NDA: Algorithms for Nondifferentiable Optimization

Lukšan, Ladislav
2000

Institute of Computer Science
Academy of Sciences of the Czech Republic

NDA: Algorithms for Nondifferentiable Optimization

Ladislav Lukšan, Jan Vlček

Technical report No. 797

October 30, 2000

Pod Vodárenskou věží 2, 182 07 Prague 8 phone: +420 2 688 42 44, fax: +420 2 858 57 89,
e-mail:e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

NDA: Algorithms for Nondifferentiable Optimization

Ladislav Lukšan, Jan Vlček [1]

Technical report No. 797

October 30, 2000

Abstract:

We present four basic FORTRAN subroutines for nondifferentiable optimization with simple bounds and general linear constraints. Subroutine PMIN, intended for minimax optimization, is based on a sequential quadratic programming variable metric algorithm. Subroutines PBUN and PNEW, intended for general nonsmooth problems, are based on bundle type methods. Subroutine PVAR is based on special nonsmooth variable metric methods. Besides the description of methods and codes, we propose computational experiments which demonstrate the efficiency of this approach.

Keywords:
Minimax optimization, discrete Chebyshev approximation, sequential quadratic programming methods, variable metric methods, general linear constraints

[1]This work was supported by the Czech Republic Grant Agency, project code 201/00/0080. The first author is also from Technical University of Liberec, Hálkova 6, 461 17 Liberec

1 Introduction

We propose four basic subroutines which implement selected nondifferentiable optimization algorithms. The efficiency of these subroutines is demonstrated in [12], [13], [21], by comparing them with similar (usually not free) codes known from literature.

The double-precision FORTRAN 77 subroutine `PMIN` is designed to find a close approximation to a local minimum of a special minimax objective function

$$F(x) = \max_{1 \le i \le n_a} f_i(x).$$

Here $x \in R^n$ is a vector of n variables and $f_i : R^n \to R$, $1 \le i \le n_a$, are twice continuously differentiable functions. Subroutine `PMIN` is based on the sequential quadratic programming variable metric method described in [10] (see also [4], [19] for theoretical foundation).

The double-precision FORTRAN 77 subroutines `PBUN, PNEW, PVAR` are designed to find a close approximation to a local minimum of a nonlinear nonsmooth function $f(x)$. Here $x \in R^n$ is a vector of n variables and function $f : R^n \to R$, assumed to be Lipschitz continuous. We assume that for each $x \in R^n$ we can compute $f(x)$, an arbitrary subgradient $g(x)$, i.e. one element of the subdifferential $\partial f(x)$ (called the generalized gradient in [1]). Subroutine `PBUN` is based on the proximal bundle method described in [20] (see also [6], [8], [9], [14], [15], [16] for theoretical foundation), which only uses first-order information. Subroutine `PNEW` is based on the bundle-Newton method described in [12], which uses second-order information as well, i.e. an $n \times n$ symmetric matrix $G(x)$ as a substitute for the Hessian matrix. Subroutine `PVAR` is based on variable metric methods described in [13] and [21] which use variable metric updates for obtaining an approximation of the Hessian matrix.

All the above subroutines allow us to work with simple bounds and general linear constraints. Simple bounds are assumed in the form

$$\begin{aligned} x_i - \text{unbounded} \quad &, \quad I_i^x = 0, \\ x_i^l \le x_i \quad &, \quad I_i^x = 1, \\ x_i \le x_i^u \quad &, \quad I_i^x = 2, \\ x_i^l \le x_i \le x_i^u \quad &, \quad I_i^x = 3, \\ x_i = x_i^l = x_i^u \quad &, \quad I_i^x = 5, \end{aligned}$$

where $1 \le i \le n$. General linear constraints are assumed in the form

$$\begin{aligned} a_i^T x - \text{unbounded} \quad &, \quad I_i^c = 0, \\ c_i^l \le a_i^T x \quad &, \quad I_i^c = 1, \\ a_i^T x \le c_i^u \quad &, \quad I_i^c = 2, \\ c_i^l \le a_i^T x \le c_i^u \quad &, \quad I_i^c = 3, \\ a_i^T x = c_i^l = c_i^u \quad &, \quad I_i^c = 5, \end{aligned}$$

where $1 \le i \le n_c$ and n_c is the number of general linear constraints (I^x, I^c correspond to arays `IX, IC` in the subroutines).

To simplify the user's work, additional easy-to-use subroutines are added. These subroutines call general subroutines `PMIN, PBUN, PNEW, PVAR`:

PMINU - unconstrained minimax optimization.
PMINS - minimax optimization with simple bounds.
PMINL - minimax optimization with simple bounds and general linear constraints.
PBUNU, PNEWU, PVARU - unconstrained nonsmooth optimization.
PBUNS, PNEWS, PVARS - nonsmooth optimization with simple bounds.
PBUNL, PNEWL, PVARL - nonsmooth optimization with simple bounds and general linear constraints.

Each subroutine contains a description of formal parameters and extensive comments. Moreover, text files PMIN.TXT, PBUN.TXT, PNEW.TXT, PVAR.TXT are added, which contain a detailed description of all important subroutines (including indications of required storage). Finally, test programs TMINU, TMINL, TBUNU, TBUNL, TNEWU, TNEWL, TVARU, TVARL are included, which contain sets of test problems. These test programs serve as examples for using the subroutines, verify their correctness and demonstrate their efficiency.

2 Sequential quadratic programming methods for nonlinear minimax optimization

To simplify the description of the method, we consider the particular linearly constrained problem written in the following form

$$x^\star = \arg\min_{x \in L_C} \{\max_{i \in M_1} f_i(x)\}, \tag{1}$$

where

$$L_C = \{x \in R^n : a_i^T x \le b_i, i \in M_2\}$$

with $M_1 \cap M_2 = \emptyset$ and $M_1 \cup M_2 = M = \{1, \ldots, m\}$ (L_C is a feasible set determined by linear constraints). It is clear that the application of the method described below to the problem with general linear constraints stated in Section 1 is straightforward, but this requires considering each type of constraints separately as is realized in subroutine PMIN.

2.1 Variable metric method for nonlinear minimax optimization

If we introduce a new variable u, then the problem (1) can be reformulated as a nonlinear programming problem

$$(x^\star, u^\star) = \arg\min_{(x,u) \in N^{n+1}} \{u\}, \tag{2}$$

where

$$N^{n+1} = \{(x, u) \in R^{n+1} : f_i(x) \le e_i u, i \in M\}$$

with $e_i = 1$ for $i \in M_1$ and $e_i = 0$, $f_i(x) = a_i^T x - b_i$ for $i \in M_2$. This nonlinear programming problem can be solved by a sequential quadratic programming method that uses a quadratic approximation of the Lagrangian and a linear approximation of constraints in each iteration. Let $x^k \in R^n$ be a current approximation to the minimizer $x^\star$. Then the resulting quadratic programming subproblem has the form

$$(d^k, u^k) = \arg\min_{(d,u) \in L_k^{n+1}} \left\{\frac{1}{2} d^T G^k d + u\right\}, \tag{3}$$

where G^k is an approximation of the Hessian matrix of the Lagrangian and

$$L_k^{n+1} = \{(d,u) \in R^{n+1} : f_i^k + (a_i^k)^T d \le e_i u, i \in M\}$$

with $f_i^k = f_i(x^k)$, $a_i^k = \nabla f_i(x^k)$ for $i \in M_1$ and $f_i^k = a_i^T x^k - b_i$, $a_i^k = a_i$ for $i \in M_2$. The solution of the quadratic programming subproblem (3) has to satisfy the Karush-Kuhn-Tucker conditions

$$\begin{aligned} d^k &= -H^k g^k, \\ e^T \lambda^k &= 1, \\ \lambda^k &\ge 0, \\ \mu^k &\ge 0, \\ (\mu^k)^T \lambda^k &= 0, \end{aligned}$$

where λ^k is the vector of Lagrange multipliers, $H^k = (G^k)^{-1}$, $A^k = [a_1^k, \ldots, a_m^k]$, $e = [e_1, \ldots, e_m]^T$, $f^k = [f_1^k, \ldots, f_m^k]^T$, $g^k = A^k \lambda^k$ is the gradient of the Lagrangian function and $\mu^k = u^k e - f^k - (A^k)^T d^k$ is the vector of constraint violations. Note that if $g^k = 0$, then we obtain the Karush-Kuhn-Tucker conditions for the nonlinear programming problem (2) exactly so that the minimax problem (1) is solved if each f_i is convex. Therefore, the condition $\|g^k\|_\infty \le$ `TOLG` is used in subroutine `PMIN` as the basic stopping criterion (when it is fulfilled, then `ITERM` $= 4$).

The direction vector $d^k \in R^n$ obtained as the solution to the quadratic programming subproblem (3) is used for the definition of the new approximation x^{k+1} to the minimizer $x^\star$ by the formula

$$x^{k+1} = x^k + t^k d^k,$$

where $0 < t^k \le 1$ is a steplength, which is chosen in such a way that

$$F(x^k + t^k d^k) - F(x^k) \le m_L t^k (d^k)^T g^k,$$

where $0 < m_L < 1/2$ is a tolerance for function decrease in the line search (value $m_L = 10^{-4}$ is used in subroutine `PMIN`). The steplength t^k is chosen iteratively by the bisection.

Having the new approximation x^{k+1} to the minimum $x^\star$, we can compute the new matrix $A^{k+1} = [a_1^{k+1}, \ldots, a_m^{k+1}]$, where $a_i^{k+1} = \nabla f_i(x^{k+1})$ for $i \in M_1$ and $a_i^{k+1} = a_i$ for $i \in M_2$. If we denote $s^k = x^{k+1} - x^k$ and $z^k = A^{k+1}\lambda^k - A^k \lambda^k = A^{k+1}\lambda^k - g^k$, then the BFGS method [2] consists in the following update

$$G^{k+1} = \frac{1}{\gamma^k}\left(G^k + \gamma^k \frac{z^k (z^k)^T}{(s^k)^T z^k} - \frac{G^k s^k (G^k s^k)^T}{(s^k)^T G^k s^k}\right) = \frac{1}{\gamma^k}\left(G^k + \gamma^k \frac{z^k (z^k)^T}{(s^k)^T z^k} + t^k \frac{g^k (g^k)^T}{(s^k)^T g^k}\right),$$

where $\gamma^k > 0$ is a self-scaling parameter. The special value $\gamma^k = (s^k)^T G^k s^k / (s^k)^T z^k = -t^k (s^k)^T g^k / (s^k)^T z^k$ is used in the first iteration (or in the iteration after a restart). This special value is also used in the other iterations whenever it lies in the interval $[0.5, 4.0]$. The BFGS method requires condition $(s^k)^T z^k > 0$ to be satisfied, which guarantees a positive definiteness of matrix G^{k+1}. Unfortunately, this condition does not hold in the minimax optimization automatically. If $(s^k)^T z^k \le 0$ and `MEC` $= 1$, we set $G^{k+1} = G^k$. If $(s^k)^T z^k \le 0$ and `MEC` $= 2$, we use the Powell correction [18] (`MEC` is the parameter of subroutine `PMIN`). At the first iteration, we set $H^1 = I$ (the unit matrix). This setting is also performed if $-(d^k)^T g^k \le \underline{\varepsilon} \|d^k\| \|g^k\|$, where $\underline{\varepsilon}$ is a restart tolerance (value $\underline{\varepsilon} = 10^{-8}$ is used in the subroutine `PMIN`).

2.2 Dual range space method for a special quadratic programming subproblem

Consider a quadratic programming problem in which we seek a pair $(d^\star, u^\star) \in R^{n+1}$ in such a way that

$$(d^\star, u^\star) = \arg \min_{(d,u)\in L^{n+1}} \phi(s,u), \tag{4}$$

where

$$\phi(d,u) = \frac{1}{2} d^T G d + u$$

and

$$L^{n+1} = \{(d,u) \in R^{n+1} : f_i + a_i^T d \le e_i u, i \in M\}$$

(see (3)). The assumption that the matrix G is positive definite implies that problem (4) is convex and we can apply duality theory to obtain a dual quadratic programming problem which consists in seeking a vector $\lambda^\star \in R^m$ (vector of Lagrange multipliers of (4)) so that

$$\psi(\lambda^\star) = \min_{\lambda \in L_D} \psi(\lambda), \tag{5}$$

where

$$\psi(\lambda) = \frac{1}{2} \lambda^T A^T H A \lambda - f^T \lambda$$

and

$$L_D = \{\lambda \in R^m : e^T \lambda = 1, \lambda \ge 0\}.$$

Here $H = G^{-1}$, $A = [a_1, \ldots, a_m]$, $f = [f_1, \ldots, f_m]^T$, $e = [e_1, \ldots, e_m]^T$. The solution of (4) can be obtained from the solution of (5) by formulas

$$d^\star = -HA\lambda^\star \tag{6}$$

and

$$u^\star = f^T \lambda^\star - (\lambda^\star)^T A^T H A \lambda^\star. \tag{7}$$

The solution $\lambda^\star$ of (5) is the optimal Lagrange multiplier vector of (4). Since problem (5) is convex, $\lambda^\star$ is its solution if and only if the Karush-Kuhn-Tucker conditions are valid, i.e. if and only if

$$e^T \lambda^\star = 1, \quad \lambda^\star \ge 0 \tag{8}$$

and there exists a scalar $u^\star$ such that

$$\mu^\star = A^T H A \lambda^\star - f + u^\star e \ge 0, \quad (\mu^\star)^T \lambda^\star = 0. \tag{9}$$

Vector $\mu^\star$ is the Lagrange multiplier vector of problem (5). Conditions (6) and (9) imply that $u^\star$ in (9) is identical with $u^\star$ in (7). This in turn implies that $\mu^\star$ is, at the same time, the vector of constraint values of problem (4).

Consider any subset $I \subset M$ and denote the vectors of elements λ_i, f_i, e_i, $i \in I$ by λ, f, e, respectively. Similarly, let A be the matrix of columns a_i, $i \in I$. To simplify the investigation of the dual range space method, we denote

$$\tilde{A} = \begin{bmatrix} A \\ -e^T \end{bmatrix}, \quad \tilde{H} = \begin{bmatrix} H & 0 \\ 0 & 1 \end{bmatrix}$$

and assume that the subset $I \subset M$ was chosen in such a way that the columns of $\tilde{A}$ are linearly independent.

If $I = I^*$ were the set of active constraints at the solution of problem (4), then we could compute the dual variables $u^\star$ and $\lambda^\star$ from (8)-(9). Unfortunately, this set is not known a priory. Therefore, we start with the set $I = \{k\}$, where $k \in M_1$ is arbitrary. Then $u = f_k - a_k^T H a_k$ and $\lambda = [1]$. Suppose that $I \subset M$ is a current subset and u, λ are corresponding dual variables. Then we can proceed in the following way. First we compute the direction vector $d = -HA\lambda$ and the value of the most violated primal constraint

$$\mu_k = ue_k - f_k - a_k^T d = \min_{i \in M \setminus I} \{ue_i - f_i - a_i^T d\}.$$

If $\mu_k \geq 0$ then the set of active constraints has been detected and the solutions of (4) and (5) have been found. Otherwise, we set $\lambda_k = 0$ and compute the primal and dual steplengths

$$\begin{aligned} t_k^P &= -\frac{\mu_k}{\beta_k \gamma_k + \delta_k} \\ t_k^D &= \frac{\lambda_j}{q_{kj} + \gamma_k p_j} = \min_{i \in \overline{I}} \frac{\lambda_i}{q_{ki} + \gamma_k p_i}, \end{aligned}$$

where $p = (\tilde{A}^T \tilde{H} \tilde{A})^{-1} e$, $q_k = (\tilde{A}^T \tilde{H} \tilde{A})^{-1} \tilde{A}^T \tilde{H} \tilde{a}_k$, $\beta_k = e_k - e^T q_k$, $\gamma_k = \beta_k / p^T e$, $\delta_k = \tilde{a}_k^T (\tilde{H} - \tilde{H}\tilde{A}(\tilde{A}^T \tilde{H} \tilde{A})^{-1} \tilde{A}^T \tilde{H}) \tilde{a}_k$ (with $\tilde{a}_k = [a_k, -e_k]^T$) and $\overline{I} = \{i \in I : q_{ki} + \gamma_k p_i > 0\}$. If $\beta_k \gamma_k + \delta_k = 0$, then we set $t_k^P = \infty$. If $\overline{I} = \emptyset$, we set $t_k^D = \infty$. If simultaneously $t_k^P = \infty$ and $t_k^D = \infty$, the problem has no feasible solution. Otherwise we set $t_k = \min\{t_k^P, t_k^D\}$ and compute $u := u + t_k \gamma_k$, $\lambda := \lambda - t_k(q_k + \gamma_k p)$, $\lambda_k := \lambda_k + t_k$, $\mu_k := (1 - t_k / t_k^P)\mu_k$.

If $t_k^P \leq t_k^D$, then the primal step is realized, i.e. we set $I := I \cup \{k\}$, $\lambda := [\lambda^T, \lambda_k]^T$, $e := [e^T, e_k]^T$, $A := [A, a_k]$, $\tilde{A} := [\tilde{A}, \tilde{a}_k]$, recompute $d = -HA\lambda$ and determine a new value of the most violated primal constraint and a new index k.

If $t_k^P > t_k^D$, then the dual step is realized, i.e. we set $I := I \setminus \{j\}$, $\lambda := \lambda^{(j)}$, $e := e^{(j)}$, $A := A^{(j)}$, $\tilde{A} := \tilde{A}^{(j)}$, where the upper index in parentheses denotes an element or column which is deleted. Now, two cases can occur. If $I \cap M_1 \neq \emptyset$, we recompute the primal and dual steplengths and repeat the process with the same index k. If $I \cap M_1 = \emptyset$, then we compute $u := u - \mu_k$, set $I := I \cup \{k\}$, $\lambda := [\lambda^T, \lambda_k]^T$, $e := [e^T, e_k]^T$, $A := [A, a_k]$, $\tilde{A} := [\tilde{A}, \tilde{a}_k]$, recompute $d = -HA\lambda$ and determine the new value of the most violated primal constraint and the new index k.

In [11] it has been proved that the above dual range space method finds solutions of quadratic programming problems (4) and (5) after a finite number of steps. This method is also used for solving quadratic programming subproblems in the bundle type methods described in the next section.

3 Bundle type methods for nonsmooth optimization

To simplify the description of the method, we consider the particular linearly constrained problem written in the following form

$$x^\star = \arg \min_{x \in L_C} \{f(x)\}, \tag{10}$$

where

$$L_C = \{x \in R^n : a_j^T x \leq b_j, j \in M_2\}.$$

It is clear that the application of the methods described below to the problem with general linear constraints stated in Section 1 is straightforward, but this requires considering each type of constraint separately as is realized in subroutines `PBUN` and `PNEW`.

The idea behind the bundle methods is that they use a bundle of information obtained at points $y^j \in L_C$, $j \in J_k$, where $J_k \subset \{1, \ldots, k\}$. The bundle of information serves for building a simple nonsmooth model which is utilized for direction determination. Having the direction vector $d \in R^n$, a special line search procedure which produces either serious or short or null steps is used in such a way that

$$x^{k+1} = x^k + t_L^k d^k, \quad y^{k+1} = x^k + t_R^k d^k, \tag{11}$$

where $0 \leq t_L^k \leq t_R^k \leq 1$. Serious steps, characterized by the relation $t_R^k = t_L^k$, i.e. $y^{k+1} = x^{k+1}$, are typical for classical optimization methods. For nonsmooth minimization, special null steps are essential. Both short and null steps with $t_R^k \neq t_L^k$, i.e., $y^{k+1} \neq x^{k+1}$ obtain bundle information from a larger domain which can include points lying on the opposite sides of a possible discontinuity of the objective gradient. The difference between the bundle methods described below consists in the choice of a nonsmooth model. The proximal bundle method uses a piecewise linear function with a special quadratic penalty term while the bundle-Newton method uses a piecewise quadratic function.

Notice that bundle methods can only handle locally Lipschitz, weakly upper semismooth objectives.

3.1 The proximal bundle method

The piecewise linear function used in the proximal bundle method is based on the cutting-plane model

$$\hat{f}_k(x) = \max_{j \in J_k}\{f(y^j) + (g^j)^T(x - y^j)\} = \max_{j \in J_k}\{f(x^k) + (g^j)^T(x - x^k) - \beta_j^k\},$$

where $g^j \in \partial f(y^j)$, $j \in J_k$, are subgradients and $\beta_j^k = f(x^k) - f(y^j) - (g^j)^T(x^k - y^j)$, $j \in J_k$, are linearization errors. If the objective function were convex, then the cutting plane model would underestimate it, i.e. $\hat{f}_k(x) \leq f(x)$ for all $x \in L_C$. This is not valid in general since β_j^k may be negative in a nonconvex case. Therefore, the linearization error β_j^k is replaced by the so-called subgradient locality measure

$$\alpha_j^k = \max\{|\beta_j^k|, \gamma(s_j^k)^2\}, \tag{12}$$

where

$$s_j^k = \|x^j - y^j\| + \sum_{i=j}^{k-1} \|x^{i+1} - x^i\|$$

is the distance measure approximating $\|x^k - y^j\|$ without the need to store the bundle point y^j, $\gamma \geq 0$ is the distance measure parameter (parameter ETA of subroutine `PBUN`). We can set $\gamma = 0$ in the convex case. Obviously, now $\min_{L_C} \hat{f}_k \leq f(x^k)$ from $\alpha_j^k \geq 0$ and

$x^k \in L_C$. In order to respect the above considerations, we can define the following local subproblem for the direction determination

$$d^k = \arg \min_{x^k+d \in L_C} \{\hat{f}_k(x^k + d) + \frac{1}{2}\sigma^k d^T d\},$$

where the regularizing quadratic penalty term $(1/2)\sigma^k d^T d$ is added to guarantee the existence of the solution d^k and to keep the approximation local enough.

The choice of weights σ^k is very important. Weights which are too large imply a small $\|d^k\|$, almost all serious steps and a slow descent. Weights which are too small imply a large $\|d^k\|$ and many null steps. The weight updating method depends on the parameter `MET` of subroutine `PBUN`:

- Quadratic interpolation (`MET` = 1): The idea is based on a simplified case $n = 1$ and f quadratic, where σ^k estimates the second-order derivative of f (see [5]). By letting $\sigma^{k+1} = \min\{\max\{\sigma_{int}^{k+1}, \sigma^k/10, \sigma_{min}\}, 1/\sigma_{min}, 10\sigma^k\}$, where σ_{min} is a small positive constant, we safeguard the value σ_{int}^{k+1} obtained by quadratic interpolation (see [20] for details).

- Minimum localization (`MET` = 2): The quadratic interpolation is not suitable for f of the polyhedral type. Since the second-order derivative of the single-variable quadratic function $ax^2 + bx + c$, b fixed, is inversely proportional to the coordinate of the minimum, we set $\sigma_{loc}^{k+1} = \sigma^k/x_{min}$, where x_{min} is a computed estimate of the minimum of f in the direction d^k. We again safeguard σ_{loc}^{k+1} similarly as σ_{int}^{k+1}.

- Quasi-Newton condition (`MET` = 3): If we approximate the Hessian matrix of f by $\sigma_{con}^{k+1} \cdot I$, then the quasi-Newton condition with aggregate subgradient g_0^{k+1} (see below) can be written in the form $\sigma_{con}^{k+1}\|d^k\|^2 = (d^k)^T(g_0^{k+1} - g_0^k)$. We safeguard σ_{con}^{k+1} by setting $\sigma^{k+1} = \min\{\max\{\sigma_{con}^{k+1}, 10^{-3}\}, 10^3\}$.

The above local subproblem is still a nonsmooth optimization problem. However, due to the piecewise linear nature it can be rewritten as a (smooth) quadratic programming subproblem

$$(d^k, u^k) = \arg \min_{(d,u)\in L_k} \{u + \frac{1}{2}\sigma^k d^T d\}, \tag{13}$$

where

$$L_k = \{(d, u) : -\alpha_j^k + (g^j)^T d \le e_j u,\ j \in J_k \cup M_2\}$$

with α_j^k given by (12), $g^j \in \partial f(y^j)$, $e_j = 1$ for $j \in J_k$ and $\alpha_j^k = b_j - a_j^T x$, $g^j = a_j$, $e_j = 0$ for $j \in M_2$ (we suppose that $J_k \cap M_2 = \emptyset$, which is easily ensured in our implementation). This quadratic programming subproblem can be efficiently solved by the dual range space method described in Section 2.2.

The above derivation was slightly simplified since the aggregation of constraints was not included. In fact we add the element $\{0\}$ to J_k, letting

$$\begin{aligned}
\tilde{f}_0^{k-1} &= \sum_{j\in J_{k-1}\setminus\{0\}} \lambda_j^{k-1}\left(f(y^j) + (g^j)^T(x^{k-1} - y^j)\right) + \lambda_0^{k-1} f_0^{k-1}, \\
\tilde{s}_0^{k-1} &= \sum_{j\in J_{k-1}} \lambda_j^{k-1} s_j^{k-1},
\end{aligned}$$

$$
\begin{aligned}
f_0^k &= \tilde{f}_0^{k-1} + (g_0^k)^T(x^k - x^{k-1}),\\
s_0^k &= \tilde{s}_0^{k-1} + |x^k - x^{k-1}|,\\
g_0^k &= \sum_{j \in J_{k-1}\setminus\{0\}} \lambda_j^{k-1} g^j + \lambda_0^{k-1} g_0^{k-1},\\
\alpha_0^k &= \max\{|f(x^k) - f_0^k|, \gamma(s_0^k)^2\}
\end{aligned}
$$

and $e_0 = 1$, $f_0^1 = f(x^1)$, $s_0^1 = 0$, $g_0^1 = g^1$. The values λ_j^{k-1}, $j \in J_{k-1}$ are Lagrange multipliers of the quadratic programming subproblem from the previous iteration.

Having the pair (d^k, u^k) determined as a solution to the quadratic programming subproblem (13), we can obtain points (11) using a suitable line search (such a line search is guaranteed to be finite only in the weakly upper semismooth case). The line search consists in the initial setting $t_L^k = 0$ and the construction of the sequence $t_i^k > 0$, $i \in N$ (N is the set of natural numbers), $t_1^k = 1$, using a cubic interpolation method and a suitable backtracking. Let $0 < m_L < 1/2$, $m_L < m_R < 1$ be line search tolerances (values $m_L = 10^{-2}$ and $m_R = 1/2$ are used in subroutines `PBUN` and `PNEW`) and $0 < \underline{t} < 1$. If

$$
f(x^k + t_i^k d^k) \le f(x^k) + m_L t_i^k v^k, \tag{14}
$$

where $v^k = u^k + \sum_{j \in J_k} \lambda_j^k \alpha_j^k - \tilde{\alpha}_0^k$, $\tilde{\alpha}_0^k = \max\{|\tilde{f}_0^k - f(x^k)|, \gamma(\tilde{s}_0^k)^2\}$, then we set $t_L^k = t_i^k$. If $t_L^k \ge \underline{t}$, then we set $t_R^k = t_L^k$ and terminate the line search (serious step). Otherwise, if

$$
-\alpha_{k+1}^{k+1} + (g^{k+1})^T d^k \ge m_R v^k, \tag{15}
$$

where

$$
\begin{aligned}
\alpha_{k+1}^{k+1} &= \max\{|\beta_{k+1}^{k+1}|, \gamma(s_{k+1}^{k+1})^2\},\\
\beta_{k+1}^{k+1} &= f(x^k + t_L^k d^k) - f(x^k + t_i^k d^k) - (t_L^k - t_i^k)(g^{k+1})^T d^k,\\
s_{k+1}^{k+1} &= \|(t_L^k - t_i^k) d^k\|
\end{aligned}
$$

and $g^{k+1} \in \partial f(x^k + t_i^k d^k)$, then we set $t_R^k = t_i^k$ and terminate the line search. Otherwise, the line search continues with i increased by 1.

The iteration is terminated if $w^k \le$ `TOLG`, where $w^k = (1/2)|g_0^k|^2 + \tilde{\alpha}_0^k$ (`TOLG` is a parameter of subroutines `PBUN` and `PNEW`).

3.2 The bundle-Newton method

The bundle-Newton method is based on the following piecewise quadratic model

$$
\begin{aligned}
\tilde{f}_k(x) &= \max_{j \in J_k}\{f(y^j) + (g^j)^T(x - y^j) + \frac{1}{2}\rho^j (x - y^j)^T G^j (x - y^j)\}\\
&= \max_{j \in J_k}\{f(x^k) + (g_j^k)^T(x - x^k) + \frac{1}{2}\rho^j (x - x^k)^T G^j (x - x^k) - \beta_j^k\},
\end{aligned}
$$

where G^j are symmetric positive definite matrices, $\rho^j \in [0, 1]$ are values defined below, $g_j^k = g^j + \rho^j G^j (x^k - y^j)$ and

$$
\beta_j^k = f(x^k) - f(y^j) - (g^j)^T(x^k - y^j) - \frac{1}{2}\rho^j (x^k - y^j)^T G^j (x^k - y^j)
$$

for $j \in J_k$. Note that even in the convex case β_j^k might be negative. Therefore, we replace the error β_j^k by the locality measure (12) again so that $\min_{L_C} \tilde{f}_k \leq f(x^k)$. But $\gamma > 0$ is now required for the distance measure parameter (parameter `ETA` of the subroutines `PNEW`). The local subproblem for direction determination has the form

$$d^k = \arg \min_{x^k+d \in L_C} \{\tilde{f}_k(x^k + d)\}.$$

This local subproblem is in fact a nonlinear minimax problem which can be solved approximately by the Lagrange-Newton method (see [2]). Thus, we solve the following (smooth) quadratic programming subproblem

$$(d^k, v^k) = \arg \min_{(d,v) \in L_k} \{v + \frac{1}{2} d^T W^k d\}, \tag{16}$$

where $W^k = \sum_{j \in J_{k-1}} \lambda_j^{k-1} \rho^j G^j$ and λ_j^{k-1}, $j \in J_{k-1}$ are Lagrange multipliers of the quadratic programming subproblem from the previous iteration and

$$L_k = \{(d, v) : -\alpha_j^k + (g_j^k)^T d \leq e_j v, j \in J_k \cup M_2\}$$

with α_j^k given by (12), $g_j^k = g^j + \rho^j G^j (x^k - y^j)$, $e_j = 1$ for $j \in J_k$ and $\alpha_j^k = b_j - a_j^T x$, $g_j^k = a_j$, $e_j = 0$ for $j \in M_2$. This quadratic programming subproblem can be efficiently solved by the dual range space method described in Section 2.2.

The above derivation is not full since the aggregation of constraints is not included. The aggregation of constraints is based on the same principle that was used in the proximal bundle method. We refer to [12] for details.

Having the pair (d^k, v^k) determined as a solution to the quadratic programming subproblem (16), we can obtain the points (11) using a line search which is in fact the same as in the proximal bundle method. Again, conditions (14) and (15) are used, where

$$\begin{aligned}
\alpha_{k+1}^{k+1} &= \max\{|\beta_{k+1}^{k+1}|, \gamma(s_{k+1}^{k+1})^2\}, \\
\beta_{k+1}^{k+1} &= f(x^k + t_L^k d^k) - f(x^k + t_i^k d^k) - (t_L^k - t_i^k)(g_{k+1}^{k+1})^T d^k \\
&\quad -(\rho^{k+1}/2)(t_L^k - t_i^k)^2 (d^k)^T G(x^k + t_i^k d^k) d^k, \\
s_{k+1}^{k+1} &= \|(t_L^k - t_i^k) d^k\|
\end{aligned}$$

and g^{k+1} is replaced by $g_{k+1}^{k+1} = g(x^k + t_i^k d^k) + \rho^{k+1}(t_L^k - t_i^k) G(x^k + t_i^k d^k) d^k$. Here $g(x^k + t_i^k d^k) \in \partial f(x^k + t_i^k d^k)$ and $G(x^k + t_i^k d^k)$ is a second-order matrix computed at the point $x^k + t_i^k d^k$. The stopping criterion is in fact the same as in the proximal bundle method.

The damping parameters ρ^j, $j \in J_k$, have unit values on most iterations and are zeroed if many short or null steps occur, since a quadratic model is inefficient in this case.

4 Variable metric methods for nonsmooth optimization

The main deficiency of standard bundle methods is the necessity of solving a rather extensive QP subproblem in every iteration, which is a time-consuming procedure. On the other hand, standard variable metric methods are relatively robust and efficient when they are applied to nonsmooth convex problems (see e.g. [7]). This fact indicates that special nonsmooth modifications of variable metric methods, not containing time-consuming

operations, could be developed. Roughly speaking, three basic ideas of bundle methods can be applied to variable metric methods for improving their efficiency and robustness. The essential feature is the utilization of null steps for obtaining sufficient information about a nondifferentiable function. Furthermore, a simple aggregation of subgradients and application of modified linearization errors are used that guarantee convergence of subgradients to zero and allow us to evaluate a termination criterion.

4.1 Globally convergent variable metric methods

Globally convergent variable metric methods based on these ideas are proposed in [13], [21]. These methods, which utilize a simple three-term aggregation at null steps can be described by the following simplified procedure (we consider the unconstrained case in this subsection).

Starting with x^1, $f(x^1)$, $g^1 \in \partial f(x^1)$, H^1 positive definite (e.g. $H^1 = I$), $\tilde{g}_1 = g^1$, $\tilde{\alpha}_1 = 0$, the k-th iteration begins by testing whether matrix H^k is sufficiently positive definite (if not, correction $\varrho^k I$, $\varrho^k > 0$ is added to H^k). Then the determination of the direction vector $d^k = -H^k \tilde{g}_k$ and the computation of the stationarity measure $w^k = (\tilde{g}^k)^T H^k \tilde{g}^k + 2\tilde{\alpha}^k$ follow. If $w^k \leq$ `TOLG` (`TOLG` is a parameter of subroutine `PVAR`), then x^k is a good approximation of a stationary point. Otherwise, a steplength t^k is selected, e.g. using a piecewise linear approximation (moreover, in the nonconvex case [21], a special line-search procedure is used), together with $y^{k+1} = x^k + t^k d^k$, $f(y^{k+1})$ and $g^{k+1} \in \partial f(y^{k+1})$. Let $0 < m_L < 1/2$ be a line search tolerance (we use the value $m_L = 10^{-4}$ in subroutine `PVAR`). If

$$f(y^{k+1}) - f(x^k) \leq -m_L t^k w^k \tag{17}$$

(descent step), then we set $x^{k+1} = y^{k+1}$, $\tilde{g}^{k+1} = g^{k+1}$, compute $s^k = x^{k+1} - x^k$, $z^k = g^{k+1} - g^m$, where m is the index of the iteration after the latest serious step, determine H^{k+1} from H^k by the BFGS update [2]

$$H^{k+1} = H^k + \left(1 + \frac{(z^k)^T H^k z^k}{(z^k)^T s^k}\right) \frac{s^k (s^k)^T}{(z^k)^T s^k} - \frac{H^k z^k (s^k)^T + s^k (z^k)^T H^k}{(z^k)^T s^k}$$

finishing the k-th iteration. If (17) is not satisfied (null step), then we set $x^{k+1} = x^k$, compute $\alpha^{k+1} = (f(x^k) - f(y^{k+1}))/t^k + (d^k)^T g^{k+1}$ in the convex case [13] or $\alpha^{k+1} = \max[(f(x^k) - f(y^{k+1})) + t^k (d^k)^T g^{k+1}, \gamma |t^k d^k|^2]$, $\gamma > 0$, in the nonconvex case [21], determine multipliers $\lambda_j^k \geq 0$, $j \in \{1, 2, 3\}$, $\lambda_1^k + \lambda_2^k + \lambda_3^k = 1$, which minimize the function

$$\varphi(\lambda_1, \lambda_2, \lambda_3) = |\lambda_1 W^k g^m + \lambda_2 W^k g^{k+1} + \lambda_3 W^k \tilde{g}^k|^2 + 2[\lambda_2 \alpha^{k+1} + \lambda_3 \tilde{\alpha}^k],$$

where $W^k = (H^k)^{1/2}$ and set

$$\tilde{g}^{k+1} = \lambda_1^k g^m + \lambda_2^k g^{k+1} + \lambda_3^k \tilde{g}^k, \qquad \tilde{\alpha}^{k+1} = \lambda_2^k \alpha^{k+1} + \lambda_3^k \tilde{\alpha}^k.$$

After this simple aggregation we compute $s^k = y^{k+1} - x^k$, $z^k = g^{k+1} - g^m$. If $(\tilde{g}^k)^T (s^k - H^k z^k) > 0$, then we construct H^{k+1} from H^k by the SR1 update [2]

$$H^{k+1} = H^k + \frac{(s^k - H^k z^k)(s^k - H^k z^k)^T}{(z^k)^T (s^k - H^k z^k)}$$

finishing the k-th iteration.

More details concerning these methods are given in [13] and [21].

4.2 Active set strategy for linear constraints

Since variable metric methods do not use quadratic programing subalgorithms, which allow us to handle linear constraints automatically, we have to apply the active set strategy to these methods directly (see [17]). To simplify the description of active set strategy, we again consider the particular problem (10). Suppose $x^k \in R^n$ is a feasible point so that $a_i^T x^k = b_i$, $i \in I_k$ and $a_i^T x^k < b_i$, $i \in M_2 \backslash I_k$, where $I_k \subset M_2$ is a set of indices of active constraints. Then we can restrict to minimization on the manifold

$$L_k = \{x : (A^k)^T x = b^k\} = \{x : x^k + Z^k \hat{s}\} \tag{18}$$

where $A^k = [a_i]_{i \in I_k}$, $(b^k)^T = [b_i]_{i \in I_k}$, $(A^k)^T Z^k = 0$, $(Z^k)^T Z^k = I$ and $\mathrm{Range}([A^k, Z^k]) = R^n$. Considering this reduced problem, we can easily see that the reduced subgradient and the reduced Hessian matrix are given by $\hat{g}(\hat{s}) = (Z^k)^T g(x)$ and $\hat{G}(\hat{s}) = (Z^k)^T G(x) Z^k$, respectively. If we have an approximation $\hat{H}^k$ of $[(Z^k)^T G(x^k) Z^k]^{-1}$, we can improve it by a variable metric update where the vectors z^k and s^k are replaced by the reduced vectors $\hat{z}^k = (Z^k)^T z^k$ and $\hat{s}^k = t^k \hat{d}^k = -t^k \hat{H}^k \hat{g}^k$.

If the solution to the problem (10) lies on L_k, we can find it by using methods described in Section 4.1, remembering that all vectors have to be replaced by corresponding reduced vectors. If the solution to the problem (10) does not lie on L_k, the active constraints have to be changed subsequently. The test for constraint deletion at iteration k employs the Lagrange multiplier vector $\lambda^k = ((A^k)^T A^k)^{-1} (A^k)^T \hat{g}^k$. Let $e_{i_k}^T \lambda^k = \min_{i \in I_k} e_i^T \lambda^k$. Then i_k is deleted from I_k if $e_{i_k}^T \lambda^k \leq -c \|\hat{g}^k\|$, where $c > 0$ (value $c = 0.8$ is used in subroutine `PVAR`). On the other hand, constraint addition is performed at the end of the iteration. For this purpose, the upper bound

$$t_{max}^k = \min_{\substack{i \in M_2 \backslash I_k \\ a_i^T s^k > 0}} \frac{b_i - a_i^T x^k}{a_i^T s^k}$$

is determined and the suitable stepsize $t^k \leq t_{max}^k$ is found. After the variable metric update is carried out, all indices $i \in M_2 \backslash I_k$ satisfying $|b_i - a_i^T x^{k+1}| \leq 10^{-8}$ are added to I_k.

If the set of active constraints is changed, then the representation of the corresponding linear manifold has to be updated, see e.g. [3]. To simplify the notation, we omit index k in the rest of this section and denote by $+$ and $-$ quantities after addition and deletion of an active constraint, respectively. Then the current (active) linear manifold is represented by the set of constraint indices I, the matrix of constraint normals A, the upper triangular matrix R satisfying $R^T R = A^T A$ and the orthonormal basis Z.

After addition of the constraint normal a_+ to matrix A, we obtain

$$R_+ = \begin{bmatrix} R & r_+ \\ 0 & \rho_+ \end{bmatrix},$$

where $R^T r_+ = A^T a_+$ and $\rho_+ = \|Z^T a_+\| = \sqrt{a_+^T a_+ - r_+^T r_+}$. Furthermore, let P be the orthogonal matrix (we use a product of the Givens rotation matrices) such that $P^T Z^T a_+ = \|Z^T a_+\| e_1$, where e_1 is the first column of the unit matrix. Then Z_+ is obtained from ZP by deletion of its first column and

$$\hat{H}_+ = \hat{H}_* - \frac{\hat{h}\hat{h}^T}{\hat{\eta}}$$

where

$$P^T \hat{H} P = \begin{bmatrix} \hat{H}_* & \hat{h} \\ \hat{h}^T & \hat{\eta} \end{bmatrix}.$$

After deletion of the constraint normal a from matrix A, let M be the permutation matrix which interchanges column a with the last column of A so that $AM = [A_- \; a]$ and the matrix RM is upper Hessenberg. Let Q be the orthogonal matrix (product of the Givens rotation matrices) which anihilates the subdiagonal elements of RM so that

$$QRM = \begin{bmatrix} R_- & r \\ 0 & \rho \end{bmatrix}.$$

Then R_- is a part of the upper triangular matrix QRM. Furthermore, $Z_- = [Z \; z_-]$ where

$$z_- = AM \begin{bmatrix} R_- & r \\ 0 & \rho \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Since the second-order information along the direction z_- is not contained in matrix $\hat{H}$, we set

$$\hat{H}_- = \begin{bmatrix} \hat{H} & 0 \\ 0 & 1 \end{bmatrix}.$$

5 Description of subroutines

In this section we describe easy-to-use subroutines which can be called from the user's program. In the description of formal parameters we introduce a type of the argument denoted by two letters. The first letter is either `I` for integer arguments or `R` for double precision real arguments. The second letter specifies whether the argument must have a value defined on the entry to the subroutine (`I`), whether it is a value which will be returned (`O`), or both (`U`), or whether it is an auxiliary value (`A`). Notice that the input type arguments can be changed on the output under some circumstances, especially if improper input values were given. Besides the formal parameters, we use a `COMMON /STAT/` block containing statistical information. This block, used in each subroutine, has the following form:

```
COMMON /STAT/ NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
```

Its elements have the following meanings:

Element	Type	Significance
NDECF	IO	Number of matrix decompositions.
NRES	IO	Number of restarts.
NRED	IO	Number of reductions.
NREM	IO	Number of constraint deletions during the QP solutions.
NADD	IO	Number of constraint additions during the QP solutions.
NIT	IO	Number of iterations.
NFV	IO	Number of function evaluations.
NFG	IO	Number of gradient evaluations.
NFH	IO	Number of Hessian evaluations.

Easy-to-use subroutines are called by the following statements:

```
 CALL PMINU(NF,NA,X,AF,IPAR,RPAR,F,GMAX,IEXT,IPRNT,ITERM)
 CALL PMINS(NF,NA,NB,X,IX,XL,XU,AF,IPAR,RPAR,F,GMAX,IEXT,IPRNT,ITERM)
 CALL PMINL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,AF,IPAR,RPAR,F,
& GMAX,IEXT,IPRNT,ITERM)
 CALL PBUNU(NF,NA,X,IPAR,RPAR,F,GMAX,IPRNT,ITERM)
 CALL PBUNS(NF,NA,NB,X,IX,XL,XU,IPAR,RPAR,F,GMAX,IPRNT,ITERM)
 CALL PBUNL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,IPAR,RPAR,F,
& GMAX,IPRNT,ITERM)
 CALL PNEWU(NF,NA,X,IPAR,RPAR,F,GMAX,IHES,IPRNT,ITERM)
 CALL PNEWS(NF,NA,NB,X,IX,XL,XU,IPAR,RPAR,F,GMAX,IHES,IPRNT,ITERM)
 CALL PNEWL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,IPAR,RPAR,F,
& GMAX,IHES,IPRNT,ITERM)
 CALL PVARU(NF,NA,X,IPAR,RPAR,F,GMAX,IPRNT,ITERM)
 CALL PVARS(NF,NA,NB,X,IX,XL,XU,IPAR,RPAR,F,GMAX,IPRNT,ITERM)
 CALL PVARL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,IPAR,RPAR,F,
& GMAX,IPRNT,ITERM)
```

Their arguments have the following meanings:

Argument	Type	Significance
NF	II	Number of variables of the objective function.
NA	II	Number of functions in the minimax criterion for subroutines PMINU, PMINS, PMINL or the maximum bundle dimension for the other subroutines (choice NA = 0 causes that the default value NA =NF+3 will be taken in a later case)
NB	II	Specification whether the simple bounds are suppressed (NB = 0) or accepted (NB > 0).
NC	II	Number of linear constraints; if NC = 0 the linear constraints are suppressed.
X(NF)	RU	On input, vector with the initial estimate to the solution. On output, the approximation to the minimum.
IX(NF)	II	Vector containing the simple bound types (significant only if NB > 0):

		IX(I) = 0: the variable X(I) is unbounded, IX(I) = 1: the lower bound X(I) $\geq$ XL(I), IX(I) = 2: the upper bound X(I) $\leq$ XU(I), IX(I) = 3: the two-side bound XL(I) $\leq$ X(I) $\leq$ XU(I), IX(I) = 5: the variable X(I) is fixed (given by its initial estimate).
XL(NF)	RI	Vector with lower bounds for variables (significant only if NB > 0).
XU(NF)	RI	Vector with upper bounds for variables (significant only if NB > 0).
CF(NC)	RA	Vector which contains values of constraint functions (significant only if NC > 0).
IC(NC)	II	INTEGER vector which contains constraint types (significant only if NC > 0): IC(K) = 0: the constraint CF(K) is not used, IC(K) = 1: the lower constraint CF(K) $\geq$ CL(K), IC(K) = 2: the upper constraint CF(K) $\leq$ CU(K), IC(K) = 3: the two-side constraint CL(K) $\leq$ CF(K) $\leq$ CU(K), IC(K) = 5: the equality constraint CF(K) = CL(K).
CL(NC)	RI	Vector with lower bounds for constraint functions (significant only if NC > 0).
CU(NC)	RI	Vector with upper bounds for constraint functions (significant only if NC > 0).
CG(NF*NC)	RI	Matrix whose columns are normals of the linear constraints (significant only if NC > 0).
AF(NA)	RO	Vector which contains the values of functions in the minimax criterion.
IPAR(5)	IA	Integer parameters (see Table 5.1).
RPAR(6)	RA	Real parameters (see Table 5.1).
F	RO	Value of the objective function at the solution X.
GMAX	RO	value indicating the termination ($\|g^k\|_\infty$ in PMIN or w^k in PBUN, PNEW and PVAR).
IEXT	II	Variable that specifies the minimax criterion: IEXT < 0: maximum of function values, IEXT = 0: maximum of absolute function values (l_∞ approximation),
IHES	II	Variable that specifies a way for computing second derivatives: IHES = 0: numerical computation, IHES = 1: analytical computation by the user supplied subroutine HES.
IPRNT	II	Print specification: IPRNT = 0: print is suppressed, IPRNT = 1: basic print of final results, IPRNT = -1: extended print of final results, IPRNT = 2: basic print of intermediate and final results, IPRNT = -2: extended print of intermediate and final results,
ITERM	IO	Variable that indicates the cause of termination: ITERM = 1: if $\|x - x_{old}\|$ was less than or equal to TOLX in MTESX subsequent iterations,

ITERM = 2: if $|F - F_{old}|$ was less than or equal to TOLF in MTESF subsequent iterations,
ITERM = 3: if F is less than or equal to TOLB,
ITERM = 4: if GMAX is less than or equal to TOLG,
ITERM = 11: if NFV exceeded MFV,
ITERM = 12: if NIT exceeded MIT,
ITERM < 0: if the method failed (ITERM = −6 if the required precision was not achieved, ITERM = −10 if two consecutive restarts were required, ITERM = −12 if the quadratic programming subroutine failed).

The integer and real parameters are listed in the following table:

Parameter	PMIN	PBUN	PNEW	PVAR
IPAR(1)	MIT	MIT	MIT	MIT
IPAR(2)	MFV	MFV	MFV	MFV
IPAR(3)	MEC	MET	-	MEX
IPAR(4)	-	MTESX	MTESX	MTESX
IPAR(5)	-	MTESF	MTESF	MTESF
RPAR(1)	XMAX	XMAX	XMAX	XMAX
RPAR(2)	TOLX	TOLX	TOLX	TOLX
RPAR(3)	TOLF	TOLF	TOLF	TOLF
RPAR(4)	TOLB	TOLB	TOLB	TOLB
RPAR(5)	TOLG	TOLG	TOLG	TOLG
RPAR(6)	-	ETA	ETA	ETA

Table 5.2 - Integer and real parameters

Integer and real parameters have the following meanings:

Argument Type Significance

MIT II Maximum number of iterations; the choice MIT = 0 causes that the default value 200 will be taken.

MFV II Maximum number of function evaluations; the choice MFV = 0 causes that the default value 500 will be taken.

MEC II Variable that specifies correction of variable metric updates if negative curvature occurs:
MEC = 1: correction is suppressed,
MEC = 2: Powell's correction is used.
The choice MEC = 0 causes that the default value MEC = 1 will be taken.

MET II Variable that specifies the weight updating method:
MET = 1: quadratic interpolation,
MET = 2: local minimization,
MET = 3: quasi-Newton condition.
The choice MET = 0 causes that the default value MET = 1 will be taken.

MEX II Variable that specifies version of nonsmooth variable metric method:
MEX = 1: convex version is used,
MEX = 2: nonconvex version is used.

The choice MEX = 0 causes that the default value MEX = 2 will be taken.

MTESX	II	Maximum number of iterations with changes of the coordinate vector X smaller than TOLX; the choice MTESX = 0 causes that the default value 20 will be taken.
MTESF	II	Maximum number of iterations with changes of function values smaller than TOLF; the choice MTESF = 0 causes that the default value 2 will be taken.
XMAX	RI	Maximum stepsize; the choice XMAX = 0 causes that the default value 10^3 will be taken.
TOLX	RI	Tolerance for the change of the coordinate vector X; the choice TOLX = 0 causes that the default value 10^{-16} will be taken.
TOLF	RI	Tolerance for the change of the function value; the choice TOLF = 0 causes that the default value 10^{-8} will be taken.
TOLB	RI	Minimum acceptable function value; the choice TOLB = 0 causes that the default value -10^{60} will be taken.
TOLG	RI	Tolerance for the gradient of the Lagrangian function; the choice TOLG = 0 causes that the default value 10^{-6} will be taken.
ETA	RI	Distance measure parameter γ (ETA=0 is the default value).

The choice of parameters ETA and XMAX is rather delicate. It can considerably influence the efficiency of the method. Therefore, these parameters should be tuned carefully. Briefly, the parameter ETA should be smaller (e.g. $10^{-12} - 10^{-6}$) for convex problems and larger (e.g. $10^{-4} - 10^{2}$) for nonconvex problems. The parameter XMAX reduces the stepsize so that it plays an important role in the neighborhood of the kink. The other parameters are not so important, but small MTESX or MTESF can lead to premature termination of the iterative process.

Subroutines PMINU, PMINS, PMINL require the user supplied subroutines FUN and DER which define the values and the gradients of the functions in the minimax criterion and have the form

```
SUBROUTINE FUN(NF,KA,X,FA)
SUBROUTINE DER(NF,KA,X,GA)
```

Subroutines PBUNU, PBUNS, PBUNL, PNEWU, PNEWS, PNEWL, PVARU, PVARS, PVARL require the user supplied subroutine FUNDER which defines the objective function and its subgradient and has the form

```
SUBROUTINE FUNDER(NF,X,F,G)
```

Subroutines PNEWU, PNEWS, PNEWL require the additional user supplied subroutine HES which defines the matrix of the second-order information (usually the Hessian matrix) and has the form

```
SUBROUTINE HES(NF,X,H)
```

If IHES=0, then the user supplied subroutine HES can be empty.

The arguments of user supplied subroutines have the following meanings:

Argument	Type	Significance
NF	II	Number of variables of the objective function.
KA	II	Index of a function in the minimax criterion.

X(NF)	RI	An estimate to the solution.
FA	RO	Value of a function with the index KA at point X.
GA(NF)	RO	Gradient of a function with the index KA at point X.
F	RO	Value of the objective function at point X.
G(NF)	RO	Subgradient of the objective function at point X.
H(NH)	RO	Matrix of the second-order information at point X (NH is equal to NF*(NF+1)/2).

6 Verification of subroutines

In this section we report the results obtained by using test programs TMINU, TMINL, TBUNU, TBUNL, TNEWU, TNEWL, TVARU, TVARL which serve for demonstration, verification and testing of subroutines PMINU, PMINL, PBUNU, PBUNL, PNEWU, PNEWL, PVARU, PVARL. These results are listed in the following tables (rows corresponding to individual test problems contain the number of iterations NIT, the number of function evaluations NFV, the number of gradient evaluations NFG, the final value of the objective function F, the value of the termination criterion G and the cause of termination ITERM). All computations reported were performed on a Pentium PC computer, under the Windows 2000 system using the Digital Visual Fortran (Version 6) compiler, in double precision arithmetic. All subroutines were checked with a Fortran verifier and also implemented and tested on various UNIX workstations (Digital, Silicon Graphics, Hewlet Packard).

Problem	NIT	NFV	NFG	F	G	ITERM
1	7	8	8	1.95222449	0.104E-07	4
2	7	8	8	0.216079417E-09	0.178E-13	4
3	93	180	94	0.250685495E-10	0.651E-06	4
4	13	15	14	3.59971930	0.250E-07	4
5	11	16	12	-44.0000000	0.251E-06	4
6	12	21	13	-44.0000000	0.854E-06	4
7	8	9	9	0.420214268E-02	0.644E-09	4
8	5	6	6	0.508163266E-01	0.150E-06	4
9	10	12	11	0.808436839E-02	0.187E-08	4
10	11	11	11	115.706440	0.708E-08	4
11	35	113	36	0.263597350E-02	0.149E-07	4
12	34	86	35	0.201607548E-02	0.157E-08	4
13	7	8	8	0.996651439E-05	0.453E-06	4
14	6	8	7	0.122371255E-03	0.684E-07	4
15	17	57	17	0.223404960E-01	0.326E-12	4
16	21	53	22	0.349049265E-01	0.252E-07	4
17	11	16	12	0.197290621	0.482E-06	4
18	18	91	19	0.618528478E-02	0.192E-06	4
19	19	45	20	680.630057	0.574E-06	4
20	13	19	14	24.3062091	0.937E-07	4
21	19	30	20	133.728276	0.555E-06	4
22	41	106	41	54.5981500	0.303E-05	-6
23	22	25	23	261.082581	0.315E-06	4
24	18	20	19	0.911538955E-07	0.547E-06	4
25	67	286	68	0.480296951E-01	0.862E-06	4
Σ	525	1249	539	TIME = 0.06		

Table 6.1 - Results obtained by program `TMINU`

Problem	NIT	NFV	NFG	F	G	ITERM
1	6	7	7	-0.389659516	0.613E-08	4
2	5	5	5	-0.330357143	0.222E-15	4
3	8	8	8	-0.448910786	0.203E-10	4
4	75	75	75	-0.429280613	0.445E-10	4
5	9	9	9	-1.85961870	0.830E-12	4
6	7	9	8	0.101830889	0.821E-06	4
7	7	10	8	0.710542736E-14	0.660E-06	4
8	15	23	16	24.3062091	0.350E-06	4
9	23	37	24	133.728276	0.286E-07	4
10	15	16	15	0.506947996	0.149E-10	4
11	38	40	39	0.276078379E-03	0.532E-07	4
12	157	864	158	-1768.80696	0.357E-07	4
13	15	22	16	1227.22608	0.171E-06	4
14	147	270	148	7049.24802	0.103E-06	4
15	65	109	65	174.786994	0.217E-08	4
Σ	592	1504	601	TIME = 0.06		

Table 6.2 - Results obtained by program `TMINL`

Problem	NIT	NFV	NFG	F	G	ITERM
1	45	48	48	0.198400179E-07	0.325E-03	4
2	29	31	31	0.435801362E-13	0.354E-06	4
3	31	33	33	1.95222452	0.445E-03	2
4	13	15	15	2.00000000	0.192E-06	4
5	17	19	19	-3.00000000	0.974E-10	2
6	13	15	15	7.20000149	0.221E-02	4
7	18	21	21	-1.41421356	0.370E-11	2
8	54	56	56	-0.999999830	0.535E-03	2
9	13	15	15	-1.00000000	0.986E-07	4
10	44	47	47	-7.99999992	0.469E-02	2
11	43	45	45	-43.9999991	0.373E-02	2
12	27	29	29	22.6001621	0.145E-03	4
13	59	61	61	-32.3486789	0.243E-03	2
14	115	116	116	-2.91969280	0.137E-02	2
15	155	156	156	0.559814839	0.123E-02	2
16	74	75	75	-0.841408290	0.729E-03	2
17	146	148	148	9.78593906	0.493E-02	2
18	128	151	151	16.7038653	0.734E-02	2
19	148	149	149	0.167123812E-06	0.781E-04	2
20	39	40	40	0.272746651E-12	0.125E+00	2
Σ	1211	1270	1270	TIME = 0.06		

Table 6.3 - Results obtained by program `TBUNU`

Problem	NIT	NFV	NFG	F	G	ITERM
1	10	11	11	-0.389659516	0.453E-04	4
2	4	5	5	-0.330357143	0.389E-14	4
3	8	10	10	-0.448910786	0.698E-03	4
4	84	85	85	-0.429280614	0.139E-07	2
5	16	17	17	-1.85961382	0.202E-10	2
6	16	17	17	0.101830889	0.343E-06	2
7	54	57	57	0.108743445E-08	0.144E-08	2
8	73	75	75	24.3062154	0.456E-02	4
9	148	151	151	133.728344	0.194E-01	2
10	81	82	82	0.506947996	0.264E-06	2
11	568	733	733	0.604084510E-03	0.687E-02	2
12	231	233	233	-1687.55166	0.456E-01	2
13	128	130	130	1227.22963	0.210E-01	2
14	201	203	203	7051.22733	0.287E-01	2
15	390	393	393	174.791184	0.340E-01	2
Σ	2012	2202	2202	TIME = 0.11		

Table 6.4 - Results obtained by program `TBUNL`

Problem	NIT	NFV	NFG	F	G	ITERM
1	58	59	59	0.134104250E-18	0.895E-05	2
2	7	8	8	0.167657006E-10	0.579E-05	4
3	15	17	17	1.95222450	0.304E-03	4
4	10	11	11	2.00000682	0.216E-04	4
5	14	15	15	-2.99999999	0.676E-04	4
6	3	5	5	7.20000000	0.000E+00	4
7	16	17	17	-1.41421356	0.565E-07	4
8	11	13	13	-1.00000000	0.416E-07	4
9	10	11	11	-1.00000000	0.456E-06	4
10	24	25	25	-7.99999997	0.322E-02	4
11	13	15	15	-44.0000000	0.421E-05	4
12	7	8	8	22.6001727	0.126E-02	4
13	22	24	24	-32.3486790	0.341E-02	4
14	83	84	84	-2.91970018	0.108E-02	4
15	116	123	123	0.559813085	0.240E-05	4
16	12	14	14	-0.841408335	0.673E-06	4
17	68	72	72	9.78577208	0.107E-03	4
18	40	42	42	16.7038553	0.178E+00	4
19	36	37	37	0.383737024E-08	0.609E-08	2
20	24	25	25	0.452894273E-08	0.100E-07	2
Σ	589	625	625	TIME = 0.05		

Table 6.5 - Results obtained by program `TNEWU`

Problem	NIT	NFV	NFG	F	G	ITERM
1	6	7	7	-0.389659516	0.165E-07	4
2	2	11	11	-0.330357143	0.111E-15	4
3	37	38	38	-0.448910785	0.134E-06	4
4	9	10	10	-0.429280609	0.311E-04	4
5	28	29	29	-1.85961870	0.741E-07	4
6	9	10	10	0.101830889	0.114E-06	4
7	69	70	70	0.813571432E-12	0.737E-05	4
8	15	16	16	24.3062091	0.130E-06	4
9	43	45	45	133.728305	0.409E-03	4
10	94	98	98	0.506947996	0.407E-06	4
11	1148	1192	1192	0.282340278E-03	0.315E-01	2
12	1239	1241	1241	-1768.80243	0.205E-02	2
13	71	80	80	1227.22608	0.635E-02	2
14	54	55	55	7049.24803	0.949E-04	2
15	1382	1385	1385	174.869642	0.322E-01	2
Σ	4206	4287	4287	TIME = 0.41		

Table 6.6 - Results obtained by program `TNEWL`

Problem	NIT	NFV	NFG	F	G	ITERM
1	56	57	57	0.103091036E-08	0.207E-08	4
2	15	16	16	0.948941202E-10	0.474E-10	4
3	26	26	26	1.95222473	0.779E-06	4
4	17	17	17	2.00000000	0.292E-07	4
5	22	22	22	-2.99999998	0.402E-07	4
6	22	22	22	7.20000067	0.492E-06	4
7	14	14	14	-1.41421356	0.175E-06	4
8	58	63	63	-0.999999928	0.103E-06	4
9	62	62	62	-0.999999852	0.574E-07	4
0	39	39	39	-7.99999983	0.848E-06	4
11	74	75	75	-43.9999976	0.433E-06	4
12	49	49	49	22.6001627	0.241E-06	4
13	52	53	53	-32.3486784	0.100E-05	2
14	32	32	32	-2.91970037	0.341E-06	4
15	114	114	114	0.559818540	0.624E-06	4
16	112	112	112	-0.841396818	0.474E-06	4
17	158	158	158	9.78600636	0.890E-06	4
18	105	105	105	16.7038379	0.446E-06	4
19	128	129	129	0.160715615E-05	0.355E-06	4
20	22	22	22	0.00000000	0.000E+00	4
Σ	1177	1187	1187	TIME = 0.03		

Table 6.7 - Results obtained by program `TVARU`

Problem	NIT	NFV	NFG	F	G	ITERM
1	11	11	11	-0.389659516	0.163E-07	4
2	5	8	8	-0.330357143	0.487E-30	2
3	27	27	27	-0.448910784	0.416E-06	4
4	86	86	86	-0.429280615	0.297E-07	4
5	24	24	24	-1.85961862	0.294E-06	4
6	25	25	25	0.101830939	0.943E-07	4
7	94	94	94	0.526114262E-05	0.777E-06	4
8	159	159	159	24.3064637	0.652E-06	4
9	226	227	227	133.728285	0.174E-05	2
10	144	145	145	0.506950749	0.598E-05	2
11	658	658	658	0.297444605E-03	0.727E-06	4
12	203	212	212	-1768.57476	0.210E-04	2
13	451	448	448	1227.28556	0.143E-04	2
14	294	479	479	7049.25599	0.217E-03	2
15	181	180	180	174.787082	0.374E-03	2
Σ	2588	2783	2783	TIME = 0.06		

Table 6.8 - Results obtained by program `TVARL`

Computational comparisons of subroutines `PBUNU, PNEWU, PVARU` with other algorithms described in literature can be found in [12], [13] and [21].

References

[1] Clarke F.H. Optimization and Nonsmooth Analysis. Wiley-Interscience, New York, 1983.

[2] Fletcher R.: Practical Methods of Optimization (second edition). Wiley, New York, 1987.

[3] Gill P.E., Murray W.: Newton type methods for unconstrained and linearly constrained optimization. Math. Programming 7 (1974) 311-350.

[4] Han S.P.: Variable Metric Methods for Minimizing a Class of Nondifferentiable Functions. Math. Programming 20 (1981) 1, 1-13 Derived by Variational Means. Math. Comput. 24 (1970) 23-26.

[5] Kiwiel K.C. Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization, Mathematical Programming 46 (1980), 105-122.

[6] Kiwiel K.C. Methods of Descent for Nondifferentiable Optimization, Lecture Notes in Mathematics 1133, Springer-Verlag, Berlin, 1985.

[7] Lemarechal C.: Numerical experiments in nonsmooth optimization. Proc. IIASA workshop "Progress in Nondifferentiable Optimization", December 1978.

[8] Lemarechal C.: Nondifferentiable Optimization. In: Optimization (G.L.Nemhauser, A.H.G.Rinnooy Kan, M.J.Todd, eds.), Elsevier Science Publishers, North-Holland, Amsterdam 1989.

[9] Lemarechal C., Zowe J.: A condensed introduction to bundle methods in nonsmooth optimization. In: Algorithms for Continuous Optimization (E.Spedicato, ed.), Kluwer Academic Publishers, Dordrecht, 1994.

[10] Lukšan L.: 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.

[11] Lukšan L.: An Implementation of Recursive Quadratic Programming Variable Metric Methods for Linearly Constrained Nonlinear Minimax Approximation, Kybernetika 21 (1985), 1, 22-40.

[12] Lukšan L., Vlček J.: A Bundle-Newton Method for Nonsmooth Unconstrained Minimization. Math. Programming 83 (1998) 373-391.

[13] Lukšan L., Vlček J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. Journal of Optimization Theory and Applications Vol.102, 1999, pp.593-613.

[14] Mäkelä M.M., Neittaanmäki P.: Nonsmooth Optimization. World Scientific Publishing Co., London, 1992.

[15] Mifflin R.: An algorithm for constrained optimization with semismooth functions. Mathematics of Operations Research 2 (1977), 191-207.

[16] Mifflin R.: A modification and an extension of Lemarechal's algorithm for nonsmooth minimization, Mathematical Programming Study 17 (1982) 77-90.

[17] Panier E.: An active set method for solving linearly constrained nonsmooth optimization problems. Mathematical Programming 37 (1987) 269-292.

[18] Powell M.J.D.: A fast algorithm for nonlinearly constrained optimization calculations. In:"Numerical analysis" (G.A.Watson ed.). Springer Verlag, Berlin 1977.

[19] Pschenychny B.N.: Method of Linearization. Nauka, Moscow, 1983.

[20] Vlček J.: Bundle Algorithms for Nonsmooth Unconstrained Minimization. Research Report V-608, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 1995.

[21] Vlček J., Lukšan L.: Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. Technical Report B-8/1999. Laboratory of Scientific Computing, University of Jyvaskyla, Jyvaskyla 1999.