_i \in [-1, 1]$, $i \in K$ $(\mathbf{x})$ *such that*

$$\sum_{i \notin K(\mathbf{x})} \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i \in K(\mathbf{x})} \alpha_i \nabla f_i(\mathbf{x}) = \mathbf{0}.$$

Proof ((S1) $\Rightarrow$ (S2)). We actually show that if (S2) is false, then (S1) is false. Let

$$\mathbf{v} = \sum_{i \in \notin K(\mathbf{x})} \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})], \tag{2.23}$$

and define the set

$$S = \left\{\mathbf{u} : \mathbf{u} = \sum_{i \in K(\mathbf{x})} \alpha_i \nabla f_i(\mathbf{x}), \alpha_i \in [-1, 1]\right\}. \tag{2.24}$$

Then S is convex, closed, and it is also symmetric, i.e.

$$\mathbf{u} \in S \Rightarrow -\mathbf{u} \in S. \tag{2.25}$$

(S2) says that $\mathbf{v} \in S$. Suppose (S2) is false; then by the separation theorem, there exists $\mathbf{h} \in R^n$, $b \in R$ such that

$$\mathbf{v}^T\mathbf{h} < b, \qquad \mathbf{u}^T\mathbf{h} \geqq b \quad \forall \mathbf{u} \in S. \tag{2.26}$$

In other words

$$\mathbf{v}^T\mathbf{h} < b, \qquad -\mathbf{u}^T\mathbf{h} \leqq -b \quad \forall \mathbf{u} \in S. \tag{2.27}$$

In particular, let

$$\mathbf{u} = \sum_{i \in K(\mathbf{x})} \alpha_i \nabla f_i(\mathbf{x}) \tag{2.28}$$

where

$$\alpha_i = 1 \quad \text{if} \quad \mathbf{h}^T\nabla f_i(\mathbf{x}) \leqq 0, \tag{2.29a}$$

$$\alpha_i = -1 \quad \text{if} \quad \mathbf{h}^T\nabla f_i(\mathbf{x}) > 0; \tag{2.29b}$$

then

$$-\mathbf{u}^T\mathbf{h} = \sum_{i \in K(\mathbf{x})} -\alpha_i \mathbf{h}^T\nabla f_i(\mathbf{x}) = \sum_{i \in K(\mathbf{x})} |\mathbf{h}^T\nabla f_i(\mathbf{x})| \leqq -b. \tag{2.30}$$

Therefore

$$\mathbf{v}^T\mathbf{h} - \mathbf{u}^T\mathbf{h} < b - b = 0. \tag{2.31}$$

This shows that (S1) is false.

((S2) $\Rightarrow$ (S1)). Suppose that (S2) is true. For $\alpha_i \in [-1, 1]$ we have

$$\alpha_i \mathbf{h}^T\nabla f_i(\mathbf{x}) < |\mathbf{h}^T\nabla f_i(\mathbf{x})| \quad \forall \mathbf{h} \in R^n; \tag{2.32}$$

therefore

$$\begin{aligned} 0 &= \mathbf{h}^T\left\{\sum_{i \notin K(\mathbf{x})} \nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i \in K(\mathbf{x})} \alpha_i \nabla f_i(\mathbf{x})\right\} \\ &\leqq \sum_{i \notin K(\mathbf{x})} \mathbf{h}^T\nabla f_i(\mathbf{x})[\operatorname{sign} f_i(\mathbf{x})] + \sum_{i \in K(\mathbf{x})} |\mathbf{h}^T\nabla f_i(\mathbf{x})|. \end{aligned} \tag{2.33}$$

From this (S1) follows directly. □

3. Osborne and Watson algorithm. In this section, we discuss an algorithm for nonlinear l_1-approximation due to Osborne and Watson [1]; we show that this algorithm is of a steepest descent type for which the convergence is ensured but the rate of convergence is ultimately slow. For the sake of clarity, we restate the problem as an l_1-minimization problem and then restate their algorithm in the same notation followed throughout this paper.

Let $\mathbf{f}: R^n \to R^m$ be continuously differentiable. Find $\mathbf{x}^* \in R^n$ to minimize

$$F(\mathbf{x}) = \|\mathbf{f}(\mathbf{x})\|_1 = \sum_{i=1}^{m} |f_i(\mathbf{x})|. \tag{3.1}$$

Let $\mathbf{H}$ be an $m \times n$ matrix with columns ∇f_i. Assume

(A1) $m > n$,

(A2) $f_i(\mathbf{x} + \beta\mathbf{h}) = f_i(\mathbf{x}) + \beta\mathbf{h}^T\nabla f_i(\mathbf{x}) + o(\beta\mathbf{h})$,

(A3) $\mathbf{H}$ has rank n.

Now, we state the algorithm.

Step 1. Pick $\mathbf{x}_0 \in R^n$; set $k=0$.

Step 2. Calculate $\mathbf{h}$ to minimize

$$\sum_{i=1}^{m} |f_i(\mathbf{x}_k)+\mathbf{h}^T\nabla f_i(\mathbf{x}_k)|.$$

Let the minimum be attained at $\mathbf{h}=\mathbf{h}_k$.

Step 3. Calculate β_k to minimize

$$\sum_{i=1}^{m} |f_i(\mathbf{x}_k+\beta\mathbf{h}_k)|.$$

Set $\mathbf{x}_{k+1}=\mathbf{x}_k+\beta_k\mathbf{h}_k$; $k=k+1$ and go to Step 2.

We make the following remarks:

1) In Step 2, the algorithm calculates the direction to follow. Clearly, this direction is a steepest-descent direction at the point $\mathbf{x}_k$.

2) To calculate this direction one has to solve a linear programming problem. The limitation due to assumptions (A1) and (A2) is a direct consequence of the linear programming approach.

As a conclusion, although Osborne and Watson in their paper gave a convergence proof for their algorithm, computational evidence shows the typical behavior of a steepest-descent technique. In §§ 5 and 6 where some numerical examples are solved, the algorithm showed very slow convergence near the optimum in one example and no convergence at all for another example.

4. Transformation to a sequence of minimizations.

Among the many numerical methods for solving unconstrained optimization problems, the methods that use the information gained from evaluating the gradient of the objective function are, in general, superior to the others in practice. Considering our problem (Problem I), the function $F(\mathbf{x})$ is not necessarily differentiable at all $\mathbf{x}$; therefore, we cannot directly use any of the existing powerful gradient methods. In this section, we transform Problem I to a sequence of minimizations of continuously differentiable functions. Consider first

Problem IV. Given m continuously differentiable functions $f_i(\mathbf{x})$, $i=1,\cdots,m$, minimize

$$P(\mathbf{x},\varepsilon)=\sum_{i=1}^{m}[f_i^2(\mathbf{x})+\varepsilon]^{1/2}, \qquad \varepsilon>0. \tag{4.1}$$

Clearly, the function $P(\mathbf{x},\varepsilon)$ is a differentiable function in $\mathbf{x}$ for $\varepsilon>0$. A necessary condition for $\mathbf{x}^*$ to minimize $P(\mathbf{x},\varepsilon)$ for a fixed $\varepsilon>0$ is

$$\sum_{i=1}^{m}\frac{f_i(\mathbf{x}^*)}{[f_i^2(\mathbf{x}^*)+\varepsilon]^{1/2}}\cdot\nabla f_i(\mathbf{x}^*)=\mathbf{0}. \tag{4.2}$$

We claim that if Problem IV is solved with decreasing values of ε, the resulting sequence of solutions converges to a solution of Problem I as $\varepsilon\to 0$, under suitable conditions. The following lemmas justify this claim.

LEMMA 4.1. *For each fixed* $\mathbf{x}$, $P(\mathbf{x},\varepsilon)$ *approaches* $F(\mathbf{x})$ *of Problem I as* ε *approaches.*

LEMMA 4.2. *Let* $\mathbf{x}_\varepsilon^*$ *minimize* $P(\mathbf{x},\varepsilon)$; *i.e. suppose*

$$P(\mathbf{x}_\varepsilon^*,\varepsilon)\leqq P(\mathbf{x},\varepsilon)\quad \forall\mathbf{x}\in R^n. \tag{4.3}$$

Let $\{\varepsilon_i\}$ be any sequence converging to zero. Then every limit point of the sequence $\{\mathbf{x}^*_{\varepsilon_i}\}$ is a solution of Problem I.

Proof. Suppose $\mathbf{x}^*$ is a limit point of the sequence $\{\mathbf{x}^*_{\varepsilon_i}\}$; then there exists a subsequence which we can renumber as $\{\varepsilon_i\}$, such that $\varepsilon_i \to 0$ and $\mathbf{x}^*_{\varepsilon_i} \to \mathbf{x}^*$ as $i \to \infty$. Let $\mathbf{x}$ be any element of R^n; then

$$P(\mathbf{x}^*_{\varepsilon_i}, \varepsilon_i) \leqq P(\mathbf{x}, \varepsilon_i) \quad \forall i. \tag{4.4}$$

Letting $i \to \infty$ gives

$$F(\mathbf{x}^*) \leqq F(\mathbf{x}). \tag{4.5}$$

Since this inequality holds for every $\mathbf{x} \in R^n$, it follows that $\mathbf{x}^*$ solves Problem I. □

LEMMA 4.3. *Suppose at least one of the functions $f_i(\mathbf{x})$ has the property that*

$$|f_i(\mathbf{x})| \to \infty \quad \textit{as} \quad \|\mathbf{x}\| \to \infty; \tag{4.6}$$

then, for each $\varepsilon > 0$, the function $P(\mathbf{x}, \varepsilon)$ actually attains its infimum, i.e. the set

$$\psi_\varepsilon = \{\mathbf{x}_\varepsilon : P(\mathbf{x}_\varepsilon, \varepsilon) = \inf_{\mathbf{x}} P(\mathbf{x}, \varepsilon)\} \tag{4.7}$$

is nonempty for each $\varepsilon > 0$.

Proof. Since $P(\mathbf{x}, \varepsilon)$ is continuous, and radially unbounded, it attains its infimum. □

Combining Lemmas 4.2 and 4.3, we get the following useful result.

LEMMA 4.4. *Suppose at least one of the functions f_i satisfies* (4.6), *and let $\{\varepsilon_i\}$ be a sequence that converges to zero; then every sequence $\{\mathbf{x}_i\}$ with $\mathbf{x}_i$ belonging to the set ψ_{ε_i} $\forall i$ contains a subsequence that converges to a solution of Problem* I.

LEMMA 4.5. *Let $\{\varepsilon_i\}$ be a monotonically decreasing sequence converging to zero and let $\mathbf{x}^*_{\varepsilon_i}$ minimize $P(\mathbf{x}, \varepsilon_i)\, \forall \mathbf{x} \in R^n$; then $P(\mathbf{x}^*_{\varepsilon_i}, \varepsilon_i)$ is a monotonically decreasing function in ε_i.*

Proof. We have

$$P(\mathbf{x}^*_{\varepsilon_i}, \varepsilon_i) \leqq P(\mathbf{x}^*_{\varepsilon_{i-1}}, \varepsilon_i) \leqq P(\mathbf{x}^*_{\varepsilon_{i-1}}, \varepsilon_{i-1}). \quad □$$

Using the above lemmas, we can give an alternate derivation of the necessary conditions for an l_1-optimum. Suppose $\{\mathbf{x}_i\}$ is a sequence converging to $\mathbf{x}^*$, and that $\mathbf{x}_i$ minimizes $P(\mathbf{x}, \varepsilon_i)$. Since $P(\mathbf{x}, \varepsilon_i)$ is continuously differentiable for all $\varepsilon_i > 0$, $\mathbf{x}_i$ must satisfy (4.2), i.e.

$$\sum_{j=1}^{m} \frac{f_j(\mathbf{x}_i)}{[f_j^2(\mathbf{x}_i) + \varepsilon_i]^{1/2}} \cdot \nabla f_j(\mathbf{x}_i) = \mathbf{0}. \tag{4.8}$$

Now let $i \to \infty$. For $j \notin K(\mathbf{x}^*)$, we see that

$$\frac{f_j(\mathbf{x}_i)}{[f_j^2(\mathbf{x}_i) + \varepsilon_i]^{1/2}} \to \operatorname{sign} f_j(\mathbf{x}^*) \quad \text{as } i \to \infty. \tag{4.9}$$

On the other hand, if $j \in K(\mathbf{x}^*)$, the sequence $\{f_j(\mathbf{x}_i)/[f_j^2(\mathbf{x}_i) + \varepsilon_i]^{1/2}\}$ does not have a definite limit in general; however, it is a bounded sequence (since all terms in the sequence are between -1 and 1), and hence some subsequence converges to a number between -1 and 1. Thus, at $\mathbf{x}^*$ we have

$$\sum_{j \notin K(\mathbf{x}^*)} [\operatorname{sign} f_j(\mathbf{x}^*)] \cdot \nabla f_j(\mathbf{x}^*) + \sum_{j \in K(\mathbf{x}^*)} \alpha_j \cdot \nabla f_j(\mathbf{x}^*) = \mathbf{0} \tag{4.10}$$

for some $\alpha_j \in [-1, 1]$, $j \in K(\mathbf{x}^*)$.

5. An algorithm for l_1-norm minimization. In this section we propose an algorithm for l_1-norm minimization problems. The basic algorithm is based on the material presented in the previous sections and is an iterative technique for minimizing the function $P(\mathbf{x}, \varepsilon)$ with decreasing values of ε.

BASIC ALGORITHM.

Step 1. Pick $\mathbf{x}_0 \in R^n$, $\varepsilon_1 > 0$ (small number); set $k = 1$.

Step 2. Minimize $P(\mathbf{x}, \varepsilon_k)$ given by equation (4.1). Denote the solution by $\mathbf{x}_k^*$.

Step 3. Set $\varepsilon_{k+1} = \varepsilon_k / L$ where L is a prespecified number greater than 1.

Step 4. If $\varepsilon_{k+1} \leqq \delta$ and/or if $\|\mathbf{x}_k^* - \mathbf{x}_{k-1}^*\| \leqq \beta$, where δ and β are prespecified small numbers depending on the accuracy desired, STOP. Otherwise, set $k = k+1$ and go back to Step 2.

Some remarks and questions are appropriate at this stage:

1. How can we choose an initial value for the parameter ε_1?

2. For smaller and smaller values of ε_k, the problem tends to become more and more ill-conditioned; i.e., the function $P(\mathbf{x}, \varepsilon)$, although theoretically differentiable, becomes in some cases, nondifferentiable in practice. This ill-conditioning occurs when at least one of the f_i's is close to or identically zero, and results in erroneous gradient information. As a result, the minimization of the function $P(\mathbf{x}, \varepsilon)$, employing a gradient method, takes an unnecessarily large number of function evaluations. Also, for some problems, the algorithm, although leading to a point which is very close to the minimum, does not converge to the minimum to within the desired accuracy.

3. The Hessian of the objective function $P(\mathbf{x}, \varepsilon)$ is given by

$$\nabla^2 P(\mathbf{x}, \varepsilon) = \sum_{i=1}^{m} \{[f_i^2(\mathbf{x}) + \varepsilon]^{-1/2} \cdot [f_i(\mathbf{x}) \cdot \nabla^2 f_i(\mathbf{x}) + \nabla f_i(\mathbf{x}) \cdot \nabla^T f_i(\mathbf{x})] - [f_i^2(\mathbf{x}) + \varepsilon]^{-3/2} \cdot f_i^2(\mathbf{x}) \cdot \nabla f_i(\mathbf{x}) \cdot \nabla^T f_i(\mathbf{x})\}. \tag{5.1}$$

It is clear, from the above expression, that if at least one of the f_i's becomes zero as $\varepsilon \to 0$, then the inverse of the Hessian becomes singular.

One can think of these difficulties as analogous to the difficulties occurring in some of the penalty function methods for constrained optimization problems (e.g. SUMT [2]) where, at the optimal point, the augmented objective function may not necessarily be differentiable.

The choice of an appropriate starting value for ε_1 is related to the scaling of the particular problem under consideration. In general, numerical experience with the algorithm shows that a good choice of ε_1 is one tenth of either the largest absolute value of the f_i's or the average absolute value of the f_i's, i.e.,

$$\varepsilon_1 = \tfrac{1}{10} \max_{i \in [1, \cdots, m]} |f_i(\mathbf{x}_0)| \tag{5.2}$$

or

$$\varepsilon_1 = \frac{1}{10m} \sum_{i=1}^{m} |f_i(\mathbf{x}_0)|. \tag{5.3}$$

In some rare cases, the spread in the values of different f_i, $i = 1, \cdots, m$, can be very large. In this case we associate an appropriate value ε_{1i} with every f_i or every particular set of f_i's, i.e., the function $P(\mathbf{x}, \varepsilon)$ is chosen to be

$$P(\mathbf{x}, \varepsilon) = \sum_{i=1}^{m} [f_i^2(\mathbf{x}) + \varepsilon_i]^{1/2}, \tag{5.4}$$

where

(5.5)
$$\varepsilon_i = \varepsilon \cdot c_i$$

and $c_i > 0$ are the weighting constants. In this case the theoretical results obtained in the previous sections still apply.

We turn now to the ill-conditioning problem. To avoid some of the difficulties arising as ε_k becomes close to zero, we use the extrapolation technique reported by Fiacco and McCormick [2], [3, p. 188], which can be summarized as follows:

Suppose the function $P(\mathbf{x}, \varepsilon)$ has been minimized for $\varepsilon_i > \varepsilon_2 > \cdots > \varepsilon_k > 0$ at $\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_k^*$. Then every $\mathbf{x}_j^*$, $j = 1, 2, \cdots, k$ can be expressed as a polynomial in $\varepsilon^{1/2}$ given by

(5.6)
$$\mathbf{x}_j^* = \sum_{i=0}^{k-1} \mathbf{a}_i (\varepsilon_j^{1/2})^i, \qquad j = 1, 2, \cdots, k,$$

where the $\mathbf{a}_i$'s are n-component vectors. Through some theoretical reasoning, we can obtain an estimate $\hat{\mathbf{x}}_{k+1}$ to $\mathbf{x}_{k+1}^*$, given that k minima have been obtained. This estimate is expressed as

(5.7)
$$\hat{\mathbf{x}}_{k+1} = \sum_{i=1}^{k} \mathbf{a}_{i-1} \left(\frac{\varepsilon_1}{L^k}\right)^{(i-1)/2}.$$

Note that the $\mathbf{a}_i$'s are unique and need not be calculated explicitly. For linear estimates, i.e., given two previous minima, only the first two terms in the last equation appear. For the theoretical details of this technique, see [2] and [3, p. 188].

The implementation of this technique in the new algorithm results in the following advantages:

1. The estimate $\hat{\mathbf{x}}_{k+1}$ to $\mathbf{x}_{k+1}^*$ is obtained as a linear combination of previous minima $\mathbf{x}_j^*$, $j = 1, 2, \cdots, k$, i.e., the new estimate depends on the minima obtained while the problem was better-conditioned.

2. At the $(k+1)$th iteration the starting point is $\hat{\mathbf{x}}_{k+1}$ and not $\mathbf{x}_k^*$ (previous minimum). So if $\hat{\mathbf{x}}_{k+1}$ is a good estimate to $\mathbf{x}_{k+1}^*$, the number of function evaluations in the minimization of $P(\mathbf{x}, \varepsilon_{k+1})$ is reduced. As a result a saving in the overall CPU time (in some cases up to 35%) occurs.

3. Let $(x_k^*)_i$ denote the ith component of $\mathbf{x}_k^*$. If, for all i, $i = 1, 2, \cdots, n$, each sequence $(x_k^*)_i$, $k = 1, 2, \cdots$ is a monotone sequence, the estimation of new minima at the last few iterations agrees with the exact minima to an accuracy approaching the desired accuracy of the problem. In most of the examples that have been solved using this algorithm, these sequences become monotone at the last few iterations, though this result seems to be difficult to prove in general.

Now, let us assume that the extrapolation technique is working in such a way that more and more accurate estimates are generated as the iterations proceed, which is a very mild assumption in practice; then the sequence of minima will hopefully converge to the minimum of the problem although the inverse of the Hessian is singular at the minimum. As a matter of curiosity, a nongradient method (e.g. Powell [11]) was used to minimize the function $P(\mathbf{x}, \varepsilon)$ as ε becomes very small, i.e., at the last few iterations. Numerical results show that this approach is generally slower than just using a gradient method (e.g. Fletcher [12]) together with the extrapolation scheme.

Although the technique of Fiacco and McCormick was designated to accelerate the convergence properties for some penalty function methods [3], the results obtained by employing it in conjunction with the new algorithm are encouraging; this gives their technique a wider application. One drawback of the technique is the increased computer storage requirement, which becomes critical only for large problems.

We present next an improved version of the algorithm, based on the previous discussion.

IMPROVED ALGORITHM.

Step 1. Pick $\hat{\mathbf{x}}_1 \in R^n$; set ε_1 as in (5.2) or (5.3); and set $k = 1$.

Step 2. Minimize $P(\mathbf{x}, \varepsilon_k)$ with starting point $\hat{\mathbf{x}}_k$. Denote the solution by $\mathbf{x}_k^*$.

Step 3. Set $\varepsilon_{k+1} = \varepsilon_k / L$ where L is a prespecified number greater than 1.

Step 4. If $\varepsilon_{k+1} \leqq \delta$ and/or $\|\mathbf{x}_k^* - \mathbf{x}_{k-1}^*\| \leqq \beta$, where δ and β are prespecified small numbers depending on the accuracy desired, STOP.

Step 5. If $k = 1$, set $\hat{\mathbf{x}}_{k+1} = \mathbf{x}_k^*$. If $k \geqq 2$, find an estimate $\hat{\mathbf{x}}_{k+1}$ to $\mathbf{x}_{k+1}^*$ by the above extrapolation technique.

Step 6. Set $k = k + 1$ and go back to Step 2.

We mention here that an approach similar to the one used by Abdelmalek [13] for linear l_1-approximation problems, can be applied to Problem I. Let

$$Z(\mathbf{x}, p) = \left\{ \sum_{i=1}^{m} |f_i(\mathbf{x})|^p \right\}^{1/p}, \qquad p > 1, \tag{5.8}$$

where $f_i(\cdot)$ is defined as in Problem I. For $p > 1$, the sequence of solutions obtained by minimizing the differentiable function $Z(\mathbf{x}, p)$ will hopefully converge to the l_1-solution as p approaches 1. However numerical evidence shows that this approach is numerically unstable. The function $Z(\mathbf{x}, p)$ becomes practically nondifferentiable for $p < 2$, which leads to wrong gradient calculations at the earlier stages of the computation.

A final point needs to be clarified. Suppose that for all i the functions f_i's are linear in some of the variables x_j, $j = 1, 2, \cdots, l$, and nonlinear in the others x_j, $j = l+1, \cdots, n$, and suppose the following procedure is applied:

1. Fix the nonlinear variables and solve the linear l_1 problem using Barrodale and Roberts algorithm [8].

2. Fix the linear variables obtained in the previous step and solve the nonlinear problem using the algorithm presented in this section.

3. Repeat the process until convergence is obtained.

Numerical results show that this approach leads in all cases to local minima; this is due to the fact that the search is along two orthogonal subspaces.

We close this section by giving two numerical examples, where at the minimal points the function $F(\mathbf{x}) = \sum_{i=1}^{m} |f_i(\mathbf{x})|$ is not differentiable, and we show that the algorithm converges to a point where the necessary conditions obtained in § 2 are satisfied.

Example 5.1. Minimize $F(x) = \sum_{i=1}^{3} |f_i(x)|$ where

$$f_1(\mathbf{x}) = x_1^2 + x_2 - 10, \qquad f_2(\mathbf{x}) = x_1 + x_2^2 - 7, \qquad f_3(\mathbf{x}) = x_1^2 - x_2^3 - 1.$$

The starting point is $\mathbf{x}_0 = [1 \;\; 1]^T$ where the value of $F(\mathbf{x}_0)$ is 14. The problem is solved using the basic algorithm as well as the improved algorithm. Table 5.1 summarizes the progress of the computation. At the point where the computation ended the values of the f_i's are 0, $-.4704$, and 0, which shows that $F(\mathbf{x})$ is not differentiable. In Table 5.2 we give the gradient vectors of the functions f_i's at this particular point. The necessary conditions for a minimum are satisfied with the multipliers $\alpha_1 = .4809$ and $\alpha_3 = -.305$. The computation is done on a CDC 6400 computer. The accuracy β (see Step 4 of the

TABLE 5.1

		$P(\mathbf{x}_k^*, \varepsilon_k)$	$F(\mathbf{x}_k^*)$	$\mathbf{x}_k^*$ (basic)		$\hat{\mathbf{x}}_k$ (estimates)		$\mathbf{x}_k^*$ (improved)		$P(\mathbf{x}_k^*, \varepsilon_k)$	$F(\mathbf{x}_k^*)$	No. of Func. Evaluations	
k	ε_k†	(basic)		x_1^*	x_2^*	$\hat{x}_1$	$\hat{x}_2$	x_1^*	x_2^*	(improved)		(basic)	(improved)
1	8×10^{-1}	2.774178	.626162	2.867128	1.941842	1.0	1.0	2.867128	1.941842	2.774178	.626162	21	21
2	8×10^{-2}	1.081869	.582102	2.860555	1.935813	2.867128	1.941842	2.860555	1.935813	1.081869	.582102	7	7
3	8×10^{-3}	.643181	.514535	2.849752	1.9264	2.858476	1.933907	2.849752	1.926400	.643181	.514535	9	9
4	8×10^{-4}	.523061	.484707	2.844857	1.9222	2.845463	1.922673	2.844857	1.922196	.523061	.484707	10	8
5	8×10^{-5}	.486880	.474952	2.843250	1.920817	2.843143	1.920729	2.843250	1.920816	.486880	.474952	11	7
6	8×10^{-6}	.475609	.471857	2.842740	1.920378	2.842739	1.920377	2.842739	1.920378	.475609	.471857	10	4
7	8×10^{-7}	.472062	.470877	2.842578	1.920239	2.842577	1.820239	2.842577	1.920239	.472062	.470877	11	4
8	8×10^{-8}	.470942	.470566	2.842527	1.920195	2.842526	1.920195	2.842526	1.920195	.470942	.470566	9	5
9	8×10^{-9}	.470588	.470470	2.842511	1.920182	2.842510	1.920181	2.842510	1.920181	.470588	.470470	10	5
10	8×10^{-10}	.470476	.470438	2.842506	1.920177	2.842505	1.920177	2.842505	1.920177	.470476	.470438	10	6
11	8×10^{-11}	.470443	.470433	2.842506	1.920176	2.842504	1.920176	2.842504	1.920176	.470441	.470431	6	6
12	8×10^{-12}	.470435	.470432	2.842506	1.920176	2.842504	1.920176	2.842504	1.920176	.470429	.470424	6	6

†$\varepsilon_1 = \frac{1}{10} \max_i |f_i(\mathbf{x}_0)|$.

TABLE 5.2

g_1	g_2	g_3
5.685006	1.	5.685006
1.	3.840351	−11.061219

algorithm) is 10^{-6}. The value ε_{12} in Table 5.1 is of the order of 10^{-12}, because ε should be comparable to f_i^2. As is clear from Table 5.1, the value of $F(\mathbf{x}^*)$ using the improved algorithm is slightly lower than the value obtained using only the basic algorithm. The necessary conditions for the minimum are satisfied in both cases, but to a higher accuracy when the improved algorithm is used. Third-order estimates were used in solving this example as well as Example 5.2.

Example 5.2. Minimize $F(\mathbf{x})=\sum_{i=1}^{6}|f_i(\mathbf{x})|$ where

$$f_1(\mathbf{x})=x_1^2+x_2^2+x_3^2-1. \qquad f_2(\mathbf{x})=x_1^2+x_2^2+(x_3-2)^2,$$
$$f_3(\mathbf{x})=x_1+x_2+x_3-1. \qquad f_4(\mathbf{x})=x_1+x_2-x_3+1.$$
$$f_5(\mathbf{x})=2x_1^3+6x_2^2+2(5x_3-x_1+1)^2, \qquad f_6(\mathbf{x})=x_1^2-9x_3.$$

The starting point is $[1 \quad 1 \quad 1]^T$. The results are given in Table 5.3 and 5.4. The Osborne and Watson algorithm is also used, and after 10 seconds of CPU time no convergence is obtained; and the value of the objective function is 8.446.

6. Nonlinear l_1-approximation problem. In this section we consider several nonlinear l_1-approximation problems to illustrate some of the theoretical ideas, and to show the effectiveness of the new algorithm together with the implementation of the extrapolation technique. We also show that the Osborne–Watson algorithm suffers from very slow convergence.

Example 6.1. We want to approximate the function

$$f(t)=\tfrac{1}{2}e^{-t}-e^{-2t}+\tfrac{1}{2}e^{-3t}+\tfrac{3}{2}e^{-(3/2)t}\sin(7t)+e^{-(5/2)t}\sin(5t) \tag{6.1}$$

by

$$K(A,t)=a_1e^{-a_2t}\cos(a_3t+a_4)+a_5e^{-a_6t}. \tag{6.2}$$

This problem can be thought of as that of finding a third-order model for a seventh-order system, in the best l_1-norm sense. The problem is discretized into 51 uniformly spaced points in the interval 0 to 5 seconds. The objective function is then

$$e(A)=\sum_{i=1}^{51}|f(t_i)-K(A,t_i)|, \tag{6.3}$$

which has exactly the same form as Problem II, where $e(A)$ is now the summation of absolute values of the errors in the approximation. In the application of the algorithm to find A^* that minimizes (6.3), Fletcher's method [12] was used to minimize the continuously differentiable function $P(A,\varepsilon)$. The starting point is $[2,2,7,0,-2,1]^T$ where the value of $e(A)$ is 24.254416. The choice of the starting point is not completely arbitrary, and corresponds to the so-called "dominant pole approximation" to the original system; however, with other starting points the algorithm converges to the same optimal point, but at the expense of increased CPU time. In Table 6.1 we summarize the progress of the computation.

TABLE 5.3

		$P(\mathbf{x}_k^*, \varepsilon_k)$	$F(\mathbf{x}_k^*)$		$\mathbf{x}_k^*$ (basic)		No. of Funct. Evaluations			$\mathbf{x}_k^*$ (improved)		$P(\mathbf{x}_k^*, \varepsilon_k)$	$F(\mathbf{x}_k^*)$
k	ε_k†	(basic)		x_1^*	x_2^*	x_3^*	(basic)	(improved)	x_1^*	x_2^*	x_3^*	(improved)	
1	5.8	17.58145	8.03995	.367175	−.039695	.001897	26	26	.367175	−.039695	.001897	17.58145	8.03995
2	$\varepsilon_1 \times 10^{-1}$	9.82947	8.00377	.441527	−.029461	−.001957	11	11	.441527	−.029461	−.001957	9.82947	8.00377
3	$\varepsilon_1 \times 10^{-2}$	8.23684	7.95112	.502993	−.007589	.0112971	11	11	.502993	−.007589	.0112971	8.23684	7.95112
4	$\varepsilon_1 \times 10^{-3}$	7.96597	7.91471	.526836	−.001609	.023681	14	13	.526836	−.0010609	.023681	7.96597	7.91471
5	$\varepsilon_1 \times 10^{-4}$	7.91289	7.90103	.533277	−.000115	.029027	14	11	.533277	−.000115	.029027	7.91289	7.90103
6	$\varepsilon_1 \times 10^{-5}$	7.89971	7.89642	.535134	−.000012	.030965	13	11	.535134	−.000012	.030965	7.89971	7.89642
7	$\varepsilon_1 \times 10^{-6}$	7.89592	7.89492	.535708	−.000001	.031612	14	10	.535708	−.000001	.031612	7.89592	7.89492
8	$\varepsilon_1 \times 10^{-7}$	7.89476	7.89445	.535888	0.0	.031821	14	7	.535888	0.0	.031821	7.89476	7.89445
9	$\varepsilon_1 \times 10^{-8}$	7.89439	7.89429	.535944	0.0	.031887	14	3	.535944	0.0	.031887	7.89439	7.89429
10	$\varepsilon_1 \times 10^{-9}$	7.89428	7.89425	.535963	0.0	.031908	18	3	.535962	0.0	.031909	7.89428	7.89424
11	$\varepsilon_1 \times 10^{-10}$	7.89424	7.89423	.535968	0.0	.031915	14	3	.535968	0.0	.031915	7.89424	7.89423
12	$\varepsilon_1 \times 10^{-11}$	7.89423	7.89423	.535970	0.0	.031917	14	4	.535970	0.0	.031917	7.89423	7.89423
13	$\varepsilon_1 \times 10^{-12}$	7.89423	7.89423	.535971	0.0	.031918	17	4	.535971	0.0	.031918	7.89423	7.89423
14	$\varepsilon_1 \times 10^{-13}$	7.89423	7.89423	.535971	0.0	.031918	6	4	.535971	0.0	.031918	7.89423	7.89423

† $\varepsilon_1 = \frac{1}{10} \max_i |f_i(\mathbf{x}_0)|$.

TABLE 5.4

$g_1(x^*)$	$g_2(x^*)$	$g_3(x^*)$	$g_4(x^*)$	$g_5(x^*)$	$g_6(x^*)$†
1.071942	1.071942	1.0	1.0	−.770894	1.071942
0.0	0.0	1.0	1.0	0.0	0.0
0.063837	−3.936163	1.0	−1.0	12.472411	−9.0

† $f_6(x^*)=0.0$; $\alpha_6=0.71915$.

An attempt was made to solve the same problem using the Osborne–Watson algorithm. The typical behavior of the steepest-descent type technique was observed (very fast convergence at the beginning of the computation followed by very slow progress).

Example 6.2. We consider here the rational approximation problem. We want to find a rational approximant of the form

$$K(A,x)=\frac{a_0+a_1x+a_2x^2}{1+b_1x+b_2x^2}$$

to the functions

(i) $\sqrt{x}$ in the interval $[0,1]$,
(ii) $e^x\cos x$ in the interval $[0,2]$,
(iii) $\sin x$ in the interval $[0,2\pi]$.

Case (i) was treated by Osborne and Watson [1]. They discretize the problem into 51 uniformly spaced samples on the given interval. Their algorithm converges to a point where the l_1-norm of the error in approximation was 0.089333 as reported in their paper. Starting with the same initial values and applying our algorithm, the optimal l_1-error is found to be 0.0707195. The results using the improved algorithm are given in Table 6.2.

7. Conclusion.

Although the problem of nonlinear l_1-minimization is not new, no practical algorithms to solve it were previously available. The algorithm proposed in this paper gives a new approach to deal with such problem. The implementation of the extrapolation technique of Fiacco and McCormick renders the algorithm numerically stable and accelerates the convergence. One advantage of the new algorithm is that it is possible to use the very powerful gradient methods such as the one of Fletcher [12] although the objective function may not be differentiable everywhere.

Acknowledgment. The authors thank Professor I. Barrodale and the referee of this paper for their valuable comments and suggestions.

TABLE 6.1

		Without Extrapolation (basic)					Function evaluations		With Extrapolation (improved)				
k	ε_k	$P(A_k, \varepsilon_k)$	$e(A_k)$	Parameters					Parameters			$P(A_k, \varepsilon_k)$	$e(A_k)$
1	.335744*	29.56385	.610844	2.259258 −1.651950	1.892493 .164569	6.820873 .702816	35	35	2.259258 −1.651950	1.892493 .164569	6.820873 .702816	29.56385	.610844
2	$\varepsilon_1/16$	7.437931	.609294	2.259399 −1.65183	1.892605 .164532	6.819793 .704510	14	14	2.259399 −1.65183	1.892605 .164532	6.819793 .704510	7.437931	.609294
3	$\varepsilon_1/(16)^2$	2.019614	.594582	2.260413 −1.650497	1.892787 .163936	6.808479 .718380	13	13	2.260413† −1.650497	1.892787 .163936	6.808479 .718380	2.019614	.594582
4	$\varepsilon_1/(16)^3$	.823673	.569856	2.257807 −1.646999	1.885035 .162506	6.781303 .733393	18	18	2.257807‡ −1.646999	1.885035 .162506	6.781303 .733393	.823673	.569856
5	$\varepsilon_1/(16)^4$	.600441	.561924	2.248622 −1.645658	1.870877 .164487	6.771466 .749205	18	18	2.248622§ −1.645658	1.870877 .164487	6.771466 .749205	.600441	.561924
6	$\varepsilon_1/(16)^5$	.565678	.560805	2.243051 −1.645317	1.861943 .165896	6.770729 .750880	24	20	2.243051§ −1.645317	1.861943 .165896	6.770729 .750880	.565678	.560805
7	$\varepsilon_1/(16)^6$	.560810	.559994	2.241403 −1.644923	1.858676 .165716	6.769981 .742489	27	18	2.241403§ −1.644923	1.858676 .165716	6.769981 .742488	.560810	.559994
8	$\varepsilon_1/(16)^7$	.560000	.559855	2.240965 −1.644845	1.857881 .165719	6.769862 .740868	26	16	2.240965§ −1.644845	1.857881 .165719	6.769862 .740868	.560000	.559854
9	$\varepsilon_1/(16)^8$	.559853	.559825	2.240825 −1.644858	1.857711 .165790	6.769924 .741257	31	4	2.240854§ −1.644827	1.857686 .165724	6.769838 .740508	.559825	.559823
10	$\varepsilon_1/(16)^9$	.559828	.559825	2.240825 −1.644858	1.857711 .165790	6.769924 .741257	6	4	2.240826§ −1.644823	1.857637 .165725	6.769832 .740422	.559824	.559818
11	$\varepsilon_1/(16)^{10}$							6	2.240826§ −1.644823	1.857637 .165725	7.769832 .740422	.559818	.559817

* $\varepsilon_1 = \frac{1}{10} \max_i |f(t_i) - K(A_0, t_i)|$.
† Linear extrapolation.
‡ Quadratic extrapolation.
§ Cubic extrapolation.

TABLE 6.2

Function	Number of Samples	Value of L	Value of ε_1	Starting Point					Optimal Point					No. of iterations	No. of funct. eval.	l_1-norm of error
				a_0	a_1	a_2	b_1	b_2	a_0^*	a_1^*	a_2^*	b_1^*	b_2^*			
$\sqrt{x}$	.51	10	$\varepsilon_1 = \frac{1}{10} \max_i \lvert f_i \rvert$	.1706	1.7578	0.0	.9537	0.0	0.0	8.563022	29.314374	24.738527	12.229623	11	301	.0707195
$e^x \cos x$	51	10	$\varepsilon_1 = \frac{1}{10} \max_i \lvert f_i \rvert$	1.0	1.0	1.0	1.0	1.0	.982518	.567515	−.759189	−.570984	.110569	10	129	.170838
	11	10	$\varepsilon_1 = \frac{1}{10} \max_i \lvert f_i \rvert$	1.0	1.0	1.0	1.0	1.0	.984748	.551478	−.750001	−.574838	.111592	10	142	.040701
$\sin x$	11	10	$\varepsilon_1 = \frac{1}{10} \max_i \lvert f_i \rvert$	0.0	1.0	1.0	1.0	1.0	.596412	−.228811	.013419	−.524558	.076288	11	189	1.969563
	51	10	$\varepsilon_1 = \frac{1}{10} \max_i \lvert f_i \rvert$	0.0	1.0	1.0	1.0	1.0	.641239	−.204113	0.0	−.529810	.084322	10	172	7.373005

REFERENCES

[1] M. R. Osborne and G. A. Watson, *On an algorithm for discrete nonlinear L_1 approximation*, Comput. J., 14 (1971), pp. 184–188.

[2] A. V. Fiacco and G. P. McCormick, *Extensions of SUMT for nonlinear programming: Equality constraints and extrapolation*, Management Sci., 12 (1966), pp. 816–829.

[3] ———, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York, 1968.

[4] R. A. El-Attar, M. Vidyasagar and S. R. K. Dutta, *Optimality conditions for l_1-norm minimization*, Proc. 19th Midwest Symposium on Circuits and Systems (Aug. 1976), pp. 272–275.

[5] I. Barrodale, F. D. K. Roberts and C. R. Hunt, *Computing best l_p-approximations by functions nonlinear in one parameter*, Comput. J., 13 (1970), pp. 382–386.

[6] M. Aoki, *Introduction to Optimization Techniques*, Macmillan, New York, 1971.

[7] I. Barrodale, *On computing best L_1 approximations*, Approximation Theory, Talbot, A., ed., Academic Press, New York, 1970, pp. 205–215.

[8] I. Barrodale and F. D. K. Roberts, *An Improved algorithm for discrete L_1 linear approximation*, this Journal, 10 (1973), pp. 839–848.

[9] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.

[10] O. L. Mangasarian, *Nonlinear Programming*, McGraw-Hill, New York, 1969.

[11] M. J. D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comput. J., 15 (1964), pp. 155–162.

[12] R. Fletcher, *A new approach to variable metric algorithms*, Ibid., 13 (1970), pp. 317–322.

[13] N. N. Abdelmalek, *Linear L_1 Approximation for a Discrete Point Set and L_1 Solutions of Overdetermined Linear Equations*, J. Assoc. Comput. Mach., 18 (1971), pp. 41–74.