KYBERNETIKA — VOLUME *21* (1985), NUMBER 1

AN IMPLEMENTATION OF RECURSIVE QUADRATIC PROGRAMMING VARIABLE METRIC METHODS FOR LINEARLY CONSTRAINED NONLINEAR MINIMAX APPROXIMATION

LADISLAV LUKŠAN

The paper contains a description of three algorithms for linearly constrained nonlinear minimax approximation. These algorithms use a dual method for solving quadratic programming subproblem together with variable metric updates for the Hessian matrix of the Lagrangian function. Moreover, a new line search procedure is described which is shown to be efficient in connection with a basic algorithm. The efficiency of all algorithms is demonstrated on test problems.

1. INTRODUCTION

We are concerned with the problem where a point $x^* \in R_n$ is sought such that

$$(1.1)\qquad \begin{cases} F(x^*) = \min\limits_{x \in L_n} \left(\max\limits_{i \in M_1} f_i(x) \right) \\ \text{where} \\ L_n = \{x \in R_n : a_i^T x \leqq b_i,\ i \in M_2\} \end{cases}$$

In (1.1), $f_i(x)$, $i \in M_1$, are real valued functions defined in the n-dimensional vector space R_n with continuous second-order derivatives and $M_1 \cup M_2 = \{1, \dots, m\}$, $M_1 \cap M_2 = \emptyset$.

Let $x \in L_n$ be a feasible point and let $a_i = a_i(x)$, $i \in M_1$, be the gradients of the functions $f_i(x)$, $i \in M_1$, at the point $x \in L_n$. Consider the quadratic programming subproblem with a symmetric positive definite matrix G which consists in the determination of the pair $(s^*, z^*) \in R_{n+1}$ such that

$$(1.2)\qquad \begin{cases} (s^*, z^*) = \arg\min\limits_{(s,z) \in L_{n+1}} \left(\frac{1}{2} s^T G s + z\right) \\ \text{where} \\ L_{n+1} = \{(s, z) \in R_{n+1} : f_i + a_i^T s \leqq e_i z,\ i \in M_1 \cup M_2\} \end{cases}$$

and where $f_i = f_i(x)$, $e_i = 1$ for $i \in M_1$ and $f_i = a_i^T x - b_i$, $e_i = 0$ for $i \in M_2$. Let $A = [a_1, \dots, a_m]$ be a matrix the columns of which are vectors a_i, $i \in M_1 \cup M_2$ and let

$$f = \begin{bmatrix} f_1 \\ \text{---} \\ f_m \end{bmatrix}, \quad e = \begin{bmatrix} e_1 \\ \text{---} \\ e_m \end{bmatrix}$$

where $f_i = f_i(x)$, $e_i = 1$ for $i \in M_1$ and $f_i = a_i^T x - b_i$, $e_i = 0$ for $i \in M_2$. If the pair $(s^*, z^*) \in R_{n+1}$ is a solution of the quadratic programming subproblem (1.2) then there exists a Lagrange multiplier vector $u^* \in R_m$ such that

$$(1.3) \qquad \begin{cases} s^* = -HAu^* \\ e^T u^* = 1 \\ u^* \geqq 0 \\ v^* \geqq 0 \\ (v^*)^T u^* = 0 \end{cases}$$

where $H = G^{-1}$ and $v^* = z^* e - f - A^T s^*$ (see [7]).

The first order (Kuhn-Tucker) necessary conditions for the original problem (1.1) have the form

$$(1.4) \qquad \begin{cases} Au^* = 0 \\ e^T u^* = 1 \\ u^* \geqq 0 \\ v^* \geqq 0 \\ (v^*)^T u^* = 0 \end{cases}$$

where $v^* = z^* e - f$ (see [8]). Comparing (1.4) with (1.3) we can see that (1.4) holds if and only if $(0, z^*) \in R_{n+1}$ is a solution of the quadratic programming subproblem (1.2). In the other way, if the first order necessary conditions (1.4) are not satisfied and if $(s^*, z^*) \in R_{n+1}$ is a solution of the quadratic programming subproblem (1.2) then, necessarily, $s^* \neq 0$. We show that this vector is a descent feasible direction for the problem (1.1).

Theorem 1.1. Let $x \in L_n$ be a feasibe point. Let $(s^*, z^*) \in R_{n+1}$ be a solution of the quadratic programming subproblem (1.2) such that $s^* \neq 0$. Then $s^* \in R_n$ is a feasible descent direction for the problem (1.1) at $x \in L_n$ provided $f_i(x)$, $i \in M_1$, have bounded Hessian matrices in a neighbourhood of $x \in L_n$.

Proof. Let $0 < \alpha \leqq 1$ be a sufficiently small steplength and let $k \in M_1$ be the index such that $F(x + \alpha s^*) = f_k(x + \alpha s^*)$ where $F(x + \alpha s^*) = \max_{i \in M_1} f_i(x + \alpha s^*)$. The function $f_k(x)$ has a bounded Hessian matrix in a neighbourhood of the point $x \in L_n$. Therefore, there exists a constant K such that

$$F(x + \alpha s^*) \leqq f_k + \alpha a_k^T s^* + \frac{\alpha^2}{2} K \|s^*\|^2 \leqq f_k + \alpha(z^* - f_k) + \frac{\alpha^2}{2} K \|s^*\|^2 \leqq$$
$$\leqq F + \alpha(z^* - F) + \frac{\alpha^2}{2} K \|s^*\|^2$$

since $f_k \leqq F = \max_{i \in M_1} f_i$ and $\alpha \leqq 1$ by the assumption, and $a_k^T s^* \leqq z^* - f_k$ by (1.3). Thus

$$(1.5) \qquad \lim_{\alpha \to 0+} \frac{F(x + \alpha s^*) - F(x)}{\alpha} \leqq z^* - F .$$

Let us denote $g^* = Au^*$. Then necessarily $g^* \neq 0$ since $s^* = -Hg^* \neq 0$ and we get

$$(s^*)^T g^* = -(g^*)^T Hg^* < 0 .$$

Therefore

(1.6)
$$z^* - F = (z^*e - Fe)^T u^* \leqq (z^*e - f)^T u^* =$$
$$= (z^*e - f - A^Ts^*)^T u^* + (A^Ts^*)^T u^* = (s^*)^T g^* < 0$$

since $f_i \leqq F$, $e_i = 1$ for $i \in M_1$ and $f_i \leqq 0$, $e_i = 0$ for $i \in M_2$ by the assumption, and, since $e^Tu^* = 1$, $u^* \geqq 0$, and $(z^*e - f - A^Ts^*)^T u^* = 0$ by (1.3). The inequalities (1.5) and (1.6) together prove that $s^* \in R_n$ is a descent direction. Furthermore $x \in L_n$ is a feasible point and $x + s^* \neq x$ is also a feasible point by (1.2). Therefore $s^* \in R_n$ is a feasibe direction. □

The direction vector $s^* \in R_n$ given by (1.3) can be determined by means of the dual method described in [7]. Having the feasible point $x \in L_n$ and the descent feasible direction $s^* \in R_n$ we can determine a new feasible point $x + \alpha s^* \in R_n$ where $0 < \alpha \leqq$ $\leqq 1$. To ensure convergence, we choose the steplength α in such a way that $F(x +$ $+ \alpha s^*)$ should be sufficiently smaller than $F(x)$. This problem is investigated in the next section where a new line search procedure is described.

The above mentioned idea was firstly presented in [5] for $L_n = R_n$ (unconstrained case) and it forms a basis for the first algorithm stated in Section 3. However, this algorithm may be inefficient in some cases as it will be shown in Section 4. Therefore two additional algorithms are given which are based on the ideas described in [1] and [3]. All algorithms use variable metric corrections for the matrix that appears in the quadratic programming subproblems which have to be solved.

The last section reviews numerical experiments with all algorithms given in Section 3. It is shown that the basic algorithm is efficient especially in the connection with a new line search procedure. It is comparable, in number of function evaluations, with other two algorithms which are more complicated and, therefore, more time-consuming.

2. A SPECIAL LINE SEARCH PROCEDURE

The steplength determination strongly affects the efficiency of algorithms for linearly constrained nonlinear minimax approximation. There are three problems which have to be solved, namely the initial steplength estimation, the line search termination, and the interpolation to obtain a new steplength in case the previous one has not been successful. To solve these problems, we use the same notation as in the previous section, except for the fact that the superscript "*" will be omitted. Moreover, we define

$$I = \{i \in M_1 \cup M_2 : v_i = ze_i - f_i - a_i^Ts = 0\}$$

and $I_1 = I \cap M_1$, $I_2 = I \cap M_2$.

The initial steplength is most frequently determined by means of linearization of the functions $f_i(x)$, $i \in M_1$. Let

$$(2.1)\qquad \left\{\begin{aligned} &\tilde{\alpha}_1 = \min_{i \in \tilde{I}_1} \left(\frac{F - f_i}{s^T a_i - s^T g} \right) \\ &\text{where} \\ &\tilde{I}_1 = \{ i \in M_1 \setminus I_1 : s^T a_i > s^T g \} \end{aligned}\right.$$

and where $g = Au$ as in the previous section. Then

$$(2.2)\qquad \alpha_1 = \min(1, \tilde{\alpha}_1)$$

is a suitable initial steplength.

As regards the line search termination, we use the same termination criterion as Han, namely the condition

$$(2.3)\qquad F(x + \alpha s) \leqq F(x) + \varepsilon_2 \alpha s^T g$$

where $0 < 2\varepsilon_2 < 1$. The condition (2.3) guarantees the global convergence of the algorithm for nonlinear minimax approximation (see [5]).

When the current steplength α_k does not satisfy the condition (2.3), we have to determine a new steplength α_{k+1}. The quadratic interpolation of the line search function $\tilde{F}(\alpha) = F(x + \alpha s)$ is usually used in this case (see [4]). But function $\tilde{F}(\alpha)$ need not be smooth. Therefore the quadratic interpolation of such a function is not advantageous. A better way consists in the approximation of each function $\tilde{f}_i(\alpha) = f_i(x + \alpha s)$, $i \in M_1$, by a special parabola $\varphi_i(\alpha) = p_i \alpha^2 + q_i \alpha + r_i$. Thus the new steplength α_{k+1} can be determined by solving the problem

$$(2.4)\qquad \left\{\begin{aligned} &\alpha_{k+1} = \arg\min_{\alpha \geq 0} \varphi(\alpha) \\ &\text{where} \\ &\varphi(\alpha) = \max_{i \in M_1} \varphi_i(\alpha) \,. \end{aligned}\right.$$

Each parabola $\varphi_i(\alpha) = p_i \alpha^2 + q_i \alpha + r_i$, $i \in M_1$, is specified by the values $\varphi_i(0) = f_i(x)$, $\varphi_i'(0) = s^T a_i(x)$ and $\varphi_i(\alpha_k) = f_i(x + \alpha_k s)$. Then $p_i = (\varphi_i(\alpha_k) - \varphi_i(0) - \varphi_i'(0)\,\alpha_k)/\alpha_k^2$, $q_i = \varphi_i'(0)$ and $r_i = \varphi_i(0)$ for $i \in M_1$ hold.

The following algorithm summarizes the above considerations.

Algorithm 2.1.

Step 1: Compute the initial steplength α_1 by (2.1)–(2.2). Set $k := 1$.

Step 2: Compute the values $\varphi_i(\alpha_k) = f_i(x + \alpha_k s)$ for all $i \in M_1$. Compute the value $F(x + \alpha_k s) = \max_{i \in M_1} f_i(x + \alpha_k s)$.

Step 3: If the condition (2.3) is satisfied for $\alpha = \alpha_k$ then terminate the computation ($\alpha = \alpha_k$ is a suitable steplength).

Step 4: Set $\tilde{\alpha}_1 := 0$ and $\tilde{\alpha}_2 := \alpha_k$. Choose the indices $i_1 \in M_1$ and $i_2 \in M_1$ such that

$$\varphi_{i_1}(\tilde{\alpha}_1) = \max_{j \in M_1} \varphi_j(\tilde{\alpha}_1),$$
$$\varphi_{i_2}(\tilde{\alpha}_2) = \max_{j \in M_1} \varphi_j(\tilde{\alpha}_2).$$

Compute the coefficients $p_{i_1}, q_{i_1}, r_{i_1}$ and $p_{i_2}, q_{i_2}, r_{i_2}$ of the parabolas $\varphi_{i_1}(\alpha)$ and $\varphi_{i_2}(\alpha)$. If $2p_{i_2}\alpha_k + q_{i_2} < 0$ then terminate the computation ($\alpha = \alpha_k$ is a suitable steplength).

Step 5: If $i_1 = i_2$ and $p_{i_1} = 0$ then set $\alpha := \tilde{\alpha}_1$ if $q_{i_1} > 0$ or $\alpha := \tilde{\alpha}_2$ if $q_{i_1} < 0$ and go to Step 7. If $i_1 = i_2$ and $p_{i_1} \neq 0$ then set $\alpha := -q_{i_1}/(2p_{i_1})$ and go to Step 7. If $i_1 \neq i_2$ and $p_{i_1} = p_{i_2}$ then set $\alpha := -(r_{i_1} - r_{i_2})/(q_{i_1} - q_{i_2})$ and go to Step 7. If $i_1 \neq i_2$ and $p_{i_1} \neq p_{i_2}$ then continue.

Step 6: Compute the values $A := -(q_{i_1} - q_{i_2})/(p_{i_1} - p_{i_2})$ and $B := -(r_{i_1} - r_{i_2}) / (p_{i_1} - p_{i_2})$. If $\tilde{\alpha}_1 < A/2 + \sqrt{((A/2)^2 + B)} < \tilde{\alpha}_2$ then set $\alpha := A/2 + \sqrt{((A/2)^2 + B)}$ and go to Step 7. If $\tilde{\alpha}_1 < A/2 - \sqrt{((A/2)^2 + B)} < \tilde{\alpha}_2$ then set $\alpha := A/2 - \sqrt{((A/2)^2 + B)}$ and go to Step 7. If the above conditions are not satisfied then set $\alpha := (\tilde{\alpha}_1 + \tilde{\alpha}_2)/2$ and go to Step 7.

Step 7: If $(\tilde{\alpha}_2 - \tilde{\alpha}_1) \leqq \varepsilon_0$ go to Step 12.

Step 8: Choose the index $i \in M_1$ such that

$$\varphi_i(\alpha) = \max_{j \in M_1} \varphi_j(\alpha).$$

If $i = i_1$ or $i = i_2$ then go to Step 9 else go to Step 10.

Step 9: If $i_1 = i_2$ or $(2p_{i_1}\alpha + q_{i_1})(2p_{i_2}\alpha + q_{i_2}) \leqq 0$ then go to Step 12 else go to Step 11.

Step 10: Compute the coefficients p_i, q_i, r_i of the parabola $\varphi_i(\alpha)$.

Step 11: If $2p_i\alpha + q_i < 0$ then set $\tilde{\alpha}_1 := \alpha$ and $i_1 := i$ else set $\tilde{\alpha}_2 := \alpha$ and $i_2 := i$. Go to Step 5.

Step 12: Set $\alpha_{k+1} := \max(\alpha, 10^{-2}\alpha_k)$. Set $k := k + 1$ and go to Step 2.

Algorithm 2.1 uses two tolerances ε_0 and ε_2. The values $\varepsilon_0 = 10^{-2}$ and $\varepsilon_2 = 10^{-2}$ were used in the implementation of this algorithm on an IBM 370/135 computer in double precision arithmetic. Efficiency of Algorithm 2.1 is shown in Section 4.

3. DESCRIPTION OF ALGORITHMS

In this section we are describing three algorithms which are based on the solution of quadratic programming subproblems. These subproblems can be solved by means of the dual method proposed in [7]. This method uses a positive definite matrix H which should approximate the inverze of the Hessian matrix of the Lagrangian function as closely as possible. Therefore, it is advantageous to use variable metric updates which belong to Broyden's class. The best known are the DFP method, the BFGS method, and the method of Hoshino which use the recurrence formulae

(3.1a) $$H^+ = H + \frac{1}{\sigma} dd^T - \frac{1}{\tau} Hy(Hy)^T,$$

(3.1b) $$H^+ = H - \frac{1}{\sigma + \tau} Hy(Hy)^T + \frac{1}{\sigma + \tau}\left(\frac{\sigma + \tau}{\sigma} d - Hy\right)\left(\frac{\sigma + \tau}{\sigma} d - Hy\right)^T$$

and

(3.1c) $$H^+ = H + \frac{2}{\sigma} dd^T - \frac{1}{\sigma + \tau}(d + Hy)(d + Hy)^T$$

respectively. Here $d = x^+ - x$, $y = g^+ - g$ and $\sigma = y^T d$, $\tau = y^T Hy$. At the same time $x^+ = x + \alpha s$ and

(3.2) $$\begin{cases} g = \sum\limits_{i \in M_1 \cup M_2} a_i u_i \\ g^+ = \sum\limits_{i \in M_1 \cup M_2} a_i^+ u_i \end{cases}$$

where $a_i^+ = a_i(x^+)$ for $i \in M_1$ and $a_i^+ = a_i$ for $i \in M_2$.

The matrix H^+ is positive definite if and only if the matrix H is positive definite and $\sigma > 0$ holds. The condition $\sigma > 0$ cannot be satisfied automatically since the steplength α is chosen to reduce the minimax objective function $F(x)$ while σ is computed from the difference between the gradients of the Lagrangian function. Therefore, the computation of the matrix H^+ has to be slightly modified. Two procedures have shown efficient during the implementation of the algorithms:

(a) The matrix H is updated by (3.1) only if

(3.3) $$\sigma \geqq \varepsilon_3 \tau$$

where $0 < \varepsilon_3 < 1$. In the opposite case we set $H^+ = H$.

(b) The procedure described in [9] is used. In this case we replace, in the formulae (3.1), d and σ by $\tilde{d} = \mu d + (1 - \mu) Hy$ and $\tilde{\sigma} = \mu\sigma + (1 - \mu)\tau$, respectively, where

(3.4) $$\mu = \min\left(1, \frac{(1 - \varepsilon_4)\tau}{\tau - \sigma}\right)$$

The first algorithm is based on the idea described in Section 1 (see also [5]).

Algorithm 3.1.

Step 1: Determine an initial feasible point $x \in L_n$ (it can be determined by the procedure described in [2]). Compute the values $f_i := f_i(x)$, $i \in M_1$, and the gradients $a_i := a_i(x)$, $i \in M_1$. Compute the value of the objective function $F := \max\limits_{i \in M_1} f_i(x)$. Set $K := 0$.

Step 2: (Restart.) Determine an initial symmetric positive definite matrix H of order n (usually set $H := I$, where I is the unit matrix of order n). Go to Step 5.

Step 3: Set $d := x - x_1$, $y := g - g_1$ and compute $\sigma := y^T d$, $\tau := y^T H y$. If $MOD = 0$ and $\sigma \geqq \varepsilon_3 \tau$, go to Step 4. If $MOD = 0$ and $\sigma < \varepsilon_3 \tau$, go to Step 5. If $MOD = 1$ then set $\mu := \min(1, (1 - \varepsilon_4) \tau / (\tau - \sigma))$ and compute $d := \mu d + (1 - \mu) Hy$, $\sigma := \mu \sigma + (1 - \mu) \tau$.

Step 4: (VM update.) Compute the matrix H^+ by (3.1a), (3.1b) or (3.1c) according to whether $MET = 1$, $MET = 2$ or $MET = 3$. Set $H := H^+$.

Step 5: (Basic QP subproblem.) Solve the quadratic programming subproblem (1.2) to obtain the pair $(s, z) \in R_{n+1}$ and the Lagrange multiplier vector $u \in R_m$. If the quadratic programming subproblem (1.2) has no solution then terminate the computation (the algorithm fails).

Step 6: Set $g_1 := Au$ and $p_1 := s^T g_1$, where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns (see (3.2)). If $|p_1| \leqq \varepsilon_1^2$ then terminate the computation (the solution of the problem (1.1) has been found with required precision).

Step 7: If $p_1 > 0$, go to Step 2.

Step 8: (Line search.) Set $x_1 := x$ and $F_1 := F$. Determine the steplength α to satisfy the condition (2.3). Use either the usual quadratic interpolation procedure (case A) or Algorithm 2.1 (case B). Set $x := x_1 + \alpha s$. Compute the values $f_i := f_i(x)$, $i \in M_1$, and the gradients $a_i := a_i(x)$, $i \in M_1$. Compute the value of the objective function $F := \max_{i \in M_1} f_i(x)$.

Step 9: If either $\|x - x_1\| \leqq TOLX . \|x\|$ or $|F - F_1| \leqq TOLF . |F|$ in two immediately consecutive iterations then terminate the computation (slow convergence was indicated) else set $g := Au$ where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns (see (3.2)).

Step 10: Set $K := K + 1$. If the restart is required (after a certain number of iterations) then go to Step 2 else go to Step 3.

Comments:

1) Algorithm 3.1 can be controlled by the integers MET and MOD. The parameter MET serves for the selection of the variable metric update from (3.1) (standard value is $MET = 2$). The parameter MOD determines a modification of the variable metric method. If $MOD = 0$ then the procedure (a) is used (see (3.3)) while if $MOD = 1$ then the procedure (b) is used (see (3.4)).
2) Algorithm 3.1 uses an additional integer K that is an iteration count.
3) Algorithm 3.1 uses several tolerances. The values $\varepsilon_1 = 10^{-10}$, $\varepsilon_2 = 10^{-2}$, $\varepsilon_3 = 10^{-2}$ and $\varepsilon_4 = 10^{-1}$ were used in the implementation of this algorithm on an IBM 370/135 computer in double precision arithmetic.
4) The tolerances $TOLX = 10^{-8}$ and $TOLF = 10^{-6}$ were used for unconstrained problems while the tolerances $TOLX = 10^{-8}$ and $TOLF = 10^{-30}$ were used for constrained problems. The results presented in the next section correspond to these tolerances.

5) Algorithm 3.1 can be restarted after a certain number of iterations (NR, say). It is not sensitive to the frequency of restart and the standard value $NR = 12n$ works well in practice.

Algorithm 3.1 can be considered as a basic algorithm for solving the problem (1.1). However, there are difficulties for this algorithm if it encounters a "steep sided curved groove" in the function $F(x)$, that is to say a nonlinear curve in the domain, across which $F(x)$ has a large jump discontinuity of derivative. This can cause slow convergence in general since the subproblem (1.2) involves a linearization of the discontinuity and only a limited progress can be made along this linearization if the function $F(x)$ is to be reduced. An associated difficulty is the "Maratos effect" in which for some x (on or close to a curved groove) arbitrarily close to x^*, a unit step $\alpha = 1$ fails to reduce $F(x)$. Superlinear convergence of Algorithm 3.1 depends on taking the unit step at every iteration, so it is no longer possible to guarantee superlinear convergence if every iteration is required to reduce $F(x)$.

The first way to overcome difficulties associated with the step reduction in Algorithm 3.1 is based on the "watchdog technique" described in [1]. The main idea of this technique consists in the use of watchdog attempts to chose steplengths that are much longer than those that would be allowed normally using (2.3). The following algorithm realize this technique for solving the problem (1.1).

Algorithm 3.2.

Step 1: The same as in Algorithm 3.1.

Step 2: (Restart.) Determine an initial symmetric positive definite matrix H of order n (usually set $H := I$, where I is the unit matrix of order n). Set $KL := K$, $KU := K + NU$ and $RS := 0$.

Step 3: The same as in Algorithm 3.1.

Step 4: The same as in Algorithm 3.1.

Step 5: The same as in Algorithm 3.1.

Step 6: The same as in Algorithm 3.1.

Step 7: The same as in Algorithm 3.1.

Step 8: If $K = KL$ then set $x_0 := x$, $F_0 := F$, $s_0 := s$, $u_0 := u$ and $g_0 := g_1$.

Step 9: If $RS = 0$ then go to Step 10 else go to Step 11.

Step 10: (Basic step.) Set $x_1 := x$ and $F_1 := F$. Determine the steplength α to satisfy the condition (2.3). Use either the usual quadratic interpolation procedure (case A) or Algorithm 2.1 (case B). Set $x := x_1 + \alpha s$. Go to Step 12.

Step 11: (Relaxed step.) Set $x_1 := x$ and $F_1 := F$. Set $\alpha := 1$ and $x := x_1 + \alpha s$.

Step 12: Compute the values $f_i := f_i(x)$, $i \in M_1$, and the gradients $a_i := a_i(x)$, $i \in M_1$. Compute the value of the objective function $F := \max_{i \in M_1} f_i(x)$. If $K = KL$ then set $\tilde{F}_0 := F_0 + \varepsilon_2 \alpha s^T g$.

Step 13: If $F \leqq \tilde{F}_0$ and $K \geqq KU$ then set $RS := 1$ else set $RS := 0$.

Step 14: If $F \leqq F_0$ then set $KL := K + 1$.

Step 15: If $K < KL + NL$, go to Step 17.

Step 16: (Back tracking.) Set $x := x_0$, $F := F_0$, $s := s_0$, $u := u_0$ and $g_1 := g_0$. Set $KL := K$, $KU := K + NU$ and go to Step 10.

Step 17: If either $\|x - x_1\| \leqq TOLX \,.\, \|x\|$ or $|F - F_1| \leqq TOLF.\, |F|$ in two immediately consecutive iterations then terminate the computation (slow convergence was indicated) else set $g := Au$ where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns (see (3.2)).

Step 18: Set $K := K + 1$. If the restart is required (after a certain number of iterations) then go to Step 2 else go to Step 3.

Comments:

1) All comments written after Algorithm 3.1 refer to Algorithm 3.2 as well.
2) Algorithm 3.2 can be controlled, in addition, by the integers NL and NU. The parameter NL gives maximal number of unsuccessful steps before back-tracking (standard value is $NL = 2$). The parameter NU restricts the frequency of unsuccessful watchdog attemts (standard value is $NU = 5$).
3) Algorithm 3.2 uses additional integers KL, KU and RS. Here KL and KU are working integers and RS is an integer indicating whether basic step ($RS = 0$) or relaxed step ($RS = 1$) is to be executed.

Another way to overcome the step reduction in Algorithm 3.1 was described in [3]. The main idea of this approach is to use projection steps returning the current point to the neighbourhood of a curved groove, which allows us to make basic steps much longer than those in the standard case. Moreover a trust region approach is used instead of line searches. Therefore (1.2) has to be replaced by

$$(3.5)\quad \begin{cases} (s^*, z^*) = \underset{(s,z)\in L_{n+1}\cap B_{n+1}}{\arg\min} \left(\tfrac{1}{2}s^{\mathrm{T}}Gs + z\right) \\ \text{where} \\ L_{n+1} = \{(s, z) \in R_{n+1} : f_i + a_i^{\mathrm{T}}s \leqq e_i z,\ i \in M_1 \cup M_2\} \\ \text{and} \\ B_{n+1} = \{(s, z) \in R_{n+1} : |s_j| \leqq h,\ 1 \leqq j \leqq n\}\,. \end{cases}$$

The constraint $\|s\|_\infty \leqq h$ defines a trust region with the radius h. If $F(x + s^*)$ is not sufficiently less than $F(x)$, the basic step has to be replaced by the projection step which uses the solution of the following subproblem.

$$(3.6)\quad \begin{cases} (s^+, z^+) = \underset{(s,z)\in \tilde{L}_{n+1}\cap B_{n+1}}{\arg\min} \left(\tfrac{1}{2}s^{\mathrm{T}}Gs + z\right) \\ \text{where} \\ \tilde{L}_{n+1} = \{(s, z) \in R_{n+1} : \tilde{f}_i + a_i^{\mathrm{T}}s \leqq e_i z,\ i \in M_1 \cup M_2\} \\ \text{and} \\ B_{n+1} = \{(s, z) \in R_{n+1} : |s_j| \leqq h,\ 1 \leqq j \leqq n\}\,, \end{cases}$$

and where $\tilde{f}_i = f_i(x + s^*) - a_i^{\mathrm{T}}s^*$, $i \in M_1 \cup M_2$. The following algorithm summarizes the above considerations. It contains more details concerning decision between basic and projection steps.

Algorithm 3.3.

Step 1: Determine an initial feasible point $x \in L_n$ (it can be determined by the procedure described in [2]). Compute the values $f_i := f_i(x)$, $i \in M_1$, and the gradients $a_i := a_i(x)$, $i \in M_1$. Compute the value of the objective function $F := \max_{i \in M_1} f_i(x)$. Set $h := h_0$ and $K := 0$.

Step 2: The same as in Algorithm 3.1.

Step 3: The same as in Algorithm 3.1.

Step 4: The same as in Algorithm 3.1.

Step 5: (Basic QP subproblem.) Solve the quadratic programming subproblem (3.5) to obtain the pair $(s, z) \in R_{n+1}$ and the Lagrange multiplier vector $u \in R_m$. If the quadratic programming subproblem (3.5) has no solution then terminate the computation (the algorithm fails).

Step 6: The same as in Algorithm 3.1.

Step 7: Set $QP := 1$. If $\|s\|_\infty < h$ then set $LS := 0$ else set $LS := 1$. Set $x_1 := x$, $F_1 := F$ and $f_{i1} := f_i$ for all $i \in M_1$. Set $u_1 := u$. Set $x := x_1 + s$. Compute the values $f_i := f_i(x)$, $i \in M_1$, and the value of the objective function $F := \max_{i \in M_1} f_i(x)$. Compute the value $D := F_1 - z + p_1/2$ and the ratio $r := (F_1 - F)/D$.

Step 8: If $r > 0{\cdot}75$, go to Step 19.

Step 9: (Projection QP subproblem.) Solve the quadratic programming subproblem (3.6) to obtain the pair $(s, z) \in R_{n+1}$ and the Lagrange multiplier vector $u \in R_m$. If the quadratic programming subproblem (3.6) has no solution then terminate the computation (the algorithm fails).

Step 10: Set $QP := 2$. Set $p := s^T A u$ where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns. Compute the ratio $\tilde{r} := (F_1 - z + {} + (p - p_1)/2)/D$.

Step 11: If $r < 0{\cdot}25$, go to Step 13.

Step 12: If $0{\cdot}9 \leqq \tilde{r} \leqq 1{\cdot}1$ then set $h := 2h$. Go to Step 20.

Step 13: If $0{\cdot}75 \leqq \tilde{r} \leqq 1{\cdot}25$ then go to Step 14 else go to Step 16.

Step 14: Set $QP := 3$. If $\|s\|_\infty < h$ then set $LS := 0$ else set $LS := 1$. Set $g_1 := Au$ where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns. Set $x := x_1 + s$. Compute the values $f_i := f_i(x)$, $i \in M_1$, and the value of the objective function $F := \max_{i \in M_1} f_i(x)$. Compute the ratio $r := {} := (F_1 - F)/D$.

Step 15: If $r > 0{\cdot}75$, go to Step 19. If $0{\cdot}25 \leqq r \leqq 0{\cdot}75$, go to Step 20. If $r < 0{\cdot}25$ go to Step 16.

Step 16: If $LS = 0$ then set $\beta := (1 - r/2)/2$ else set $\beta := (1 - r)/2$. If $\beta < 0{\cdot}1$ then set $\beta := 0{\cdot}1$. If $\beta > 0{\cdot}5$ then set $\beta := 0{\cdot}5$. Set $h := \beta h$.

Step 17: If $r > 0$, go to Step 20.

Step 18: Set $x := x_1$, $F := F_1$ and $f_i := f_{i1}$ for all $i \in M_1$. Go to Step 5.

Step 19: If $r > 0{\cdot}9$ and $LS = 1$ then set $h := 4h$. If $0{\cdot}75 < r \leqq 0{\cdot}9$ and $LS = 1$ then set $h := 2h$.

Step 20: If $QP = 2$ then set $u := u_1$. Compute the gradients $a_i := a_i(x)$, $i \in M_1$.

Step 21: If either $\|x - x_1\| \leqq TOLX . \|x\|$ or $|F - F_1| \leqq TOLF . |F|$ in two immediately consecutive iterations then terminate the computation (slow convergence was indicated) else set $g := Au$ where A is the matrix which contains the vectors a_i, $i \in M_1 \cup M_2$ as its columns (see (3.2)).

Step 22: Set $K := K + 1$. If the restart is required (after a certain number of iterations) then go to Step 2 else go to Step 3.

Comments:

1) All comments written after Algorithm 3.1 refer to Algorithm 3.3 as well.
2) Algorithm 3.3 makes use of the parameter h_0 that is the initial trust region radius.
3) Algorithm 3.3 uses additional integers QP and LS. Here QP gives information about the stage of the iteration and LS is an integer indicating whether unlimited ($LS = 0$) or limited ($LS = 1$) step is to be executed.

The efficiency of all above algorithms is investigated numerically in the next section.

4. NUMERICAL EXPERIMENTS

The efficiency of the algorithms described in Section 3 has been tested by means of test problems which are given in Appendix. The precision that has been obtained is shown in Table 1 together with the minimum values of the objective functions.

Table 1.

	Optimum function value	Precision
U1	$F(x^*) = 1{\cdot}9522245$	$P \sim 10^{-8}$
U2	$F(x^*) = -44{\cdot}00000000$	$P \sim 10^{-10}$
U3	$F(x^*) = 0{\cdot}122371 . 10^{-3}$	$P \sim 10^{-6}$
U4	$F(x^*) = 0{\cdot}19729062$	$P \sim 10^{-8}$
U5	$F(x^*) = 680{\cdot}63006$	$P \sim 10^{-8}$
U6	$F(x^*) = 24{\cdot}306209$	$P \sim 10^{-8}$
U7	$F(x^*) = 133{\cdot}72825$	$P \sim 10^{-8}$
L1	$F(x^*) = -0{\cdot}3896595161$	$P \sim 10^{-10}$
L2	$F(x^*) = -0{\cdot}3303571428$	$P \sim 10^{-10}$
L3	$F(x^*) = -0{\cdot}44891078$	$P \sim 10^{-8}$
L4	$F(x^*) = -0{\cdot}4292806146$	$P \sim 10^{-10}$
L5	$F(x^*) = 0{\cdot}1018308888$	$P \sim 10^{-10}$
L6	$F(x^*) = 0{\cdot}50694799$	$P \sim 10^{-8}$

Results of the tests are given in several tables. Each row of each table corresponds to one example numbered as in Appendix. Each column of each table corresponds to one run of the algorithm. Three numbers NI – NF – NG are given for each run and each example where

NI = number of iterations (value of integer K after termination),

NF = number of different points at which the values $f_i(x)$, $i \in M_1$, were computed,

NG = number of different points at which the gradients $a_i(x)$, $i \in M_1$, were computed,

with exception of the cases when the algorithm fails. Unsuccessful runs are denoted by the following way:

F1 – more than 300 iterations were required to obtain the solution of the test problem,

F2 – the algorithm found a point which was not a solution of the test problem.

Tests of Algorithm 3.1 have been performed with two different line search procedures. The letter A indicates the line search procedure in which the usual quadratic interpolation has been used while the letter B refers to that where each function $f_i(x)$, $i \in M_1$, has been approximated by a special parabola (Algorithm 2.1).

Table 2 contains results of the basic tests of Algorithm 3.1 for different values of the controlling parameter MOD. This table corresponds to the choice $MET = 2$.

Table 2.

	Line search A		Line search B	
	$MOD = 0$	$MOD = 1$	$MOD = 0$	$MOD = 1$
U1	8 – 10 – 10	8 – 10 – 10	8 – 10 – 10	8 – 10 – 10
U2	12 – 19 – 14	13 – 20 – 15	9 – 13 – 11	10 – 16 – 12
U3	298 – 589 – 300	15 – 20 – 17	F2	22 – 43 – 24
U4	14 – 19 – 16	15 – 18 – 17	14 – 17 – 16	15 – 18 – 17
U5	249 – 493 – 251	F1	17 – 30 – 19	42 – 93 – 44
U6	15 – 20 – 17	18 – 29 – 20	16 – 20 – 18	15 – 20 – 17
U7	21 – 33 – 23	42 – 82 – 44	19 – 30 – 21	46 – 88 – 48
L1	6 – 8 – 8	5 – 7 – 7	6 – 8 – 8	5 – 7 – 7
L2	4 – 6 – 6	4 – 6 – 6	4 – 6 – 6	4 – 6 – 6
L3	7 – 9 – 9	8 – 10 – 10	7 – 9 – 9	8 – 10 – 10
L4	75 – 77 – 77	10 – 12 – 12	75 – 77 – 77	10 – 12 – 12
L5	10 – 14 – 12	12 – 19 – 14	10 – 14 – 12	9 – 13 – 11
L6	14 – 16 – 16	18 – 24 – 20	14 – 16 – 16	14 – 17 – 16

Table 3 contains results of the basic tests of Algorithm 3.2 for different values of the controlling parameters MOD and NU. This table corresponds to the choice $MET = 2$ and $NL = 2$.

Table 4 contains results of the basic tests of Algorithm 3.3 for different values

Table 3.

	$NU = 1$		$NU = 5$	
	$MOD = 0$	$MOD = 1$	$MOD = 0$	$MOD = 1$
U1	8 — 10 — 10	8 — 10 — 10	8 — 10 — 10	8 — 10 — 10
U2	13 — 21 — 16	14 — 22 — 17	12 — 19 — 14	14 — 23 — 17
U3	85 — 162 — 105	16 — 23 — 19	80 — 159 — 91	16 — 23 — 19
U4	13 — 15 — 15	15 — 17 — 17	13 — 16 — 15	15 — 18 — 17
U5	110 — 243 — 144	F1	F1	F2
U6	16 — 19 — 18	17 — 20 — 19	15 — 19 — 17	18 — 25 — 20
U7	24 — 41 — 29	82 — 200 — 107	22 — 36 — 25	47 — 102 — 55
L1	6 — 8 — 8	5 — 7 — 7	6 — 8 — 8	5 — 7 — 7
L2	4 — 6 — 6	4 — 6 — 6	4 — 6 — 6	4 — 6 — 6
L3	7 — 9 — 9	8 — 10 — 10	7 — 9 — 9	8 — 11 — 10
L4	75 — 77 — 77	10 — 12 — 12	75 — 77 — 77	10 — 12 — 12
L5	9 — 11 — 11	8 — 10 — 10	10 — 13 — 12	10 — 14 — 12
L6	14 — 16 — 16	15 — 17 — 17	14 — 16 — 16	15 — 17 — 17

Table 4.

	$MOD = 0$			
	$h_0 = 0{\cdot}01$	$h_0 = 0{\cdot}1$	$h_0 = 1{\cdot}0$	$h_0 = 10{\cdot}0$
U1	9 — 11 — 11	7 — 9 — 9	8 — 10 — 10	8 — 10 — 10
U2	F2	F2	9 — 13 — 11	11 — 16 — 13
U3	13 — 58 — 15	12 — 50 — 14	14 — 56 — 16	12 — 46 — 14
U4	12 — 16 — 14	14 — 23 — 16	13 — 16 — 15	13 — 16 — 15
U5	14 — 39 — 16	15 — 44 — 17	12 — 45 — 14	13 — 48 — 15
U6	14 — 25 — 16	14 — 27 — 16	14 — 22 — 16	12 — 18 — 14
U7	19 — 40 — 21	21 — 45 — 23	18 — 46 — 20	18 — 51 — 20
L1	7 — 9 — 9	7 — 9 — 9	6 — 8 — 8	6 — 8 — 8
L2	5 — 7 — 7	5 — 7 — 7	4 — 6 — 6	4 — 6 — 6
L3	14 — 16 — 16	8 — 10 — 10	7 — 9 — 9	7 — 9 — 9
L4	75 — 77 — 77	75 — 77 — 77	75 — 77 — 77	75 — 77 — 77
L5	10 — 12 — 12	11 — 13 — 13	9 — 15 — 11	10 — 12 — 12
L6	33 — 42 — 35	34 — 51 — 36	35 — 48 — 37	34 — 36 — 36

of the parameter h_0 (initial trust region radius). This table corresponds to the choice $MET = 2$ and $MOD = 0$.

Table 5 contains results of the additional tests of algorithm 3.1. This table corresponds to the choice $MOD = 0$ and it demonstrates the influence of the controlling parameter MET. Table 5 shows that the choice of the variable metric method has no expressive influence on the efficiency of Algorithm 3.1. Similar results have been obtained for Algorithm 3.2 and Algorithm 3.3.

Table 5.

	Line search A			Line search B		
	$MET = 1$	$MET = 2$	$MET = 3$	$MET = 1$	$MET = 2$	$MET = 3$
	NI = 8	NI = 8	NI = 8	NI = 8	NI = 8	NI = 8
U1	NF = 10	NF = 10	NF = 10	NF = 10	NF = 10	NF = 10
	NG = 10	NG = 10	NG = 10	NG = 10	NG = 10	NG = 10
	NI = 11	NI = 12	NI = 11	NI = 9	NI = 9	NI = 9
U2	NF = 17	NF = 19	NF = 17	NF = 13	NF = 13	NF = 13
	NG = 13	NG = 14	NG = 13	NG = 11	NG = 11	NG = 11
	NI = 122	NI = 298	NI = 87	NI = 63		NI = 19
U3	NF = 197	NF = 589	NF = 167	NF = 85	F2	NF = 34
	NG = 124	NG = 300	NG = 88	NG = 65		NG = 21
	NI = 34	NI = 14	NI = 15	NI = 32	NI = 14	NI = 15
U4	NF = 46	NF = 19	NF = 17	NF = 38	NF = 17	NF = 17
	NG = 36	NG = 16	NG = 17	NG = 34	NG = 16	NG = 17
	NI = 254	NI = 249	NI = 254	NI = 17	NI = 17	NI = 16
U5	NF = 506	NF = 493	NF = 505	NF = 28	NF = 30	NF = 27
	NG = 256	NG = 251	NG = 256	NG = 19	NG = 19	NG = 18
	NI = 14	NI = 15	NI = 15	NI = 14	NI = 16	NI = 16
U6	NF = 19	NF = 20	NF = 22	NF = 18	NF = 20	NF = 18
	NG = 16	NG = 17	NG = 17	NG = 16	NG = 18	NG = 16
	NI = 19	NI = 21	NI = 20	NI = 19	NI = 19	NI = 17
U7	NF = 32	NF = 33	NF = 36	NF = 27	NF = 30	NF = 27
	NG = 21	NG = 23	NG = 22	NG = 21	NG = 21	NG = 19

Results of the tests show that Algorithm 3.1 (with usual quadratic interpolation) may be ineffective in case the problems U3 and U5 are solved. Algorithm 3.2 and Algorithm 3.3 give better results for these problems but they are more complicated and hence more time-consuming than Algorithm 3.1. Therefore it is advantageous to use Algorithm 3.1 with a new line search procedure (Algorithm 2.1), which gives good results for all problems.

The proposed algorithms were implemented as Fortran subroutines POMX 66, POMX 68 and POMX 67 in the Software Package for Optimization and Nonlinear Approximation SPONA (see [6]). All results given in this section have been obtained using these subroutines.

APPENDIX

Test problems for unconstrained nonlinear minimax approximation

Problem U1.

$$\text{Minimize } F(x) = \max_{1 \leq i \leq 3} f_i(x)$$

with

$$f_1(x_1, x_2) = x_1^2 + x_2^4$$
$$f_2(x_1, x_2) = (2 - x_1)^2 + (2 - x_2)^2$$
$$f_3(x_1, x_2) = 2 \exp(x_2 - x_1)$$

and with the starting point $x_0 = [2, 2]^T$

Problem U2 (Rosen - Suzuki).

$$\text{Minimize } F(x) = \max_{1 \leq i \leq 4} f_i(x)$$

with

$$f_1(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$
$$f_2(x) = f_1(x) + 10(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8)$$
$$f_3(x) = f_1(x) + 10(x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10)$$
$$f_4(x) = f_1(x) + 10(2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5)$$

and with the starting point $x_0 = [0, 0, 0, 0]^T$.

Problem U3.

$$\text{Minimize } F(x) = \max_{1 \leq i \leq 21} |f_i(x)|$$

with

$$f_i(x) = \frac{x_1 + x_2 t_i}{1 + x_3 t_i + x_4 t_i^2 + x_5 t_i^3} - \exp(t_i)$$

where

$$t_i = (i - 1)/10 - 1$$

and with the starting point $x_0 = [0{\cdot}5, 0, 0, 0, 0]^T$.

Problem U4.

We consider the optimization of a three-section cascaded transmission-line 10 : 1 transformer over a 100-percent bandwidth. The objective function is to minimize $\max_i |\varrho_i(x)|^2$ over 11 frequency points in the band 0·5–1·5 GHz; the sample points are taken as {0·5, 0·6, 0·7, 0·77, 0·9, 1·0, 1·1, 1·23, 1·3, 1·4, 1·5} GHz, where $\varrho_i(x)$ is the reflection coefficient at the i-th sample point. The design parameters are the characteristic impedances Z_1, Z_2, Z_3 and the lengths l_1, l_2, l_3. The lengths have been normalized with respect to quarter-wave length l_q at the centre frequency. The starting point is $l_1 = 0{\cdot}8$, $Z_1 = 1{\cdot}5$, $l_2 = 1{\cdot}2$, $Z_2 = 3{\cdot}0$, $l_3 = 0{\cdot}8$ and $Z_3 = 6{\cdot}0$.

Problem U5 (Wong 1).

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 5} f_i(x)$$

with

$$f_1(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + \\ + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

$$f_2(x) = f_1(x) + 10(2x_1^2 + 3x_2^4 + x_3 + 4x_2^2 + 5x_5 - 127)$$

$$f_3(x) = f_1(x) + 10(7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282)$$

$$f_4(x) = f_1(x) + 10(23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196)$$

$$f_5(x) = f_1(x) + 10(4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7)$$

and with the starting point $x_0 = [1, 2, 0, 4, 0, 1, 1]^T$

Problem U6 (Wong 2).

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 9} f_i(x)$$

with

$$f_1(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + \\ + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + \\ + (x_{10} - 7)^2 + 45$$

$$f_2(x) = f_1(x) + 10(3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120)$$

$$f_3(x) = f_1(x) + 10(5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40)$$

$$f_4(x) = f_1(x) + 10(0{\cdot}5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30)$$

$$f_5(x) = f_1(x) + 10(x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6)$$

$$f_6(x) = f_1(x) + 10(4x_1 + 5x_2 - 3x_7 + 9x_8 - 105)$$

$$f_7(x) = f_1(x) + 10(10x_1 - 8x_2 - 17x_7 + 2x_8)$$

$$f_8(x) = f_1(x) + 10(-3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10})$$

$$f_9(x) = f_1(x) + 10(-8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12)$$

and with the starting point $x_0 = [2, 3, 5, 5, 1, 2, 7, 3, 6, 10]^T$.

Problem U7 (Wong 3).

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 18} f_i(x)$$

with

$$f_1(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + \\ + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + \\ + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + (x_{11} - 9)^2 + \\ + 10(x_{12} - 1)^2 + 5(x_{13} - 7)^2 + 4(x_{14} - 14)^2 + \\ + 27(x_{15} - 1)^2 + x_{16}^4 + (x_{17} - 2)^2 + 13(x_{18} - 2)^2 + \\ + (x_{19} - 3)^2 + x_{20}^2 + 95$$

$$f_2(x) = f_1(x) + 10(3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120)$$

$$f_3(x) = f_1(x) + 10(5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40)$$

$$
\begin{aligned}
f_4(x) &= f_1(x) + 10(0{\cdot}5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30)\\
f_5(x) &= f_1(x) + 10(x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6)\\
f_6(x) &= f_1(x) + 10(4x_1 + 5x_2 - 3x_7 + 9x_8 - 105)\\
f_7(x) &= f_1(x) + 10(10x_1 - 8x_2 - 17x_7 + 2x_8)\\
f_8(x) &= f_1(x) + 10(-3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10})\\
f_9(x) &= f_1(x) + 10(-8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12)\\
f_{10}(x) &= f_1(x) + 10(x_1 + x_2 + 4x_{11} - 21x_{22})\\
f_{11}(x) &= f_1(x) + 10(x_1^2 + 15x_{11} - 8x_{12} - 28)\\
f_{12}(x) &= f_1(x) + 10(4x_1 + 9x_2 + 5x_{13}^2 - 9x_{14} - 87)\\
f_{13}(x) &= f_1(x) + 10(3x_1 + 4x_2 + 3(x_{13} - 6)^2 - 14x_{14} - 10)\\
f_{14}(x) &= f_1(x) + 10(14x_1^2 + 35x_{15} - 79x_{16} - 92)\\
f_{15}(x) &= f_1(x) + 10(15x_2^2 + 11x_{15} - 61x_{16} - 54)\\
f_{16}(x) &= f_1(x) + 10(5x_1^2 + 2x_2 + 9x_{17}^4 - x_{18} - 68)\\
f_{17}(x) &= f_1(x) + 10(x_1^2 - x_9 + 19x_{19} - 20x_{20} + 19)\\
f_{18}(x) &= f_1(x) + 10(7x_1^2 + 5x_2^2 + x_{12}^2 - 30x_{20})
\end{aligned}
$$

and with the starting point $x_0 = [2, 3, 5, 5, 1, 2, 7, 3, 6, 10, 2, 2, 6, 15, 1, 2, 1, 2, 1, 3]^T$.

Test problems for linearly constrained nonlinear minimax approximation

Problem L1.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 3} f_i(x)$$

with

$$
\begin{aligned}
f_1(x_1, x_2) &= x_1^2 + x_2^2 + x_1x_2 - 1\\
f_2(x_1, x_2) &= \sin x_1\\
f_3(x_1, x_2) &= -\cos x_2
\end{aligned}
$$

subject to

$$x_1 + x_2 - 0{\cdot}5 \geqq 0$$

and with the starting point $x_0 = [1, 2]^T$.

Problem L2.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 3} f_i(x)$$

with $f_i(x)$, $1 \leqq i \leqq 3$ as in Problem L1 subject to

$$-3x_1 - x_2 - 2{\cdot}5 \geqq 0$$

and with the starting point $x_0 = [-2, -1]^T$

Problem L3.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 3} f_i(x)$$

with

$$f_1(x_1, x_2) = -\exp(x_1 - x_2)$$
$$f_2(x_1, x_2) = \sinh(x_1 - 1) - 1$$
$$f_3(x_1, x_2) = -\ln(x_2) - 1$$

subject to

$$0{\cdot}05x_1 - x_2 + 0{\cdot}5 \geqq 0$$

and with the starting point $x_0 = [-1, 0{\cdot}01]^T$.

Problem L4.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 3} f_i(x)$$

with $f_i(x)$, $1 \leqq i \leqq 3$ as in Problem L3 subject to

$$-0{\cdot}9x_1 + x_2 - 1 \geqq 0$$

and with the starting point $x_0 = [-1, 3]^T$.

Problem L5.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 163} f_i(x)$$

with

$$f_i(x) = \tfrac{1}{15} + \tfrac{2}{15}\sum_{j=1}^{7} \cos(2\pi x_j \sin \vartheta_i), \quad 1 \leqq i \leqq 163$$

where

$$\vartheta_i = \frac{\pi}{180}(8{\cdot}5 + i0{\cdot}5), \quad 1 \leqq i \leqq 163$$

subject to

$$\begin{aligned} x_1 &\geqq 0{\cdot}4 \\ -x_1 + x_2 &\geqq 0{\cdot}4 \\ -x_2 + x_3 &\geqq 0{\cdot}4 \\ -x_3 + x_4 &\geqq 0{\cdot}4 \\ -x_4 + x_5 &\geqq 0{\cdot}4 \\ -x_5 + x_6 &\geqq 0{\cdot}4 \\ -x_6 + x_7 &\geqq 0{\cdot}4 \\ -x_4 + x_6 &= 1 \\ x_7 &= 3{\cdot}5 \end{aligned}$$

and with the starting point $x_0 = [0{\cdot}5, 1{\cdot}0, 1{\cdot}5, 2{\cdot}0, 2{\cdot}5, 3{\cdot}0, 3{\cdot}5]^T$.

Problem L6.

$$\text{Minimize } F(x) = \max_{1 \leqq i \leqq 38} |f_i(x)|$$

with

$$f_1(x) = -1 + x_1^2 + \sum_{j=2}^{20} x_j$$

$$f_i(x) = -1 + c_i x_k^2 + \sum_{\substack{j=1 \\ j \neq k}}^{20} x_j \,, \quad 1 < i < 38$$

$$f_{38}(x) = -1 + x_{20}^2 + \sum_{j=1}^{19} x_j$$

where

$$k = (i + 2)/2 \,, \quad c_i = 1 \,, \quad i = 2, 4, \ldots, 36$$

$$k = (i + 1)/2 \,, \quad c_i = 2 \,, \quad i = 3, 5, \ldots, 37$$

subject to

$$x_j \geqq 0{\cdot}5 \,, \quad 1 \leqq j \leqq 10$$

and with the starting point $x_j = 100$, $1 \leqq j \leqq 20$.

(Received May 12, 1984.)

REFERENCES

[1] R. M. Chamberlain, M. J. D. Powell, C. Lemarechal and H. C. Pedersen: The watchdog technique for forcing convergence in algorithms for constrained optimization. Math. Programming Study *16* (1982), 1—17.

[2] R. Fletcher: The calculation of feasible points for linearly constrained optimisation problems. A.E.R.E. Harwell Report No. R-6354 (1970).

[3] R. Fletcher: Second order corrections for non-differentiable optimization. In: Numerical Analysis, Dundee 1981 (G. A. Watson ed.), Lecture Notes in Mathematics 912, Springer-Verlag, Berlin 1982.

[4] P. E. Gill and W. Murray: Safeguarded steplength algorithms for optimization using descent methods. National Physical Lab. Report No. NAC-37 (1974).

[5] S. P. Han: Variable metric methods for minimizing a class of nondifferentiable functions. Math. Programming *20* (1981), 1, 1—13.

[6] L. Lukšan: Software package for optimization and nonlinear approximation. In: Proc. of the 2nd IFAC/IFIP Symposium on Software for Computer Control, Prague 1979.

[7] L. Lukšan: Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. Kybernetika *20* (1984), 6, 445—457.

[8] L. Lukšan: A compact variable metric algorithm for linearly constrained nonlinear minimax approximation. Kybernetika *21* (1985), to appear.

[9] M. J. D. Powell: A fast algorithm for nonlinearly constrained optimization calculations. In: Numerical Analysis, Dundee 1977 (G. A. Watson ed.), Lecture Notes in Mathematics 630, Springer-Verlag, Berlin 1978.

Ing. Ladislav Lukšan, CSc., Středisko výpočetní techniky ČSAV (General Computing Centre — Czechoslovak Academy of Sciences), Pod vodárenskou věží 4, 182 07 Praha 8. Czechoslovakia.