PBUN, PNEW - A Bundle-Type Algorithms for Nonsmooth Optimization

Lukšan, Ladislav
1997

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Prague

PBUN, PNEW - A Bundle-Type Algorithms for Nonsmooth Optimization

L. Lukšan, J. Vlček

Technical Report No. V-718

September 1997

Akademie věd České republiky

ÚSTAV INFORMATIKY A VÝPOČETNÍ TECHNIKY

Institute of Computer Science, Academy of Sciences of the Czech Republic

Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic

E-mail: ICS@uivt.cas.cz

Fax: (+422) 8585789 Phone: (+422) 846669, (+422) 66051111

PBUN, PNEW - A Bundle-Type Algorithms for Nonsmooth Optimization [1]

L. Lukšan and J. Vlček
Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic

Abstract. We present FORTRAN subroutines for minimizing multivariate nonsmooth functions with simple bounds and general linear constraints by bundle-type algorithms.

Categories and Subject Descriptors:

General Terms: Algorithms

Additional Key Words and Phrases: Nondifferentiable minimization, bundle methods, general linear constraints

---

1. Introduction

The double-precision FORTRAN 77 basic subroutines PBUN and PNEW are designed to find a close approximation to a local minimum of a nonlinear nonsmooth function $f(x)$ with simple bounds on variables and general linear constraints. Here $x \in R^n$ is a vector of n variables and $f : R^n \to R$ is assumed to be a locally Lipschitz continuous function. 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]). The subroutine PBUN is based on the proximal bundle method desdcribed in [9] (see also [8] for theoretical foundation), which only uses first order information. The subroutine PNEW is based on the bundle-Newton method described in [5], which uses second order information as well, i.e. an $n \times n$ symmetric matrix $G(x)$ as a substitute for the Hessian matrix. Simple bounds are assumed in the form (I^x, I^c correspond to the arrays `IX, IC` in Section 3)

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

[1] This work was supported under grant No. 201/96/0918 given by the Czech Republic Grant Agency

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 a number of general linear constraints. To simplify user's work, six additional easy to use subroutines are added. They call the basic general subroutines PBUN and PNEW:

PBUNU and PNEWU - unconstrained nonsmooth optimization

PBUNS and PNEWS - nonsmooth optimization with simple bounds

PBUNL and PNEWL - nonsmooth optimization with simple bounds and general linear constraints

All subroutines contain a description of formal parameters and extensive comments. Furthermore, two test programs TBUNU and TNEWU are included. They contain 23 standard test problems (see e.g. [8]). These test programs serve as examples for using the subroutines, verify their correctness and demonstrate their efficiency.

2. The bundle methods

To simplify the description of the bundle methods, we will consider a simpler problem written in the following compact form

$$x^\star = \arg\min_{x \in L^n}(f(x)), \tag{1}$$

where

$$L^n = \{x \in R^n : a_j^T x \le b_j, j \in K\}.$$

It is clear that the application of the methods described below to the general problem stated in the previous section is straightforward, but this requires to consider each type of constraint separately as it is realized in the subroutines PBUN and PNEW.

The idea behind the bundle methods is that they use a bundle of information obtained at the points y_j, $j \in J_k$, where $J_k \subset \{1, \ldots, k\}$. The bundle of information serves for building a simple nonsmooth model which is utilized for the 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

$$x_{k+1} = x_k + t_L^k d_k, \quad y_{k+1} = x_k + t_R^k d_k, \tag{2}$$

where $0 \le t_L^k \le t_R^k \le 1$. 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, especially null steps are essential. In short and null steps $t_R^k \neq t_L^k$ holds, i.e. $y_{k+1} \neq x_{k+1}$ and the bundle information is obtained from a larger domain which can include points lying on the opposite sides of a possible discontinuity of the objective function. Difference between the bundle methods described below consists in the choice of the 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.

2.1 The proximal bundle method

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 $\beta_j^k = f(x_k) - f(y_j) - g_j^T(x_k - y_j)$, $j \in J_k$, 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 $x \in L^n$. 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

$$\alpha_j^k = \max\{|\beta_j^k|, \gamma(s_j^k)^\omega\}, \tag{3}$$

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 the subroutines PBUN and PNEW) and ω is the distance measure exponent (parameter MOS of the subroutine PNEW). We suppose that $\omega = 2$ for the proximal bundle method and $\gamma = 0$ in the convex case. Obviously, now $\min_{x \in L^n} \hat{f}_k(x) \leq f(x_k)$ by $\alpha_j^k \geq 0$. In order to respect the above considerations, we can define the following local subproblem for the direction determination

$$d_k = \arg \min_{x_k + d \in L^n} \{\hat{f}_k(x_k + d) + \frac{1}{2} u_k d^T d\},$$

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

The choice of the weights u_k is very important. Weights which are too large imply a small $\|d_k\|$, almost all serious steps and 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 the subroutine PBUN:

- Quadratic interpolation (MET=1): The idea is based on a simplified case $n = 1$ and f quadratic, where u_k represents the second order derivative of f (see [3]). By letting $u_{k+1} = \min\{\max\{u_{k+1}^{int}, u_k/10, u_{min}\}, 1/u_{min}, 10u_k\}$, where u_{min} is a small positive constant, we safeguard our quadratic interpolation (u_{k+1}^{int}) (see [9] 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 $u_{k+1}^{loc} = u_k/x_{min}$, where x_{min} is the estimation (derived empirically) of the one-dimensional minimum of f. We again safeguard u_{k+1}^{loc} similarly as u_{k+1}^{int}.

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

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

$$(d_k, \hat{v}_k) = \arg \min_{(d,\hat{v})\in L} \{\hat{v} + \frac{1}{2}u_k d^T d\}, \tag{4}$$

where

$$L = \{(d, \hat{v}) : -\alpha_j^k + g_j^T d \le e_j \hat{v},\ j \in J_k \cup K\}$$

with α_j^k given by (3), $g_j \in \partial f(y_j)$, $e_j = 1$ for $j \in J_k$ and $\alpha_j^k = b_j - a_j^T x$, $g_j = a_j$, $e_j = 0$ for $j \in K$ (we suppose that $J_k \cap K = \emptyset$, which can be easily assured in the program realization). This quadratic programming subproblem can be efficiently solved by the dual range space method proposed in [4], which is also applied and shortly described in [6].

The above derivation is slightly simplified since aggregation of constraints is not included. In fact we add the element $\{0\}$ to J_k, where $\alpha_0^k = \max\{|f(x_k) - f_0^k|, \gamma(s_0^k)^\omega\}$,

$$\begin{aligned}
f_0^k &= \tilde{f}_0^{k-1} + (g_0^k)^T(x_k - x_{k-1}), \\
s_0^k &= \tilde{s}_0^{k-1} + |x_k - x_{k-1}|, \\
g_0^k &= \sum_{j\in J_{k-1}\setminus\{0\}} \lambda_j^{k-1} g_j + \lambda_0^{k-1} g_0^{k-1}, \\
\tilde{f}_0^{k-1} &= \sum_{j\in J_{k-1}\setminus\{0\}} \lambda_j^{k-1} \left(f(y_j) + g_j^T(x_{k-1} - y_j)\right) + \lambda_0^{k-1} f_0^{k-1}, \\
\tilde{s}_0^{k-1} &= \sum_{j\in J_{k-1}} \lambda_j^{k-1} s_j^{k-1},
\end{aligned}$$

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 $(d_k, \hat{v}_k)$ determined as a solution to the quadratic programming subproblem (4), we can obtain the points (2) using a suitable line search. 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 an interpolation method (bisection if MES=1 or two point quadratic interpolation if MES=2, where MES is the parameter of the subroutines PBUN and PNEW) and a suitable backtracking. Let $0 < m_L < 1/2$, $m_L < m_R < 1$ and $0 < \underline{t} < 1$. If

$$f(x_k + t_i^k d_k) \le f(x_k) + m_L t_i^k v_k, \tag{5}$$

where $v_k = \hat{v}_k + \sum_{j \in J_k} \lambda_j^k \alpha_j^k - \max\{|\tilde{f}_0^k - f(x_k)|, \gamma(\tilde{s}_0^k)^\omega\}$, 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). If

$$-\alpha_{k+1}^{k+1} + g_{k+1}^T d_k \ge m_R v_k, \tag{6}$$

where $\alpha_{k+1}^{k+1} = \max\{|\beta_{k+1}^{k+1}|, \gamma(s_{k+1}^{k+1})^\omega\}$, $\beta_{k+1}^{k+1} = f(x_k + t_L^k d_k) - f(x_k + t_i^k d_k) - (t_L^k - t_i^k) g_{k+1}^T d_k$, $s_{k+1}^{k+1} = \|(t_L^k - t_i^k) d_k\|$ 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.

The iteration is terminated if $-v_k$ is less than the final accuracy tolerance supplied by the user.

2.2 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 $g_j^k = g_j + \rho_j G_j (x_k - y_j)$ 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 (3) again so that $\min_{x \in L^n} \tilde{f}_k(x) \le f(x_k)$. The local subproblem for the direction determination has now the form

$$d_k = \arg \min_{x_k + d \in L^n} \{\tilde{f}_k(x_k + d)\},$$

where no regularizing penalty term is used since the function $\tilde{f}_k(x_k + d)$ already contains second order information. This local subproblem is in fact a nonlinear minimax problem which can be solved by the Lagrange-Newton (see [2]). Thus, we obtain the following (smooth) quadratic programming subproblem

$$(d_k, v_k) = \arg \min_{(d,v) \in L} \{v + \frac{1}{2} d^T W d\}, \tag{7}$$

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

$$L = \{(d, v) : -\alpha_j^k + (g_j^k)^T d \le e_j v, j \in J_k \cup K\}$$

with α_j^k given by (3), $g_j^k = g_j + \rho_j G_j(x_k - y_j)$, $e_j = 1$ for $j \in J_k$ and $\alpha_j^k = b_j - a_j^T x$, $g_j^k = a_j$, $e_j = 0$ for $j \in K$ (we suppose that $J_k \cap K = \emptyset$, which can be easily assured in the program realization). This quadratic programming subproblem can be efficiently solved by the dual range space method proposed in [4], which is also applied and shortly described in [6].

The above derivation is not full since the aggregation of constraints is not included. Aggregation of constraints is based on the same principle that was used in the proximal bundle method. We refer to [5] for details.

Having the pair (d_k, v_k) determined as a solution to the quadratic programming subproblem (7), we can obtain the points (2) using a line search which is in fact the same as in the proximal bundle method. Again, conditions (5) and (6) are used, where now $\alpha_{k+1}^{k+1} = \max\{|\beta_{k+1}^{k+1}|, \gamma(s_{k+1}^{k+1})^\omega\}$, $\beta_{k+1}^{k+1} = f(x_k + t_L^k d_k) - f(x_k + t_i^k d_k) - (t_L^k - t_i^k)(g_{k+1}^{k+1})^T d_k - (\rho_{k+1}/2)(t_L^k - t_i^k)^2 d_k^T G(x_k + t_i^k d_k) d_k$, $s_{k+1}^{k+1} = \|(t_L^k - t_i^k) d_k\|$ and $g_{k+1}^{k+1} = g(x_k + t_i^k d_k) + \rho_{k+1}(t_L^k - t_i^k) G(x_k + t_i^k d_k) d_k$. At the same time $g(x_k + t_i^k d_k) \in \partial f(x_k + t_i^k d_k)$ 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 also the same as in the proximal bundle method.

In the above text we use damping parameters ρ_j, $j \in J_k$. In fact, the value $\rho_j = 1$ is used in most iterations. If many nonserious (short and null) iterations would appear, then we set $\rho_j = 0$ since quadratic model is inefficient in this case.

3. Description of the subroutines

In this section we describe all subroutines which can be called from the user's programs. In the description of formal parameters we introduce a type of the argument that specifies whether the argument must have a value defined on subroutine entry (I), or whether it is a value which will be returned (O), or both (U), or whether it is an auxiliary value (A). Note that the arguments of the type I can be changed on output in some circumstances, especially if improper input values were given. Besides formal parameters, we can 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
```

The arguments have the following meanings.

Argument	Type	Significance
NDECF	O	Positive INTEGER variable that indicates the number of matrix decompositions.
NRES	O	Positive INTEGER variable that indicates the number of restarts.

NRED	O	Positive INTEGER variable that indicates the number of reductions.
NREM	O	Positive INTEGER variable that indicates the number of constraint deletions during the QP solutions.
NADD	O	Positive INTEGER variable that indicates the number of constraint additions during the QP solutions.
NIT	O	Positive INTEGER variable that indicates the number of iterations.
NFV	O	Positive INTEGER variable that indicates the number of function evaluations.
NFG	O	Positive INTEGER variable that specifies the number of gradient evaluations.
NFH	O	Positive INTEGER variable that specifies the number of Hessian evaluations.

3.1 Subroutines PBUNU, PBUNS, PBUNL, PNEWU, PNEWS, PNEWL

The calling sequences are

```
      CALL PBUNU(NF,NA,X,IA,RA,IPAR,RPAR,FP,GMAX,ITERM)

      CALL PBUNS(NF,NA,NB,X,IX,XL,XU,IA,RA,IPAR,RPAR,FP,GMAX,ITERM)

      CALL PBUNL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,IA,RA,IPAR,RPAR,
     & FP,GMAX,ITERM)

      CALL PNEWU(NF,NA,X,IA,RA,IPAR,RPAR,FP,GMAX,IHES,ITERM)

      CALL PNEWS(NF,NA,NB,X,IX,XL,XU,IA,RA,IPAR,RPAR,FP,GMAX,IHES,ITERM)

      CALL PNEWL(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,IA,RA,IPAR,RPAR,
     & FP,GMAX,IHES,ITERM)
```

The arguments have the following meanings.

Argument	Type	Significance
NF	I	Positive INTEGER variable that specifies the number of variables of the objective function.
NA	I	INTEGER variable that specifies the maximum bundle dimension (NA ≥ 2). The choice NA $= 0$ causes that the default value NA =NF+3 will be taken.
NB	I	INTEGER variable that specifies whether the simple bounds are suppressed (NB $= 0$) or accepted (NB $=$ NF).

NC	I	INTEGER variable that specifies the number of linear constraints; if NC = 0 the linear constraints are suppressed.
X(NF)	U	On input, DOUBLE PRECISION vector with the initial estimate to the solution. On output, the approximation to the minimum.
IX(NF)	I	INTEGER vector which contains the simple bounds types (significant only if NB > 0): IX(I) = 0: the variable X(I) is unbounded, IX(I) = 1: the lower bound X(I) ≥ XL(I), IX(I) = 2: the upper bound X(I) ≤ XU(I), IX(I) = 3: the two side bound XL(I) ≤ X(I) ≤ XU(I), IX(I) = 5: the variable X(I) is fixed (given by its initial estimate).
XL(NF)	I	DOUBLE PRECISION vector with lower bounds for variables (significant only if NB > 0).
XU(NF)	I	DOUBLE PRECISION vector with upper bounds for variables (significant only if NB > 0).
CF(NC)	A	DOUBLE PRECISION vector containing values of the constraint functions (only if NC > 0).
IC(NC)	I	INTEGER vector containing the constraints types (significant only if NC > 0): IC(K) = 0: the constraint CF(K) is not used, IC(K) = 1: the lower constraint CF(K) ≥ CL(K), IC(K) = 2: the upper constraint CF(K) ≤ CU(K), IC(K) = 3: the two side constraint CL(K) ≤ CF(K) ≤ CU(K), IC(K) = 5: the equality constraint CF(K) = CL(K).
CL(NC)	I	DOUBLE PRECISION vector with lower bounds for constraint functions (significant only if NC > 0).
CU(NC)	I	DOUBLE PRECISION vector with upper bounds for constraint functions (significant only if NC > 0).
CG(NF*NC)	I	DOUBLE PRECISION matrix whose columns are normals of the linear constraints (significant only if NC > 0).
IA(NIA)	A	INTEGER working array of the dimension of at least NIA=NF+NA+1.
RA(NRA)	A	DOUBLE PRECISION working array of the dimension NRA, where at least NRA=NF*(NF+1)/2+NF*(NA+5)+ 5*NA+4 for the subroutines PBUNU, PBUNS, PBUNL and at least NRA=NF*(NF+1)*(NA+3)/2+NF*(NA+6)+ 6*NA+4 for the subroutines PNEWU, PNEWS, PNEWL.

IPAR(7)	A	INTEGER parameters. IPAR(1)=MET for the subroutines PBUNU, PBUNS, PBUNL, IPAR(1)=MOS for the subroutines PNEWU, PNEWS, PNEWL, IPAR(2)=MES, IPAR(3)=MTESX, IPAR(4)=MTESF, IPAR(5)=MIT, IPAR(6)=MFV, IPAR(7)=IPRNT. These parameters (MET, MOS, MES, MTESX, MTESF, MIT, MFV, IPRNT) are described in Section 3.2.
RPAR(9)	A	DOUBLE PRECISION parameters. RPAR(1)=TOLX, RPAR(2) =TOLF, RPAR(3)=TOLB, RPAR(4)=TOLG, RPAR(5)=TOLD, RPAR(6) =TOLS, RPAR(7)=TOLP, RPAR(8)=ETA, RPAR(9)=XMAX. These parameters (TOLX, TOLF, TOLG, TOLB, TOLD, TOLS, TOLP, ETA, XMAX) are described in Section 3.2.
FP	O	DOUBLE PRECISION value of the objective function at the solution X.
GMAX	O	DOUBLE PRECISION maximum absolute value of a partial derivative of the Lagrangian function.
IHES	I	INTEGER variable that specifies a way for computing second derivatives: IHES = 0: numerical computation, IHES = 1: analytical computation by the user supplied subroutine HES.
ITERM	O	INTEGER 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 a standard termination criterion (see Section 2) is satisfied, ITERM = 11: if NFV exceeded MFV, ITERM = 12: if NIT exceeded MIT, ITERM $<$ 0: if the method failed.

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

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

The subroutines PNEWU, PNEWS, PNEWL require the additional user supplied subroutine HES that defines a matrix of the second order information (usually the Hessian matrix) and has the form

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

The arguments of user supplied subroutines have the following meaning.

Argument	Type	Significance
NF	I	Positive INTEGER variable that specifies the number of variables of the objective function.
X(NF)	I	DOUBLE PRECISION an estimate to the solution.
F	O	DOUBLE PRECISION value of the objective function at the point X.
G(NF)	O	DOUBLE PRECISION subgradient of the objective function at the point X.
H(NF*(NF+1)/2)	O	DOUBLE PRECISION matrix of the second order information at the point X.

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

3.2 Subroutines PBUN, PNEW

The general subroutine PBUN is called from the subroutines PBUNU, PBUNS, PBUNL described in Section 3.1. The calling sequence is

```
      CALL PBUN(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,AF,IA,AFD,AG,
     & IAA,AR,AZ,G,H,S,XO,GO,XS,GS,TOLX,TOLF,TOLB,TOLG,TOLD,TOLS,TOLP,
     & ETA,XMAX,GMAX,FP,MET,MES,MTESX,MTESF,MIT,MFV,IPRNT,ITERM).
```

The general subroutine PNEW is called from the subroutines PNEWU, PNEWS, PNEWL described in Section 3.1. The calling sequence is

```
      CALL PNEW(NF,NA,NB,NC,X,IX,XL,XU,CF,IC,CL,CU,CG,AF,IA,AFD,AG,
     & IAA,AR,AZ,G,H,HF,AH,S,SO,XO,GO,TOLX,TOLF,TOLB,TOLG,TOLD,TOLS,TOLP,
     & ETA,XMAX,GMAX,FP,MOS,MES,MTESX,MTESF,MIT,MFV,IPRNT,IHES,ITERM).
```

The arguments NF, NA, NB, NC, X, IX, XL, XU, CF, IC, CL, CU, CG, GMAX, FP, IHES, ITERM have the same meaning as in Section 3.1. Other arguments have the following meanings.

Argument	Type	Significance
AF(NAF)	A	DOUBLE PRECISION vector of bundle function values; NAF=4*NA in the case of the subroutine PBUN or NAF=5*NA in the case of the subroutine PNEW.
IA(NA)	A	INTEGER vector which contains types of bundle functions.
AFD(NA)	A	DOUBLE PRECISION vector of bundle functions increments.
AG(NF*NA)	A	DOUBLE PRECISION matrix whose columns are bundle gradients.

`IAA(NA)`	A	INTEGER vector which contains indices of active functions.
`AR((NF+1)*(NF+2)/2)`	A	DOUBLE PRECISION matrix which contains triangular decomposition of kernel of the orthogonal projection.
`AZ(NF+1)`	A	DOUBLE PRECISION vector of Lagrange multipliers.
`G(NF)`	A	DOUBLE PRECISION subgradient of the objective function.
`H(NH)`	A	DOUBLE PRECISION diagonal matrix of weight parameters (`NH`=`NF`) in the case of the subroutine PBUN or aggregate Hessian matrix (`NH`=`NF*(NF+1)/2`) in the case of the subroutine PNEW.
`HF(NF*(NF+1)/2`	A	DOUBLE PRECISION Hessian matrix of the objective function.
`AH(NA*NF*(NF+1)/2`	A	DOUBLE PRECISION Bundle of Hessian matrices.
`S(NF+1)`	A	DOUBLE PRECISION direction vector.
`SO(NF)`	A	DOUBLE PRECISION auxiliary vector.
`XO(NF)`	A	DOUBLE PRECISION vector containing increments of variables.
`GO(NF+1)`	A	DOUBLE PRECISION gradient of the Lagrangian function.
`XS(NF)`	A	DOUBLE PRECISION auxiliary vector.
`GS(NF)`	A	DOUBLE PRECISION auxiliary vector.
`TOLX`	I	DOUBLE PRECISION tolerance for the change of the coordinate vector `X`; the choice `TOLX` $= 0$ causes that the default value 10^{-16} will be taken.
`TOLF`	I	DOUBLE PRECISION tolerance for the change of function values; the choice `TOLF` $= 0$ causes that the default value 10^{-8} will be taken.
`TOLB`	I	DOUBLE PRECISION minimum acceptable function value; the choice `TOLB` $= 0$ causes that the default value -10^{60} will be taken.
`TOLG`	I	DOUBLE PRECISION tolerance for the gradient of the Lagrangian function; the choice `TOLG` $= 0$ causes that the default value 10^{-6} will be taken.
`TOLD`	I	DOUBLE PRECISION tolerance for a descent direction; the choice `TOLD` $= 0$ causes that the default value 10^{-4} will be taken.
`TOLS`	I	DOUBLE PRECISION tolerance parameter for a function decrease in the line search; the choice `TOLS` $= 0$ causes that the default value 10^{-2} will be taken.

TOLP	I	DOUBLE PRECISION tolerance parameter for a significant modification of the next line search direction; the choice TOLP = 0 causes that the default value 0.5 will be taken.
ETA	I	DOUBLE PRECISION distance measure parameter.
XMAX	I	DOUBLE PRECISION maximum stepsize; the choice XMAX = 0 causes that the default value 10^3 will be taken.
MET	I	INTEGER 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.
MOS	I	INTEGER distance measure exponent (MOS = 1 or MOS = 2). The choice MOS = 0 causes that the default value MOS = 1 will be taken.
MES	I	INTEGER variable that specifies the interpolation method selection in a line search (until a sufficient function decrease is reached; then only bisection will be used): MES = 1: bisection, MES = 2: two point quadratic interpolation. The choice MES = 0 causes that the default value MES = 2 will be taken.
MTESX	I	INTEGER variable that specifies the 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	I	INTEGER variable that specifies the 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.
MIT	I	INTEGER variable that specifies the maximum number of iterations; the choice MIT = 0 causes that the default value 200 will be taken.
MFV	I	INTEGER variable that specifies the maximum number of function evaluations; the choice MFV = 0 causes that the default value 500 will be taken.
IPRNT	I	INTEGER variable that specifies PRINT: 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,

Subroutines PBUN and PNEW have a modular structure. The following list contains their most important subroutines.

- `UF1HS1` Numerical computation of the Hessian matrix.
- `PDDBQ1` Determination of the descent direction using quadratic programming routines and bundle updating for the subroutine PBUN.
- `PDDBQ2` Determination of the descent direction using quadratic programming routines and bundle updating for the subroutine PNEW.
- `PLQDF1` Dual range space method for solving the quadratic programming problem with linear constraints (see [4]) and [6].
- `PS1L05` Line search using function values and derivatives.

Subroutines PBUN, PNEW require the user supplied subroutine FUNDER. Subroutine PNEW requires the additional user supplied subroutine HES. User supplied subroutines FUNDER and HES are described in Section 3.1.

3.3 Form of the printed results

The form of the printed results is specified by the parameter IPRNT as is described above. Here we demonstrate individual forms of printed results by the simple use of the program TNEWU described in the next section (with NEXT=12). If we set IPRNT=1, then the printed results will have the form

```
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06  ITERM=  4
```

If we set IPRNT=-1, then the printed results will have the form

```
EXIT FROM PNEW :
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06  ITERM=  4
X= -.1262566D+00 -.3437830D-01 -.6857198D-02  .2636066D-01  .6729492D-01
   -.2783995D+00  .7421866D-01  .1385240D+00  .8403122D-01  .3858031D-01
```

If we set IPRNT=2, then the printed results will have the form

```
ENTRY TO PNEW :
NIT=   0  NFV=   1  NFG=   1  F=  .00000000D+00  G= .1000D+61
NIT=   1  NFV=   3  NFG=   3  F=  .53370664D+04  G= .1200D+05
NIT=   2  NFV=   4  NFG=   4  F=  .66499712D+02  G= .2610D+02
NIT=   3  NFV=   5  NFG=   5  F=  .33934270D+02  G= .3771D+02
NIT=   4  NFV=   6  NFG=   6  F=  .12040341D+01  G= .3214D+01
NIT=   5  NFV=   7  NFG=   7  F=  .51324695D+00  G= .1459D+01
NIT=   6  NFV=   8  NFG=   8  F= -.76915236D+00  G= .2347D+01
NIT=   7  NFV=   9  NFG=   9  F= -.83859100D+00  G= .2683D+00
```

```
NIT=   8  NFV=  10  NFG=  10  F= -.84140726D+00  G= .2491D-02
NIT=   9  NFV=  11  NFG=  11  F= -.84140726D+00  G= .3213D-03
NIT=  10  NFV=  12  NFG=  12  F= -.84140726D+00  G= .4862D-03
NIT=  11  NFV=  13  NFG=  13  F= -.84140726D+00  G= .5265D-05
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06
EXIT FROM PNEW :
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06  ITERM=  4
```

If we set IPRNT=-2, then the printed results will have the form

```
ENTRY TO PNEW :
NIT=   0  NFV=   1  NFG=   1  F=  .00000000D+00  G= .1000D+61
NIT=   1  NFV=   3  NFG=   3  F=  .53370664D+04  G= .1200D+05
NIT=   2  NFV=   4  NFG=   4  F=  .66499712D+02  G= .2610D+02
NIT=   3  NFV=   5  NFG=   5  F=  .33934270D+02  G= .3771D+02
NIT=   4  NFV=   6  NFG=   6  F=  .12040341D+01  G= .3214D+01
NIT=   5  NFV=   7  NFG=   7  F=  .51324695D+00  G= .1459D+01
NIT=   6  NFV=   8  NFG=   8  F= -.76915236D+00  G= .2347D+01
NIT=   7  NFV=   9  NFG=   9  F= -.83859100D+00  G= .2683D+00
NIT=   8  NFV=  10  NFG=  10  F= -.84140726D+00  G= .2491D-02
NIT=   9  NFV=  11  NFG=  11  F= -.84140726D+00  G= .3213D-03
NIT=  10  NFV=  12  NFG=  12  F= -.84140726D+00  G= .4862D-03
NIT=  11  NFV=  13  NFG=  13  F= -.84140726D+00  G= .5265D-05
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06
EXIT FROM PNEW :
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06  ITERM=  4
X= -.1262566D+00 -.3437830D-01 -.6857198D-02  .2636066D-01  .6729492D-01
   -.2783995D+00  .7421866D-01  .1385240D+00  .8403122D-01  .3858031D-01
```

4. Verification of the subroutines

In this section we introduce the main programs TBUNU and TNEWU, which serve as demonstration, verification and testing of the subroutines PBUNU and PNEWU.

4.1 Program TBUNU

The following main program demonstrates the usage of the subroutine PBUNU.

```
C
C     TEST PROGRAM FOR THE SUBROUTINE PBUNU
C
      INTEGER NF,NA,IA(100),IPAR(7),ITERM
      REAL*8 X(30),RA(2000),RPAR(9),GMAX,F
      REAL*8 FMIN
```

```
      INTEGER NEXT,IERR,I
      COMMON /PROB/ NEXT
      INTEGER NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
      COMMON /STAT/ NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
C
C     LOOP FOR 23 TEST PROBLEMS
C
      DO 3 NEXT=1,23
C
C     CHOICE OF INTEGER AND REAL PARAMETERS
C
      DO 1 I=1,7
      IPAR(I)=0
    1 CONTINUE
      DO 2 I=1,9
      RPAR(I)=0.0D 0
    2 CONTINUE
      IF (NEXT.LE.8.OR.NEXT.EQ.17.OR.NEXT.GE.21) RPAR(8)=0.25D 0
      IF (NEXT.EQ.1.OR.NEXT.EQ.6.OR.NEXT.EQ.8.OR.NEXT.EQ.15) IPAR(1)=2
      IF (NEXT.GE.17) IPAR(1)=2
      IF (NEXT.EQ.14) IPAR(4)=7
      IPAR(7)=1
C
C     PROBLEM DIMENSION
C
      NF=30
      NA=0
C
C     INITIATION OF X AND CHOICE OF RPAR(9)
C
      CALL TIUD19(NF,X,FMIN,RPAR(9),NEXT,IERR)
      IF (IERR.NE.0) GO TO 3
C
C     SOLUTION
C
      CALL PBUNU(NF,NA,X,IA,RA,IPAR,RPAR,F,GMAX,ITERM)
    3 CONTINUE
      STOP
      END
C
C     USER SUPPLIED SUBROUTINE (CALCULATION OF F AND G)
C
      SUBROUTINE FUNDER(NF,X,F,G)
```

```
      INTEGER NF
      REAL*8 X(*),F,G(*)
      INTEGER NEXT
      COMMON /PROB/ NEXT
C
C     FUNCTION EVALUATION
C
      CALL TFFU19(NF,X,F,NEXT)
C
C     GRADIENT EVALUATION
C
      CALL TFGU19(NF,X,G,NEXT)
      RETURN
      END
```

This main program uses subroutines TIUD19 (initiation), TFFU19 (function evaluation) and TFGU19 (subgradient evaluation) containing 21 standard test problems, which have at most 30 variables, taken from the UFO system [7]. Results obtained by this main program have the following form.

```
NIT=  42  NFV=  45  NFG=  45  F=  .38117068D-06  G= .1135D-02  ITERM=  2
NIT=  18  NFV=  20  NFG=  20  F=  .22292495D-15  G= .6552D-08  ITERM=  2
NIT=  31  NFV=  33  NFG=  33  F=  .19522245D+01  G= .3085D-03  ITERM=  2
NIT=  14  NFV=  16  NFG=  16  F=  .20000000D+01  G= .1921D-06  ITERM=  2
NIT=  17  NFV=  19  NFG=  19  F= -.30000000D+01  G= .5564D-08  ITERM=  4
NIT=  13  NFV=  15  NFG=  15  F=  .72000015D+01  G= .2212D-02  ITERM=  4
NIT=  11  NFV=  12  NFG=  12  F= -.14142136D+01  G= .1437D-04  ITERM=  4
NIT=  66  NFV=  68  NFG=  68  F= -.99999941D+00  G= .1089D-02  ITERM=  4
NIT=  13  NFV=  15  NFG=  15  F= -.10000000D+01  G= .9859D-07  ITERM=  4
NIT=  43  NFV=  45  NFG=  45  F= -.43999999D+02  G= .3734D-02  ITERM=  2
NIT=  27  NFV=  29  NFG=  29  F=  .22600162D+02  G= .1451D-03  ITERM=  4
NIT=  74  NFV=  75  NFG=  75  F= -.84140829D+00  G= .7236D-03  ITERM=  2
NIT= 150  NFV= 151  NFG= 151  F=  .16712381D-06  G= .7782D-04  ITERM=  2
NIT=  39  NFV=  40  NFG=  40  F=  .27274665D-12  G= .1000D+01  ITERM=  2
NIT=  92  NFV=  93  NFG=  93  F=  .55981567D+00  G= .8266D-03  ITERM=  4
NIT=  43  NFV=  46  NFG=  46  F= -.80000000D+01  G= .1282D-02  ITERM=  4
NIT=  18  NFV=  19  NFG=  19  F=  .42366324D-08  G= .9986D-07  ITERM=  2
NIT=  19  NFV=  20  NFG=  20  F=  .98993016D-09  G= .1024D-07  ITERM=  2
NIT= 160  NFV= 162  NFG= 162  F=  .97857723D+01  G= .1618D-03  ITERM=  4
NIT= 104  NFV= 113  NFG= 113  F=  .16703883D+02  G= .8670D-02  ITERM=  4
NIT=  60  NFV=  62  NFG=  62  F= -.32348679D+02  G= .2190D-02  ITERM=  2
```

The 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 criterion for the termination G and the cause of termination ITERM.

4.2 Program TNEW

The following main program demonstrates the usage of the subroutine PNEWU.

```
C
C     TEST PROGRAM FOR THE SUBROUTINE PNEWU
C
      INTEGER NF,NA,IA(100),IPAR(7),ITERM
      REAL*8 X(40),RA(19000),RPAR(9),F,GMAX
      REAL*8 FMIN
      INTEGER NEXT,IERR,I
      COMMON /PROB/ NEXT
      INTEGER NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
      COMMON /STAT/ NDECF,NRES,NRED,NREM,NADD,NIT,NFV,NFG,NFH
C
C     LOOP FOR 23 TEST PROBLEMS
C
      DO 3 NEXT=1,23
C
C     CHOICE OF INTEGER AND REAL PARAMETERS
C
      DO 1 I=1,7
      IPAR(I)=0
    1 CONTINUE
      DO 2 I=1,9
      RPAR(I)=0.0D 0
    2 CONTINUE
      IPAR(1)=2
      IPAR(2)=4
      RPAR(8)=1D-10
      IF(IPAR(1).EQ.1)THEN
        IF(NEXT.EQ.1) RPAR(8)=0.5D 0
        IF(NEXT.EQ.3.OR.NEXT.EQ.21) RPAR(8)=0.25D 0
        IF(NEXT.EQ.5.OR.NEXT.EQ.8) RPAR(8)=1.0D-1
        IF(NEXT.EQ.4.OR.NEXT.EQ.22) RPAR(8)=1.0D-2
        IF(NEXT.EQ.2.OR.NEXT.EQ.12.OR.NEXT.EQ.18) RPAR(8)=1.0D-4
        IF(NEXT.EQ.23) RPAR(8)=4.0D-2
        IF(NEXT.EQ.24) RPAR(8)=5.0D-2
      ELSE
        IF(NEXT.EQ.1) RPAR(8)=1.3D 0
        IF(NEXT.EQ.5.OR.NEXT.EQ.18.OR.NEXT.EQ.21.OR.NEXT.EQ.23)
```

```
     &  RPAR(8)=0.25D0
        IF(NEXT.EQ.3.OR.NEXT.EQ.4) RPAR(8)=1.0D-1
        IF(NEXT.EQ.24) RPAR(8)=5.0D-2
        IF(NEXT.EQ.8.OR.NEXT.EQ.12.OR.NEXT.EQ.22) RPAR(8)=1.0D-2
        IF(NEXT.EQ.2.OR.NEXT.EQ.17) RPAR(8)=1.0D-3
      ENDIF
      IPAR(7)=1
C
C     PROBLEM DIMENSION
C
      NF=30
      NA=0
C
C     INITIATION OF X AND CHOICE OF RPAR(9)
C
      CALL TIUD19(NF,X,FMIN,RPAR(9),NEXT,IERR)
      IF(NEXT.EQ.3.OR.NEXT.EQ.5.OR.NEXT.EQ.21) RPAR(9)=1.0D0
      IF(NEXT.EQ.18) RPAR(9)=1.0D1
      IF (IERR.NE.0) GO TO 3
      IHES=1
C
C     SOLUTION
C
      CALL PNEWU(NF,NA,X,IA,RA,IPAR,RPAR,F,GMAX,IHES,ITERM)
    3 CONTINUE
      STOP
      END
C
C     USER SUPPLIED SUBROUTINE (CALCULATION OF F AND G)
C
      SUBROUTINE FUNDER(NF,X,F,G)
      INTEGER NF
      REAL*8 X(*),F,G(*)
      INTEGER NEXT
      COMMON /PROB/ NEXT
C
C     FUNCTION EVALUATION
C
      CALL TFFU19(NF,X,F,NEXT)
C
C     GRADIENT EVALUATION
C
      CALL TFGU19(NF,X,G,NEXT)
```

```
      RETURN
      END
C
C     USER SUPPLIED SUBROUTINE (CALCULATION OF H)
C
      SUBROUTINE HES(NF,X,H)
      INTEGER NF
      REAL*8 X(*),H(*)
      INTEGER NEXT
      COMMON /PROB/ NEXT
C
C     HESSIAN EVALUATION
C
      CALL TFHD19(NF,X,H,NEXT)
      RETURN
      END
```

This main program uses subroutines TIUD19 (initiation), TFFU19 (function evaluation), TFGU19 (subgradient evaluation) and TFHD19 (Hessian matrix evaluation) containing 21 standard test problems with at most 30 variables, which were taken from the UFO system [7]. Results obtained by this main program have the following form.

```
NIT=  58  NFV=  59  NFG=  59  F=  .22533114D-15  G= .8624D-05  ITERM=  2
NIT=   7  NFV=   8  NFG=   8  F=  .16765701D-10  G= .5792D-05  ITERM=  4
NIT=   9  NFV=  10  NFG=  10  F=  .19522245D+01  G= .1544D-04  ITERM=  4
NIT=  11  NFV=  12  NFG=  12  F=  .20000000D+01  G= .1560D-03  ITERM=  4
NIT=  14  NFV=  15  NFG=  15  F= -.30000000D+01  G= .5398D-08  ITERM=  2
NIT=   4  NFV=   6  NFG=   6  F=  .72000000D+01  G= .1445D-08  ITERM=  4
NIT=  16  NFV=  17  NFG=  17  F= -.14142136D+01  G= .5653D-07  ITERM=  4
NIT=  12  NFV=  14  NFG=  14  F= -.10000000D+01  G= .2015D-08  ITERM=  4
NIT=  10  NFV=  11  NFG=  11  F= -.10000000D+01  G= .4562D-06  ITERM=  4
NIT=  13  NFV=  15  NFG=  15  F= -.44000000D+02  G= .4215D-05  ITERM=  4
NIT=   7  NFV=   8  NFG=   8  F=  .22600173D+02  G= .1263D-02  ITERM=  4
NIT=  12  NFV=  14  NFG=  14  F= -.84140833D+00  G= .6734D-06  ITERM=  4
NIT=  36  NFV=  37  NFG=  37  F=  .38373702D-08  G= .5758D-08  ITERM=  2
NIT=  24  NFV=  25  NFG=  25  F=  .45289427D-08  G= .1100D-09  ITERM=  2
NIT=  89  NFV=  91  NFG=  91  F=  .55981330D+00  G= .1528D-05  ITERM=  4
NIT=  25  NFV=  26  NFG=  26  F= -.79999999D+01  G= .3813D-02  ITERM=  4
NIT=  14  NFV=  15  NFG=  15  F=  .48008312D-08  G= .7723D-07  ITERM=  2
NIT=  13  NFV=  14  NFG=  14  F=  .11049975D-08  G= .3952D-07  ITERM=  2
NIT=  52  NFV=  53  NFG=  53  F=  .97857721D+01  G= .2964D-03  ITERM=  4
NIT=  40  NFV=  42  NFG=  42  F=  .16703855D+02  G= .1783D+00  ITERM=  4
NIT=  22  NFV=  24  NFG=  24  F= -.32348679D+02  G= .3409D-02  ITERM=  4
```

The 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 criterion for the termination G and the cause of termination ITERM.

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

[4] 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.

[5] Lukšan L., Vlček J. A Bundle-Newton Method for Nonsmooth Unconstrained Minimization. To appear in Mathematical Programming.

[6] Lukšan L., Vlček J. Algorithm AAA. PMIN - A Recursive Quadratic Programming Variable Metric Algorithm for Minimax Optimization. Submitted to ACM Trans. on Math. Software.

[7] Lukšan L., Šiška M., Tůma M., Vlček J., Ramešová N. Interactive System for Universal Functional Optimization (UFO), Version 1996. Research Report No. V-701, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 1996.

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

[9] 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.