JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 71, No. 1, OCTOBER 1991

Discretization Methods for the Solution of Semi-Infinite Programming Problems

R. REEMTSEN[1]

Communicated by E. Polak

Abstract. In the first part of this paper, we prove the convergence of a class of discretization methods for the solution of nonlinear semi-infinite programming problems, which includes known methods for linear problems as special cases. In the second part, we modify and study this type of algorithms for linear problems and suggest a specific method which requires the solution of a quadratic programming problem at each iteration. With this algorithm, satisfactory results can also be obtained for a number of singular problems. We demonstrate the performance of the algorithm by several numerical examples of multivariate Chebyshev approximation problems.

Key Words. Nonlinear programming, semi-infinite programming, discretization of semi-infinite programming problems, Chebyshev approximation.

1. Introduction

In this paper, we consider the following model. We let $A \subseteq \mathbb{R}^n$ be a closed set of parameters and f be a continuous functional on A. We further assume that B is a compact set in $\mathbb{R}^s$ and that $g: A \to C(B)$ is a continuous mapping from A into $C(B)$ where $C(B)$ is equipped with the sup-norm. For each subset $D \subseteq B$, we finally define a set $M(D)$,

$$M(D) := \{a \in A \mid g(a, x) \leq 0, x \in D\}, \tag{1}$$

which is the set of feasible points of the following optimization problem:

$P[D]$: Minimize $f(a)$ over all $a \in M(D)$.

[1]Hochschulassistent (Assistant Professor), Fachbereich Mathematik, Technische Universität Berlin, Berlin, Germany.

0022-3239/91/1000-0085$06.50/0

Assuming $M(D) \neq \emptyset$, we set here

$$\mu(D) := \inf\{f(a) \mid a \in M(D)\}$$

for the minimal value of $P[D]$, and we say that $\hat{a} \in A$ is a solution of $P[D]$ if $f(\hat{a}) = \mu(D)$. Obviously, $P[D]$ is a finite optimization problem for $|D| < \infty$ and a semi-infinite programming problem when $|D| = \infty$. If f and g are affine linear mappings and if A can be described by linear constraints we will speak of the linear case.

For the sake of simplicity, we let here $M(D)$ be defined by a single restriction function $g(a, \cdot)$. With some additional writing effort, the results in this paper can be easily extended to the case where the set of feasible points is given as the intersection of finitely many sets

$$\{a \in A \mid g_j(a, x) \leq 0, x \in D^j\}, \qquad j \in J,$$

where, for each $j \in J$, D^j is a subset of a compact set $B^j \subset \mathbb{R}^{s_j}$ and g_j is a continuous mapping from A into $C(B^j)$. (With regard to the algorithms below, such a representation is also preferable to the one with the single restriction function

$$g(a, x) := \max_{j \in J} g_j(a, x),$$

if $D^j = D$ and $B^j = B$ for all $j \in J$ and some $D \subseteq B \subseteq \mathbb{R}^s$ as in the case of Chebyshev approximation problems. See Section 4.)

We call a method a discretization method for the solution of the semi-infinite programming problem $P[B]$ ($|B| = \infty$ assumed) whenever $P[B]$ is approximately solved by the solution of finite problems $P[D_k]$ with $D_k \subset B$. Discretization methods provide a necessary tool in order to obtain at least a sufficiently good approximate solution of $P[B]$, which in a second phase may be improved by a method with a faster local rate of convergence (see, e.g., Ref. 1). We remark that a solution of a problem $P[D_k]$ may not be an element of $M(B)$ and that, therefore, *a priori* or *a posteriori* manipulations of the problem [resp. this solution] may be necessary in order to guarantee its feasibility for $P[B]$.

For the solution of linear problems, a number of discretization methods have been proposed in the past; see, e.g., Refs. 1–3; for other methods and further references, see, for example, Refs. 4–6. Because of the similar structure of semi-infinite programming and Chebyshev approximation problems, many of the semi-infinite programming methods have been derived from the existing algorithms for such problems. It is a characteristic of those methods that they usually require the solution of a continuous optimization problem in each iteration. This lack has motivated the authors of Refs. 2 and 7 to suggest algorithms which only need the implicit solution of finite problems.

We mention in this connection that, considering the large number of algorithms which has been developed for the solution of semi-infinite programming problems, only relatively few numerical examples of a realistic size can be found in the literature.

In Section 2, we will transfer the algorithm of Ref. 7 to the solution of general semi-infinite programming problems. In this way, we can ensure in particular that also nonlinear semi-infinite programming problems can be solved by discretization techniques. However, in the nonlinear situation, the assumption for this discretization result is relatively strong and often difficult to verify in practice. Therefore, we show further in Section 2 how the requested convergence properties can be enforced by additional constraints, provided that some *a priori* knowledge about the solution of the problem is available.

We continue in Section 3 with modifying the algorithm of Section 2 and providing a class of algorithms for the solution of linear problems in this way. We further suggest there a particular algorithm which has proved to be efficient in practice. A new feature of this algorithm is that it employs solutions of certain quadratic programming problems. Beside other consequences, this fact permits also a satisfactory solution of many singular problems without the application of a second locally convergent method. We conclude the paper in Section 4 by applying this algorithm to a number of multivariate Chebyshev approximation problems.

2. Discretization Method

Throughout this section, we make the following two assumptions. For that, we let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ and

$$L(a_0, D) := \{a \in A \mid f(a) \leq f(a_0)\} \cap M(D)$$

be the level set related to $a_0 \in A$ and $D \subseteq B$.

Assumption A1. $\{B_k\}$ is a sequence of compact subsets of B with $B_k \subseteq B_{k+1}$ for all $k \in \mathbb{N}_0$ for which $\lim_{k\to\infty} h(B_k, B) = 0$, where

$$h(B_k, B) := \sup_{x \in B} \inf_{y \in B_k} \|x - y\|_\infty \tag{2}$$

and $\|\cdot\|_\infty$ is the maximum norm in $\mathbb{R}^s$.

Assumption A2. $M(B)$ is nonempty, and there is an $a_0 \in M(B)$ such that the level set $L(a_0, B_0)$ is bounded and hence compact in $\mathbb{R}^n$.

The sets B_k will usually be finite sets, in which case we will also speak of grids. Concerning A2, we note that, for the linear case, a number of equivalent conditions can be found in Ref. 1. For linear problems, Assumption A2 implies that the restriction matrix of the linear program $P[B_0]$ has rank n provided that $n \leq |B_0| < \infty$. For linear Chebyshev approximation problems, also the converse of this statement holds true; see also Ref. 7.

We consider now the following algorithm.

Algorithm 2.1.

Step 0. Set $D_0 := B_0$ and $k := 0$.

Step 1. Find a solution $\hat{a}_k \in M(D_k)$ of $P[D_k]$.

Step 2. Choose $D_{k+1} \subseteq B_{k+1}$ with $D_{k+1} \supseteq D_k \cup \{x_k\}$, where $x_k \in B_{k+1}$ is a point such that

$$g(\hat{a}_k, x_k) = \max_{x \in B_{k+1}} g(\hat{a}_k, x).$$

Step 3. Substitute $k+1$ for k, and go to Step 1.

In analogy to Ref. 7, we can prove now that this algorithm can be carried out successfully and that it converges.

Theorem 2.1. Under Assumptions A1 and A2, the following holds true for Algorithm 2.1:

(a) $P[B]$ and $P[D_k]$, $k \in \mathbb{N}_0$, possess solutions $\hat{a} \in M(B)$ and $\hat{a}_k \in M(D_k)$, respectively.

(b) $\mu(D_k) \leq \mu(D_{k+1}) \leq \mu(B)$, $k \in \mathbb{N}_0$, and

$$\lim_{k \to \infty} \mu(D_k) = \mu(B).$$

(c) Each sequence $\{\hat{a}_k\}$ of solutions has at least one accumulation point, and each accumulation point of $\{\hat{a}_k\}$ solves $P[B]$.

(d) If $\hat{a} \in M(B)$ is the unique solution of $P[B]$, then

$$\lim_{k \to \infty} \|\hat{a} - \hat{a}_k\|_2 = 0,$$

where $\|\cdot\|_2$ is the Euclidean norm in $\mathbb{R}^n$.

Proof. We first note that, due to A1, we have $D_k \subseteq B_{k+1}$, and hence $D_k \cup \{x_k\} \subseteq B_{k+1}$. From $D_0 \subseteq D_k \subseteq B$ for all $k \in \mathbb{N}$, we now conclude that

$$\{a_0\} \subseteq L(a_0, B) \subseteq L(a_0, D_k) \subseteq L(a_0, D_0), \tag{3}$$

which by A2 implies statement (a). Because of $D_k \subseteq D_{k+1}$, for $k \in \mathbb{N}_0$, we arrive next at

$$f(\hat{a}_0) \leq f(\hat{a}_k) \leq f(\hat{a}_{k+1}) \leq f(\hat{a}) \leq f(a_0), \qquad k \in \mathbb{N}. \tag{4}$$

From (3), we further get that $\hat{a}_k$ is an element of $L(a_0, D_0)$ for all $k \in \mathbb{N}_0$. Hence, there exists a subsequence $\{\hat{a}_{k(i)}\}$ of $\{\hat{a}_k\}$ which converges to an element $\tilde{a} \in L(a_0, D_0)$. From (4), we know that $f(\tilde{a}) \leq f(\hat{a})$. Thus, if we can show that $\tilde{a}$ belongs to $M(B)$, we have $f(\tilde{a}) = f(\hat{a})$, which implies (b) and (c). Then, (d) follows by a standard argument.

In order to prove that $\tilde{a} \in M(B)$, we first let

$$\|h\|_D := \sup\{|h(x)| \mid x \in D\}, \qquad D \subseteq B, \tag{5}$$

and

$$\omega(h, \delta) := \sup\{|h(x) - h(y)| \mid \|x - y\|_\infty \leq \delta, x, y \in B\},$$

for $h \in C(B)$. We further define

$$\Delta(\delta) := \sup\{\|g(a) - g(b)\|_B \mid \|a - b\|_\infty \leq \delta, a, b \in L(a_0, D_0)\},$$

$$\Omega(\delta) := \sup\{\omega(g(a), \delta) \mid a \in L(a_0, D_0)\},$$

where from A2, we can infer that $\Delta(\delta) \to 0$ and $\Omega(\delta) \to 0$ for $\delta \to 0$. Hence, assuming that

$$\Phi := \max_{x \in B} g(\tilde{a}, x) > 0,$$

we can fix $\delta > 0$ such that $\Omega(\delta) + 3\Delta(\delta) < \Phi$.

For δ, there exists now a number $i_0 \in \mathbb{N}$ such that

$$\|\hat{a}_{k(i)} - \tilde{a}\|_2 \leq \delta \quad \text{and} \quad h(B_{k(i)}, B) \leq \delta$$

hold true for all $i \geq i_0$. Therefore, for each $a \in L(a_0, D_0)$, we can find an element $\hat{x} \in B$ with

$$g(a, \hat{x}) = \max\{|g(a, x)| \mid x \in B\}$$

and an element $x_i \in B_{k(i)}$ with $\|\hat{x} - x_i\|_\infty \leq \delta$, so that

$$\max_{x \in B} g(a, x) - \max_{x \in B_{k(i)}} g(a, x) \leq g(a, \hat{x}) - g(a, x_i) \leq \Omega(\delta), \tag{6}$$

for all $i \geq i_0$. Because of $D_{k(i)+1} \subseteq D_{k(i+1)}$, we can further conclude that, for each $i \geq i_0$,

$$\begin{aligned}
&\max_{x \in B_{k(i)}} g(\hat{a}_{k(i)}, x) \leq \max_{x \in B_{k(i)+1}} g(\hat{a}_{k(i)}, x) \\
&= \max_{x \in D_{k(i)+1}} g(\hat{a}_{k(i)}, x) \leq \max_{x \in D_{k(i+1)}} g(\hat{a}_{k(i)}, x) \\
&= \max_{x \in D_{k(i+1)}} [g(\hat{a}_{k(i+1)}, x) + (g(\hat{a}_{k(i)}, x) - g(\hat{a}_{k(i+1)}, x))] \\
&\leq \max_{x \in B} [g(\hat{a}_{k(i)}, x) - g(\hat{a}_{k(i+1)}, x)] \leq 2\Delta(\delta), \qquad (7)
\end{aligned}$$

where we used the fact that $\hat{a}_{k(i+1)}$ is in $M(D_{k(i+1)})$. Then, (6) together with (7) leads to the contradiction

$$\Phi = \max_{x \in B} g(\tilde{a}, x) \leq \max_{x \in B} g(\hat{a}_{k(i)}, x) + \Delta(\delta) \leq \Omega(\delta) + 3\Delta(\delta) < \Phi. \quad \square$$

Concerning a stopping criterion for Algorithm 2.1, we refer to the criteria given in Ref. 7, which can be transferred to the situation considered here.

Remark 2.1. In this general form, Algorithm 2.1 contains known methods for the solution of linear problems (see Ref. 1) as special cases and generalizes them to the nonlinear situation. These are (i) the full-grid method, where $D_k := B_k$ for all $k \in \mathbb{N}_0$; and (ii) the exchange algorithm, where $B_{k+1} := B$ for all $k \in \mathbb{N}$.

In practice, it may be difficult to verify A2, or A2 may not be fulfilled at all (cf. Ref. 7). In such a case, the convergence of Algorithm 2.1 can be enforced by a regularization of the problem, provided that some *a priori* information on the problem is available (see also Ref. 8).

Theorem 2.2. Let $P[B]$ have a solution $\hat{a} \in M(B)$, and let a constant C be known such that $\|\hat{a}\|_\infty \leq C$. Let

$$M_C(D_k) := M(D_k) \cap \{a \in \mathbb{R}^n \mid \|a\|_\infty \leq C\},$$

and let $P_c(D_k)$ be the regularized optimization problem

$$P_C(D_k): \text{minimize } f(a) \text{ over all } a \in M_C(D_k).$$

Let further Step 1 of Algorithm 2.1 be exchanged for

Step 1'. Find a solution $\hat{a}_k \in M_c(D_k)$ of $P_c(D_k)$.

Then, Theorem 2.1 remains true without A2 if $M(D_k)$ and $P[D_k]$ in (a) are replaced by $M_c(D_k)$ and $P_c(D_k)$ and if here

$$\mu(D_k) := \inf\{f(a) \mid a \in M_c(D_k)\}.$$

Proof. Since A in our model is an arbitrary closed subset of $\mathbb{R}^n$, we can replace A everywhere by the set

$$\bar{A} = \{a \in A \mid \|a\|_\infty \leq C\}.$$

Then for the set $\bar{A}$, Assumption A2 is satisfied with $a_0 := \hat{a}$, so that Theorem 2.2 is a consequence of Theorem 2.1 with $\bar{A}$. For, $\hat{a}$ also solves $P_c(B)$ and, moreover, every solution of $P_c(B)$ is also a solution of $P[B]$. □

3. Linear Case

In this section, we generally let

$$f(a) := c^T a, \qquad g(a, x) := a^T v(x) - b(x),$$

with given $c \in \mathbb{R}^n$ and continuous mappings $v: B \to \mathbb{R}^n$ and $b: B \to \mathbb{R}$. We remark that, in this way, we can also cover linear constraints, which do not depend on x if we associate them formally with isolated points in B. Thus, for a linear model, we can set here $A = \mathbb{R}^n$.

It is well-known that, under appropriate assumptions, a solution of $P[B]$ is characterized by $m \leq n$ points of B (cf. Ref. 1). Thus, discretization methods may be understood as methods which consecutively determine better approximations of such points with progressing discretization. So, in the ideal case, only $m \leq n$ points of B_k are needed in each iteration of Algorithm 2.1; and indeed for special situations (Refs. 1, 7) and also for quite general circumstances (Ref. 3), algorithms have been suggested where $|D_k| = n$ for all $k \in \mathbb{N}_0$. In the latter case, the rules for determining D_{k+1} from D_k are rather complicated, and little is known about the quality of these algorithms. Hence, one obvious disadvantage of Algorithm 2.1 is the monotonic growth of the D_k's, $k = 0, 1, \ldots$, since in this way a number of useless points may have to be dragged along within the iteration process. In the following, we show that, for linear problems, Algorithm 2.1 can be modified in such a way that the used sets $D_k \subseteq B_k$ do not necessarily increase at each iteration.

For $D \subseteq B$ and $a \in \mathbb{R}^n$ arbitrary, we let

$$E(a, D) := \{x \in D \mid g(a, x) = 0\},$$

and define [$M(D) \neq \varnothing$ is assumed]

$$S(D) := \{a \in M(D) \mid c^T a = \mu(D)\}$$

to be the set of solutions of $P[D]$. For each $\hat{a} \in S(D)$ with $|D| < \infty$, we then have, as is well known,

$$\mu(D) = \inf_{a \in M(D)} c^T a = \inf_{a \in M(E(\hat{a}, D))} c^T a. \tag{8}$$

Note that (8) can be proved with the Kuhn–Tucker theorem for problems with linear constraints (e.g., Ref. 9). Hence, if $\hat{a}_k$ is a solution of $P[D_k]$, for any set $D_{k+1} \subseteq B_{k+1}$ with $D_{k+1} \supseteq E(\hat{a}_k, D_k)$, we reach $\mu(D_k) \leq \mu(D_{k+1})$. Note that $E(\hat{a}_k, D_k) \subseteq D_k \subseteq B_k \subseteq B_{k+1}$. Finally, we make the following assumption.

Assumption A3. $\{G_i\}$ is a sequence of grids in B with $\lim_{i \to \infty} h(G_i, B) = 0$ [cf. (2)] and G_i a proper subset of G_{i+1} for each $i \in \mathbb{N}_0$.

With these preliminaries, we can set up the following class of algorithms.

Algorithm 3.1.

Step 0. Choose $\{\tilde{\delta}_i\}$ with $\tilde{\delta}_i \geq 0$, $i \in \mathbb{N}$, and $\lim_{i \to \infty} \tilde{\delta}_i = 0$ and set $D_0 := G_0$, $k := 0$ and $i := 1$.

Step 1. Determine a solution $\hat{a}_k \in \mathbb{R}^n$ of $P[D_k]$.

Step 2. Set $B_{k+1} := G_i$ and $\delta_{k+1} := \tilde{\delta}_i$. If there is an $x_k \in B_{k+1}$ so that $g(\hat{a}_k, x_k) > \delta_{k+1}$, choose $D_{k+1} \subseteq B_{k+1}$ with $D_{k+1} \supseteq E(\hat{a}_k, D_k) \cup \{x_k\}$. Else, set $\tilde{a}_i := \hat{a}_k$, $i := i+1$ and repeat this step (and so redefine δ_{k+1}, B_{k+1}, and D_{k+1}).

Step 3. Set $k := k+1$, and go to Step 1.

For this general class of algorithms, we can prove now the following theorem.

Theorem 3.1. Let A2 and A3 be satisfied for $B_0 := G_0$, and in Algorithm 3.1 let a strategy be inserted which guarantees the following:

(S) For a fixed grid index i of G_i, the case $c^T \hat{a}_k = c^T \hat{a}_{k-1}$ can occur only for finitely many $k \in \mathbb{N}$ in succession.

Then, statements (a) and (b) of Theorem 2.1 remain valid for $\{\hat{a}_k\}$ generated by Algorithm 3.1. Further, (c) and (d) of Theorem 2.1 hold true for $\{\tilde{a}_i\}$ (instead of $\{\hat{a}_k\}$ there).

Proof. By A2, problem $P[D_0]$ has a solution, and by our remarks above we have $\mu(D_0) \leq \mu(D_1)$. Because of $M(B) \subseteq M(D_1)$, we further have $M(D_1) \neq \varnothing$ so that also $P[D_1]$ possesses a solution. In this way, we can obtain $S(D_k) \neq \varnothing$ for each $k \in \mathbb{N}$ and the monotonicity of $\{\mu(D_k)\}$. $S(B) \neq \varnothing$ is a consequence of A2, since $L(a_0, B) \subseteq L(a_0, B_0)$.

Let now

$$\delta = \sup_{i \in \mathbf{N}} \tilde{\delta}_i.$$

(For $\delta = 0$, the remainder of the proof could be immediately concluded from Theorem 2.1 by setting $D_k := G_k$ there, since due to (S) problem $P[G_i]$ is solved after finitely many iterations.) Then, we next show that, with A2,

$$L_\delta(a_0, B_0) := \{a \in \mathbb{R}^n \mid c^T a \leq c^T a_0\} \cap M_\delta(B_0)$$

is also bounded, where

$$M_\delta(B_0) := \{a \in \mathbb{R}^n \mid a^T v(x) - b(x) \leq \delta,\ x \in B_0\}.$$

If there would exist a sequence $\{a_j\}$ in $L_\delta(a_0, B_0)$ such that $\|a_j\|_\infty \to \infty$ for $j \to \infty$, then we could set $\zeta_j := a_j - a_0$ and assume without loss of generality the existence of a $\zeta \in \mathbb{R}^n$ such that $\zeta_j / \|\zeta_j\|_\infty \to \zeta$ for $j \to \infty$. From the definition of $L_\delta(a_0, B_0)$, we could further conclude that ζ satisfies

$$c^T \zeta \leq 0, \qquad \zeta^T v(x) \leq 0, \quad \forall x \in B_0. \tag{9}$$

(9), however, cannot be true for any $\zeta \in \mathbb{R}^n$, since it implies $a_0 + r\zeta \in L(a_0, B_0)$ for each $r \geq 0$ and hence contradicts A2.

If now $k := k(i)$ is that index for which $\tilde{a}_i := \hat{a}_k$, then because of (S) and the monotonicity of $\{\mu(D_k)\}$, the problem $P[D_{k(i)}]$ is solved after finitely many iterations since G_i has only finitely many pairwise distinct subsets. Using A3 and the fact that $\max\{g(\hat{a}_{k(i)}, x) \mid x \in G_i\} \leq \tilde{\delta}_i$, we further know that $L_\delta(a_0, D_{k(i)}) \subseteq L_\delta(a_0, D_0)$. Thus, since $\tilde{a}_i \in L_\delta(a_0, D_{k(i)})$, the sequence $\{\tilde{a}_i\}$ lies in the compact set $L_\delta(a_0, D_0)$. With little modifications, we can follow now the proof of Theorem 2.1 for $\tilde{a}_i$ and derive the requested results concerning $\{\tilde{a}_i\}$. In addition, we get $\mu(D_{k(i)}) \to \mu(B)$ and hence $\mu(D_k) \to \mu(B)$ for $k \to \infty$, since $\{\mu(D_k)\}$ is monotonic. □

By the following lemma, a strategy ensuring (S) only applies if $P[D_k]$ in Algorithm 3.1 has more than one solution.

Lemma 3.1. If in Algorithm 3.1 $P[D_k]$ has a unique solution $\hat{a}_k \in \mathbb{R}^n$ and if $g(\hat{a}_k, x_k) > 0$, then for each $\hat{a}_{k+1} \in S(D_{k+1})$ we have $c^T \hat{a}_{k+1} > c^T \hat{a}_k$.

Proof. Since $S(D_k) = \{\hat{a}_k\}$, we also have $S(E(\hat{a}_k, D_k)) = \{\hat{a}_k\}$. For, we know that $S(E(\hat{a}_k, D_k)) \supseteq S(D_k)$ [cf. (8)] and that further for $\hat{a}_k$,

$\tilde{a}_k \in S(E(\hat{a}_k, D_k))$ and $\lambda \in [0, 1]$ also $a_\lambda = \lambda \tilde{a}_k + (1-\lambda)\hat{a}_k$ is in $S(E(\hat{a}_k, D_k))$; so, for $\lambda > 0$ sufficiently small, a_λ is in $M(D_k)$ and by (8) also in $S(D_k)$. Using now the fact that each $\hat{a}_{k+1} \in S(D_{k+1})$ is in $M(E(\hat{a}_k, D_k))$ and that $\hat{a}_{k+1} \neq \hat{a}_k$ because of $g(\hat{a}_{k+1}, x_k) \leq 0$, we arrive at $c^T \hat{a}_k < c^T \hat{a}_{k+1}$ by employing once more (8). □

The algorithms in Refs. 2 and 7 are of the type of Algorithm 3.1. In particular, they employ the following rule securing (S):

(S1) If $c^T \hat{a}_k = c^T \hat{a}_{k-1}$ for a $k \in \mathbb{N}$, choose $D_{k+1} \subseteq B_{k+1}$ not as in Step 2, but so that $D_{k+1} \supseteq D_k \cup \{x_k\}$.

Obviously, (S1) can be passed through only finitely often in succession for $B_{k+1} = G_i$ with $i \in \mathbb{N}$ fixed, since D_k is increased by at least one point in each iteration.

If $P[D_k]$ has more than one solution, or if this seems to be the case at least numerically (see below), a solution $\hat{a}_{k+1}$ of $P[D_{k+1}]$ may be computed which is far away from $\hat{a}_k$. By our experiments, the convergence of Algorithm 3.1 can be considerably accelerated if $\hat{a}_k$ is selected so that

$$\|\hat{a}_k\|_2 := \min_{a \in S(D_k)} \|a\|_2, \qquad k \geq 0, \tag{10}$$

or

$$\|\hat{a}_k - \hat{a}_{k-1}\|_2 := \min_{a \in S(D_k)} \|a - \hat{a}_{k-1}\|_2, \qquad k \geq 1, \tag{11}$$

where (11) in general yielded the least number of iterations in order to determine, for example, $P[G_l]$ for a given $l \in \mathbb{N}$ (set $\tilde{\delta}_l = 0$ then). The solution $\hat{a}_k$ of (10) [resp. (11)] is uniquely determined since $S(D_k)$ is a nonempty, closed, and convex set. If $\tilde{a}_k$ is any solution of $P[D_k]$ (e.g., computed by the simplex algorithm), then $\mu(D_k) = f(\tilde{a}_k)$ and the minimization problems in (10) and (11) become quadratic optimization problems with linear constraints. The additional numerical effort to compute (10) or (11) is relatively small, since $\tilde{a}_k$ is a feasible point of $S(D_k)$ which in the case of the unique solvability of $P[D_k]$ is equal to $\hat{a}_k$.

The use of (10) or (11) has some side effects. We can first prove the following lemma.

Lemma 3.2. If $\hat{a}_k \in S(D_k)$ in Step 1 of Algorithm 3.1 is given by (10), then (S) is satisfied.

Proof. Let $c^T \hat{a}_k = c^T \hat{a}_{k-1}$ for some $k \in \mathbb{N}$. From $g(\hat{a}_{k-1}, x_{k-1}) > 0$, $D_k \supseteq \{x_{k-1}\}$ and $g(\hat{a}_k, x) \leq 0$ for all $x \in D_k$, we first conclude that $\hat{a}_k \neq \hat{a}_{k-1}$. Next, we observe that $\hat{a}_k$ is in $M(D_k) \subseteq M(E(\hat{a}_{k-1}, D_{k-1}))$ and that

$g(\hat{a}_{k-1}, x) < 0$ for $x \in D_{k-1} \setminus E(\hat{a}_{k-1}, D_{k-1})$. Hence, we can find $\lambda \in (0, 1)$ such that $a_\lambda := \lambda \hat{a}_k + (1-\lambda)\hat{a}_{k-1}$ is in $M(D_{k-1})$. Since $c^T a_\lambda = c^T \hat{a}_{k-1}$, we know that a_λ also belongs to $S(D_{k-1})$. Therefore, observing that $a_\lambda \neq \hat{a}_{k-1}$, we arrive at

$$\|\hat{a}_{k-1}\|_2 < \|a_\lambda\|_2 \leq \lambda \|\hat{a}_k\|_2 + (1-\lambda)\|\hat{a}_{k-1}\|_2,$$

which implies $\|\hat{a}_{k-1}\|_2 < \|\hat{a}_k\|_2$. This guarantees (S), since G_i possesses only finitely many pairwise distinct subsets so that, for i fixed, there can exist only finitely many pairwise distinct elements $\hat{a}_k$, $k = 1, 2, \ldots$, satisfying (10). □

We do not know whether Lemma 3.2 still holds true if (10) there is replaced by (11). In practice, however, application of (11) led to the fastest convergence, and a cycling in Algorithm 3.1 never occurred.

The determination of $\hat{a}_k$ by (10) or (11) has another consequence. In case Algorithm 3.1 is combined with (S1), it generates sets D_k, where $D_k \supseteq E(\hat{a}_{k-1}, D_{k-1})$ for all $k \in \mathbb{N}$. Thus, if the simplex algorithm is employed in order to solve $P[D_k]$ for all $k \in \mathbb{N}_0$, we can conclude that, for each $k \in \mathbb{N}$, the restriction matrix of this problem has rank n; therefore, at least n of the constraints are active in the obtained solutions $\hat{a}_k$, $k \in \mathbb{N}$. (For, by A2, the rank of the restriction matrix belonging to D_0 is n, so that the simplex algorithm finds a solution $\hat{a}_0$ of $P[D_0]$ for which $|E(\hat{a}_0, D_0)| \geq n$ and for which the matrix connected with $E(\hat{a}_0, D_0)$ has rank n. Because of $D_k \supseteq E(\hat{a}_{k-1}, D_{k-1})$, the same holds true for all following $\hat{a}_k$, $k \in \mathbb{N}$.) So, if $P[B]$ possesses a solution (which is the case under A2) and if in each solution of $P[B]$ there are less than n constraints active (we call such problems singular), then the linear programming problems $P[D_k]$ become increasingly ill-conditioned with growing k, i.e., with progressing discretization of B. This phenomenon has been repeatedly described and has served as one motivation for the development of so-called two phase algorithms (cf., e.g., the example given in Ref. 10 [resp. Ref. 1, p. 158]). The singular situation, for example, almost always arises in Chebyshev approximation problems on two-dimensional and higher-dimensional regions B.

In case Algorithm 3.1 is employed, then at the beginning of the iteration process, i.e., for some small k, the problems $P[D_k]$ can normally be solved without any numerical difficulties whether $P[B]$ is singular or not. If now in some later iteration $P[D_k]$ does not have a unique solution or if this seems at least to be the case numerically due to the ill-conditioning of the problem (i.e., of the matrices related to optimal vertices), the use of solutions (10) or (11) obviously has some stabilizing effect on the choice of the solutions $\hat{a}_k$ and hence of the sets D_k. Furthermore, in this situation $\hat{a}_k$ as in (10) or (11) is usually not attained anymore at a vertex of the feasible region $M(D_k)$, so that less than n constraints can become active in $\hat{a}_k$ and the restriction matrix

generated by D_{k+1} typically has a rank smaller than n. Note that the simplex algorithm can still be employed then to find a solution of $P[D_{k+1}]$. Thus, by using (10) or (11), Algorithm 3.1 may be freed from the necessity to work with vertex solutions in the aforementioned situation. Nevertheless, also then the number of active constraints in solutions $\hat{a}_k$ of the discrete problems may still be larger than the number of active constraints in the accumulation points of $\{\hat{a}_k\}$. However, application of (10) [resp. (11)] can only improve the situation, and indeed Algorithm 3.1 (with the specifications given at the end of this section) has produced satisfying numerical results also for many singular problems.

If (S1) is used and if, for each $k \in \mathbb{N}_0$, $\hat{a}_k$ is obtained by application of the simplex algorithm to the dual problem of $P[D_k]$, then because of $D_{k+1} \supseteq E(\hat{a}_k, D_k)$ the (dual) vertex solution of $P[D_k]$ can be taken as a starting vertex for the solution of $P[D_{k+1}]$ (cf. also Ref. 2). This is not possible anymore if $\hat{a}_k$ is given by (10) or (11) and not achieved at a vertex. (For a number of problems, e.g., Chebyshev approximation problems, it is easy to provide at least a feasible point for the primal problem $P[D_{k+1}]$, which in general is close to $\hat{a}_k$.) In this regard it has to be noted, however, that the total computing time for the solution of the $P[D_k]$'s, $k = 0, 1, \ldots$, is normally far exceeded by the computing time which is required for the evaluation of g on the grids G_i, $i = 0, 1, \ldots$, in Step 2 of Algorithm 3.1. (Obviously, g has to be computed at least once on the total grid G_i for each $i \in \mathbb{N}_0$.) Therefore, the faster convergence and the gain of stability which is reached by the use of (10) or (11), if the discrete problems are not uniquely solvable, should be valued higher than the possibility of obtaining good starting points for these problems.

The rate of convergence of Algorithm 3.1 depends also strongly on the choice of D_{k+1} in Step 2. Since there are no theoretical results available, the choice of the D_k's becomes a matter of intuition and experiments. Supported by our numerical results, we suggest here the following specification of Step 2, where $\tilde{\epsilon}_i \in (0, 1)$, $i \in \mathbb{N}$, are given numbers.

Step 2″. Set $B_{k+1} := G_i$, $\epsilon_{k+1} := \tilde{\epsilon}_i$, and $\delta_{k+1} := \tilde{\delta}_i$ and determine

$$D_{k+1} := \left\{ x \in B_{k+1} \mid g(\hat{a}_k, x) \geq \epsilon_{k+1} \min_{x \in D_k} g(\hat{a}_k, x) \right\}, \tag{12}$$

together with some $x_k \in B_{k+1}$ for which

$$g(\hat{a}_k, x_k) = \max_{x \in B_{k+1}} g(\hat{a}_k, x).$$

In case $g(\hat{a}_k, x_k) \leq \delta_{k+1}$, set $\tilde{a}_i := \hat{a}_k$, $i := i+1$, and repeat this step.

D_{k+1} [Eq. (12)] includes all $x \in B_{k+1}$ for which g is active and for which g is possibly violated in $\hat{a}_k$. If $\min\{g(\hat{a}_k, x) \mid x \in D_k\} < 0$, it contains also a certain number of points for which g is inactive, but almost active, in $\hat{a}_k$. Concerning the choice of the numbers $\tilde{\epsilon}_i$, we refer to Section 4. A general rule for the selection of the $\tilde{\epsilon}_i$ is that they should be all the larger the farther $\hat{a}_k$ is away from a solution of $P[B]$.

The determination of D_{k+1} [Eq. (12)] requires the computation of g on one total grid B_{k+1}. Note that, because of $G_i \subseteq G_{i+1}$, this is also true for the case where i is increased and Step 2″ is repeated. In order to reduce the numerical effort, (12) may alternatively be replaced by the intersection of (12) and neighborhoods of the active points $x \in E(\hat{a}_k, D_k)$ as long as an element x_k is found in that set which satisfies $g(\hat{a}_k, x_k) > \tilde{\delta}_i$. Such a strategy, however, complicates the programming code for the algorithm considerably.

By our observations, the number of points in (12) usually grows when i is increased and then diminishes again gradually; i.e., other than in Algorithm 2.1, the D_k's in Algorithm 3.1 with Step 2″ in general do not grow monotonically. In case a set D_{k+1} becomes too large for numerical purposes, it may be replaced by an arbitrary subset which preserves the convergence of the algorithm, i.e., which has the properties of D_{k+1} in Step 2 of Algorithm 3.1.

4. Application to the Chebyshev Approximation Problem

If $z: B \to \mathbb{R}^{n-1}$ and $r: B \to \mathbb{R}$ are given mappings and if $\|\cdot\|_D$ is defined by (5), then the linear Chebyshev approximation problem on B is as follows:

$$\text{minimize } \|r - a^T z\|_B, \text{ over all } a \in \mathbb{R}^{n-1}. \tag{13}$$

We let now $a := (a_1, \ldots, a_n)^T \in \mathbb{R}^n$, $f(a) := c^T a$, with $c := (0, \ldots, 0, 1)^T \in \mathbb{R}^n$ and

$$g_1(a, x) := \sum_{i=1}^{n-1} a_i z_i(x) - r(x) - a_n,$$

$$g_2(a, x) := -\sum_{i=1}^{n-1} a_i z_i(x) + r(x) - a_n,$$

for $(a, x) \in \mathbb{R}^n \times B$. Associating D with sets $D^1 \subseteq B$ and $D^2 \subseteq B$, we define here further

$$M(D) := \bigcap_{j=1,2} \{a \in A \mid g_j(a, x) \leq 0, x \in D^j\}.$$

We can use now former definitions with these specifications. In particular, by these agreements, a linear semi-infinite programming problem $P[B]$ is defined which is known to be equivalent to (13); see Ref. 1.

We assume now that A2 and A3 are satisfied and that $B_0 := G_0$. (It can be proved that A2 is here equivalent to the condition that the restriction matrix of the problem $P[B_0]$ has rank n (see also Ref. 7).) Then, we consider the following algorithm.

Algorithm 4.1.

Step 0. Choose $\tilde{\epsilon}_i \in (0, 1)$, $i \in \mathbb{N}$, and $\tilde{\delta}_i \geq 0$, $i \in \mathbb{N}$, such that $\lim_{i \to \infty} \tilde{\delta}_i = 0$. Set $D_0^j := G_0$ for $j = 1, 2$, $k := 0$, and $i := 1$.

Step 1. Determine a solution $\hat{a}_k \in \mathbb{R}^n$ of $P[D_k]$. In particular, for $k > 1$, compute $\hat{a}_k$ as in (11).

Step 2. Set $B_{k+1} := G_i$, $\epsilon_{k+1} := \tilde{\epsilon}_i$ and $\tilde{\delta}_{k+1} := \delta_i$, and determine

$$D_{k+1}^j = \{x \in B_{k+1} \mid g_j(\hat{a}_k, x) \geq -\epsilon_{k+1} \mu(D_k)\}, \text{ for } j = 1, 2.$$

If

$$\max_{j=1,2} \max_{x \in B_{k+1}} g_j(\hat{a}_k, x) \leq \delta_{k+1}, \tag{14}$$

set $\tilde{a}_i := \hat{a}_k$, $i := i + 1$, and repeat this step.

Step 3. Set $k := k + 1$ and go to Step 1.

Algorithm 4.1 can be considered as a specification of Algorithm 3.1 so that Theorem 3.1 is applicable here where in practice, a strategy (S) has not been needed. With respect to (12), we note that here

$$\min_{j=1,2} \min_{x \in D_k^j} g_j(\hat{a}_k, x) \geq -2\mu(D_k).$$

It is further easy to show that $D_{k+1}^1 \cap D_{k+1}^2 \neq \varnothing$ if $\mu(D_k) > 0$ and that

$$D_{k+1}^1 \cup D_{k+1}^2 = \left\{ x \in B_{k+1} \;\middle|\; \left| \sum_{i=1}^{n-1} a_i z_i(x) - r(x) \right| \geq (1 - \epsilon_{k+1}) \mu(D_k) \right\};$$

see (13) in Ref. 7. Thus, for $\mu(D_k) > 0$ and the same $\hat{a}_k$ and $\tilde{\epsilon}_i$, $P[D_k]$ in Algorithm 4.1 contains only half as many constraints as $P[D_k]$ in Algorithm (L) in Ref. 7. However, we note that

$$\mu(D_k) \leq \min_{a \in R^{n-1}} \| r - \sum_{i=1}^{n-1} a_i z_i \|_{D_{k+1}^1 \cup D_{k+1}^2}$$

and that the minimal values of both problems may not coincide. We finally

remark in this connection that δ_{k+1} in (14) may also be replaced by $\delta_{k+1}\mu(D_k)$; see Ref. 7.

We have applied Algorithm 4.1 in order to determine the polynomial of best approximation to several functions in two or three variables on fine grids. So, for $x=(x_1,\ldots,x_s)^T\in\mathbb{R}^s$ and $i_1,\ldots,i_s\in\mathbb{N}$, we chose the z_i, $i=1,\ldots,n-1$, to be here the $n(d)$ functions

$$x_1^{i_1}\cdots x_s^{i_s}, \qquad i_1+\cdots+i_s\le d, \tag{15}$$

for $s=2$ or $s=3$ or alternatively, the $n(t)$ functions

$$x_1^{i_1}x_2^{i_2}, \qquad 0\le i_1, i_2\le t, \tag{16}$$

for $s=2$. The dimension of $P[B]$ is then given by

$$n(d)=\binom{d+s}{s}+1 \text{ or } n(t)=(t+1)^2+1.$$

In all cases we used equispaced grids. For G_0 with a stepsize vector $h_0\in\mathbb{R}^s$, G_{i+1} was obtained from G_i by dividing the stepsizes of G_i by z_{i+1} where normally $z_{i+1}=2$ was chosen for each $i\in\mathbb{N}_0$. We set $z_2=z_3=3$ in accordance with Ref. 2 only for Examples 4.1 and 4.2. As in Ref. 7, we prescribed $\tilde{\epsilon}_1$ and then defined

$$\tilde{\epsilon}_{i+1}=\tilde{\epsilon}_i/(z_{i+1})^s, \qquad i\in\mathbb{N}.$$

We further fixed $\tilde{\delta}_i=0$ for all $i\in\mathbb{N}$ and terminated the algorithm when $i>l$ was reached in Step 2 for $l\in\mathbb{N}$ given. Hence, the algorithm stopped when the discrete problem $P[G_l]$ had been solved completely.

In addition, we employ the following definitions.

- K: number of iterations;
- N: $=K+1=$number of solved problems $P[D_k]$, $k=0,1,\ldots,K$, $=$number of evaluations of g on total grids.
- M: average number of constraints in $P[D_k]$, $k=1,\ldots,K$;
- J: number of active constraints in $\tilde{a}_l$ on G_l;
- κ: number of evaluations of functions $g(\hat{a}_k,\cdot)$ on the finest grid G_l.

$P[D_k]$ was first solved in its dual version by the routine given in Ref. 11, a quite stable revised simplex algorithm. Afterward, $\hat{a}_k$ as in Eq. (11) was computed with the quadratic programming algorithm DQPROG of the IMSL library, which is Powell's version of the algorithm in Ref. 12. For a permanent implementation of the algorithm, a stable simplex algorithm should be used; moreover, codes should be applied which allow the option of providing a feasible starting point. All computations were carried out on the CYBER 180-860 of the Technische Universität Berlin.

Table 1. Numerical results, Example 4.1.

d	$n(d)$	$\tilde{\epsilon}_1$	M	N	κ	$\mu(G_3)$
2	7	0.01	21 (24)	4 (5)	2	2.80626×10^{-2}
3	11	0.01	22 (38)	6 (8)	3	3.47440×10^{-3}
4	16	0.01	67 (88)	4 (12)	2	6.96156×10^{-4}
5	22	0.01	76 (90)	11 (13)	5	1.62373×10^{-4}
6	29	0.01	123 (140)	8 (15)	3	3.96538×10^{-5}
7	37	0.01	186 (221)	13 (12)	6	1.00478×10^{-5}

We begin again with the two examples of Ref. 2 (cf. also Ref. 7). Other numerical examples of a comparable size are not known to us. For these examples, the algorithm performed for all choices of d in the same way whether the quadratic programming problem (11) was added to Step 1 or not. This seems to indicate the unique solvability [resp. the well-conditioning] of all discrete problems and, in particular, implies $J \geq n(d)$. The same observations have been made for Example 4.3.

Example 4.1. See Ref. 2.

Data: $r(x_1, x_2) = \log(x_1 + x_2) \sin x_1$,
$B = [0, 1] \times [1, 2.5]$,
z_i as in (15).

Parameters: $h_0 = (0.1, 0.15)^T$, $l = 3$;
$\Rightarrow |G_0| = 121$, $|G_3| = 32761$.

Results: See Table 1. The numbers in parenthesis are the corresponding numbers in Ref. 1.

Example 4.2. See Ref. 2.

Data: $r(x_1, x_2) = (1 + x_1)^{x_2}$, $B = [0, 1] \times [1, 2.5]$, z_i as in (15).

Parameters: See Example 4.1.

Results: See Table 2.

Table 2. Numerical results, Example 4.2.

d	$n(d)$	$\tilde{\epsilon}_1$	M	N	κ	$\mu(G_3)$
2	7	0.01	24 (23)	4 (6)	2	1.77657×10^{-1}
3	11	0.02	23 (33)	5 (5)	2	3.65746×10^{-2}
4	16	0.03	64 (56)	5 (11)	3	4.67753×10^{-3}
5	22	0.04	47 (79)	7 (11)	2	7.38653×10^{-4}
6	29	0.05	76 (108)	6 (12)	2	7.67641×10^{-5}
7	37	0.05	85 (145)	8 (10)	3	8.80605×10^{-6}

Table 3. Numerical results, Example 4.3.

d	$n(d)$	$\tilde{\epsilon}_1$	M	N	κ	$\mu(G_3)$
2	11	0.02	24	4	2	1.52486×10^{-1}
3	21	0.03	40	7	4	3.11125×10^{-2}
4	36	0.03	67	12	5	4.87583×10^{-3}
5	57	0.04	102	11	4	7.08744×10^{-4}

Example 4.3. See Ref. 7.

Data: $r(x_1, x_2, x_3) = \cos x_3(1+x_1)^{x_2}$,
$B = [0, 1] \times [1, 2] \times [0, 1]$, z_i as in (15).
Parameters: $h_0 = (0.2, 0.2, 0.2)^T$, $l = 3$;
$\Rightarrow |G_0| = 216$, $|G_3| = 68921$.
Results: See Table 3.

For all of the following examples, the use of the solution (11) of $P[D_k]$ in Step 1 influences the behavior of the algorithm considerably and leads to solutions $\tilde{a}_l$ for which $J < n(d)$ [resp. $J < n(t)$]. All of these examples are probably singular. (By our knowledge, there is no criterion to ensure this.)

Examples 4.4 to 4.6. See Ref. 13.

Data: $r(x_1, x_2) = 1/(x_1 + 2x_2 + 4)$, Example 4.4;
$r(x_1, x_2) = \exp(x_1^2 + x_1 x_2)$, Example 4.5;
$r(x_1, x_2) = \sqrt{x_1 + 2x_2 + 4}$, Example 4.6;
$B = [-1, 1] \times [-1, 1]$, z_i as in (16).
Parameters: $h_0 = (2/3, 2/3)^T$, $l = 8$;
$\Rightarrow |G_0| = 16$, $|G_8| = 591361$.
Results: See Table 4.

In order to show that we did not meet any stability problems, we went up to relatively fine grids in Examples 4.4 to 4.6. When alternatively the solution obtained by the simplex algorithm was used instead of the one in (11), the rate of convergence became very slow and because of stability problems computations were only possible for grids up to several hundreds of points. We remark that, in Example 4.6, we have $M < n(2)$ and $N < l + 1$.

Table 4. Numerical results, Examples 4.4 to 4.6.

Example	$n(2)$	$\tilde{\epsilon}_1$	M	N	J	κ	$\mu(G_8)$
4.4	10	0.01	11	12	4	2	5.835897×10^{-2}
4.5	10	0.01	16	9	9	2	7.354679×10^{-1}
4.6	10	0.01	9	6	4	2	1.140057×10^{-2}

Table 5. Numerical results, Example 4.7.

t	$n(t)$	$\tilde{\epsilon}_1$	M	N	J	κ	$\mu(G_6)$
3	17	0.01	15	6	5	2	2.747442×10^{-3}
4	26	0.02	36	13	14	3	7.400313×10^{-4}
5	37	0.02	73	13	16	3	2.132079×10^{-4}

The latter is explained by the fact that $\mu(G_i)$ remained constant for several indices i in succession. Similar facts also hold true for the following example.

Example 4.7.

Data: See Example 4.6.
Parameters: $h_0 = (2/7, 2/7)^T$, $l = 8$;
$\Rightarrow |G_0| = 64$, $|G_6| = 201601$.
Results: See Table 5 (cf. Ref. 13 for $t = 3$).

We finally present an example which seems to be difficult and for which we encountered severe stability problems when we went up to higher dimensions. The use of a stable simplex algorithm could possibly improve the situation.

Example 4.8.

Data: $r(x_1, x_2, x_3) = |\log[(x_1 x_2 + 1)/(x_1 + 0.5)]|\, x_2^{(x_3+1)/2}$;
$B = [0, 1] \times [0, 1] \times [0, 1]$, z_i as in (15).
Parameters: $h_0 = (0.25, 0.25, 0.25)^T$, $l = 4$;
$\Rightarrow |G_0| = 125$, $|G_4| = 274625$.
Results: See Table 6.

5. Conclusions

In this paper, we presented a result on the discretization of nonlinear semi-infinite programming problems, and we showed how the discrete problems can be regularized if the assumption needed for this result is not satisfied. We further proved the convergence of a class of algorithms for the

Table 6. Numerical results, Example 4.8.

d	$n(d)$	$\tilde{\epsilon}_1$	M	N	J	κ	$\mu(G_4)$
2	11	0.01	74	5	7	2	8.893175×10^{-2}
3	21	0.01	50	12	11	5	4.811702×10^{-2}

solution of linear problems and exhibited the behavior of a particular algorithm of this class for several multivariate Chebyshev approximation problems. By using also nonvertex solutions of the linear programming problems arising in this algorithm, we were able to obtain satisfactory numerical results also for a number of singular problems. These results suggest that (approximate) nonvertex solutions of the occurring discrete problems should be employed in order to stabilize discretization methods for singular problems. More knowledge on the right choice of such solutions would be desirable.

References

1. HETTICH, R., and ZENCKE, P., *Numerische Methoden der Approximation und Semi-infiniten Optimierung*, B. G. Teubner, Stuttgart, Germany, 1982.
2. HETTICH, R., *An Implementation of a Discretization Method for Semi-Infinite Programming*, Mathematical Programming, Vol. 34, pp. 354–361, 1986.
3. JUERGENS, U., *Zur Konvergenz Semiinfiniter Mehrfachaustauschalgorithmen*, PhD Thesis, Universität Hamburg, 1986.
4. HETTICH, R., Editor, *Semi-Infinite Programming*, Springer-Verlag, Berlin, Germany, 1979.
5. FIACCO, A. V., and KORTANEK, K. D., Editors, *Semi-Infinite Programming and Applications*, Springer-Verlag, Berlin, Germany, 1983.
6. TICHATSCHKE, R., and NEBELING, V., *A Cutting-Plane Method for Quadratic Semi-Infinite Programming Problems*, Optimization, Vol. 19, pp. 803–817, 1988.
7. REEMTSEN, R., *Modifications of the First Remez Algorithm*, SIAM Journal on Numerical Analysis, Vol. 27, pp. 507–518, 1990.
8. REEMTSEN, R., *A Note on the Regularization of Discrete Approximation Problems*, Journal of Approximation Theory, Vol. 58, pp. 352–357, 1989.
9. WERNER, J., *Optimization Theory and Applications*, Friedrich Vieweg und Sohn, Braunschweig, Germany, 1984.
10. CURTIS, A. R., and POWELL, J. M. D., *Necessary Conditions for a Minimax Approximation*, Computer Journal, Vol. 8, pp. 358–361, 1966.
11. WETTERLING, W., *Procedure RVSA (Revidierte Simplexmethode A)*, Preprint, University of Enschede, Enschede, Holland, 1974.
12. GOLDFARB, D., and IDNANI, A., *A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs*, Mathematical Programming, Vol. 27, pp. 1–33, 1983.
13. WATSON, G. A., *A Multiple Exchange Algorithm for Multivariate Chebyshev Approximation*, SIAM Journal on Numerical Analysis, Vol. 12, pp. 46–52, 1975.