Ladislav Lukšan

Efficient trust region method for nonlinear least squares

Kybernetika, Vol. 32 (1996), No. 2, 105--120

Persistent URL: http://dml.cz/dmlcz/124177

KYBERNETIKA — VOLUME 32 (1996), NUMBER 2, PAGES 105 - 120

EFFICIENT TRUST REGION METHOD FOR NONLINEAR LEAST SQUARES[1]

LADISLAV LUKŠAN

The main purpose of this paper is to show that suitable transformations and decompositions lead to an efficient trust region method that uses one decomposition in each iteration only. Convergence properties of the resulting algorithm are comparable with convergence properties of the trust region method with optimal locally constrained step (OLCS) that uses more than one decomposition in each iteration and, therefore, that needs a longer time for obtaining results. This fact is demonstrated by numerical experiments.

1. INTRODUCTION

Let $f_i : I\!R^n \to I\!R$, $1 \leq i \leq m$, be real-valued functions with continuous second order derivatives on the open set $X \subset I\!R^n$. Let us denote

$$F(x) = \frac{1}{2} \sum_{i=1}^{m} f_i^2(x). \tag{1.1}$$

We are concerned with the finding of a local minimum $x^* \in I\!R^n$ of the function $F : I\!R^n \to I\!R$ given by (1.1) on an open set $X \subset I\!R^n$, i.e. a point $x^* \in I\!R^n$ that satisfies the inequality $F(x^*) \leq F(x)$ $\forall x \in B(x^*, \varepsilon)$ for some $\varepsilon > 0$, where $B(x^*, \varepsilon) = \{x \in I\!R^n : \|x - x^*\| < \varepsilon\} \subset X$ is an open ball contained in $X \subset I\!R^n$.

If we denote $g_i(x)$ and $G_i(x)$ the gradients and the Hessian matrices of the functions $f_i : I\!R^n \to I\!R$, $1 \leq i \leq m$, respectively, and $g(x)$ and $G(x)$ the gradient and the Hessian matrix of the function $F : I\!R^n \to I\!R$ respectively then, using (1.1), we obtain

$$g(x) = \sum_{i=1}^{m} f_i(x)\, g_i(x) \tag{1.2}$$

and

$$G(x) = \sum_{i=1}^{m} g_i(x)\, g_i^T(x) + \sum_{i=1}^{m} f_i(x)\, G_i(x). \tag{1.3}$$

[1] This work was supported under the grant No. 23012 given by the Grant Agency of the Academy of Sciences of the Czech Republic.

Numerical methods for local minimization of the objective function $F : \mathbb{R}^n \to \mathbb{R}$ are usually derived from the Newton method. These methods are iterative and their iteration step has the form

$$x^+ = x + d, \tag{1.4}$$

where x and x^+ are old and new vectors of variables respectively and $d = x^+ - x$ is the direction vector obtained as a minimizer of the quadratic function

$$Q(d) = \frac{1}{2} d^T B d + g^T d, \tag{1.5}$$

where B is an approximation of the Hessian matrix $G(x)$ and $g = g(x)$.

If the objective function $F : \mathbb{R}^n \to \mathbb{R}$ has the form (1.1) then we usually choose

$$B = \sum_{i=1}^{m} g_i(x)\, g_i^T(x). \tag{1.6}$$

One reason for this choice is the fact that often $F(x^*) = 0$ so that the second term of (1.3) is negligible in $B(x^*, \varepsilon)$. Another reason follows from the linearization of (1.1). In this case

$$\begin{aligned} F(x^+) &\approx \frac{1}{2} \sum_{i=1}^{m} (f_i(x) + g_i^T(x)\, d)^2 \\ &= \frac{1}{2} \sum_{i=1}^{m} (f_i^2(x) + 2 f_i(x)\, g_i^T(x)\, d + d^T g_i(x)\, g_i^T(x)\, d) \\ &= F(x) + g^T(x)\, d + \tfrac{1}{2}\, d^T B d = F(x) + Q(d) \end{aligned}$$

with B given by (1.6). The method which uses the matrix (1.6) instead of the Hessian matrix $G(x)$ is called the Gauss–Newton method.

The simple realization (1.4) of the Gauss–Newton method is unusable since it is not globally convergent. One approach which guarantee the global convergence of the Gauss–Newton method is based on the line-search strategy. In this case

$$x^+ = x + \alpha d, \tag{1.7}$$

where α is a positive stepsize chosen so that

$$F^+ - F \leq \varepsilon_1 \alpha d^T g \tag{1.8a}$$

and

$$d^T g^+ \geq \varepsilon_2 d^T g \tag{1.8b}$$

with $0 < \varepsilon_1 < 1/2$ and $\varepsilon_1 < \varepsilon_2 < 1$. This approach is very efficient for the so-called variable metric methods (see [9]) but it has some disadvantages in connection with the Gauss–Newton method. Frequently the matrix, whose columns are the gradients $g_i(x)$, $1 \leq i \leq m$, has nearly linearly dependent rows so that the matrix (1.6) is nearly singular. This leads to the excessive growth of $\|d\|$ and, therefore, the usual first attempt $\alpha = 1$ in (1.7) is unsuitable (too many line-search steps are

needed for conditions (1.8) to be fulfilled). Because of this reason the Gauss–Newton method is frequently combined with the variable metric methods (see [2], [1], [5]).

Another approach that makes the Gauss–Newton method to be globally convergent is based on the trust region strategy. In this case the unconstrained minimization of (1.5) is substituted by the constrained problem with the objective function (1.5) and with the constraint

$$||Td|| \leq \Delta, \tag{1.9}$$

where T is some nonsingular transformation matrix. This approach partially appeared already in papers of Levenberg [8] and Marquardt [10] but the current state was started from the papers of Powell [16], Dennis and Mei [3], Hebden [7], Moré [13], Moré and Sorensen [15]. The accurate minimizer $d^* \in \mathbb{R}^n$ of the function (1.5) over the constraint (1.9) is usually replaced by the approximate solution which has to satisfy the conditions

$$||Td|| \leq \delta_2 \Delta \tag{1.10}$$

$$||Td|| \leq \delta_1 \Delta \Rightarrow Bd = -g, \tag{1.11}$$

where $0 < \delta_1 < 1 < \delta_2$, and which has to guarantee a sufficient decrease of the function (1.5) such that

$$Q(d) \leq \delta_1^2 Q(d^*). \tag{1.12}$$

Function (1.5) and the constraint (1.9) can be replaced by

$$\tilde{Q}(\tilde{d}) = \frac{1}{2}\tilde{d}^T \tilde{B}\tilde{d} + \tilde{g}^T \tilde{d}, \tag{1.13}$$

and

$$||\tilde{d}|| \leq \Delta, \tag{1.14}$$

where $\tilde{d} = Td$, $\tilde{g} = (T^T)^{-1}g$, $\tilde{B} = (T^T)^{-1}BT^{-1}$. In this case the conditions (1.10), (1.11), (1.12) have the form

$$||\tilde{d}|| \leq \delta_2 \Delta \tag{1.15}$$

$$||\tilde{d}|| < \delta_1 \Delta \Rightarrow \tilde{B}\tilde{d} = -\tilde{g} \tag{1.16}$$

$$\tilde{Q}(\tilde{d}) \leq \delta_1^2 \tilde{Q}(\tilde{d}^*), \tag{1.17}$$

where $\tilde{d}^* \in \mathbb{R}^n$ is a minimizer of the function (1.13) over the constraint (1.14).

A typical trust region realization of the Gauss–Newton method is represented by the following algorithm:

Algorithm 1.1

Data: $0 < \beta_1 < \beta_2 < 1 < \gamma_1 < \gamma_2$, $0 < \delta_1 < 1 < \delta_2$, $0 < \rho_1 < \rho_2 < 1$, $0 < \varepsilon_1 < \varepsilon_2 < 1$, $0 < \Delta_{\max}$, $k_1 \in N$, $l_1 \in N$.

Step 1: Choose an initial point $x \in \mathbb{R}^n$. Compute the values $f_i := f_i(x)$ of the functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $1 \leq i \leq m$, at the point $x \in \mathbb{R}^n$. Determine the value $F := F(x)$ of the objective function $F : \mathbb{R}^n \to \mathbb{R}$ at the point $x \in \mathbb{R}^n$ by (1.1). Choose an initial trust region bound $0 < \Delta < \Delta_{\max}$ and set $k := 1$.

Step 2: Compute the gradients $g_i := g_i(x)$ of the functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $1 \leq i \leq m$, at the point $x \in \mathbb{R}^n$. Determine the gradient $g := g(x)$ of the objective function $F : \mathbb{R}^n \to \mathbb{R}$ at the point $x \in \mathbb{R}^n$ by (1.2). If either $F \leq \varepsilon_1$ or $\|g\| \leq \varepsilon_2$ then stop, otherwise set $l := 1$.

Step 3: Choose a transformation matrix T and compute the transformed gradient $\tilde{g} := (T^T)^{-1}g$ and the transformed matrix $\tilde{B} := (T^T)^{-1}BT^{-1}$ where B is an approximation of the Hessian matrix given by (1.6). Determine the transformed direction vector $\tilde{d} \in \mathbb{R}^n$ that satisfies the conditions (1.15), (1.16), (1.17) where $\tilde{Q}(\tilde{d})$ is a quadratic function defined by (1.13). Set $d := T^{-1}\tilde{d}$.

Step 4: Set $x^+ := x + d$. Compute the values $f_i^+ := f_i(x^+)$ of the functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $1 \leq i \leq m$, at the point $x^+ \in \mathbb{R}^n$. Determine the value $F^+ := F(x^+)$ of the objective function $F : \mathbb{R}^n \to \mathbb{R}$ at the point $x^+ \in \mathbb{R}^n$ by (1.1). Compute the value $\tilde{Q}(\tilde{d})$ by (1.13) and set $\rho := (F^+ - F)/\tilde{Q}(\tilde{d})$. When $\rho \leq \rho_1$ then compute $\alpha := (F^+ - F)/\tilde{g}^T\tilde{d}$, $\beta := 1/(2(1-\alpha))$ and set $\Delta := \beta_1\|\tilde{d}\|$ if $\beta < \beta_1$, $\Delta := \beta\|\tilde{d}\|$ if $\beta_1 \leq \beta \leq \beta_2$, $\Delta := \beta_2\|\tilde{d}\|$ if $\beta_2 < \beta$. When $\rho_1 \leq \rho \leq \rho_2$ then set $\Delta := \min(\Delta, \gamma_2\|\tilde{d}\|)$. When $\rho_2 < \rho$ then compute $\Delta := \max(\Delta, \gamma_1\|\tilde{d}\|)$ and set $\Delta := \min(\Delta, \gamma_2\|\tilde{d}\|, \Delta_{\max})$.

Step 5: If $\rho \leq 0$ and $l \geq l_1$ then stop (too many reductions). If $\rho \leq 0$ and $l < l_1$ then set $l := l + 1$ and go to Step 3. If $\rho > 0$ and $k \geq k_1$ then stop (too many iterations). If $\rho > 0$ and $k < k_1$ then set $x := x^+$, $F := F^+$, set $k := k + 1$ and go to Step 2.

The maximum number of iterations $k_1 \in N$ serves as an alternative termination criterion in the case when the convergence is too slow. The maximum number of reductions $l_1 \in N$ serves as a safeguard against possible infinite cycle which can arise for large residual problems when a presence of round-off errors do not allow us to obtain a solution with the required gradient norm $\|g\| \leq \varepsilon_2$. Minimizer of the function (1.13) over the constraint (1.14) can be obtained iteratively by means of the Newton method applied to the nonlinear equation

$$\frac{1}{\Delta} - \frac{1}{\|\tilde{d}(\lambda)\|} = 0, \tag{1.18}$$

where

$$(\tilde{B} + \lambda I)\tilde{d}(\lambda) = -\tilde{g} \tag{1.19}$$

(see [15] as an example). This iterative process is terminated whenever the conditions (1.15), (1.16), (1.17) are satisfied. It leads to the repeated solution of the equation (1.19) for selected values of the parameter $\lambda \geq 0$ which can be time consuming in general. The main purpose of this paper is to show that there exists a suitable transformation matrix T, which makes the matrix $\tilde{B}$ to be diagonal (so that the repeated solution of (1.19) is not time consuming), while the resulting trust region method has still good convergence properties.

2. TRANSFORMATIONS, DECOMPOSITIONS, AND TRUST REGION SOLUTIONS

The most time consuming part of Algorithm 1.1 is a repeated solution of the equation (1.19). Consider the decomposition

$$X^{-1}B(X^T)^{-1} + C = PLDL^TP^T, \tag{2.1}$$

where X is a diagonal positive definite scaling matrix, C is a small diagonal positive semidefinite (usually zero) correction matrix, which is chosen automatically during the decomposition process, such that $X^{-1}B(X^T)^{-1} + C$ is positive definite (see [6], [17]), P is a permutation matrix, L is a lower triangular matrix with unit diagonal elements, and D is a diagonal positive definite matrix. The matrix $B + XCX^T$ will be used in Algorithm 1.1 instead of the matrix B, given by (1.6), which may not be positive definite.

Choose a diagonal positive definite weighting matrix Y and set

$$T = Y^TL^TP^TX^T. \tag{2.2}$$

Then

$$\begin{aligned} \tilde{B} &= (T^T)^{-1}(B + XCX^T)T^{-1} \\ &= Y^{-1}L^{-1}P^{-1}X^{-1}XPLDL^TP^TX^T(X^T)^{-1}(P^T)^{-1}(L^T)^{-1}(Y^T)^{-1} \\ &= Y^{-1}D(Y^T)^{-1} \end{aligned} \tag{2.3}$$

is a diagonal positive definite matrix and, therefore, the solution of (1.19) is trivial.

If the vector $\tilde{d}(\lambda)$ is a solution of the equation (1.19) then, by (2.3), the vector $d(\lambda) = T^{-1}\tilde{d}(\lambda)$ is a solution of the equation

$$(B + XCX^T + \lambda T^TT)d(\lambda) = -g. \tag{2.4}$$

Trust region methods for nonlinear least squares are usually designed such a way that $T^TT = E$ is a diagonal positive definite matrix and either

$$E = I \tag{2.5a}$$

or

$$E = \sum_{i=1}^{n} e_ie_i^TBe_ie_i^T, \tag{2.5b}$$

where $e_i, 1 \leq i \leq n$, are columns of the unit matrix. Because $T^TT = XPLYY^TL^TP^TX^T$ by (2.2), it would be advantageous to choose, the matrices X and Y so that $X = E^{1/2}$ and LYY^TL^T be as close to the unit matrix as possible.

Lemma 2.1. Let L be a constant nonsingular matrix and let Z be a diagonal matrix which can vary. Then the function

$$\frac{1}{2}\|LZL^T - I\|_F^2 = \frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n}(e_k^T(LZL^T - I)\,e_l)^2 \tag{2.6}$$

reaches its minimum if and only if the elements $Z_i = e_i^T Z e_i$, $1 \leq i \leq n$, satisfy the equations

$$\sum_{j=1}^{n} (e_i^T L^T L e_j)^2 Z_j = e_i^T L^T L e_i \tag{2.7}$$

for $1 \leq i \leq n$.

Proof. The quadratic function (2.6) is convex since it is bounded from below. Therefore it reaches its minimum if and only if its derivatives are zero. Using (2.6) we get the equation

$$\begin{aligned} \frac{1}{2} \frac{\partial \|LZL^T - I\|_F^2}{\partial z_i} &= \sum_{k=1}^{n} \sum_{l=1}^{n} (e_k^T (LZL^T - I) e_l e_k^T L e_i e_i^T L^T e_l \\ &= \sum_{k=1}^{n} \sum_{l=1}^{n} e_i L^T e_k e_k^T (LZL^T - I) e_l e_l^T L e_i \\ &= e_i^T L^T (LZL^T - I) L e_i = 0 \end{aligned}$$

for $1 \leq i \leq n$, or

$$\sum_{j=1}^{n} e_i^T L^T L e_j e_j^T Z e_j e_j^T L^T L e_i = e_i^T L^T L e_i$$

for $1 \leq i \leq n$, which gives (2.7). □

Lemma 2.1 has no practical significance since the values $Z_i, 1 \leq i \leq n$, obtained from (2.7) can be nonpositive. Moreover, the solution of (2.7) requires another matrix decomposition. Therefore we use a simpler way. We choose either

$$Z = I \tag{2.8a}$$

or

$$Z = \sum_{i=1}^{n} \frac{e_i e_i^T}{e_i^T L^T L e_i} \tag{2.8b}$$

and then we set $Y = Z^{1/2}$. Note that (2.8b) can be obtained by neglecting the off-diagonal elements in (2.7).

Another reasonable condition, which should be satisfied for the matrix $T^T T$ in (2.4), is well-conditioning. If we set $X = Y = I$ in (2.2), then $T^T T = PLL^T P$. Thus well-conditioning usually depends on the permutation matrix P. This is shown in the following example.

Example 2.1. Consider the decomposition

$$B_1 = \begin{bmatrix} \varepsilon^2, & \varepsilon \\ \varepsilon, & 1+\varepsilon \end{bmatrix} = \begin{bmatrix} 1, & 0 \\ \frac{1}{\varepsilon}, & 1 \end{bmatrix} \begin{bmatrix} \varepsilon^2, & 0 \\ 0, & \varepsilon \end{bmatrix} \begin{bmatrix} 1, & \frac{1}{\varepsilon} \\ 0, & 1 \end{bmatrix} = L_1 D_1 L_1^T.$$

Then

$$L_1 L_1^T = \begin{bmatrix} 1, & 0 \\ \frac{1}{\varepsilon}, & 1 \end{bmatrix} \begin{bmatrix} 1, & \frac{1}{\varepsilon} \\ 0, & 1 \end{bmatrix} = \begin{bmatrix} 1, & \frac{1}{\varepsilon} \\ \frac{1}{\varepsilon}, & 1 + \left(\frac{1}{\varepsilon}\right)^2 \end{bmatrix}$$

so that $\kappa(L_1 L_1^T) \to \infty$ as $\varepsilon \to 0$. On the other hand consider the decomposition

$$B_2 = \begin{bmatrix} 1+\varepsilon, & \varepsilon \\ \varepsilon, & \varepsilon^2 \end{bmatrix} = \begin{bmatrix} 1, & 0 \\ \frac{\varepsilon}{1+\varepsilon}, & 1 \end{bmatrix} \begin{bmatrix} 1+\varepsilon, & 0 \\ 0, & \frac{\varepsilon^3}{1+\varepsilon} \end{bmatrix} \begin{bmatrix} 1, & \frac{\varepsilon}{1+\varepsilon} \\ 0, & 1 \end{bmatrix} = L_2 D_2 L_2^T,$$

where $B_2 = P B_1 P$ with $P = [e_2, e_1]$. Then

$$L_2 L_2^T = \begin{bmatrix} 1, & 0 \\ \frac{\varepsilon}{1+\varepsilon}, & 1 \end{bmatrix} \begin{bmatrix} 1, & \frac{\varepsilon}{1+\varepsilon} \\ 0, & 1 \end{bmatrix} = \begin{bmatrix} 1, & \frac{\varepsilon}{1+\varepsilon} \\ \frac{\varepsilon}{1+\varepsilon}, & 1 + \left(\frac{\varepsilon}{1+\varepsilon}\right)^2 \end{bmatrix}$$

so that $\kappa(L_2 L_2^T) \to 1$ as $\varepsilon \to 0$.

The decomposition (2.1) can be obtained by means of corrective Gaussian elimination with diagonal pivoting as it is proposed in [6], [17]. We use a modification of the process proposed in [17]. Since our modifications are significant for nonlinear least squares, we give a detailed description of the algorithm (we suppose $X = I$ in (2.1) for simplicity).

Algorithm 2.1

Data: $0 < \varepsilon_3 < 1$.

Step 1: Set $\delta := 0$ and $\gamma := \max\left(\varepsilon_3, \max\limits_{1 \le j \le n} |B_{jj}|\right)$.

Step 2: Phase 1. For $k := 1$ to n do:

Step 2.1: Choice of a pivot. Set $l := k$. Determine the index i such that $B_{ii} = \max\limits_{k \le j \le n} B_{jj}$. If $B_{ii} \le 0$ then go to Step 4. If $i \ne k$ then switch rows and columns i and k of the matrix B.

Step 2.2: Test on phase 1 acceptability. If $\min\limits_{k+1 \le j \le n} (B_{jj} - B_{jk}^2 / B_{kk}) < \varepsilon_3 \gamma$ then go to Step 4.

Step 2.3: Elimination. Set $D_k := B_{kk}$ and $L_{kk} := 1$. For $j := k+1$ to n do:

Step 2.3.1: Set $L_{jk} := B_{jk} / D_k$.

Step 2.3.2: For $i := k+1$ to j compute $B_{ij} := B_{ij} - B_{jk} L_{ik}$.

Step 3: End of phase 1. Terminate the computations.

Step 4: Phase 2. If $l \le n-2$ then for $j := l$ to n compute

$$h_j := B_{jj} - \sum_{\substack{i=1 \\ i \ne j}}^{n} |B_{ij}|$$

and for $k := l$ to $n-2$ do:

Step 4.1: Choice of a pivot. Determine the index i such that $h_i = \max\limits_{k \le j \le n} h_j$. If $i \neq k$ then switch rows and columns i and k of the matrix B.

Step 4.2: Choice of a correction. Compute

$$\beta := \sum_{j=k+1}^{n} |B_{jk}|$$

and $\rho := \max(0, -B_{kk} + \max(\beta, \varepsilon_3\gamma), \delta)$. Set $B_{kk} := B_{kk} + \rho$ and $\delta := \rho$.

Step 4.3: Update of Gerschgorin bound estimates. If $B_{kk} \neq \beta$ then set $\beta := 1 - \beta/B_{kk}$ and for $j := l + 1$ to n compute $h_j := h_j + \beta|B_{jk}|$.

Step 4.4: Elimination. Set $D_k := B_{kk}$ and $L_{kk} := 1$. For $j := k + 1$ to n do:

Step 4.4.1: Set $L_{jk} := B_{jk}/D_k$.

Step 4.4.2: For $i := k + 1$ to j compute $B_{ij} := B_{ij} - B_{jk}L_{ik}$.

Step 5: Final 2×2 submatrix. If $l \le n - 1$ then set $l := n - 1$ and do:

Step 5.1: Choice of a correction. Compute the values $\beta := \sqrt{(B_{nn} - B_{ll})^2/4 + B_{nl}^2}$, $\lambda := (B_{nn} + B_{ll})/2 - \beta$ and set $\rho := \max(0, -\lambda + \varepsilon_3 \max(2\beta/(1 - \varepsilon_3), \gamma))$. Set $B_{ll} := B_{ll} + \rho$, $B_{nn} := B_{nn} + \rho$ and $\delta := \max(\delta, \rho)$.

Step 5.2: Elimination. Set $D_l := B_{ll}$ and $L_{ll} := 1$. Set $L_{nl} := B_{nl}/D_l$ and $B_{nn} := B_{nn} - B_{nl}L_{nl}$. Set $D_n := B_{nn}$ and $L_{nn} := 1$.

Step 6: Final 1×1 submatrix. If $l = n$ then do:

Step 6.1: Choice of a correction. Set $\rho := \max(0, -B_{nn} + \varepsilon_3\gamma)$. Set $B_{nn} := B_{nn} + \rho$ and $\delta := \max(\delta, \rho)$.

Step 6.2: Elimination. Set $D_n := B_{nn}$ and $L_{nn} := 1$.

Step 7: End of phase 2. Terminate the computations.

Note that the matrices D and L need not be stored separately since $D_k = B_{kk}$ and L_{jk} overwrite B_{jk} for $1 \le k < j \le n$ after termination of computations.

The main difference between Algorithm 2.1 and the algorithm proposed in [17] consists in the fact that ρ can be less then the previous δ in Step 5. This difference leads to less correction matrix which considerably improves the efficiency of trust region algorithm for nonlinear least squares. Other modifications are extension of Step 2.1 (phase 1 is left if $B_{ii} \le 0$) and addition of Step 6. These modifications make Algorithm 2.1 to be more general.

Now we return to the determination of the transformed direction vector satisfying (1.15), (1.16), (1.17). If we use the transformation matrix (2.2) then the matrix (2.3) is diagonal and positive definite. Therefore the solution of the equation (1.19) has the form

$$\tilde{d}(\lambda) = -\sum_{i=1}^{n} \frac{e_i^T \tilde{g}}{e_i^T \tilde{B} e_i + \lambda} e_i \tag{2.9}$$

and, moreover, the eigenvalues $\lambda_i, 1 \le i \le n$, of the matrix $\tilde{B}$ are given by $\lambda_i = e_i^T \tilde{B} e_i$, $1 \le i \le n$. This fact considerably reduces computations in each iteration

of the Newton method applied to (1.18). The new value λ^+ of λ obtained by the Newton method is computed by the rule

$$\lambda^+ := \lambda + \frac{\|\tilde{d}(\lambda)\|^2}{\|\tilde{c}(\lambda)\|^2}\left(\frac{\|\tilde{d}(\lambda)\| - \Delta}{\Delta}\right), \tag{2.10}$$

where $\tilde{d}(\lambda)$ is given by (2.9) and

$$\|\tilde{c}(\lambda)\|^2 = \tilde{d}^T(\lambda)(\tilde{B} + \lambda I)^{-1}\tilde{d}(\lambda) = \sum_{i=1}^{n} \frac{(e_i^T \tilde{d}(\lambda))^2}{e_i^T \tilde{B} e_i + \lambda}. \tag{2.11}$$

We get the following algorithm which is a simplification of the algorithm given in [15].

Algorithm 2.2

Data: $0 < \beta_3 < 1, 0 < \delta_1 < 1 < \delta_2$.

Step 1: Choose indices i and j such that $\tilde{B}_{ii} = \min\limits_{1 \le k \le n}(\tilde{B}_{kk})$ and $\tilde{B}_{jj} = \max\limits_{1 \le k \le n}(\tilde{B}_{kk})$ where $\tilde{B}_{kk} = e_k^T \tilde{B} e_k$, $1 \le k \le n$. Compute the bounds $\lambda_l := \max(0, \|\tilde{g}\|/\Delta - \tilde{B}_{jj})$ and $\lambda_u := \max(0, \|\tilde{g}\|/\Delta - \tilde{B}_{ii})$. Set $\lambda := \lambda_l$.

Step 2: If $\lambda < \lambda_l$ then compute $\lambda^+ := \sqrt{\lambda_l \lambda_u}$ and set $\lambda := \lambda_l + \beta_3(\lambda_u - \lambda_l)$ if $\lambda^+ < \lambda_l + \beta_3(\lambda_u - \lambda_l)$, $\lambda := \lambda_u - \beta_3(\lambda_u - \lambda_l)$ if $\lambda^+ > \lambda_u - \beta_3(\lambda_u - \lambda_l)$, and $\lambda := \lambda^+$ otherwise.

Step 3: Compute the vector $\tilde{d} := \tilde{d}(\lambda)$ by (2.9). When $\delta_2 \Delta < \|\tilde{d}\|$ then set $\lambda_l := \lambda$ and go to Step 5. When $\delta_1 \Delta \le \|\tilde{d}\| \le \delta_2 \Delta$ or $\|\tilde{d}\| < \delta_1 \Delta$ and $\lambda = 0$ then stop (the conditions (1.15), (1.16), (1.17) are satisfied). When $\|\tilde{d}\| < \delta_1 \Delta$ and $\lambda > 0$ then set $\lambda_u := \lambda$ and continue.

Step 4: Determine the value α having the same sign as $\tilde{d}_i = e_i^T \tilde{d}$ such that $\|\tilde{d} + \alpha e_i\| = \Delta$. If $\alpha^2(\tilde{B}_{ii} + \lambda) \le (1 - \delta_1)^2(\lambda \Delta^2 - \tilde{g}^T \tilde{d})$ then set $\tilde{d} := \tilde{d} + \alpha e_i$ and stop (the conditions (1.15), (1.16), (1.17) are satisfied).

Step 5: Compute the value λ^+ by (2.10) and (2.11), set $\lambda := \min(\lambda^+, \lambda_u)$ and go to Step 2.

Algorithms described in this section can be used in Step 3 of Algorithm 1.1.

Step 3 of Algorithm 1.1: Determine a diagonal positive definite scaling matrix X such that $X_i := \sigma_1$ if $\sqrt{E_i} < \sigma_1$, $X_i := \sigma_2$ if $\sqrt{E_i} > \sigma_2$ and $X_i := \sqrt{E_i}$ otherwise, where $X_i = e_i^T X e_i$, $E_i = e_i^T E e_i$, $1 \le i \le n$, with E given either by (2.5a) (case $S = 1$) or by (2.5b) (case $S = 2$), and set $g := X^{-1} g$, $B := X^{-1} B X^{-1}$. Compute the decomposition $B + C := LDL^T$ using Algorithm 2.1 and set $\tilde{g} := L^{-1} g$. Determine a diagonal positive definite weighting matrix Y such that $Y_i := \sigma_1$ if $\sqrt{Z_i} < \sigma_1$, $Y_i := \sigma_2$ if $\sqrt{Z_i} > \sigma_2$ and $Y_i := \sqrt{Z_i}$ otherwise, where $Y_i = e_i^T Y e_i$, $Z_i = e_i^T Z e_i$, $1 \le i \le n$, with Z given either by (2.8a) (case $W = 1$) or by (2.8b) (case $W = 2$), and set $\tilde{g} := Y^{-1} \tilde{g}$, $\tilde{B} := Y^{-1} D Y^{-1}$. Compute the transformed direction vector $\tilde{d}$ using Algorithm 2.2. Set $\tilde{d} := Y^{-1} \tilde{d}$. Set $d := (L^T)^{-1} \tilde{d}$. Set $d := X^{-1} d$.

Numerical effectiveness of a trust region method realized by Algorithm 1.1 is investigated in Section 3. We have no theoretical justification for a good convergence of this method but numerical experiments given in Section 3 show that our considerations were reasonable.

3. COMPUTATIONAL EXPERIMENTS

In this section we present results of a comparative study of trust region methods for nonlinear least squares. These results were obtained by means of 30 standard test problems given in [14; pp. 21–28] and by means of 6 difficult problems given in the Appendix. Problems 1–19 had the same dimensions as in [14]. Problems 20–30 were considered either with 6 variables or with 20 variable.

Because Algorithm 1.1 (together with Algorithm 2.1 and Algorithm 2.2) contains several parameters, we have to specify our recommended values. We have used the values $\beta_1 = 0.05$, $\beta_2 = 0.75$, $\beta_3 = 0.1$, $\gamma_1 = 2$, $\gamma_2 = 10$, $\delta_1 = 0.9$, $\delta_2 = 1.1$, $\rho_1 = 0.1$, $\rho_2 = 0.9$, $\varepsilon_1 = 10^{-16}$, $\varepsilon_2 = 10^{-6}$, $\varepsilon_3 = 10^{-18}$, $\sigma_1 = 10^{-5}$, $\sigma_2 = 5.10^4$, $l_1 = 20$ in all numerical experiments. The value $\Delta_{\max}$ was chosen individually for each problem, as high as possible, but such that overflows could not appear.

The first three tables contain numerical tests of Algorithm 1.1 with different scaling (cases $S = 1$ and $S = 2$) and weighting (cases $W = 1$ and $W = 2$) matrices which were introduced in Section 2. Rows of these tables correspond to individual problems and columns correspond to different scaling and weighting matrices. Results are recorded in the form IT-IF-IG (P) where IT is a number of iterations, IF is a number of different points at which the values $f_i(x)$, $1 \leq i \leq m$, were computed, IG is a number of different points at which the gradients $g_i(x)$, $1 \leq i \leq m$, were computed and P is the logarithm of the obtained gradient norm. Table 1 contains results for problems 1–30 where problems 20–30 had 6 variables, Table 2 contains results for problems 20–30 with 20 variables and Table 3 contains results for problems given in the Appendix.

Results reported in these tables show that standard problems are better solved with unit scaling matrix (choice $S = 1$) while the choice $S = 2$ is necessary for difficult exponential models. Therefore, it is necessary to have both these possiblities in a general algorithm. The choice $W = 2$ gives slightly worse results for standard problems and slightly better results for difficult exponential models, but its influence has little significance.

The last three tables compare the new method (Algorithm 1.1) with two other methods for nonlinear least squares: the trust region method with optimal locally constrained step (OLCS) proposed in [15], and the trust region method with double dog-leg step (DDLS) described in [3] (see also [18]). Results are presented in the form IT-IF-IG/ID where ID is a number of decompositions performed by Algorithm 2.1. Table 4 contains results for problems 1–30 where problems 20–30 had 6 variables, Table 5 contains results for problems 20–30 with 20 variables and Table 6 contains results for problems given in the Appendix.

Table 1.

	$S=1, W=1$	$S=1, W=2$	$S=2, W=1$	$S=2, W=2$
1	14- 17- 15(99)	12- 14- 13(-13)	23- 28- 24(-99)	16- 20- 17(-13)
2	25- 40- 26(-6)	30- 50- 31(-6)	13-22- 14(-6)	11- 21- 12(-7)
3	33- 34- 34(-6)	33- 34- 34(-6)	47- 52- 49(-6)	40- 44- 41(-7)
4	22- 24- 21(-10)	20- 24- 21(-10)	15- 19- 16(-10)	16- 20- 17(-10)
5	6- 7- 7(-10)	6- 7- 7(13)	7- 9- 8(-9)	7- 9- 8(14)
6	19- 60- 19(-5)	17- 48- 17(-5)	29- 60- 21(-4)	19- 31- 20(-6)
7	12- 16- 13(-5)	11- 14- 12(-7)	9- 11- 10(-11)	13- 20- 14(-7)
8	5- 6- 6(-8)	5- 6- 6(-8)	5- 6- 6(-8)	5- 6- 6(-8)
9	1- 2- 2(-7)	1- 2- 2(-7)	1- 2- 2(-7)	1- 2- 2(-7)
10	125-136-126(-4)	126-151-126(-3)	130-169-130(-1)	154-190-154(-3)
11	34- 41- 35(-6)	35- 41- 36(-6)	248-312-249(-6)	IT > 400
12	12- 14- 13(-8)	12- 14- 13(-7)	10- 11- 11(-6)	overflow
13	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)
14	36- 40- 37(-7)	34- 39- 35(-7)	72- 81- 73(-11)	70- 79- 71(-8)
15	16- 17- 17(-6)	16- 17- 17(-6)	14- 17- 15(-6)	12- 15- 13(-6)
16	29- 64- 29(-3)	26- 62- 26(-3)	IT > 400	IT > 400
17	19- 22- 20(-7)	19- 22- 20(-7)	19- 22- 20(-8)	12- 15- 13(-8)
18	28- 32- 29(-8)	39- 46- 40(-9)	23- 29- 24(-7)	12- 13- 13(-11)
19	13- 16- 14(-6)	13- 15- 14(-6)	11- 13- 12(-6)	12- 16- 13(-6)
20	7- 8- 8(-6)	7- 8- 8(-6)	7- 8- 8(-6)	7- 8- 8(-6)
21	14- 17- 15(-99)	12- 14- 13(-99)	19- 24- 20(-13)	18- 23- 19(-15)
22	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)
23	17- 24- 18(-6)	17- 24- 18(-6)	94- 116- 95(-6)	33- 42- 34(-6)
24	36- 45- 37(-7)	39- 48- 40(-6)	IT > 400	IT > 400
25	8- 9- 9(-10)	8- 9- 9(-10)	8- 9- 9(-9)	8- 9- 9(-9)
26	7- 9- 8(-7)	7- 9- 8(-9)	5- 6- 6(-6)	6- 8- 7(-8)
27	5- 6- 6(-7)	5- 6- 6(-6)	10- 12- 11(-9)	7- 8- 8(-9)
28	4- 5- 5(-10)	4- 5- 5(-9)	4- 5- 5(-9)	4- 5- 5(-8)
29	2- 3- 3(-6)	2- 3- 3(-6)	2- 3- 3(-6)	2- 3- 3(-6)
30	4- 5- 5(-8)	4- 5- 5(-8)	4- 5- 5(-8)	4- 5- 5(-8)
$\sum$	571-741-599	580-759-607	IT > 1641	IT > 1709

Table 2.

	$S=1, W=1$	$S=1, W=2$	$S=2, W=1$	$S=2, W=2$
20	7- 8- 8(-12)	8- 9- 9(-11)	4- 5- 5(-6)	6- 7- 7(-13)
21	14- 17- 15(-99)	12- 14- 13(-13)	22- 28- 23(-8)	22- 27- 23(-99)
22	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)	10- 11- 11(-6)
23	28- 35- 29(-6)	64- 73- 65(-6)	IT > 300	IT > 300
24	104-118-105(-6)	141-163-142(-6)	IT > 300	IT > 300
25	11- 12- 12(-6)	11- 12- 12(-6)	30- 31- 31(-99)	15- 16- 16(-99)
26	42- 53- 43(-6)	7- 9- 8(-6)	58- 75- 59(-6)	6- 7- 7(-9)
27	10- 14- 11(-8)	9- 11- 10(-8)	11- 14- 12(-7)	11- 13- 12(-7)
28	5- 6- 6(-6)	5- 6- 6(-6)	6- 7- 7(-7)	6- 7- 7(-6)
29	2- 3- 3(-6)	2- 3- 3(-6)	2- 3- 3(-6)	2- 3- 3(-6)
30	4- 5- 5(-8)	4- 5- 5(-8)	4- 5- 5(-8)	4- 5- 5(-8)
$\sum$	237-282-248	273-316-284	IT > 747	IT > 682

Table 3.

	$S = 1, W = 1$	$S = 1, W = 2$	$S = 2, W = 1$	$S = 2, W = 2$
A1	48- 49- 49(-6)	48- 49- 49(-7)	38- 41- 49(-6)	40- 44- 41(-6)
A2	17- 50- 17(-5)	17- 48- 17(-5)	21- 60- 21(-4)	19- 31- 20(-6)
A3	125-136-126(-4)	126-151-126(-3)	130-169-130(-1)	154-190-154(-3)
A4	IT > 900	IT > 900	28- 30- 29(-9)	21- 23 22(-6)
A5	IT > 900	IT > 900	234-263-235(-6)	144-158-145(-6)
A6	IT > 900	IT > 900	581-605-582(-8)	576-594-577
$\sum$	IT > 2890	IT > 2891	1032-1168-1036	954-1040-959

Table 4.

	New ($S = 1, W = 1$)	OLCS($S = 1$)	DDLS($S = 1$)
1	14- 17- 15/14	12- 15- 13/28	18- 21- 19/18
2	25- 40- 26/25	30- 63- 30/106	20- 30- 21/20
3	33- 34- 34/33	33- 34- 34/64	52- 53- 53/52
4	19- 23- 20/19	13- 16- 14/27	4- 6- 6/4
5	6- 7- 7/6	6- 7- 7/8	7- 8- 8/7
6	16- 28- 17/16	11- 21- 12/36	19- 56- 19/19
7	11- 13- 12/11	7- 8- 8/11	11- 14- 12/11
8	5- 6- 6/5	5- 6- 6/8	4- 5- 5/4
9	1- 2- 2/1	1- 2- 2/1	1- 2- 2/1
10	124-129-125/125	124-126-125/263	46- 76- 46/46
11	70- 99- 70/34	70- 97- 70/207	51- 60- 52/16
12	12- 14- 13/12	12- 14- 13/29	16- 20- 17/16
13	10- 11- 11/10	10- 11- 11/14	10- 11- 11/10
14	36- 40- 37/36	69- 75- 70/99	42- 46- 43/42
15	16- 17- 17/16	17- 20- 18/35	15- 17- 16/15
16	29- 64- 29/29	28- 65- 28/132	157-202-157/157
17	19- 22- 20/19	21- 23- 22/60	16- 17- 17/16
18	28- 32- 29/28	17- 20- 18/161	196-226-197/196
19	13- 16- 14/13	13- 15- 14/29	13- 16- 14/13
20	6- 7- 7/6	7- 8- 8/15	8- 9- 9/8
21	14- 17- 15/14	12- 15- 13/28	18- 21- 19/18
22	10- 11- 11/10	10- 11- 11/14	10- 11- 11/10
23	22- 30- 23/22	20- 25- 21/42	18- 21- 19/18
24	76- 85- 77/76	28- 36- 29/111	59- 72- 60/59
25	10- 11- 11/10	10- 11- 11/10	8- 9- 9/8
26	9- 16- 10/9	9- 13- 10/21	32- 43- 33/32
27	7- 8- 8/7	6- 7- 7/9	4- 5- 5/4
28	4- 5- 5/4	7- 8- 8/18	6- 7- 7/6
29	2- 3- 3/2	2- 3- 3/2	2- 3- 3/2
30	4- 5- 5/4	5- 6- 6/6	5- 6- 6/5
$\sum$	651-812-679/651	615-780-642/1712	868-1093-895/871
Time	1:08.22	1:24.14	1:56.72

Table 5.

	New ($S = 1, W = 1$)	OLCS($S = 1$)	DDLS($S = 1$)
20	7- 8- 8/7	8- 9- 9/46	10- 11- 11/10
21	14- 17- 15/14	12- 15- 13/28	18- 21- 19/18
22	10- 11- 11/10	10- 11- 11/14	10- 11- 11/10
23	28- 35- 29/28	20- 26- 21/44	30- 35- 31/30
24	104-118-105/104	42- 53- 43/157	IT > 300
25	11- 12- 12/11	11- 12- 12/11	11- 12- 12/11
26	42- 53- 43/42	37- 48- 38/124	IT > 300
27	10- 14- 11/10	7- 8- 8/10	4- 5- 5/4
28	5- 6- 6/5	8- 9- 9/24	8- 9- 9/8
29	2- 3- 3/2	2- 3- 3/2	2- 3- 3/2
30	4- 5- 5/4	5- 6- 6/6	5- 6- 6/5
$\sum$	237-282-248/237	162-200-173/466	IT > 698
Time	4:40.37	6:01.57	>15:04.51

Table 6.

	New ($S = 1, W = 2$)	OLCS($S = 1$)	DDLS($S = 2$)
A1	40- 44- 41/40	43- 46- 44/125	40- 44- 41/40
A2	19- 31- 20/19	14- 25- 15/43	28- 67- 28/28
A3	154-190-154/154	137-161-137/310	32- 69- 32/32
A4	21- 23- 22/21	32- 34- 33/116	38- 40- 39/38
A5	144-158-145/144	135-149-136/434	72- 78- 73/72
A6	576-594-577/576	522-551-523/1110	IT > 900
$\sum$	954-1040-959/954	883-966-888/2138	IT > 1110
Time	1:20.14	1:28.54	>1:34.58

Results recorded in these tables show that the convergence of the new method is almost as robust and fast as the convergence of the trust region method with optimal locally constrained step (OLCS). The OLCS method uses time-consuming repeated matrix decompositions and, therefore it needs a longer time for obtaining results. The new method is much better then the DDLS method which also does not use repeated matrix decompositions.

The results obtained by the approach introduced in Section 2 are very hopeful. Since we use a simple decomposition technique, namely a corrective Choleski factorization of a Gauss–Newton normal equation matrix, it would be possible to improve these results by some other decomposition techniques such as rank revealing QR factorization. Also a study of the matrices P and Y's influence to the condition number $\kappa(LYY^TL^T)$, which would lead to minimal $\kappa(LYY^TL^T)$, remains for further research.

APPENDIX

Difficult problems we have used were originally listed in [11]. These problems served for testing statistical packages in [12].

Problem A1. (Meyer, Roth)

$$f_i(x) = x_1 + x_2 \exp(x_3 t_i) - y_i$$

for $1 \leq i \leq 10$, where

i	t_i	y_i	i	t_i	y_i
1	1.0	16.7	6	25.0	17.4
2	5.0	26.8	7	30.0	17.6
3	10.0	16.9	8	35.0	17.9
4	15.0	17.1	9	40.0	18.1
5	20.0	17.2	10	50.0	18.7

Start $x_1 = 20.0$, $x_2 = 2.0$, $x_3 = 0.5$.

Problem A2. (Jennrich, Sampson)

$$f_i(x) = \exp(x_1 t_i) + \exp(x_2 t_i) - y_i$$

for $1 \leq i \leq 10$, where

i	t_i	y_i	i	t_i	y_i
1	1.0	4.0	6	6.0	14.0
2	2.0	6.0	7	7.0	16.0
3	3.0	8.0	8	8.0	18.0
4	4.0	10.0	9	9.0	20.0
5	5.0	12.0	10	10.0	22.0

Start $x_1 = 0.3$, $x_2 = 0.4$.

Problem A3. (Meyer, Roth)

$$f_i(x) = x_1 \exp(x_2/(x_3 + t_i)) - y_i$$

for $1 \leq i \leq 16$, where

i	t_i	y_i	i	t_i	y_i	i	t_i	y_i
1	50.0	34780.0	7	80.0	11540.0	13	110.0	4427.0
2	55.0	28610.0	8	85.0	9744.0	14	115.0	3820.0
3	60.0	23650.0	9	90.0	8261.0	15	120.0	3307.0
4	65.0	19630.0	10	95.0	7030.0	16	125.0	2872.0
5	70.0	16370.0	11	100.0	6005.0			
6	75.0	13720.0	12	105.0	5147.0			

Start $x_1 = 0.02$, $x_2 = 4000.0$, $x_3 = 250.0$.

Problem A4. (Květoň)

$$f_i(x) = x_1 \exp(-x_3 t_i) + x_2 \exp(-x_4 t_i) - y_i$$

for $1 \leq i \leq 10$, where

i	t_i	y_i	i	t_i	y_i
1	1.0	99.6	6	6.0	16.1
2	2.0	67.1	7	7.0	11.7
3	3.0	45.9	8	8.0	8.6
4	4.0	31.9	9	9.0	6.38
5	5.0	22.5	10	10.0	4.78

Start $x_1 = 1.0$, $x_2 = 1.0$, $x_3 = 1.0$, $x_4 = 1.0$.

Problem A5. (Hřebiček)

$$f_i(x) = x_1 \exp(-x_3 t_i) + x_2 \exp(-x_4 t_i) - y_i$$

for $1 \leq i \leq 15$, where

i	t_i	y_i	i	t_i	y_i	i	t_i	y_i
1	7.448	57.554	6	7.877	27.952	11	8.552	11.803
2	7.448	53.546	7	7.969	19.498	12	8.903	7.727
3	7.552	45.290	8	8.176	16.444	13	9.114	4.764
4	7.607	51.286	9	8.176	21.777	14	9.284	4.305
5	7.847	31.623	10	8.523	13.996	15	9.439	3.006

Start $x_1 = 100000.0$, $x_2 = 100000.0$, $x_3 = 1.079$, $x_4 = 1.31$.

Problem A6. (Květoň)

$$f_i(x) = x_1 t_i^{x_3} + x_2 t_i^{x_4} - y_i$$

for $1 \leq i \leq 12$, where

i	t_i	y_i	i	t_i	y_i
1	12.0	7.31	7	18.0	8.84
2	13.0	7.55	8	19.0	9.12
3	14.0	7.80	9	20.0	9.40
4	15.0	8.05	10	21.0	9.69
5	16.0	8.31	11	22.0	9.99
6	17.0	8.57	12	23.0	10.3

Start $x_1 = 1000.0$, $x_2 = 0.01$, $x_3 = 2.0$, $x_4 = 100.0$.

(Received January 31, 1995.)

REFERENCES

[1] M. Al-Baali and R. Fletcher: Variational methods for nonlinear least squares. J. Optim. Theory Appl. *36* (1985), 405–421.
[2] J. E. Dennis, D. M. Gay and R. E. Welsch: An adaptive nonlinear least-squares algorithm. ACM Trans. Math. Software *7* (1981), 348–368.
[3] J. E. Dennis and H. H. W. Mei: An Unconstrained Optimization Algorithm which Uses Function and Gradient Values. Research Report No. TR-75-246. Dept. of Computer Sci., Cornell University 1975.
[4] R. Fletcher: A Modified Marquardt Subroutine for Nonlinear Least Squares. Research Report No. R-6799, Theoretical Physics Division, A.E.R.E. Harwell 1971.
[5] R. Fletcher and C. Xu: Hybrid methods for nonlinear least squares. IMA J. Numer. Anal. *7* (1987), 371–389.
[6] P. E. Gill and W. Murray: Newton type methods for unconstrained and linearly constrained optimization. Math. Programming *7* (1974), 311–350.
[7] M. D. Hebden: An Algorithm for Minimization Using Exact Second Derivatives. Research Report No. TP515, Theoretical Physics Division, A.E.R.E. Harwell 1973.
[8] K. Levenberg: A method for the solution of certain nonlinear problems in least squares. Quart. Appl. Math. *2* (1944), 164–168.
[9] L. Lukšan: Computational experience with improved variable metric methods for unconstrained minimization. Kybernetika *26* (1990), 415–431.
[10] D. W. Marquardt: An algorithm for least squares estimation of non-linear parameters. SIAM J. Appl. Math. *11* (1963), 431–441.
[11] J. Militký: Mathematical Models Building. VI. Mineo, Technical House, Ostrava 1989.
[12] J. Militký, O. Šenkýř and L. Rudišar: Comparison of statistical software for nonlinear regression on IBM PC. In: COMPSTAT 90, Short communications, 1990, pp. 49–50.
[13] J. J. Moré: The Levenberg-Marquardt algorithm. Implementation and theory. In: Numerical Analysis (G. A. Watson ed.), Springer Verlag, Berlin 1978.
[14] J. J. Moré, B. S. Garbow and K. E. Hillström: Testing unconstrained optimization software. ACM Trans. Math. Software *7* (1981) 17–41.
[15] J. J. Moré and D. C. Sorensen: Computing a trust region step. SIAM J. Sci. Statist. Comput. *4* (1983), 553–572.
[16] M. J. D. Powell: A new algorithm for unconstrained optimization. In: Nonlinear Programming (J. B. Rosen, O. L. Mangasarian and K. Ritter, eds.), Academic Press, London 1970.
[17] R. B. Schnabel and E. Eskow: A new Choleski factorization. SIAM J. Sci. Statist. Comput. *11* (1990), 1136–1158.
[18] G. A. Shultz, R. B. Schnabel and R. H. Byrd: A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. SIAM J. Numer. Anal. *22* (1985) 47–67.

Ing. Ladislav Lukšan, DrSc., Ústav informatiky a výpočetní techniky AV ČR (Institute of Computer Science – Academy of Sciences of the Czech Republic), Pod vodárenskou věží 2, 182 07 Praha. Czech Republic.