JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 119, No. 3, pp. 459–484, December 2003 (©2003)

Algorithms with Adaptive Smoothing for Finite Minimax Problems[1]

E. POLAK,[2] J. O. ROYSET,[3] AND R. S. WOMERSLEY[4]

Communicated by D. Q. Mayne

Abstract. We present a new feedback precision-adjustment rule for use with a smoothing technique and standard unconstrained minimization algorithms in the solution of finite minimax problems. Initially, the feedback rule keeps a precision parameter low, but allows it to grow as the number of iterations of the resulting algorithm goes to infinity. Consequently, the ill-conditioning usually associated with large precision parameters is considerably reduced, resulting in more efficient solution of finite minimax problems.

The resulting algorithms are very simple to implement, and therefore are particularly suitable for use in situations where one cannot justify the investment of time needed to retrieve a specialized minimax code, install it on one's platform, learn how to use it, and convert data from other formats. Our numerical tests show that the algorithms are robust and quite effective, and that their performance is comparable to or better than that of other algorithms available in the Matlab environment.

Key Words. Finite minimax, nonsmooth optimization algorithms, smoothing techniques, feedback precision-adjustment rule.

[1]Support from the Norwegian and Australian Research Councils, National Science Foundation Grant ECS-9900985, UC Berkeley Space Sciences Laboratory, and Lockheed Martin Advanced Technology Center Minigrant Program is acknowledged.

[2]Professor, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California.

[3]Graduate Student, Department of Civil and Environmental Engineering, University of California, Berkeley, California.

[4]Associate Professor, School of Mathematics, University of New South Wales, Sydney, Australia.

0022-3239/03/1200-0459/0

1. Introduction

Many applications in engineering (see e.g. Refs. 1–2) give rise to finite minimax problems in the form

$$(\mathrm{P}) \quad \min_{x\in\mathbb{R}^n} \psi(x), \tag{1}$$

where $\psi:\mathbb{R}^n\to\mathbb{R}$ is defined by

$$\psi(x) \triangleq \max_{j\in Q} f^j(x), \tag{2}$$

with $f^j:\mathbb{R}^n\to\mathbb{R}$, $j\in Q\triangleq\{1,\ldots,q\}$, being smooth functions. In particular, (P) may occur in L_1 and L_∞ approximations problems, solution of systems of nonlinear equations, problems of finding feasible points of systems of inequalities, solution of nonlinear programming problems using non-differentiable penalty functions, or occur as subproblems in algorithms for the solution of semi-infinite minimax problems; see e.g. Refs. 3–10. This has given rise to a considerable literature on algorithms for the solution of finite minimax problems.

There are several different approaches that have been taken to solve (P). Many authors have derived algorithms for the solution of (P) by solving a sequence of either unconstrained subproblems (Refs. 11–14 and Ref. 8, Section 2.4.1) or constrained subproblems (Refs. 15–19) of the trust-region type.

The equivalent transcription of (P) into a standard nonlinear program, i.e.,

$$(\mathrm{P}') \quad \min_{(x,\,\alpha)\in\mathbb{R}^{n+1}} \{\alpha \mid f^j(x)-\alpha\le 0,\ j\in Q\}, \tag{3}$$

where $\alpha\in\mathbb{R}$, has been exploited by Refs. 20–30. Di Pillo et al. (Ref. 23) developed a differentiable penalty function for the transcribed problem (P′). In Ref. 20, (P′) is not solved directly, but it is considered in auxiliary calculations in the proposed algorithm.

In Refs. 31–32, an algorithm is developed based on a barrier function. In Ref. 33, an algorithm is constructed exploiting the special minimax problem arising in discrete Chebyshev approximation problems.

A variety of regularization approaches has been used to obtain smooth approximations to problem (P) (Refs. 34–44). The main advantage of smoothing techniques is that they convert minimax problems into simple, smooth, unconstrained optimization problems that can be solved by a standard unconstrained minimization solver; see e.g. Refs. 39 and 43. However, when the approximation accuracy is high, the smooth approximating problems become significantly ill-conditioned. Hence, the unconstrained

optimization solver may experience numerical difficulties and slow convergence. Consequently, the simple use of smoothing techniques is complicated by the need of trading-off accuracy of approximation against problem ill-conditioning.

An attempt to address these shortcomings was made in Ref. 43, where a precision parameter for the smooth approximation is initially given by a preselected value and then it is increased by a fixed factor at each consecutive iteration (specifically 2). Effectively, the algorithm is solving a sequence of gradually more accurate approximations. However, the main problem with this open-loop scheme is its sensitivity to the selection of the multiplication factor, as can be seen from our numerical experiments in Section 4. The precision parameter may become extremely large too fast and cause ill-conditioning and floating-point overflow before the algorithm has converged to an approximate solution. Additionally, Ref. 43 establishes only convergence results under the rather restrictive assumption that the max function is coercive and that the smooth approximations are strictly convex.

In this paper, we present a feedback precision-adjustment rule, that we combine with several unconstrained optimization algorithms, to obtain several robust and effective minimax algorithms whose performance is significantly superior to that of algorithms based on open-loop schemes. Our feedback precision-adjustment rule is based on tests and is implemented as a subroutine. The aim of our feedback scheme is to ensure that the precision parameter is kept small far from a solution and is increased as a solution is approached, resulting in a much better management of ill-conditioning than with open-loop schemes. Hence, the new algorithms retain the simplicity of other smoothing approaches, while reducing the numerical difficulties. Additionally, the algorithms converge under weaker assumptions than those in Ref. 43.

Although, we cannot claim that the new algorithms are superior to a number of algorithms, such as the one described Ref. 30, and whose implementations are available in C or FORTRAN, our numerical results show that their performance is comparable to or better than that of a number of algorithms available in the Matlab environment. The new algorithms are extremely simple to code and hence are particularly suitable to situations where only a few minimax problems need to be solved and it makes little sense to invest the time needed to retrieve a specialized minimax code, get it to run on one's platform, learn how to use it, and convert data from other formats.

The smooth approximations are described in Section 2. Section 3 presents the new algorithms, the subroutine for increasing the precision parameter, and the proof of convergence. Section 4 contains a series of numerical examples comparing the performance of different algorithms.

2. Smooth Approximations

We will apply a smoothing technique to (P) that can be found in Refs. 35, 39, and 43. Let $p>0$ be a smoothing precision parameter and consider the approximating problem

$$(\mathrm{P}_p) \quad \min_{x\in\mathbb{R}^n} \psi_p(x), \tag{4}$$

where $\psi_p: \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\psi_p(x) \triangleq \log\left(\sum_{j\in Q} \exp(pf^j(x))\right)\Big/p. \tag{5}$$

This approximation can be derived using Jaynes' maximum entropy principle (see Ref. 39) and is sometimes referred to as an exponential penalty function. An alterative expression for $\psi_p(\cdot)$ is

$$\psi_p(x) = \psi(x) + \log\left(\sum_{j\in Q} \exp(p(f^j(x) - \psi(x)))\right)\Big/p. \tag{6}$$

It follows from (6) that

$$0 \le \psi_p(x) - \psi(x) \le \log(q)/p; \tag{7}$$

hence,

$$\psi_p(x) \to \psi(x), \qquad \text{as } p\to\infty.$$

The following result due to Li (Theorem 1, Ref. 39) shows that the convergence is monotonic.

Lemma 2.1. $\psi_p(x)$ decreases monotonically as p increases and, for any $x\in\mathbb{R}^n$, $\psi_p(x)\to\psi(x)$, as $p\to\infty$.

Assumption 2.1. We assume that the functions $f^j(\cdot)$, $j\in Q$, are twice continuously differentiable.

Remark 2.1. Under Assumption 2.1, $\psi_p(\cdot)$ is twice continuously differentiable for arbitrary $p>0$.

In the following, for any function f, we will use the notation $f_{xx}(x)$ for the Hessian of f at x.

Lemma 2.2. Suppose that Assumption 2.1 holds. Then, for every bounded set $S\subset\mathbb{R}^n$, there exists an $L<\infty$ such that

$$\langle y, \psi_{p,xx}(x)y\rangle \le pL\|y\|^2, \tag{8}$$

for all $x\in S$, $y\in\mathbb{R}^n$, and $p\ge 1$.

Proof. By Assumption 2.1, $\psi_p(\cdot)$ is twice differentiable, with gradient

$$\nabla \psi_p(x) \triangleq \sum_{j\in Q} \mu_p^j(x) \nabla f^j(x), \tag{9}$$

where, for any $j^* \in Q$,

$$\mu_p^{j^*}(x) \triangleq \exp(pf^{j^*}(x)) \Big/ \left(\sum_{j\in Q} \exp(pf^j(x)) \right), \tag{10}$$

and Hessian

$$\psi_{p,xx}(x) \triangleq \sum_{j\in Q} [\nabla \mu_p^j(x) \nabla f^j(x)^T + \mu_p^j(x) f_{xx}^j(x)], \tag{11}$$

where, for any $j^* \in Q$,

$$\nabla \mu_p^{j^*}(x) \triangleq p\mu_p^{j^*}(x) \sum_{j\in Q} \mu_p^j(x)(\nabla f^{j^*}(x) - \nabla f^j(x)). \tag{12a}$$

Let $S \subset \mathbb{R}^n$ be bounded. Then by continuity, there exists a $K < \infty$ such that $\|\nabla f^j(x)\| \le K$ and

$$\langle y, f_{xx}^j(x) y \rangle \le K \|y\|^2, \tag{12b}$$

for all $x \in S$, $y \in \mathbb{R}^n$, and $j \in Q$. Then, for all $x \in S$,

$$\begin{aligned} \|\nabla \mu_p^{j^*}(x)\| &\le p \sum_{j\in Q} \|\nabla f^{j^*}(x) - \nabla f^j(x)\| \\ &\le 2pqK. \end{aligned} \tag{13}$$

Hence, there exists $K_1 < \infty$ such that

$$\left\langle y, \sum_{j\in Q} \nabla \mu_p^j(x) \nabla f^j(x)^T y \right\rangle \le pK_1 \|y\|^2, \tag{14}$$

for all $x \in S$ and $y \in \mathbb{R}^n$. By inspection, $0 < \mu_p^j(x) < 1$, for all $x \in \mathbb{R}^n$, $j \in Q$, and $p > 0$. Hence, for $p \ge 1$ and $x \in S$,

$$\begin{aligned} \langle y, \psi_{p,xx}(x) y \rangle &\le pK_1 \|y\|^2 + qK\|y\|^2 \\ &\le p(K_1 + qK)\|y\|^2. \end{aligned} \tag{15}$$

Hence,

$$L = K_1 + qK.$$

This completes the proof. □

An optimality condition for (P) is given in terms of the subgradient of $\psi(\cdot)$; see Ref. 8.

Theorem 2.1. Suppose that the functions $f^j(\cdot)$, $j \in Q$, are continuously differentiable. If $\hat{x} \in \mathbb{R}^n$ is a local minimizer for (P), then

$$0 \in \partial \psi(\hat{x}) \triangleq \operatorname*{conv}_{j \in \hat{Q}(\hat{x})} \{\nabla f^j(\hat{x})\}, \tag{16a}$$

where

$$\hat{Q}(\hat{x}) \triangleq \{j \in Q \mid f^j(\hat{x}) = \psi(\hat{x})\} \tag{16b}$$

and conv$\{A\}$ denotes the convex hull of A.

3. Algorithms

We present two algorithms, together with a feedback precision-adjustment subroutine, for the solution of (P). The first algorithm is based on the Armijo gradient method (see Ref. 8); the second one is based on the stabilized Newton-Armijo algorithm (see Ref. 8). Both algorithms call the subroutine to determine when and how much the precision parameter p should be increased. In Section 4 below, we also test a third algorithm based on the stabilized Newton-Armijo algorithm with the Hessian replaced by its BFGS approximation (see e.g. Ref. 8).

Algorithm 3.1. Armijo Gradient Algorithm.
Parameters. $\alpha, \beta \in (0, 1)$, $p_0 > 0$.
Data. $x \in \mathbb{R}^n$.

Step 0. Set $i = 0$, $k = 0$, and $\gamma = 1$.
Step 1. Compute the steepest decent direction $h_{p_i}(x_i) = -\nabla \psi_{p_i}(x_i)$.
Step 2. Compute the stepsize

$$\lambda_{p_i}(x_i) = \max_{l \in \mathbb{N}} \{\beta^l \mid \psi_{p_i}(x_i + \beta^l h_{p_i}(x_i)) - \psi_{p_i}(x_i) \leq -\alpha \beta^l \|h_{p_i}(x_i)\|^2\}. \tag{17}$$

Step 3. Set $x_{i+1} = x_i + \lambda_{p_i}(x_i) h_{p_i}(x_i)$.
Step 4. Enter Subroutine 3.1 and go to Step 1 when exit Subroutine 3.1.

Subroutine 3.1. Adaptive Parameter Growth.
Parameters. Functions $\epsilon_b, \epsilon_a, \tau$: $(0, \infty) \to (0, \infty)$, with $\epsilon_b(p) \geq \epsilon_a(p) \geq \tau(p)$ for all $p > 0$, $\hat{p} \gg 1$.

Substep 1. If

$$\|\nabla \psi_{p_i}(x_i)\|^2 \leq \tau(p_i), \tag{18}$$

go to Substep 2. Else, set $p_{i+1} = p_i$, replace i by $i + 1$, and exit Subroutine 3.1.

Substep 2. If (initial stage)

$$p^* \leq \hat{p} \quad \text{and} \quad \gamma = 1, \tag{19a}$$

where p^* is such that

$$\epsilon_a(p_i) \leq \|\nabla \psi_{p^*}(x_i)\|^2 \leq \epsilon_b(p_i), \tag{19b}$$

set $p_{i+1} = \max\{p^*, p_i + 1\}$, replace k by $k + 1$, i by $i + 1$, and exit Subroutine 3.1.

Else, if (switch stage)

$$p^* > \hat{p} \quad \text{and} \quad \gamma = 1, \tag{20a}$$

where p^* is such that

$$\epsilon_a(p_i) \leq \|\nabla \psi_{p^*}(x_i)\|^2 \leq \epsilon_b(p_i), \tag{20b}$$

set $\gamma = \max\{2, (\hat{p} + 2)/(k + 1)\}$, $p_{i+1} = \gamma(k + 2)$, replace k by $k + 1$, i by $i + 1$, and exit Subroutine 3.1.

Else (final stage), set $p_{i+1} = \gamma(k + 2)$, replace k by $k + 1$, i by $i + 1$, and exit Subroutine 3.1.

Algorithm 3.1 with Subroutine 3.1 solves the sequence of approximating problem $\{(P_{p_i})\}_{i=0}^{\infty}$, associate with a sequence of monotonically increasing precision parameters p_i that diverge to infinity. At a given precision level (p', say), Algorithm 3.1 computes iterates that approach a stationary point of the approximating problem ($P_{p'}$). When the current iterate is sufficiently close to a stationary point for ($P_{p'}$), as determined by the test in (18), the precision level is increased from p' to p'', say. Algorithm 3.1 will then continue by computing iterates that are approaching a stationary point of ($P_{p''}$) until the test in (18) again determines that the precision level has to be increased. The last iteration of the previous precision level is used as a warm start for calculations on the next precision level.

In our case, in contrast to open-loop schemes (see Ref. 43), the sequence of precision parameters $\{p_i\}_{i=0}^{\infty}$ is not determined a priori but is constructed by Subroutine 3.1. When (18) is satisfied, the precision level is increased by an amount determined by Substep 2 in Subroutine 3.1.

If $\tau(p_i)$ is small, then the precision parameter will not be augmented before the current iterate x_i is close to a minimizer for $\psi_{p_i}(x_i)$. In the early iterations, i.e., before the test in (19a) fails, p_{i+1} is determined according to the value of p^*; see Substep 2. If $\epsilon_a(p_i)$ and $\epsilon_b(p_i)$ are small, then x_i will also

be relatively close to a minimizer $\psi_{p_{i+1}}(x_i)$. Hence, x_i is a good warm start for the new approximating problem $\min_{x\in\mathbb{R}^n}\psi_{p_{i+1}}(x)$.

When the test in (19a) fails, γ is set to be larger than 1 in the switch stage of Subroutine 3.1. Hence, for all subsequent iterations, the increase of the precision parameter will be determined by the final stage of Subroutine 3.1. Note that, in the switch stage and final stage, the precision parameter is augmented by a fixed amount γ each time $p_{i+1} > p_i$. This is to ensure the technical property that

$$\sum_{i=0}^{\infty} 1/p_i = +\infty;$$

see Lemma 3.3 and the proofs of convergence of Algorithms 3.1 and 3.2.

Let $t > 0$ be the required tolerance on the minimax solution. Then, every $p_i \gg \log(q)/t$ is associated with an error [see (7)] $\log(q)/p_i \ll t$. Hence, we recommend that the algorithm parameter $\hat{p}$, used to decide when the switch from the initial stage to the final stage should take place, be set equal to $\log(q)/t$.

Lemma 3.1. Suppose that Assumption 2.1 holds and that the sequences $\{x_i\}_{i=0}^{\infty}$ and $\{p_i\}_{i=0}^{\infty}$ are generated by Algorithm 3.1 with Subroutine 3.1. Then, the following properties hold: (i) the sequence $\{p_i\}_{i=0}^{\infty}$ is monotone nondecreasing; (ii) if $\{x_i\}_{i=0}^{\infty}$ has an accumulation point, then $p_i \to \infty$ and $\sum_{i=0}^{\infty} 1/p_i = +\infty$.

Proof.

(i) If the test in (18) fails, then $p_{i+1} = p_i$; if the test in (18) is satisfied, then $p_{i+1} \geq p_i + 1$.

(ii) Suppose that Algorithm 3.1 has generated the sequence $\{x_i\}_{i=0}^{\infty}$ with accumulation point $\hat{x}$, but $\{p_i\}_{i=0}^{\infty}$ is bounded from above. By construction, $p_{i+1} \geq p_i + 1$ whenever $p_{i+1} > p_i$. Hence, there must exist an $i^* \in \mathbb{N}$ such that, for all $i > i^*$,

$$\|\nabla \psi_{p^{i^*}}(x_i)\|^2 > \tau(p_{i^*}). \tag{21}$$

By Theorem 1.3.4 in Ref. 8, every accumulation point $\hat{x}$ of $\{x_i\}_{i=0}^{\infty}$ satisfies $\|\nabla \psi_{p_{i^*}}(\hat{x})\| = 0$. This is a contradiction, so $p_i \to \infty$.

Next, since p_i is eventually increased at a linear rate or slower, the final result follows by the fact that $\sum_{i=0}^{\infty} 1/i = +\infty$. □

Lemma 3.2. Suppose that Assumption 2.1 holds. For every bounded set $S \subset \mathbb{R}^n$ and parameters $\alpha, \beta \in (0, 1)$, there exists a $K_S < \infty$ such that, for all $p \geq 1$ and $x \in S$,

$$\psi_p(x - \lambda_p(x)\nabla \psi_p(x)) - \psi_p(x) \leq -\alpha K_S \|\nabla \psi_p(x)\|^2/p, \tag{22}$$

where $\lambda_p(x)$ is the stepsize of Algorithm 3.1; see (17).

Proof. Let $S \subset \mathbb{R}^n$ be bounded. It follows from Assumption 2.1 and (9) that there exists a constant $M < \infty$ such that

$$\|\nabla \psi_p(x)\| \leq M, \qquad \text{for all } x \in S \text{ and } p > 0.$$

Next, let

$$S_B \triangleq \{x \in \mathbb{R}^n \mid \|x - x'\| \leq M,\ x' \in S\} \tag{23a}$$

and let $L < \infty$ be the constant corresponding to S_B such that (8) holds for all $x \in S_B$, $y \in \mathbb{R}^n$ and $p \geq 1$. Then, for all $\lambda \in (0, 1]$, $x \in S$ and $p \geq 1$, we have by expansion and Lemma 2.2 that, for some $s \in [0, 1]$,

$$\begin{aligned}
&\psi_p(x - \lambda \nabla \psi_p(x)) - \psi_p(x) + \alpha\lambda \|\nabla \psi_p(x)\|^2 \\
&= -\lambda(1 - \alpha)\|\nabla \psi_p(x)\|^2 \\
&\quad + \lambda^2 \langle \nabla \psi_p(x), \psi_{p,xx}(x - s\lambda \nabla \psi_p(x)) \nabla \psi_p(x) \rangle / 2 \\
&\leq -\lambda \|\nabla \psi_p(x)\|^2 (1 - \alpha - \lambda p L/2).
\end{aligned} \tag{23b}$$

Let

$$\lambda^* \triangleq \min\{1, 2(1 - \alpha)/pL\}. \tag{23c}$$

Then, it follows from (23b) that, for every $\lambda \in (0, \lambda^*]$, we have

$$\psi_p(x - \lambda \nabla \psi_p(x)) - \psi_p(x) + \alpha\lambda \|\nabla \psi_p(x)\|^2 \leq 0. \tag{24}$$

Hence, by (24) and the stepsize rule in (17),

$$\lambda_p(x) \geq \beta \lambda^*, \tag{25}$$

for all $x \in S$ and $p \geq 1$. Consequently, by (17) and (25), we have that

$$\begin{aligned}
&\psi_p(x - \lambda_p \nabla \psi_p(x)) - \psi_p(x) \\
&\leq -\alpha \lambda_p \|\nabla \psi_p(x)\|^2 \\
&\leq -\alpha \beta \lambda^* \|\nabla \psi_p(x)\|^2 \\
&\leq -\alpha \min\{\beta, 2\beta(1 - \alpha)/L\} \|\nabla \psi_p(x)\|^2 / p,
\end{aligned} \tag{26}$$

for all $x \in S$ and $p \geq 1$. Hence, the conclusion follows with

$$K_S = \min\{\beta, 2\beta(1 - \alpha)/L\}.$$

This completes the proof. □

Theorem 3.1. Suppose that Assumption 2.1 holds and that $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence generated by Algorithm 3.1 with Subroutine 3.1. Then, there exist an infinite subset $K \subset \mathbb{N}$ and a $\hat{x} \in \mathbb{R}^n$ such that $x_i \to^K \hat{x}$ and $0 \in \partial \psi(\hat{x})$; i.e., $\hat{x}$ is a stationary point for (P).

Proof. Suppose that $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence generated by Algorithm 3.1. For the sake of a contradiction, suppose that there exists an $\epsilon > 0$ such that

$$\liminf_{i\to\infty} \|\nabla \psi_{p_i}(x_i)\| \geq \epsilon. \tag{27}$$

Since $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence, it has at least one accumulation point. Hence, by Lemma 3.1, $p_i \to \infty$, as $i \to \infty$. Next, by Lemma 3.2, there exist an $M < \infty$ such that

$$\psi_{p_i}(x_{i+1}) - \psi_{p_i}(x_i) \leq -\alpha M \|\nabla \psi_{p_i}(x_i)\|^2 / p_i, \tag{28}$$

for all $i \in \mathbb{N}$. Hence, for all $i \in \mathbb{N}$,

$$\begin{aligned} &\psi_{p_{i+1}}(x_{i+1}) - \psi_{p_i}(x_i) \\ &= \psi_{p_{i+1}}(x_{i+1}) - \psi_{p_i}(x_{i+1}) + \psi_{p_i}(x_{i+1}) - \psi_{p_i}(x_i) \\ &\leq -\alpha M \|\nabla \psi_{p_i}(x_i)\|^2 / p_i, \end{aligned} \tag{29}$$

where we have used the fact from Lemma 2.1 that

$$\psi_{p_{i+1}}(x_{i+1}) - \psi_{p_i}(x_{i+1}) \leq 0, \qquad \text{for all } i \in \mathbb{N}.$$

Since by Lemma 3.1,

$$\sum_{i=0}^{\infty} 1/p_i = +\infty,$$

it follows from (27) and (29) that

$$\psi_{p_i}(x_i) \to -\infty, \qquad \text{as } i \to \infty.$$

But for every accumulation point x^* of $\{x_i\}$, i.e., there exists as infinite subset $K^* \subset \mathbb{N}$ such that $x_i \to^{K^*} x^*$, we have by continuity that $\psi_{p_i}(x_i) \to^{K^*} \psi(x^*)$, which is a contradiction. Hence,

$$\liminf_{i\to\infty} \|\nabla \psi_{p_i}(x_i)\| = 0. \tag{30}$$

Now, by (9) and (10), we have that, for all $x \in \mathbb{R}^n$,

$$\lim_{p\to\infty} \nabla \psi_p(x) = \sum_{j \in \hat{Q}(x)} \nu^j(x) \nabla f^j(x) \in \partial \psi(x), \tag{31}$$

for some $\nu^j(x) \geq 0$, $j \in \hat{Q}(x)$ such that

$$\sum_{j \in \hat{Q}(x)} \nu^j(x) = 1.$$

Hence by (30) and (31), there have to exist an infinite subset $K \subset \mathbb{N}$ and a $\hat{x} \in \mathbb{R}^n$ such that $x_i \to^K \hat{x}$ and $0 \in \partial \psi(\hat{x})$. This completes the proof. □

We turn our attention now to the second algorithm, which is based on the stabilized Newton-Armijo algorithm; see Ref. 8. The selection of the search direction depends on the positive definiteness of the Hessian of $\psi_p(\cdot)$ [see (11)], which can be determined by computing the eigenvalues of $\psi_{p,xx}(\cdot)$. Here, we have selected to use a more efficient procedure based on the Cholesky decomposition of $\psi_{p,xx}(\cdot)$ and its reciprocal condition number. Three parameters ($\kappa_1 > 0$, κ_2, $\kappa_3 \gg 1$) are associated with the selection of the type of search direction. Note that κ_1, κ_2, κ_3 enter the proof of convergence, but they are imposed also by the numerical linear algebra computer code used to solve the equation

$$\psi_{p_i,xx}(x_i)h_{p_i}(x_i) = -\nabla\psi_{p_i}(x_i).$$

The parameter κ_1 should be selected very close to zero to increase the likelihood of a Newton-type search direction; see Algorithm 3.2, Step 1 below. Furthermore, for arbitrary $x \in \mathbb{R}^n$, $\psi_{p,xx}(x)$ can be unbounded as $p \to \infty$, and hence κ_2 and κ_3 are used to bound $\psi_{p,xx}(x)$ from above. The unboundedness can be seen from (11) and (12), which imply that $\nabla\mu_p^{j^*}(x) \to 0$ if $j^* \notin \hat{Q}(x)$, or if j^* is such that $\nabla f^{j^*}(x) = \nabla f^j(x)$ for all $j \in \hat{Q}(x)$, but otherwise $\nabla\mu_p^{j^*}(x)$ is unbounded as $p \to \infty$. In particular, for $x \in \mathbb{R}^n$ such that the cardinality of $\hat{Q}(x)$ is 1, there exists a p making $\psi_{p,xx}(x)$ positive definite whenever $f_{xx}^j(x)$, $j \in \hat{Q}(x)$, is positive definite.

Subroutine 3.1 is used in the same manner as for Algorithm 3.1 to generate the sequence of precision parameters $\{p_i\}_{i=0}^{\infty}$.

Algorithm 3.2. Stabilized Newton-Armijo Algorithm.
Parameters. $\alpha, \beta \in (0,1)$; $p_0 > 0$; $\kappa_1 \in (0,1)$, κ_2, $\kappa_3 \gg 1$.
Data. $x_0 \in \mathbb{R}^n$.

Step 0. Set $i = 0$, $k = 0$ and $\gamma = 1$.

Step 1. Determine if $\psi_{p_i,xx}(x_i)$ is positive definite by computing its Cholesky decomposition R, and compute the reciprocal condition number $c(R)$ of R. If $\psi_{p_i,xx}(x_i)$ is positive definite, $c(R) \geq \kappa_1$ and $p_i \leq \kappa_3$, go to Step 2.
Else, if $\psi_{p_i,xx}(x_i)$ is positive definite, $c(R) \geq \kappa_1$ and the largest eigenvalue $\sigma_{p_i,\max}(x_i)$, of $\psi_{p_i,xx}(x_i)$ satisfies $\sigma_{p_i,\max}(x_i) \leq \kappa_2$ go to Step 2.
Else, go to Step 3.

Step 2. Compute the search direction

$$h_{p_i}(x_i) = -\psi_{p_i,xx}(x_i)^{-1}\nabla\psi_{p_i}(x_i), \tag{32}$$

and go to Step 4.

Step 3. Compute the search direction

$$h_{p_i}(x_i) = -\nabla \psi_{p_i}(x_i). \tag{33}$$

Step 4. Compute the stepsize

$$\lambda_{p_i}(x_i) = \max_{l \in \mathbb{N}} \{\beta^l \,|\, \psi_{p_i}(x_i + \beta^l h_{p_i}(x_i)) - \psi_{p_i}(x_i) \leq \alpha \beta^l \langle \nabla \psi_{p_i}(x_i), h_{p_i}(x_i) \rangle\}. \tag{34}$$

Step 5. Set $x_{i+1} = x_i + \lambda_{p_i}(x_i) h_{p_i}(x_i)$.

Step 6. Enter Subroutine 3.1 and go to Step 1 when exit Subroutine 3.1.

Lemma 3.3. Suppose that Assumption 2.1 holds and that the sequences $\{x_i\}_{i=0}^{\infty}$ and $\{p_i\}_{i=0}^{\infty}$ are generated by Algorithm 3.2 with Subroutine 3.1. Then, the following properties hold: (i) the sequence $\{p_i\}_{i=0}^{\infty}$ is monotonically increasing; (ii) if $\{x_i\}_{i=0}^{\infty}$ has an accumulation point, then $p_i \to \infty$ and $\sum_{i=0}^{\infty} 1/p_i = +\infty$.

Proof. The proof is identical to the one for Lemma 3.1, when the reference to Theorem 1.3.4 is replaced by Theorem 1.4.17. □

Lemma 3.4. Suppose that Assumption 2.1 holds. Then, for every bounded set $S \subset \mathbb{R}^n$ and parameters $\alpha, \beta \in (0,1)$, there exists a $K_S < \infty$ such that, for all $p \geq 1$ and $x \in S$,

$$\psi_p(x - \lambda_p(x) \nabla \psi_p(x)) - \psi_p(x) \leq -\alpha K_S \|\nabla \psi_p(x)\|^2 / p, \tag{35}$$

where $\lambda_p(x)$ is the stepsize of Algorithm 3.2; see (34).

Proof. Let $S \subset \mathbb{R}^n$ be bounded. It follows from Assumption 2.1 and (9) that there exists a constant $M < \infty$ such that

$$\|\nabla \psi_p(x)\| \leq M, \qquad \text{for all } x \in S \text{ and } p > 0.$$

Next, let

$$S_B \triangleq \{x \in \mathbb{R}^n \,|\, \|x - x'\| \leq M, x' \in S\} \tag{36}$$

and let $L < \infty$ be the constant corresponding to S_B such that (8) holds for all $x \in S_B$, $y \in \mathbb{R}^n$ and $p \geq 1$.

For any real $n \times n$ matrix A, let

$$\|A\| \triangleq \sup_{\|v\|=1} \|Av\|$$

be the induced matrix norm. Suppose that A is symmetric. Then,

$$\|A\| = \sigma_{\max},$$

whenever $\sigma_{\max} \geq 0$, where $\sigma_{\max}$ is the largest eigenvalue of A. Now, suppose that $x \in S$ and $p \geq 1$ are such that

$$h_p(x) = -\psi_{p,xx}(x)^{-1} \nabla \psi_p(x).$$

Then,

$$\|h_p(x)\| \leq \|\psi_{p,xx}(x)^{-1}\| \|\nabla \psi_p(x)\| = \|\nabla \psi_p(x)\| / \sigma_{\min}(x,p), \tag{37a}$$

$$\langle \nabla \psi_p(x), h_p(x) \rangle \leq -\|\nabla \psi_p(x)\|^2 / \sigma_{\max}(x,p), \tag{37b}$$

where $\sigma_{\min}(x,p)$ and $\sigma_{\max}(x,p)$ are the smallest and the largest eigenvalue of $\psi_{p,xx}(x)$, respectively. From Steps 1, 2, 3 of Algorithm 3.2 we see that the search direction in (32) is selected only when the reciprocal condition number is greater than κ_1. Hence, there exists a $\kappa_1^* > 0$ such that $\sigma_{\min}(x,p) \geq \kappa_1^*$ for all x and p for which the search direction in (32) is used. Furthermore, we see that the search direction in (32) is selected only when $\sigma_{\max}(x,p) \leq \kappa_2$ or $p \leq \kappa_3$. Since S is bounded, there exists a $\kappa_2^* < \infty$ such that $\kappa_2^* \geq \kappa_2$ and $\sigma_{\max}(x,p) \leq \kappa_2^*$ on $S \times (0, \kappa_2]$. Hence,

$$\|h_p(x)\| \leq \|\nabla \psi_p(x)\| / \kappa_1^*, \tag{38}$$

$$\langle \nabla \psi_p(x), h_p(x) \rangle \leq -\|\nabla \psi_p(x)\|^2 / \kappa_2^*. \tag{39}$$

It follows directly from (33) that (38) and (39) also hold when $h_p(x) = -\nabla \psi_p(x)$.

Next, for all $\lambda \in (0, 1]$, $x \in S$ and $p \geq 1$, by expansion and Lemma 2.2, we have that, for some $s \in [0, 1]$,

$$\begin{aligned}
&\psi_p(x + \lambda h_p(x)) - \psi_p(x) - \alpha\lambda \langle \nabla \psi_p(x), h_p(x) \rangle \\
&= \lambda(1-\alpha) \langle \nabla \psi_p(x), h_p(x) \rangle + \lambda^2 \langle h_p(x), \psi_{p,xx}(x + s\lambda h_p(x)) h_p(x) \rangle / 2 \\
&\leq \lambda(1-\alpha) \langle \nabla \psi_p(x), h_p(x) \rangle + \lambda^2 pL \|h_p(x)\|^2 / 2 \\
&\leq -\lambda \|\nabla \psi_p(x)\|^2 ((1-\alpha)/\kappa_2^* - \lambda pL / (2\kappa_1^{*2})).
\end{aligned} \tag{40a}$$

Let

$$\lambda^* \triangleq \min\{1, 2\kappa_1^{*2}(1-\alpha)/\kappa_2^* pL\}. \tag{40b}$$

Using the same arguments as in the proof of Lemma 3.2, we obtain

$$\lambda_p(x) \geq \beta\lambda^*, \qquad \text{for all } x \in S \text{ and } p \geq 1,$$

and the result follows with

$$K_S = \min\{\beta, 2\beta\kappa_1^{*2}(1-\alpha)/L\kappa_2^*\}.$$

This completes the proof. □

Theorem 3.2. Suppose that Assumption 2.1 holds and that $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence generated by Algorithm 3.2 with Subroutine 3.1. Then, there exist an infinite subset $K \subset \mathbb{N}$ and a $\hat{x} \in \mathbb{R}^n$ such that $x_i \to^K \hat{x}$ and $0 \in \partial\psi(\hat{x})$; i.e., $\hat{x}$ is a stationary point for (P).

Proof. In view of Lemma 3.4, the result follows by the same arguments as in Theorem 3.1. □

4. Numerical Results

This section consists of two parts. First, we examine the effect of different strategies for selecting the precision parameter p on the computational efficiency of Algorithms 3.1 and 3.2. Second, we compare Algorithm 3.1 and 3.2, with Subroutine 3.1, to other algorithms available in the Matlab environment. Most of the examples are known test problems from the literature. However, we have included also results for a specific class of minimax problems, which arises in polynomial interpolation on the sphere.

We happen to know the solutions of most of the test problems. Hence, unless stated specifically, we run the algorithms until the approximating cost function is within the tolerance t of the test problem solution; i.e., we exit the algorithm when

$$|\psi_{p_i}(x_i) - \psi(\hat{x})| \leq t,$$

where $\hat{x} \in \mathbb{R}^n$ is a minimizer of (P). Obviously, this stopping criterion cannot be used in practical problem solving because the problem solution is not known in advance. Nevertheless, we consider this stopping criterion to be the most useful one when comparing the computational efficiency of different algorithms.

The results were computed on a 500 MHz PC running Matlab, Version 6.0 (Release 12). We utilized the Matlab functions rcond and chol to compute the reciprocal condition number and the Cholesky decomposition, respectively (see Algorithm 3.2, Step 1).

4.1. Strategies for Selecting the Precision Parameter. We compare the efficiency of Subroutine 3.1 with alternative strategies for determining

the precision parameter p. A preselected large precision parameter, which is kept constant for all iterations, can be incorporated into Algorithm 3.1 and 3.2 by using the following trivial subroutine.

Subroutine 4.1. Constant Parameter.

Substep 1. Set $p_{i+1} = p_i$, replace i by $i+1$, and exit Subroutine 4.1.

Similarly, a geometric increase of the precision parameter, as in Ref. 43, can be incorporated into Algorithm 3.1 and 3.2 by using the following subroutine.

Subroutine 4.2. Geometric Parameter Growth.
Parameters. $\omega > 1$.

Substep 1. Set $p_{i+1} = \omega p_i$, replace i by $i+1$, and exit Subroutine 4.2.

Note that the proofs of convergence of Algorithm 3.1 and 3.2 do not apply to the use of Subroutine 4.1 and 4.2. However, a proof of convergence of a stabilized Newton-Armijo algorithm for the solution of (P), when using Subroutine 4.2 with $\omega = 2$, can be found in Ref. 43 under a rather restrictive convexity assumption.

Based on the reasoning in the paragraphs following Subroutine 3.1, we set

$$\hat{p} = \log q/t,$$

where t is the tolerance in the objective function of the desired approximate solution. The other algorithm parameters used were

$$\alpha = 0.5, \quad \beta = 0.8, \quad \kappa_1 = 1 \cdot 10^{-7}, \quad \kappa_2 = 1 \cdot 10^{30}, \quad \kappa_3 = 1000\hat{p},$$
$$\tau(p) = 1 \cdot 10^{-4} \qquad \text{for all } p > 0.$$

For $\epsilon_a(\cdot)$ and $\epsilon_b(\cdot)$, we use the two sets of constant values 0.001 and 0.02 [case (A)] and 0.01 and 0.2 [case (B)].

In Subroutine 3.1, Substep 2, we use a simple bisection-type algorithm to find a p^* such that

$$\epsilon_a(p_i) \leq \|\nabla \psi_{p^*}(x_i)\|^2 \leq \epsilon_b(p_i). \tag{40c}$$

For Subroutine 4.1, the constant precision parameter is selected based on the formula

$$p_0 = 2 \log q/t,$$

where t is the tolerance in the objective function of the desired approximate solution. Note that this implies that the tolerance is equal to half the error associated with p_0; see (7). For Subroutine 3.1, p_0 is less important and is selected to be unity.

Table 1. CPU time for Algorithm 3.2 for tolerance level t.

	Subroutine 3.1 (A)	Subroutine 3.1 (B)	Subroutine 4.1	Subroutine 4.2
Ex. 6.1, $t = 10^{-3}$	0.21	0.08	0.15	—
Ex. 6.1, $t = 10^{-5}$	0.27	0.14	0.31	—
Ex. 6.2, $t = 10^{-3}$	0.12	0.06	134	—
Ex. 6.2, $t = 10^{-5}$	0.18	0.09	fail	see Table 2
Ex. 6.3, $t = 10^{-3}$	15.2	19.4	202	—
Ex. 6.3, $t = 10^{-5}$	16.0	20.1	376	see Table 3

Table 2. Example 6.2. CPU time for Algorithm 3.2 with Subroutine 4.2, tolerance $t = 1{\cdot}10^{-5}$.

$p_0\backslash\omega$	1.05	1.1	1.5	2.0	2.5	3.0	4.0	10	50	100	200	500
0.0001	188	95.6	20.7	11.9	8.8	7.2	5.7	3.3	1.7	1.3	1.3	0.93
0.001	164	82.8	14.9	7.0	7.4	3.0	3.7	2.1	**0.10**	0.89	**0.08**	0.47
0.01	142	67.3	17.5	9.1	5.9	5.4	3.8	2.2	0.49	**0.10**	**0.08**	**0.07**
0.1	121	61.0	13.3	7.3	4.2	2.2	2.2	**0.16**	fail	fail	fail	fail
1	102	49.6	10.0	2.3	0.25	fail	fail	fail	fail	fail	fail	fail
10	73.0	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
100	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
1000	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail

In the tables below, the CPU time (i.e., the computational time in seconds before we exit the algorithm) is reported for Examples 6.1, 6.2, and 6.3, described in the Appendix. Note that Tables 2, 3, and 5 show the CPU time as a function of the initial precision parameter p_0 and the multiplication factor ω in Subroutine 4.2. In these tables, the word "fail" means that the particular combination of p_0 and ω resulted in a failure to compute an approximating solution. Typically, this is caused by a rapid increase in the precision parameter to a level that causes strong ill-conditioning or floating-point overflow. Furthermore, the CPU time is in boldface when the result is as good or better than the corresponding worst-case result using Subroutine 3.1; i.e., boldface indicates that Subroutine 4.2 is competitive with Subroutine 3.1.

We see from Tables 1 and 4 that, for both Algorithm 3.1 and 3.2, the adaptive increase of the precision parameter p in Subroutine 3.1 is computationally more efficient than keeping p constant (Subroutine 4.1). This is particularly the case for problems with more than a few functions and/or variables.

The performance of Subroutine 4.2 depends strongly on the selection of the parameters p_0 and ω; see Tables 2–3 and 5. Subroutine 4.2 is competitive with Subroutine 3.1 only when one is lucky enough to have selected

Table 3. Example 6.3. CPU time for Algorithm 3.2 with Subroutine 4.2, tolerance $t = 1 \cdot 10^{-5}$.

$p_0 \backslash \omega$	1.05	1.1	1.5	2.0	2.5	3.0	4.0	10
0.0001	49.1	fail	fail	fail	fail	fail	fail	fail
0.001	**20.1**	fail	fail	fail	fail	fail	fail	fail
0.01	**17.8**	fail	fail	fail	fail	fail	fail	fail
0.1	**16.3**	fail	fail	fail	fail	fail	fail	fail
1	fail	fail	fail	fail	fail	fail	fail	fail
10	**9.8**	fail	fail	fail	fail	fail	fail	fail
100	fail	fail	fail	fail	fail	fail	fail	fail
1000	fail	fail	fail	fail	fail	fail	fail	fail

Table 4. CPU time for Algorithm 3.1, tolerance $t = 10^{-3}$.

	Subroutine 3.1 (A)	Subroutine 3.1 (B)	Subroutine 4.1	Subroutine 4.2
Ex. 6.1	0.36	0.32	0.83	—
Ex. 6.1	0.08	0.08	74.0	see Table 5

Table 5. Example 6.2. CPU time for Algorithm 3.1 with Subroutine 4.2, tolerance $t = 1 \cdot 10^{-3}$.

$p_0 \backslash \omega$	1.05	1.1	1.3	1.5	2.0	2.5	3.0
0.0001	2.7	2.0	0.96	0.65	0.42	fail	fail
0.001	2.6	2.0	0.82	0.58	0.33	fail	fail
0.01	2.0	1.6	0.72	0.49	0.30	fail	fail
0.1	1.9	1.3	0.59	0.43	0.27	fail	fail
1	1.5	0.83	0.49	0.29	0.16	fail	fail
10	0.77	fail	fail	fail	fail	fail	fail
100	fail	fail	fail	fail	fail	fail	fail
1000	fail	fail	fail	fail	fail	fail	fail

a favorable combination of parameters. Our tables show that such combinations are few.

The results using Subroutine 3.1 are moderately dependent on the selection of the parameter functions $\tau(\cdot), \epsilon_a(\cdot), \epsilon_b(\cdot)$. In our implementation, the parameters were selected to be constants that are related to the desired accuracy of the solution of the approximating problems[5]. Hence, efficient values of $\tau(\cdot)$, $\epsilon_a(\cdot)$, $\epsilon_b(\cdot)$ can more easily be selected than efficient values of p_0 and ω in Subroutine 4.2.

[5] In our experiments with interpolation polynomials, in Section 4.2, we found it advantageous to set $\tau(p) = 100/p^2$.

Table 6. CPU time for Algorithm 3.2. (*) BFGS approximation of the Hessian.

Ex.	Algorithm 3.2 (A)	Algorithm 3.2 (B)	Algorithm 3.2 (A)*	Algorithm 3.2 (B)*	fminimax	fmincon	PPP
6.1	**0.27**	**0.14**	**0.36**	**0.04**	0.54	0.74	1.0
6.2	**0.18**	**0.09**	**0.49**	**0.79**	1.71	fail	4.4
6.3	16.0	20.1	**3.9**	**2.8**	4.2	1.8	214
6.4	30.9	35.0	**9.4**	fail	13.5	2.1	1210
6.5	103	107	450	**18.8**	36.1	3.4	7200
6.6	**0.56**	**0.39**	1.38	**0.68**	0.75	0.78	13.3
6.7	1.3	0.97	3.1	1.6	0.91	0.97	89.7
6.8	3.7	2.7	7.2	3.7	1.3	1.4	fail
6.9	**0.29**	**0.26**	**0.80**	**0.79**	1.5	1.5	11.3
6.10	**0.38**	**0.21**	**0.29**	**0.18**	0.59	0.65	2.5
6.11	0.76	**0.34**	**0.59**	**0.29**	0.64	0.73	12.0
6.12	1.3	**0.74**	0.86	**0.50**	0.77	0.80	46.9
6.13	8.4	4.7	4.0	2.7	2.1	2.1	10518
6.14	—	—	**43.4**	**17.4**	48.2	fail	183
6.15	—	—	**141**	**127**	445	fail	2825
6.16	—	—	**5.9**	**6.2**	26.0	fail	18.9
6.17	—	—	**42.1**	**38.7**	125	fail	20.2

Table 7. CPU time for Algorithm 3.1.

Ex.	Algorithm 3.1 (A)	Algorithm 3.1 (B)	fminimax	fmincon	PPP
6.14	**1.7**	**1.7**	48.2	fail	183
6.15	**6.6**	**6.5**	445	fail	2825
6.16	**4.4**	**4.4**	26.0	fail	18.9
6.17	**2.3**	**2.4**	125	fail	20.2

4.2. Comparison with Other Algorithms. We compare Algorithms 3.1 and 3.2 with Subroutine 3.1 to the Pshenichnyi-Pironneau-Polak (PPP) minimax algorithm (Refs. 8, 11, 14), to the Matlab fminimax algorithm (Ref. 45), and to the Matlab SQP algorithm fmincon (Ref. 46) applied to (P′). The solver fminimax uses a SQP algorithm on (P′), while it takes advantage also of the special structure of (P′). In our application of fmincon, we do not take advantage of the structure of the problem. Additionally, fmincon uses a different merit function than fminimax.

First, we carry out a comparison using the artificial test examples described in the appendix. The computational time needed to reach a cost value that is within the tolerance $t = 1 \cdot 10^{-5}$ of the optimal solution is reported in Tables 6 and 7 for a range of different examples and algorithms. In Tables 6 and 7, columns 2–5 and columns 2–3, respectively, contain the CPU time for Algorithm 3.2 with Subroutine 3.1. The boldface CPU times indicate that the performance of Algorithm 3.2 is better than that of fminimax. The word

"fail" indicates that the algorithm jams or experiences extremely slow convergence on the particular example.

In Tables 6 and 7, the use of $\epsilon_a(p) = 0.001$ and $\epsilon_b(p) = 0.02$ for all $p > 0$ is referred to as (A); the use of $\epsilon_a(p) = 0.01$ and $\epsilon_b(p) = 0.2$ for all $p > 0$ is referred to as (B). Additionally, in columns 4 and 5, the Hessian is replaced by its BFGS approximation; see e.g. Ref. 8. In Subroutine 3.1, Substep 2, we use a simple bisection-type algorithm to find a p^* such that

$$\epsilon_a(p_i) \le \|\nabla \psi_{p^*}(x_i)\|^2 \le \epsilon_b(p_i). \tag{40d}$$

The initial precision parameter $p_0 = 1$ throughout. The other algorithm parameters are as given in the previous subsection.

We see from Tables 6 and 7 that Algorithms 3.1 and 3.2 are competitive with the other algorithms. Algorithms 3.1 and 3.2 are particularly efficient on problems with many variables (large n), where fminimax and fmincon are slow due the computational cost of solving the sequence of quadratic subproblems. The dash in Table 6 indicates that the corresponding algorithms were not tested due to the inefficiency of computing Hessians in large-dimensional spaces.

Our second set of test problems emanate from the numerical integration of functions on a sphere, which requires the determination of interpolation points for a class of polynomials on a sphere. The accuracy and efficiency of numerical integration and interpolation schemes on the unit sphere $S^2 \subset \mathbb{R}^3$ depends on the location of such interpolation points. In Refs. 9 and 47, it is shown that the minimum-norm polynomial interpolation points on the sphere can be calculated by solving a semi-infinite minimax problem of the form

$$\min_{x \in \mathbb{R}^n} \max_{\xi \in S^2} \sum_{j \in D} |\ell^j(x, \xi)|, \tag{41}$$

where d is the number of interpolation points, $n = 2d - 3$, and $\ell^j: \mathbb{R}^n \times S^2 \to \mathbb{R}$, $j \in D \triangleq \{1, \ldots, d\}$ are smooth functions.

The space $\mathbb{P}_m$ of all spherical polynomials of degree at most m on S^2 has dimension $d = (m+1)^2$. Let P_m be the Legendre polynomial of degree m defined on $[-1, 1]$, and let

$$J_m(\xi, \zeta) = \sum_{\ell=0}^{m} P_\ell(\langle \xi, \zeta \rangle). \tag{42}$$

A system of points $\xi^j \in S^2$, $j \in D$ is fundamental if the $d \times d$ Gram matrix with elements $G^{ij} = J_m(\xi^i, \xi^j)$ for $i, j \in D$ is nonsingular. The Lagrange polynomials $\ell^j \in \mathbb{P}_m$ can be calculated by

$$\ell(\xi) = G^{-1} g(\xi), \qquad \text{where } g(\xi) = [J_m(\xi^1, \xi), \ldots, J_m(\xi^d, \xi)]^T.$$

Table 8. CPU time for polynomial interpolation Example 6.18.

n	q	Algorithm 3.2	fminimax
29	12	**11.8**	47.4
47	12	**14.6**	17.7
69	12	**27.1**	fail
95	12	47.1	42.0
125	12	**77.9**	350
159	12	**156**	409
197	12	**254**	500
29	42	54.5	43.9
69	42	**92.0**	352
95	42	251	127
125	42	**371**	fail
159	42	**611**	fail

The Lebesgue constant or the norm on the interpolation operator as a map from $C(S^2)$ to $C(S^2)$ is given by $\max_{\xi\in S^2}\sum_{j=1}^{d}|\ell^j(\xi)|$. The Lagrange polynomials ℓ^j depend implicitly on the points $\xi^j, j\in D$ through G and g. Moving to a spherical parametrization of S^2 and using the rotational invariance means that the points $\xi^j\in S^2, j\in D$ can be represented by $n=2d-3$ variables x, giving (41).

The problem in (41) can be solved approximately by discretizing S^2 (an icosahedral grid was used in the experiments). Such approximations of (41) are of the form (1) with the number of functions q determined by the number of points in the discretization and with the number of variables n determined by the degree m of the polynomials. The performance of Algorithm 3.2, with Subroutine 3.1 and a BFGS quasi-Newton approximation to $\psi_{p_i,xx}(x_i)$, is compared to the Matlab fminimax algorithm for different values of n and q. The computational times in seconds before the directional derivative in the search direction is greater than $-1\cdot 10^{-5}$ are reported in Table 8. Numbers in boldface indicate that Algorithm 3.2 is faster than fminimax. The word "fail" indicates that the algorithm jammed on the particular example.

The algorithm parameters used were

$$\alpha=0.1,\quad \beta=0.5,\quad p_0=10\log q,$$

$$\hat{p}=1\cdot 10^{6},\quad \kappa_1=1\cdot 10^{-7},\quad \kappa_2=1\cdot 10^{30},$$

$$\kappa_3=1000\hat{p},\quad \tau(p)=100/p^2,\quad \epsilon_a(p)=\epsilon_b(p)=0.16.$$

In Subroutine 3.1, Substep 2, we used a secant algorithm to approximately find a p^* such that $\|\nabla\psi_{p^*}(x_i)\|^2=0.16$.

We see from Table 8 that Algorithm 3.2, with Subroutine 3.1, is predominantly faster than fminimax for these moderately sized examples. In particular, it appears that fminimax becomes slow and possibly unstable for problems with many variables (i.e., large n).

5. Concluding Remarks

We have developed a feedback precision-adjustment rule for the solution of finite minimax problems using smooth exponential penalty functions and combined it with several unconstrained optimization algorithms. The resulting algorithms solve a sequence of gradually more accurate, smooth, approximating problems, where the accuracy is controlled by the feedback rule in a way that reduces the effect of ill-conditioning. The algorithms are extremely simple to implement. Hence, they are particularly useful for infrequent users who cannot justify the significant amount of time needed to obtain, install, and familiarize oneself with the optimization software that may be available from various sources.

In a series of numerical examples, we have shown that our algorithms are competitive with other smoothing algorithms, based on exponential penalty functions, and with algorithms in the Matlab environment. For the infrequent user, these algorithms are the realistic alternatives to our algorithms, due to the ease with which they can be used. The competitiveness of our algorithms with various alternatives currently implemented in C, FORTRAN, or MATHEMATICA remains to be determined.

6. Appendix: Numerical Examples

Table 9 describes the numerical examples used. Components of vectors are denoted with superscripts, $x = (x^1, x^2, \ldots, x^n)$. When the problem is given in semi-infinite form [e.g. as in (45)], the set Y is discretized with equally spaced points so that the total number of functions is q [e.g., Example 6.3 uses 25 discretization points of Y, for a total of 50 functions].

$$f^1(x) = (x^1)^2 + (x^2)^4, \tag{43a}$$

$$f^2(x) = (2 - x^1)^2 + (2 - x^2)^2, \tag{43b}$$

$$f^3(x) = 2\exp(-x^1 + x^2), \tag{43c}$$

$$f^j(x) = (x^j)^2, \qquad j = \{1, \ldots, 20\}, \tag{44}$$

Table 9. Numerical examples.

Ex.	n	q	$\psi(x)$	Initial point	Solution	Reference
6.1	2	3	(2), (43a, b, c)	(0, 0)	1.952224494	Ref. 39
6.2	20	20	(2), (44)	(0.1, 0.2, . . . , 1, – 1.1, – 1.2, . . . , – 2)	0	Ref. 48
6.3	4	50	(45), (46)	(1, 1, 1, 1)	$2.63664 \cdot 10^{-3}$	Ref. 49
6.4	4	102	(45), (46)	(1, 1, 1, 1)	$2.64954 \cdot 10^{-3}$	Ref. 49
6.5	4	202	(45), (46)	(1, 1, 1, 1)	$2.64954 \cdot 10^{-3}$	Ref. 49
6.6	3	50	(45), (47)	(1, 1, 1)	$4.49977 \cdot 10^{-3}$	Ref. 49
6.7	3	102	(45), (47)	(1, 1, 1)	$4.50481 \cdot 10^{-3}$	Ref. 49
6.8	3	202	(45), (47)	(1, 1, 1)	$4.50481 \cdot 10^{-3}$	Ref. 49
6.9	2	2	(2), (48a, b)	(1.41831, – 4.79462)	0	Ref. 32
6.10	1	25	(49), (50)	5	$1.781609 \cdot 10^{-1}$	Ref. 30
6.11	1	51	(49), (50)	5	$1.783425 \cdot 10^{-1}$	Ref. 30
6.12	1	101	(49), (50)	5	$1.783844 \cdot 10^{-1}$	Ref. 30
6.13	1	501	(49), (50)	5	$1.783942 \cdot 10^{-1}$	Ref. 30
6.14	100	100	(2), (51)	(0.02, 0.04, . . . , 1, –1.02, –1.04, . . . , –2)	0	
6.15	200	200	(2), (52)	(0.01, 0.02, . . . , 1, –1.01, –1.02, . . . , –2)	0	
6.16	100	50	(2), (53)	(0.02, 0.04, . . . , 1, –1.02, –1.04, . . . , –2)	0	
6.17	200	50	(2), (54)	(0.01, 0.02, . . . , 1, –1.01, –1.02, . . . , –2)	0	

$$\psi(x) = \max_{y \in Y} |\phi(x, y)|, \tag{45}$$

$$\phi(x, y) = \sqrt{y} - (x^4 - (x^1(y)^2 + x^2 y + x^3)^2), \quad Y = [0.25, 1] \subset \mathbb{R}, \tag{46}$$

$$\phi(x, y) = \sin y - (x^3(y)^2 + x^2 y + x^1), \quad Y = [0, 1] \subset \mathbb{R}, \tag{47}$$

$$f^1(x) = (x^1 - \sqrt{(x^1)^2 + (x^2)^2}\cos((x^1)^2 + (x^2)^2))^2 + 0.005((x^1)^2 + (x^2)^2), \tag{48a}$$

$$f^2(x) = (x^2 - \sqrt{(x^1)^2 + (x^2)^2}\sin((x^1)^2 + (x^2)^2))^2 + 0.005((x^1)^2 + (x^2)^2), \tag{48b}$$

$$\psi(x) = \max_{y \in Y} \phi(x, y), \tag{49}$$

$$\phi(x, y) = (2(y)^2 - 1)x + y(1 - y)(1 - x), \quad Y = [0, 1] \subset \mathbb{R}, \tag{50}$$

$$f^j(x) = (x^j)^2, \qquad j = \{1, \ldots, 100\}, \tag{51}$$

$$f^j(x) = (x^j)^2, \qquad j = \{1, \ldots, 200\}, \tag{52}$$

$$f^j(x) = (x^{(j-1)2+1})^2 + (x^{2j})^2, \quad j = \{1, \ldots, 50\}, \tag{53}$$

$$f^j(x) = (x^{(j-1)4+1})^2 + (x^{(j-1)4+2})^2 + (x^{(j-1)4+3})^2 + (x^{4j})^2, \quad j = \{1, \ldots, 50\}. \tag{54}$$

References

1. Polak, E., *On the Mathematical Foundations of Nondifferentiable Optimization in Engineering Design*, SIAM Review, Vol. 29, pp. 21–89, 1987.
2. Polak, E., Salcudean, S., and Mayne, D. Q., *Adaptive Control of ARMA Plants Using Worst-Case Design by Semi-Infinite Optimization*, IEEE Transactions on Automatic Control, Vol. 32, pp. 388–397, 1987.
3. Clarke, F. H., Demyanov, V. F., and Giannessi, F., Editors, *Nonsmooth Optimization and Related Topics*, Plenum, New York, NY, 1989.
4. Demyanov, V. F., and Malozemov, V. N., *Introduction to Minimax*, Wiley, New York, NY, 1974.
5. Demyanov, V. F., and Vasilev, V., *Nondifferentiable Optimization*, Springer Verlag, New York, NY, 1986.

6. LEMARECHAL, C., and MIFFLIN, R., Editors, *Nonsmooth Optimization*, Pergamon, Oxford, England, 1978.
7. LEMARECHAL, C., *Nondifferentiable Optimization*, Handbooks in Operations Research and Management Science, Vol. 1: Optimization, Edited by G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, North-Holland, Amsterdam, Netherlands, 1989.
8. POLAK, E., *Optimization: Algorithms and Consistent Approximations*, Springer Verlag, New York, NY, 1997.
9. WOMERSLEY, R. S., *A Continuous Minimax Problem for Calculating Minimum Norm Polynomial Interpolation Points on the Sphere*, ANZIAM Journal, Vol. 42(E), pp. C1536–C1557, 2000 (http://anziamj.autms.org.au/).
10. ZOWE, J., *Nondifferentiable Optimization*, Computational Programming, Edited by K. Schittkowski, Springer Verlag, Berlin, Germany, 1985.
11. PIRONNEAU, O., and POLAK, E., *On the Rate of Convergence of a Certain Methods of Centers*, Mathematical Programming, Vol. 2, pp. 230–258, 1972.
12. POLAK, E., MAYNE, D. Q., and HIGGINS, J., *A Superlinearly Convergent Algorithm for Min-Max Problems*, Journal of Optimization Theory and Applications, Vol. 89, pp. 407–439, 1991.
13. POLAK, E., MAYNE, D. Q., and HIGGINS, J., *On the Extension of Newton's Method to Semi-Infinite Minimax Problems*, SIAM Journal on Control and Optimization, Vol. 30, pp. 376–389, 1992.
14. PSHENICHNYI, B. N., and DANILIN, Y. M., *Numerical Methods in External Problems*, Nauka, Moscow, USSR, 1975.
15. FLETCHER, R., *A Model Algorithm for Composite Nondifferentiable Optimization Problems*, Mathematical Programming Study, Vol. 17, pp. 67–76, 1982.
16. FLETCHER, R., *Second-Order Correction for Nondifferentiable Optimization*. Numerical Analysis, Edited by G.A. Watson, Springer Verlag, Berlin, Germany, pp. 85–114, 1982.
17. HALD, J., and MADSEN, K., *Combined LP and Quasi-Newton Methods for Minimax Optimization*, Mathematical Programming, Vol. 20, pp. 49–62, 1981.
18. MADSEN, A. K., and SCHJAER-JACOBSEN, H., *Linearly Constrained Minimax Optimization*, Mathematical Programming, Vol. 14, pp. 208–223, 1978.
19. YUAN, Y., *On the Superlinear Convergence of a Trust-Region Algorithm for Nonsmooth Optimization*, Mathematical Programming, Vol. 31, pp. 269–285, 1985.
20. CHARALAMBOUS, C., and CONN, A. R., *An Efficient Method to Solve the Minimax Problem Directly*, SIAM Journal on Numerical Analysis, Vol. 17, pp. 162–187, 1978.
21. CONN, A. R., *An Efficient Second-Order Method to Solve the Constrained Minimax Problem*, Report CORR-79-5, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada, 1979.
22. DEMYANOV, V. F., and MALOZEMOV, V. N., *On the Theory of Nonlinear Min-Max Problems*, Russian Mathematical Surveys, Vol. 26, pp. 57–115, 1971.
23. DI PILLO, G., GRIPPO, L., and LUCIDI, S., *A Smooth Method for the Finite Minimax Problem*, Mathematical Programming, Vol. 60, pp. 187–214, 1993.

24. Han, S. P., *Variable Metric Methods for Minimizing a Class of Nondifferentiable Functions*, Mathematical Programming, Vol. 20, pp. 1–13, 1981.
25. Murray, A. W., and Overton, M. L., *A Projected Lagrangian Algorithm for Nonlinear Minimax Optimization*, SIAM Journal on Scientific and Statistical Computing, Vol. 1, pp. 345–370, 1980.
26. Overton, M. L., *Algorithms for Nonlinear* l_1 and l_∞ *Fitting*, Nonlinear Optimization, Edited by M. J. D. Powell, Academic Press, London, England, pp. 213–221, 1982.
27. Vardi, A., *New Minimax Algorithm*, Journal of Optimization Theory and Applications, Vol. 75, pp. 613–634, 1992.
28. Womersley, R. S., *An Algorithm for Composite Nonsmooth Optimization Problems*, Journal of Optimization Theory and Applications, Vol. 48, pp. 493–523, 1986.
29. Zhou, J. L., and Tits, A. L., *Nonmonotone Line Search for Minimax Problems*, Journal of Optimization Theory and Applications, Vol. 76, pp. 455–476, 1993.
30. Zhou, J. L., and Tits, A. L., *An SQP Algorithm for Finely Discretized Continuous Minimax Problems and Other Minimax Problems with Many Objective Functions*, SIAM Journal on Optimization, Vol. 6, pp. 461–487, 1996.
31. Higgins, J., and Polak, E., *An* ϵ*-Active Barrier Function Method for Solving Minimax Problems*, Applied Mathematics and Optimization, Vol. 23, pp. 275–297, 1991.
32. Polak, E., Higgins, J., and Mayne, D. Q., *A Barrier Function Method for Minimax Problems*, Mathematical Programming, Vol. 54, pp. 155–176, 1992.
33. Conn, A. R., and Li, Y., *A Structure-Exploiting Algorithm for Nonlinear Minimax Problems*, SIAM Journal on Optimization, Vol. 2, pp. 242–263, 1992.
34. Bandler, J. W., and Charalambous, C., *Practical Least p*th *Optimization of Networks*, IEEE Transactions on Microwave Theory and Techniques, Vol. 20, pp. 834–840, 1972.
35. Bertsekas, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, 1982.
36. Charalambous, C., *Acceleration of the Least p*th *Algorithm for Minimax Optimization with Engineering Applications*, Mathematical Programming, Vol. 17, pp. 270–297, 1979.
37. Charalambous, C., and Bandler, J. W., *Nonlinear Minimax Optimization as a Sequence of Least p*th *Optimization with Finite Values of p*, International Journal of Systems Sciences, Vol. 7, pp. 377–391, 1976.
38. Gigola, C., and Gomez, S., *A Regularization Method for Solving Finite Convex Min-Max Problems*, SIAM Journal on Numerical Analysis, Vol. 27, pp. 1621–1634, 1990.
39. Li, X., *An Entropy-Based Aggregate Method for Minimax Optimization*, Engineering Optimization, Vol. 18, pp. 277–285, 1997.
40. Mayne, D. Q., and Polak, E., *Nondifferentiable Optimization via Adaptive Smoothing*, Journal of Optimization Theory and Applications, Vol. 43, pp. 601–614, 1984.

41. Polyak, R. A., *Smooth Optimization Method for Minimax Problems*, SIAM Journal on Control and Optimization, Vol. 26, pp. 1274–1286, 1988.
42. Guerra Vazquez, F., Gunzel, H., and Jongen, H. T., *On Logarithmic Smoothing of the Maximum Function*, Annals of Operations Research, Vol. 101, pp. 209–220, 2001.
43. Xu, S., *Smoothing Method for Minimax Problems*, Computational Optimization and Applications, Vol. 20, pp. 267–279, 2001.
44. Zang, I., *A Smoothing Technique for Min-Max Optimization*, Mathematical Programming, Vol. 19, pp. 61–77, 1980.
45. MathWorks, Natick, Massachusetts, *Matlab Version 6.0.0.88*, *Release 12*, *Optimization Toolbox 2.1*, 2001 (www.mathworks.com/access/helpdesk/help/toolbox/optim/fminimax.shtml).
46. MathWorks, Natick, Massachusetts, *Matlab Version 6.0.0.88*, *Release 12*, *Optimization Toolbox 2.1*, 2001 (www.mathworks.com/access/helpdesk/help/toolbox/optim/fmincon.shtml).
47. Womersley, R. S., and Sloan, I. H., *Now Good Can Polynomial Interpolation on the Sphere Be?*, Advances in Computational Mathematics, Vol. 14, pp. 195–226, 2001.
48. Mäkelä, M. M., *Nonsmooth Optimization*, PhD Thesis, University of Jyväskylä, Jyväskylä, Finland, 1990.
49. Oettershagen, K., *Ein Superlinear Konvergenter Algorithmus zur Losung Semi-Infiniter Optimierungsprobleme*, PhD Thesis, University of Bonn, Bonn, Germany, 1982.