INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Prague

Hybrid Methods for Large Sparse Nonlinear Least Squares

L. Lukšan

Technical Report No. V-561

October 1993

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

Hybrid Methods for Large Sparse Nonlinear Least Squares

Ladislav Lukšan

*Institute of Computer Science,
Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2,
18207 Prague 8, Czech Republic*

Technical Report No. V-561, October 1993

Abstract. Hybrid methods were developed for improving the Gauss-Newton method in the case of large residual or ill-conditioned nonlinear least squares. These methods are usually used in the form suitable for dense problems. But some old approaches are unsuitable and some new possibilities appear in the sparse case. We propose efficient hybrid methods for various representations of the sparse problems. After describing basic ideas that serves for deriving new hybrid methods, we are concerned with designing hybrid methods for sparse Jacobian, partitioned Hessian and sparse Hessian representations of the least squares problems. Efficiency of hybrid methods is demonstrated by extensive numerical experiments.

Key words. Unconstrained optimization, nonlinear least squares, line search methods, trust region methods, Gauss-Newton method, hybrid methods, sparse problems, matrix iterative methods, matrix direct methods, computational experiments.

1. Introduction

Let $f_i : R^n \to R$, $1 \le i \le r$, be real-valued functions with continuous second order derivatives on an open set $X \subset R^n$. Let us denote

$$F(x) = \frac{1}{2}\sum_{i=1}^{r} f_i^2(x) = \frac{1}{2} f^T(x) f(x) \tag{1.1}$$

where $f(x) = [f_1(x), ..., f_r(x)]^T$. We are concerned with finding a local minimum $x^\star \in X$ of the function $F : R^n \to R$ given by (1.1) i.e. a point $x^\star \in X$ that satisfies the inequality $F(x^\star) \le F(x)\ \forall x \in B(x^\star, \varepsilon)$ for some $\varepsilon > 0$, where $B(x^\star, \varepsilon) = \{x \in X : \| x - x^\star \| < \varepsilon\} \subset X$ is an open ball contained in $X \subset R^n$.

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

$$g(x) = \sum_{i=1}^{r} f_i(x) g_i(x) = J^T(x) f(x) \tag{1.2}$$

and

$$G(x) = \sum_{i=1}^{r} g_i(x) g_i^T(x) + \sum_{i=1}^{r} f_i(x) G_i(x) = J^T(x) J(x) + S(x) \tag{1.3}$$

where $J(x) = [g_1(x), ..., g_r(x)]^T$ is the Jacobian matrix of $f(x) = [f_1(x), ..., f_r(x)]^T$ and $S(x)$ is the second order term. Numerical methods for local minimization of the objective function $F : R^n \to R$ are usually derived from the Newton method. These methods are iterative and their iteration step has the form

$$x^+ = x + \alpha d$$

where x and x^+ are old and new vectors of variables respectively, α is a stepsize parameter and d is a direction vector which approximately minimizes the quadratic function

$$Q(d) = \frac{1}{2} d^T B d + g^T d \tag{1.4}$$

over some subset of R^n. Here $B = B(x)$ is an approximation of the Hessian matrix $G(x)$ and $g = g(x)$ is the gradient given by (1.2). There are three basic possibilities concerning how the matrix B in (1.4) can be constructed. The first possibility is the Newton method (or modified Newton method) defined by the substitution $B(x) = G(x)$. This method is usually quadratically convergent, but it requires second order derivatives computed either analytically or numerically. Moreover the Hessian matrix $G(x)$ can be indefinite which implies difficulties connected with its factorization and with descent direction determination.

The second possibility leads to the so-called quasi-Newton methods which use an arbitrary positive definite matrix in the first iteration and which generate subsequent matrices by simple quasi-Newton updates [16]. The main advantage of this approach is its general applicability (the objective function $F : R^n \to R$ cannot have the special form (1.1)) and the fact that the matrix $B(x)$ can be kept positive definite.

The third possibility is based on the special form (1.1) of the objective function $F : R^n \to R$ and it consists in the substitution

$$B(x) = J^T(x) J(x) = \sum_{i=1}^{r} g_i(x) g_i^T(x). \tag{1.5}$$

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

$$\begin{aligned} F(x+d) &\approx \frac{1}{2}\sum_{i=1}^{r}(f_i(x)+g_i^T(x)d)^2 = \\ &= \frac{1}{2}\sum_{i=1}^{r}(f_i^2(x)+2f_i(x)g_i^T(x)d+d^Tg_i(x)g_i^T(x)d) = \\ &= F(x)+g^T(x)d+\frac{1}{2}d^TBd = F(x)+Q(d) \end{aligned}$$

with B given by (1.6). The method which uses the matrix (1.5) instead of the Hessian matrix $G(x)$ is called the Gauss-Newton (or modified Gauss-Newton) method [13]. The main advantage of the Gauss-Newton method is its quadratic convergence for zero-residual problems. Convergence of the Guass-Newton method is usually faster then convergence of the quasi-Newton methods, but this advantage can be lost for large-residual or ill-conditioned problems.

Besides the above three possibilities there exist their various combinations (see [1], [2], [3], [8], [10], [11], [17]). These so-called hybrid methods are of prime interest to us and they are investigated in the subsequent parts of this contribution.

All the above methods can be realized in two different forms using either the line search strategy or the trust region strategy. A typical iteration step of the line search strategy has the following form.

(L1) Direction determination. Choose $d \in R^n$ so that

$$\| Bd+g \| \le \omega \| g \| \tag{1.6}$$

and

$$-g^Td \ge \bar{\varepsilon}_0 \| g \| \| d \| \tag{1.7}$$

where $0 \le \omega \le \bar{\omega} < 1$, $\bar{\varepsilon}_0 > 0$ ($\bar{\omega}$ and $\bar{\varepsilon}_0$ do not depend on the iteration step), $g = g(x)$ and $B = B(x)$.

(L2) Stepsize selection. Choose $\alpha > 0$ so that

$$F(x+\alpha d) - F \le \bar{\varepsilon}_1 \alpha g^T d \tag{1.8a}$$

and

$$g^T(x+\alpha d)d \ge \bar{\varepsilon}_2 g^T d \tag{1.8b}$$

where $0 \le \bar{\varepsilon}_1 < 1/2$, $\bar{\varepsilon}_1 < \bar{\varepsilon}_2 < 1$ ($\bar{\varepsilon}_1$ and $\bar{\varepsilon}_2$ do not depend on the iteration step) $F = F(x)$ and $g = g(x)$. Finally set

$$x^+ = x + \alpha d \tag{1.9}$$

If the conditions (1.6) and (1.7) cannot be satisfied simultaneously, we must change the matrix B (restart).

The line search strategy is very convenient for the quasi-Newton methods that generate matrices which are usually positive definite and well-conditioned. A different situation appears for the Gauss-Newton method since the matrix given by (1.5) is still positive semidefinite but very often ill-conditioned even a singular. In this case, the direction vector $d \in R^n$ can have a rather large euclidean norm and, moreover, it can be almost orthogonal to the gradient g. Therefore, too many line search steps can appear for satisfying (1.8) and, moreover, frequent restarts may occur due to violation of (1.7). Similar difficulties arise for the Newton method since the Hessian matrix $G(x)$ can be indefinite and, therefore, (1.7) may be violated again. More details about the line search strategy can be found in [16]. Our implementation is described in [25].

A typical iteration step of the trust region strategy has the following form.

(T1) Direction determination. Choose $d \in R^n$ so that

$$\| d \| \leq \Delta, \tag{1.10a}$$

$$\| d \| < \Delta \Longrightarrow \| Bd + g \| \leq \omega \| g \| \tag{1.10b}$$

and

$$-Q(d) \geq \bar{\varepsilon}_0 \| g \| \;\min(\| d \|, \| g \| \,/\, \| B \|) \tag{1.11}$$

where $\Delta > 0$ is a trust region bound, $0 \leq \omega \leq \bar{\omega} < 1$, $\bar{\varepsilon}_0 > 0$ ($\bar{\omega}$ and $\bar{\varepsilon}_0$ do not depend on the iteration step), $g = g(x)$ and $B = B(x)$ ($Q(d)$ is given by (1.4)).

(T2) Stepsize selection. Set

$$x^+ = x + d \quad \text{if} \quad F(x+d) < F(x), \tag{1.12a}$$

$$x^+ = x \quad \text{if} \quad F(x+d) \geq F(x). \tag{1.12b}$$

(T3) Trust region update. Compute

$$\rho = \frac{F(x+d) - F(x)}{Q(d)}. \tag{1.13}$$

When $\rho < \bar{\rho}_1$, then determine the value $0 < \beta < 1$ using quadratic interpolation and set $\Delta^+ = \bar{\beta}_1 \| d \|$ if $\beta < \bar{\beta}_1$, $\Delta^+ = \beta \| d \|$ if $\bar{\beta}_1 \leq \beta \leq \bar{\beta}_2$ and $\Delta^+ = \bar{\beta}_2 \| d \|$ if $\bar{\beta}_2 < \beta$. When $\bar{\rho}_1 \leq \rho \leq \bar{\rho}_2$ then set $\Delta^+ = \Delta$. When $\bar{\rho}_2 < \rho$ then set $\Delta^+ = \min\,(\max(\Delta, \bar{\gamma}_1 \| d \|), \bar{\Delta})$.

Here $0 < \bar{\beta}_1 \le \bar{\beta}_2 < 1 < \bar{\gamma}_1$, $0 < \bar{\rho}_1 < 1/2$, $\bar{\rho}_1 < \bar{\rho}_2 < 1$ and $\bar{\Delta} > 0$ (barred constants do not depend on the iteration step).

The trust region strategy is very advantageous in connection with both the Newton and the Gauss-Newton methods. The matrix (1.5) can be as indefinite as ill-conditioned, even singular, but $\| d \|$ is always defined and bounded from above according to (1.10). The trust region strategy has strong global convergence properties (see [35], [37]). More details about the trust region strategy can be found in [12], [31], [34], [38], our implementation is described in [27].

All of the above considerations hold for both dense and sparse least squares problems but some of the old approaches are unsuitable and some new possibilities appear in the sparse case. The efficiency of sparse methods depends on the problem structure representation. There exists three basic representations.

(SJ) Sparse Jacobian representation. Let n_i be the numbers of nonzero elements of gradients $g_i(x) \in R^n$, $1 \le i \le r$. Denote $\hat{g}_i(x) \in R^{n_i}$ packed gradients containing only nonzero elements of $g_i(x) \in R^n$, $ind\ \hat{g}_i \in R^{n_i}$ vectors containing indices of elements of $\hat{g}_i(x) \in R^{n_i}$ in $g_i(x) \in R^n$, $1 \le i \le r$, and $ord\ \hat{g}_i = 1 + \sum_{j=1}^{i-1} n_i$, $1 \le i \le r+1$. Let

$$\hat{J}(x) = \begin{bmatrix} \hat{g}_1(x) \\ \dots \\ \hat{g}_r(x) \end{bmatrix}, \ ind\ \hat{J} = \begin{bmatrix} ind\ \hat{g}_1 \\ \dots \\ ind\ \hat{g}_r \end{bmatrix}, \ ord\ \hat{J} = \begin{bmatrix} ord\ \hat{g}_1 \\ \dots \\ ord\ \hat{g}_{r+1} \end{bmatrix}.$$

Then sparse Jacobian representation uses two numbers n, $\hat{n} = \sum_{i=1}^{r} n_i = ord\ \hat{g}_{r+1} - 1$ and three vectors $\hat{J}(x) \in R^{\hat{n}}$ (real), $ind\ \hat{J} \in R^{\hat{n}}$ (integer) and $ord\ \hat{J} \in R^{r+1}$ (integer).

(PH) Partitioned Hessian representation. Denote $\hat{B}_i(x) \in R^{n_i \times n_i}$ packed matrices containing only nonzero elements of $B_i(x) \in R^{n \times n}$ and $\check{B}_i(x) \in R^{n_i(n_i-1)/2}$ vectors containing only upper half parts of the symmetric matrices $\hat{B}_i(x) \in R^{n_i \times n_i}$, $1 \le i \le r$. Let

$$\hat{B}(x) = \begin{bmatrix} \check{B}_1(x) \\ \dots \\ \check{B}_r(x) \end{bmatrix}.$$

Then partitioned Hessian representation uses three numbers n, $\hat{n} = \sum_{i=1}^{r} n_i$, $\hat{m} = \sum_{i=1}^{r} n_i(n_i+1)/2$ and three vectors $\hat{B}(x) \in R^{\hat{m}}$ (real), $ind\ \hat{J} \in R^{\hat{n}}$ (integer) and $ord\ \hat{J} \in R^{r+1}$ (integer). Partitioned Hessian representation was introduced in [22].

(SH) Sparse Hessian representation. Let $\tilde{m}$ be the number of nonzero elements of the upper half part of the matrix $B(x) \in R^{n \times n}$. Denote $\tilde{B}(x) \in R^{\tilde{m}}$ the vector containing rowwise ordered nonzero elements of the upper half part of the matrix

$B(x) \in R^{n \times n}$, $ind\ \tilde{B} \in R^{\tilde{m}}$ the vector containing column indices of elements of $\tilde{B}(x) \in R^{\tilde{m}}$ in $B(x) \in R^{n \times n}$ and $ord\ \tilde{B} \in R^{n+1}$ the vector with the elements $ord\ \tilde{B}_i = 1 + \sum_{j=1}^{i-1} m_i$, where m_i are the numbers of nonzero elements in the i-th row of the upper half part of the matrix $B(x) \in R^{n \times n}$, $1 \le i \le n$. Then sparse Hessian representation uses two numbers n, $\tilde{m} = \sum_{i=1}^{n} m_i = ord\ \tilde{B}_{n+1} - 1$ and three vectors $\tilde{B}(x) \in R^{\tilde{m}}$ (real), $ind\ \tilde{B} \in R^{\tilde{m}}$ (integer) and $ord\ \tilde{B} \in R^{n+1}$ (integer).

Sparse Jacobian representation is most general but it does not make possible an easy use of second order information. Moreover if $n \ll r$, then often $\tilde{m} \ll \hat{n}$, so that the matrix operations connected with sparse Hessian representation are more economical then those connected with the sparse Jacobian one. On the other hand, if some row of the Jacobian matrix is dense, i.e. if $n_i \sim n$ for some $1 \le i \le r$, then the Hessian matrix is also dense, i.e. $\tilde{m} \sim n(n+1)/2$ and since $\hat{m} \ge \tilde{m}$, both the partitioned Hessian and the sparse Hessian representations cannot be used. An advantage of sparse Hessian representation against the partitioned one is the possibility of using matrix direct methods, based on the sparse Choleski decomposition, which are, in the case of the moderate fill-in, more efficient then matrix iterative methods. We demonstrate, in subsequent parts of our contribution, that sparse Hessian representation is generally more economical then the partitioned one, even if a greater number of hybrid methods can be realized in the partitioned one.

In the sparse case, the most complicated and time consuming part of both the line search and the trust region methods is the direction determination. There are a great variety of ways how this operation can be realized, but since we are concentrated on the effect of improving the Gauss-Newton method by a hybrid approach, we use only simle ones of them.

In connection with all of the above representations, we can use iterative methods based on conjugate gradients. The basic conjugate gradient (CG) algorithm is represented by the following iterative process:

$$d_0 = 0, \quad g_0 = g, \tag{1.14a}$$

$$\gamma_1 = \| g_0 \|^2, \quad p_1 = -g_0 \tag{1.14b}$$

and

$$q_i = Bp_i, \quad \delta_i = p_i^T q_i, \tag{1.14c}$$

$$d_i = d_{i-1} + \frac{\gamma_i}{\delta_i} p_i, \quad g_i = g_{i-1} + \frac{\gamma_i}{\delta_i} q_i, \tag{1.14d}$$

$$\gamma_{i+1} = \| g_i \|^2, \quad p_{i+1} = -g_i + \frac{\gamma_{i+1}}{\gamma_i} p_i \tag{1.14e}$$

for $i \in N$. Note that $g_i = Bd_i + g$ for $i \in N$.

Using the substitution $B = J^T J$ we can transform the basic conjugate gradient algorithm to solve linear least squares problems. We obtain the so-called conjugate gradient least squares (CGLS) algorithm (see [33] as an example) which is represented by the following iterative process:

$$d_0 = 0, \qquad r_0 = -f, \tag{1.15a}$$

$$v_1 = J^T r_0, \qquad \gamma_1 = \| v_1 \|^2, \tag{1.15b}$$

$$p_1 = v_1 \tag{1.15c}$$

and

$$u_i = Jp_i, \qquad \delta_i = \| u_i \|^2, \tag{1.15d}$$

$$d_i = d_{i-1} + \frac{\gamma_i}{\delta_i} p_i, \qquad r_i = r_{i-1} - \frac{\gamma_i}{\delta_i} u_i, \tag{1.15e}$$

$$v_{i+1} = J^T r_i, \qquad \gamma_{i+1} = \| v_{i+1} \|^2, \tag{1.15f}$$

$$p_{i+1} = v_{i+1} + \frac{\gamma_{i+1}}{\gamma_i} p_i \tag{1.15g}$$

for $i \in N$. Note that $r_i = -(Jd_i + f)$ for $i \in N$. The CGLS algorithm is held to be more stable then the CG one for linear least squares problems but, for $n \ll r$, it can be slightly less efficient since it uses a greater number of large vectors.

Both the CG and the CGLS algorithms can be used in truncated forms proposed in [9] for line search and in [38] for trust region strategies respectively. Our implementation is given in [27]. In connection with sparse Jacobian or sparse Hessian representations, we can also use direct methods based on sparse QR [42] or sparse Choleski [18] decompositions respectively. The following procedures will be used in subsequent sections.

(LI) Line search strategy with iterative subalgorithm. Use truncated form of either CG or CGLS algorithms [9] for computation of the direction vector (L1) in line search strategy.

(LD) Line search strategy with direct subalgorithm. Use either sparse QR or sparse Choleski decomposition with possible correction maintaining positive definiteness [19] for computation of the direction vector (L1) in line search strategy.

(TI) Trust region strategy with iterative subalgorithm. Use truncated form of either CG or CGLS algorithms [38] for computation of the direction vector (T1) in trust region strategy.

(TD) Trust region strategy with direct subalgorithm. Use either sparse QR or sparse Choleski decomposition together with optimum step selection [31] for computation of the direction vector (T1) in trust region strategy.

Table 1a: *Test problems for nonlinear least squares.*

No.	Problem	n	r	$\hat{n}$	$\hat{m}$	$\tilde{m}$
1	Chained Rosenbrock function [7]	50	98	147	196	99
2	Chained Wood function [7]	50	144	240	336	99
3	Chained Powell singular [7]	50	96	192	288	123
4	Chained Cragg and Levy function [7]	50	120	192	264	99
5	Generalized Broyden tridiagonal function [7]	50	50	148	294	147
6	Chained Broyden banded function [7]	50	50	334	1308	329
7	Extended Freudenstein and Roth problem [24]	50	98	196	294	99
8	Wright and Holt zero residual problem [44]	48	240	480	720	72
9	Toint quadratic merging problem [39]	50	144	576	1440	171
10	Chained exponential problem [24]	50	99	246	441	147

Table 1b: *Test problems for nonlinear equations.*

No.	Problem	n	r	$\hat{n}$	$\hat{m}$	$\tilde{m}$
1	Countercurrent reactors problem 1 [5]	50	50	196	484	217
2	Countercurrent reactors problem 2 [5]	50	50	243	717	284
3	Trigonometric system [40]	50	50	250	750	150
4	Trigonometric-exponential system TRIGEXP 1 [40]	50	50	148	294	147
5	Trigonometric-exponential system TRIGEXP 2 [40]	49	49	193	501	213
6	Singular Broyden problem [21]	50	50	148	294	147
7	Tridiagonal system [23]	50	50	148	294	147
8	Five-diagonal system [23]	50	50	244	722	240
9	Seven-diagonal system [23]	50	50	338	1324	329
10	Structured Jacobian problem [21]	50	50	384	1685	372
11	Extended Rosenbrock function [30]	50	50	75	100	75
12	Extended Powell singular function [30]	48	48	96	144	96
13	Extended Cragg and Levy function [7]	48	48	84	120	84
14	Broyden tridiagonal function [7]	50	50	148	294	147
15	Broyden banded function [7]	50	50	334	1308	329
16	Extended Powell badly scaled function [30]	50	50	100	150	75
17	Discrete boundary value problem [30]	50	50	148	294	147
18	Modified Broyden tridiagonal problem [30]	50	50	148	294	147

This contribution is organized as follows. In section 2, we study basic ideas of hybrid methods, namely switches for leaving the Gauss-Newton method and techniques for construction of the matrix $B(x)$ in (1.4) from the second order information. In section 3, two hybrid methods suitable for sparse Jacobian representation are proposed. In section 4, three hybrid methods based on partitioned Hessian approximation are studied. In section 5, three hybrid methods, based on sparse Hessian approximation, are investigated. Finally, in section 6, useful comments based on numerical results are presented. Numerical results were obtained using 10 test problems of nonlinear least squares listed in [24] and 18 test problems of nonlinear equations listed in [26]. Names and sizes of these problems are given in tables 1a and 1b.

Problems given in Table 1a are more suitable for testing our hybrid methods than problems given in Table 1b since objective functions corresponding to nonlinear equations have nonzero local minima in many cases. Since a square Jacobian matrix, connected with a system of nonlinear equations, is singular in nonzero local minimum, many iterations are usually needed for finding such a point.

2. Basic ideas of hybrid methods

A typical hybrid method for nonlinear least squares is based on three ideas. The first one is an efficient switch between the Gauss-Newton method and the method based on a second order information. The second one is a technique for construction of the matrix $B(x)$ in (1.4) using a second order information. The last but not least one is an updating technique for obtaining a second order information. We shall give more details about the first two ideas here. The last one is the main purpose of subsequent sections.

First let us concentrate our attention on the conditions for leaving the Gauss-Newton method. The most simple condition of this type was proposed by Fletcher and Xu in [17] as the condition HY2. In fact Fletcher and Xu recommended two additional conditions HY1 and HY3 but the latter ones use values that can be computed from the matrix decomposition only, which is impractical in the sparse case. The condition HY2 can be written in the following form

$$F - F^{+} \leq \bar{\eta}_1 F \tag{2.1}$$

where F and F^{+} are the old and the new values of the objective function respectively and the Gauss-Newton method is left if (2.1) holds.

Another condition was introduced by Dennis and Welsch in [15]. This condition consists in comparing two predicted reductions with the actual one and it can be written in the following form

$$\left| \frac{F^{+} - F}{d^T g + \frac{1}{2} d^T B d} - 1 \right| \leq \bar{\eta}_2 \left| \frac{F^{+} - F}{d^T g + \frac{1}{2} d^T J^T J d} - 1 \right| \tag{2.2}$$

where B is a matrix containing second order information. Condition (2.2) has a disadvantage in that two additional matrix vector products have to be computed. Therefore, it is practical at most in connection with trust region strategy which always uses one of these matrix vector products. Moreover condition (2.2) cannot be used in the case when the matrix J or J^TJ is overwritten, which frequently occurs.

The last condition we have tested was introduced by Ramsin and Wedin in [36]. This condition is based on the observation that the ratio $\| P_J f \| / \| P_{J^-} f^- \|$, where $P_J = J(J^TJ)^{-1}J^T$, is a good estimate of the convergence rate of the Gauss-Newton method at least in a neighborhood of the solution (f^- and J^- are quantities from the previous iteration). A neighborhood of the solution can be detected by comparing the values $\| P_J f \|$ and $\| f \|$. If we use the Gauss-Newton step $d = -(J^TJ)^{-1}J^Tf$ then $P_J f = -Jd$ and the resulting condition has the following form

$$\| Jd \| \leq \bar{\eta}_3 \| f \| \tag{2.3a}$$

and simultaneously

$$\| Jd \| \leq \bar{\eta}_4 \| J^- d^- \| . \tag{2.3b}$$

Condition (2.3) is applicable for direction vectors computed from the Gauss-Newton equation $J^TJd = -J^Tf$ only. This limitation excludes the trust region strategy and it is also partially unsuitable for truncated iterative methods like CG and CGLS. Moreover, condition (2.3) cannot be used if the direction vector is obtained using a second order information, so that we cannot return to the Gauss-Newton method.

The above conditions form a basis for the following switches.

(FX) The Fletcher and Xu switch. Compute the values F and F^+. If condition (2.1) holds, then use a second order information in the next iteration. Otherwise use the Gauss-Newton method in the next iteration.

(DW) The Dennis and Welsch switch. Compute the values $d^Tg + \frac{1}{2}d^TBd$ and $d^Tg + \frac{1}{2}d^TJ^TJd$. If condition (2.2) holds, then use a second order information in the next iteration. Otherwise use the Gauss-Newton method in the next iteration.

(RW) The Ramsin and Wedin switch. If a second order information was used in the current iteration, then use it also in the next iteration. Otherwise compute the values $\| Jd \|$ and $\| f \|$. If conditions (2.3a) and (2.3b) are satisfied, then use a second order information in the next iteration. Otherwise use the Gauss-Newton method in the next iteration.

We anticipate the main computational experiments now and we show the relative efficiency of the three fore-mentioned switches. The simple BFGS update method (SJH1), proposed in the next section, is used for this purpose. We denote SJH1/LI/FX and SJH1/LI/RW line search methods with switches FX and RW respectively and SJH1/TI/FX and SJH1/TI/DW trust region methods with switches FX and DW respectively. Table 2a shows summary results for 10 least squares problems listed in Table 1a. Table 2b shows summary results for 18 nonlinear equations problems listed in Table 1b. These tables contain the total number of iterations NI, the total number of function evaluations NF, the total number of gradient evaluations NG, the total computational time, the number of local solutions NL which was obtained instead of global ones and the number of fails. The method failed when either more than 1000 function evaluations or 500 iterations for nonlinear least squares problems and 800 iterations for nonlinear equations respectively were required. We used the values $\bar{\eta}_1 = 0.01$ for LI/FX, $\bar{\eta}_3 = 0.005$ and $\bar{\eta}_4 = 0.5$ for LI/RW, $\bar{\eta}_1 = 0.0001$ for TI/FX, $\bar{\eta}_2 = 0.85$ for TI/DW respectively. These values were obtained using extensive numerical experiments and they are quite suitable.

Table 2a: *Results for 10 nonlinear least squares problems with 50 variables.*

Method	NI	NF	NG	TIME	NL	FAIL
SJH1/LI/FX	402	588	588	0:34.33	-	-
SJH1/LI/RW	389	567	567	0:36.58	-	-
SJH1/TI/FX	347	465	357	0:28.89	-	-
SJH1/TI/DW	335	456	345	0:31.25	-	-

Table 2b: *Results for 18 nonlinear equations problems with 50 unknowns.*

Method	NI	NF	NG	TIME	NL	FAIL
SJH1/LI/FX	1033	1492	1492	1:40.94	5	-
SJH1/LI/RW	1064	1628	1628	1:40.62	5	-
SJH1/TI/FX	1474	1785	1492	1:23.93	4	-
SJH1/TI/DW	1438	1800	1456	1:26.50	4	-

Tables 2a and 2b show that the most simple switch FX is at least as efficient as the more complicated and often unusable switches DW and RW. Therefore we merely use the switch FX in the subsequent sections.

Now let us briefly describe techniques for the construction of the matrix $B(x)$ using a second order information. We consider the following possibilities.

(SU) Simple quasi-Newton update. If the Gauss-Newton method should be left, then compute the matrix B^+ from the matrix $(J^+)^T J^+$ using a quasi-Newton update, otherwise set $B^+ = (J^+)^T J^+$. Similar procedure is applied to the matrix J^+ when the sparse Jacobian representation is used.

(CU) Cumulative quasi-Newton update. If the Gauss-Newton method should be left, then compute the matrix B^+ from the matrix B using a quasi-Newton update, otherwise set $B^+ = (J^+)^T J^+$. Similar procedure is applied to the matrix J^+ when the sparse Jacobian representation is used.

(DA) Difference approximation of the second order term. If the Gauss-Newton method should be left, then compute an approximation of the matrix S^+ using differences of gradients and set $B^+ = (J^+)^T J^+ + S^+$, otherwise set $B^+ = (J^+)^T J^+$. The matrix S^+ need not be stored separately since the second order information is immediately substituted into the matrix B^+.

(QA) Quasi-Newton approximation of the second order term. Compute the matrix S^+ from the matrix S using a quasi-Newton update. If the Gauss-Newton method should be left, then set $B^+ = (J^+)^T J^+ + S^+$, otherwise set $B^+ = (J^+)^T J^+$.

Note that technique QA requires two matrices B and S while techniques SU, CU, DA uses the matrix B only.

3. Hybrid methods for sparse Jacobian representation

In this section, we propose two hybrid methods realized as simple quasi-Newton updates (SU). The first one is based on so-called product form [6] or factorized [43] quasi-Newton updates. We need to find the updated Jacobian matrix

$$J_u^+ = J^+ + uv^T \tag{3.1}$$

with the vectors $u \in R^r$ and $v \in R^n$ chosen in such a way that the quasi-Newton condition

$$B^+ s = (J_u^+)^T J_u^+ s = y \tag{3.2}$$

is satisfied, where

$$s = x^+ - x, \qquad y = g^+ - g. \tag{3.3}$$

There exist equivalents of the form (3.1) for all members of the convex part of the Broyden family, but we shall restrict our attention only to the BFGS method that corresponds to the choice

$$u = J^+ s / \parallel J^+ s \parallel, \qquad v = y/\sqrt{y^T s} - (J^+)^T u. \tag{3.4}$$

Formulas (3.1) and (3.4) form a basis for the first hybrid method which we denote as SJH1. It consists of the update (3.1) and (3.4) which is used whenever a second order information should be considered. This method can be used together with CG based iterative subalgorithms LI and TI only since we have to solve the linear equation $J_u^T J_u d = J^T f$ which is not equivalent to the linear least squares problem with the objective function $\frac{1}{2} \parallel J_u d + f \parallel^2$. Using the CG subalgorithm, we have to compute two matrix vector products $r_i = J_u p_i = J p_i + v^T p_i u$ and $q_i = J_u^T r_i = J^T r_i + u^T r_i v$ (see (1.14c)) which is a very easy operation.

The next hybrid method is based on the rank one formula. Consider the augmented linear least squares problem with the objective function $\frac{1}{2} \parallel J_a^+ d^+ + f_a^+ \parallel^2$ where

$$J_a^+ = \begin{bmatrix} J^+ \\ w \end{bmatrix}, \quad f_a^+ = \begin{bmatrix} f^+ \\ 0 \end{bmatrix}. \tag{3.5}$$

Then using (1.4) and (1.5) we obtain $B^+ d^+ = (J^+)^T f^+$ where

$$B^+ = (J_a^+)^T J_a^+ = (J^+)^T J^+ + ww^T \tag{3.6}$$

which together with the choice

$$w = (y - (J^+)^T J^+ s) / \sqrt{s^T(y - (J^+)^T J^+ s)} \tag{3.7}$$

gives exactly the rank one quasi-Newton update. Note that (3.7) can be used only if $s^T(y - (J^+)^T J^+ s) > 0$ which slightly restricts the use of the update (3.5).

Formulas (3.5) and (3.7) form a basis for the second hybrid method which we denote as SJH2. It consists of the update (3.5) and (3.7) which is used whenever a second order information should be considered and $s^T(y - (J^+)^T J^+ s) > 0$ holds simultaneously. This method can be used together with all subalgorithms LI, LD, TI, TD since we can solve the linear least squares problem with the objective function $\frac{1}{2} \parallel J_a d + f_a \parallel^2$. The matrix J_a differs from the matrix J only in the last row which is of course dense.

Now we can give a computational comparison of two hybrid methods SJH1 and SJH2 together with the sparse Jacobian Gauss-Newton method SJGN. This comparison is shown in tables 3a and 3b which have the same meaning as tables 2a and 2b. Again 10 least squares problems and 18 nonlinear equations problems given in tables 1a and 1b were used. The results correspond to FX switch with the values $\bar{\eta}_1 = 0.01$ and $\bar{\eta}_1 = 0.0001$ in (2.1) for line search and trust region realizations respectively.

Table 3a: *Results for 10 nonlinear least squares problems with 50 variables.*

Method	NI	NF	NG	TIME	NL	FAIL
SJGN/LI	450	675	675	0:41.41	-	-
SJH1/LI	402	588	588	0:34.33	-	-
SJH2/LI	421	600	600	0:37.29	-	-
SJGN/TI	407	553	417	0:38.67	-	-
SJH1/TI	347	465	357	0:28.89	-	-
SJH2/TI	362	508	372	0:34.16	-	-

Tables 3a and 3b show that simple quasi-Newton updates usually improve an efficiency of the Gauss-Newton method. The only exception is the first row of Table 3b, where an excelent result of the Gauss-Newton method was caused by surprising success in obtaining nonzero local minimum of the TRIGEXP 2 problem.

Table 3b: *Results for 18 nonlinear equations problems with 50 unknowns.*

Method	NI	NF	NG	TIME	NL	FAIL
SJGN/LI	909	1434	1434	1:31.12	5	-
SJH1/LI	1033	1492	1492	1:40.94	5	-
SJH2/LI	1113	1631	1631	1:59.13	5	-
SJGN/TI	1965	2121	1983	1:33.49	4	1
SJH1/TI	1474	1785	1492	1:23.93	4	-
SJH2/TI	1101	1298	1119	1:01.69	4	-

4. Hybrid methods for partitioned Hessian representation

In this section, we propose three different hybrid methods. The first one is realized as a cumulative quasi-Newton update (CU). This update is in fact the partitioned rank-one update introduced in [22]. Let $\hat{g}_i^+, 1 \le i \le r$, be new packed gradients. Define either

$$\hat{B}_i^+ = \hat{B}_i + \frac{(\hat{z}_i - \hat{B}_i\hat{s}_i)(\hat{z}_i - \hat{B}_i\hat{s}_i)^T}{\hat{s}_i^T(\hat{z}_i - \hat{B}_i\hat{s}_i)}. \tag{4.1a}$$

if a second order information should be considered and $| \hat{s}_i^T(\hat{z}_i - \hat{B}_i\hat{s}_i) | > \bar{\eta}_0$ or

$$\hat{B}_i^+ = \hat{B}_i \tag{4.1b}$$

if a second order information should be considered and $| \hat{s}_i^T(\hat{z}_i - \hat{B}_i\hat{s}_i) | \le \bar{\eta}_0$ or

$$\hat{B}_i^+ = \hat{g}_i^+(\hat{g}_i^+)^T \tag{4.1c}$$

otherwise, where $\hat{z}_i = f_i^+\hat{g}_i^+ - f_i\hat{g}_i$ and where $\hat{s}_i$ are packed vectors which contains elements of the vector $s = x^+ - x$ with indices contained in $ind\ \hat{g}_i, 1 \le i \le r$. In this way we obtain packed matrices $\hat{B}_i^+, 1 \le i \le r$, which define the partitioned matrix $\hat{B}^+$ as it was shown in section 1.

Formulas (4.1) form a basis for the first hybrid method which we denote as PHH1. In connection with the switch FX, it is exactly a partitioned variant of the HY2 hybrid method proposed by Fletcher and Xu in [17]. We chose the rank-one update since the matrices $\hat{B}_i^+, 1 \le i \le r$, can be indefinite even if the matrix $\hat{B}^+$ is positive definite. In fact we have tested some other updates, BFGS update as an example, but rank-one update was found most efficient.

The next hybrid method is of QA type. It is based on the rank-one update applied to the approximations $\hat{G}_i$ of the packed Hessian matrices $\hat{G}_i(x), 1 \le i \le r$. These matrices are stored in the extra vector

$$\hat{G} = \begin{bmatrix} \check{G}_1 \\ \dots \\ \check{G}_r \end{bmatrix}$$

like the matrices $\hat{B}_i, 1 \le i \le r$, (see section 1) and they are updated in such a way that either

$$\hat{G}_i^+ = \hat{G}_i + \frac{(\hat{y}_i - \hat{G}_i\hat{s}_i)(\hat{y}_i - \hat{G}_i\hat{s}_i)^T}{\hat{s}_i^T(\hat{y}_i - \hat{G}_i\hat{s}_i)}. \tag{4.2a}$$

if $| \hat{s}_i^T(\hat{y}_i - \hat{G}_i\hat{s}_i) | > \bar{\eta}_0$ or

$$\hat{G}_i^+ = \hat{G}_i \tag{4.2b}$$

otherwise, where $\hat{y}_i = \hat{g}_i^+ - \hat{g}_i$, $1 \le i \le r$. Then we define either

$$\hat{B}_i^+ = \hat{g}_i^+(\hat{g}_i^+)^T + f_i^+\hat{G}_i^+ \tag{4.2c}$$

if a second order information should be considered or

$$\hat{B}_i^+ = \hat{g}_i^+(\hat{g}_i^+)^T \tag{4.2d}$$

otherwise. In the first iteration we set $G_i = I, 1 \le i \le r$.

Formulas (4.2) form a basis for the second hybrid method which we denote as PHH2. It is very similar to the method proposed in [8] which uses the unsymmetric Broyden update instead of (4.2a). In fact we have tested some other updates instead of (4.2a) but rank-one update was found most efficient.

The last hybrid method is of DA type. It is based on the difference approximation of the packed Hessian matrices using the formula

$$(e_k^i)^T\hat{G}_i(x^+)e_l^i \approx (e_k^i)^T\hat{G}_i^+e_l^i = \frac{1}{2\delta}((\hat{g}_i(x^+ + \delta e_l^i) - g_i(x^+))^T e_k^i + (\hat{g}_i(x^+ + \delta e_k^i) - g_i(x^+))^T e_l^i) \tag{4.3a}$$

where e_k^i and e_l^i are k-th and l-th columns of the n_i dimensional unit matrix respectively. Then we define either

$$\hat{B}_i^+ = \hat{g}_i^+(\hat{g}_i^+)^T + f_i^+\hat{G}_i^+ \tag{4.3b}$$

if a second order information should be considered or

$$\hat{B}_i^+ = \hat{g}_i^+(\hat{g}_i^+)^T \tag{4.3c}$$

otherwise, for $1 \le i \le r$.

Formulas (4.3) form a basis for the third hybrid method which we denote as PHH3. This method is inexpensive since the current matrix $\hat{G}_i^+$ of the dimension n_i have to be stored and n_i gradients have to be computed only for $1 \le i \le r$. Let $n_{max} =$

$max(n_i, 1 \leq i \leq r)$ and $\hat{n} = \sum_{i=1}^{r} n_i$. Then we need to store an extra vector of the dimension $n_{max}(n_{max}+1)/2$ and we need to evaluate $\hat{n}/r$ equivalent gradients.

Now we can give computational comparison of three hybrid methods PHH1, PHH2, PHH3 together with the partitioned Hessian variants of the modified Newton PHMN, quasi-Newton PHQN, Gauss-Newton PHGN methods respectively (the PHMN method was realized using gradient differences and the PHQN method was realized as the partitioned rank-one update). This comparison is shown in tables 4a and 4b which have the same meaning as tables 2a and 2b. Again 10 least squares problems and 18 nonlinear equations problems given in tables 1a and 1b were used. The results correspond to FX switch with values $\bar{\eta}_1 = 0.005$ and $\bar{\eta}_1 = 0.0001$ in (2.1) for line search and trust region realizations respectively (PHH3 method used the value $\bar{\eta}_1 = 0.005$ in all the cases).

Table 4a: *Results for 10 nonlinear least squares problems with 50 variables.*

Method	NI	NF	NG	TIME	NL	FAIL
PHMN/LI	380	537	1694	1:09.65	-	-
PHQN/LI	673	1074	1074	0:55.86	-	-
PHGN/LI	441	629	629	0:38.95	-	-
PHH1/LI	378	497	778	0:29.66	-	-
PHH2/LI	360	446	446	0:33.51	-	-
PHH3/LI	354	482	1080	0:38.06	-	-
PHMN/TI	296	336	956	0:58.68	-	-
PHQN/TI	1068	1332	1332	1:25.30	-	1
PHGN/TI	411	563	421	0:40.64	-	-
PHH1/TI	341	417	351	0:29.82	-	-
PHH2/TI	267	345	277	0:28.94	-	-
PHH3/TI	249	275	402	0:26.42	-	-

Table 4b: *Results for 18 nonlinear equations problems with 50 unknowns.*

Method	NI	NF	NG	TIME	NL	FAIL
PHMN/LI	839	1282	4937	1:53.37	5	-
PHQN/LI	1263	1992	1992	1:31.51	7	-
PHGN/LI	1165	1887	1887	2:52.58	5	-
PHH1/LI	768	979	1555	1:12.17	5	-
PHH2/LI	770	984	984	1:24.31	4	-
PHH3/LI	749	1182	2617	1:19.42	5	-
PHMN/TI	767	869	3372	1:51.61	4	-
PHQN/TI	2329	2798	2347	2:41.15	4	1
PHGN/TI	1969	2141	1987	2:23.14	4	1
PHH1/TI	902	1061	920	1:23.87	4	-
PHH2/TI	768	871	786	1:21.18	4	-
PHH3/TI	670	744	1076	1:09.65	4	-

Tables 4a and 4b show that the proposed hybrid methods considerably outperform pure ones, especially in connection with trust region strategy.

5. Hybrid methods for sparse Hessian representation

In this section, we propose three different hybrid methods. The first one is realized as a cummulative quasi-Newton update (CU). This update is in fact the sparse Marwill update introduced in [29]. Denote $P_i \in R^{n\times n}, 1 \le i \le n$, diagonal matrices such that $e_j^T P_i e_j = 0$ if $e_j^T B e_i = 0$ and $e_j^T P_i e_j = 1$ otherwise, for $1 \le j \le n$. Then P_i is the orthogonal projection matrix which projects any vector to the subspace of vectors having the same sparse structure as the i-th row of the matrix B. Define

$$U^+ = B + \sum_{i=1}^{n} ((P_i s)^T P_i s)^\dagger e_i^T (y - Bs) e_i (P_i s)^T \tag{5.1a}$$

where U^+ is an unsymmetric matrix which has the same sparsity pattern as the matrix B and where $a^\dagger$ is a pseudoinverse of a ($a^\dagger = 0$ if $a = 0$ and $a^\dagger = 1/a$ otherwise). Then we set either

$$B^+ = \frac{1}{2}(U^+ + (U^+)^T) \tag{5.1b}$$

if a second order information should be considered or

$$B^+ = (J^+)^T J^+ \tag{5.1c}$$

otherwise.

Formulas (5.1) form a basis for the first hybrid method which we denote SHH1. In connection with the switch FX, it is exactly a sparse variant of the HY2 hybrid method proposed by Fletcher and Xu in [17]. We chose the Marwill update since it was found to be best among all sparse updates we have tested. This observation is also mentioned in [14] and [41].

The next hybrid method, which we denote SHH2, is exactly a sparse variant of the method PHH2 described in the previous section. This method uses the formulas (4.2) again, but the matrices $\hat{B}_i^+, 1 \le i \le r$, are not stored, they are directly added into the sparse structure $\tilde{B}^+$. The SSH2 method is not suitable when we are limited by the storage since the partitioned structure $\hat{G}^+$ have to be stored.

The last hybrid method, which we denote SHH3, is exactly a sparse variant of the method PHH3 described in the previous section. This method uses the formulas (4.3) again, but the matrices $\hat{B}_i^+, 1 \le i \le r$, are not stored, they are directly added into the sparse structure $\tilde{B}^+$. Note that the SSH3 method do not use any partitioned structure, it needs an extra vector of the dimension $n_{max}(n_{max}+1)/2$ only.

Now we can give a computational comparison of the three hybrid methods SHH1, SHH2, SHH3 together with the sparse Hessian variants of the modified Newton SHMN, quasi-Newton SHQN, Gauss-Newton SHGN methods respectively (the SHMN method was realized using gradient differences and the SHQN method was realized as the sparse Marwill update). This comparison is shown in tables 4a and 4b which have the same meaning as tables 2a and 2b. Again 10 least squares problems and 18 nonlinear equations problems given in tables 1a and 1b were used. The results correspond to FX switch with values $\bar{\eta}_1 = 0.01$ and $\bar{\eta}_1 = 0.0001$ in (2.1) for line search and trust region realizations respectively (SHH3 method used the value $\bar{\eta}_1 = 0.005$ in all the cases).

Table 5a: *Results for 10 nonlinear least squares problems with 50 variables.*

Method	NI	NF	NG	TIME	NL	FAIL
SHMN/LI	304	419	1380	0:55.75	-	-
SHQN/LI	870	1298	1298	0:44.55	1	-
SHGN/LI	500	733	733	0:37.41	-	-
SHH1/LI	423	576	818	0:23.12	-	-
SHH2/LI	339	457	457	0:26.25	-	-
SHH3/LI	313	451	968	0:29.61	-	-
SHMN/LD	263	456	1339	0:54.76	1	-
SHQN/LD	1569	3338	3338	1:45.74	1	-
SHGN/LD	841	1377	1377	0:53.55	-	-
SHH1/LD	308	518	704	0:17.90	-	-
SHH2/LD	226	387	387	0:20.54	-	-
SHH3/LD	206	396	700	0:24.55	1	-
SHMN/TI	299	345	966	0:50.20	-	-
SHQN/TI	1226	1584	1236	1:03.27	1	-
SHGN/TI	467	638	477	0:30.32	-	-
SHH1/TI	347	471	357	0:19.06	-	-
SHH2/TI	297	372	307	0:24.17	-	-
SHH3/TI	252	275	395	0:19.77	-	-
SHMN/TD	238	291	779	0:43.22	1	-
SHQN/TD	1596	1929	1606	2:25.16	1	1
SHGN/TD	346	468	356	0:21.64	-	-
SHH1/TD	243	367	253	0:17.14	-	-
SHH2/TD	220	274	230	0:19.88	-	-
SHH3/TD	191	217	291	0:16.42	-	-

Tables 4a and 4b show that the proposed hybrid methods are much more efficient than pure ones. The only exception is the SHH1/LD method (see Table 5b) which failed on the Powell badly scaled function. It is interesting that the simple update variant of the SHH1/LD method was much more efficient in this case (we obtained results NI=401, NF=646, NG=646, TIME=0:15.82, NL=1, FAIL=0). For all other realizations (LI, TI, TD) the cumulative update variant was better.

Table 5b: *Results for 18 nonlinear equations problems with 50 unknowns.*

Method	NI	NF	NG	TIME	NL	FAIL
SHMN/LI	843	1356	5027	1:32.11	6	-
SHQN/LI	2373	3595	3595	1:58.42	6	1
SHGN/LI	863	1346	1346	1:05.31	5	-
SHH1/LI	792	1012	1543	0:46.41	5	-
SHH2/LI	689	877	877	0:52.29	5	-
SHH3/LI	704	1164	2404	0:53.28	5	-
SHMN/LD	636	2131	5128	1:13.88	6	1
SHQN/LD	3600	8071	8071	2:27.53	8	5
SHGN/LD	695	1463	1463	0:39.16	2	1
SHH1/LD	1021	1580	1877	0:29.55	2	1
SHH2/LD	405	665	665	0:23.51	2	-
SHH3/LD	316	667	1107	0:17.36	3	-
SHMN/TI	733	825	3229	1:19.31	4	-
SHQN/TI	4605	5334	4623	4:32.54	5	3
SHGN/TI	1970	2159	1988	1:25.85	4	1
SHH1/TI	856	994	874	0:46.58	4	-
SHH2/TI	827	939	845	0:58.11	4	-
SHH3/TI	688	771	1105	0:47.18	4	-
SHMN/TD	1184	1264	4985	2:01.22	5	1
SHQN/TD	4282	4876	4300	7:55.00	8	3
SHGN/TD	1797	1895	1815	2:29.46	3	1
SHH1/TD	707	817	729	0:47.46	3	-
SHH2/TD	462	559	480	0:47.78	3	-
SHH3/TD	267	351	453	0:24.45	3	-

6. Conclusions

Before formulating our conclusions we need to make several comments on the implementation of the above methods. All methods were implemented using modular interactive system for universal functional optimization UFO [28]. This is an extensive software system containing more then 1200 Fortran modules realizing basic parts of optimization methods. For this reason all methods were realized using the same line search or trust region strategies and with the same matrix operations. Therefore, the results are quite comparable and they show real efficiency of individual methods.

The problems used for testing our methods are given in tables 1a and 1b together with the sizes of individual representations (numbers n, r, $\hat{n}$, $\hat{m}$, $\tilde{m}$). The efficiency of the methods depended on the number of nonzero local minima (NL) in such a way that each nonzero local minimum usually increased the total number of iterations and function evaluations. Therefore, what is really comparable are only results with the same number of nonzero local minima. On the other hand, the method which gives a lower number of nonzero local minima is more suitable for the computations. It was pointed out in [26] that truncated CG and CGLS methods tend to find nonzero local minima so that better results were obtained with direct methods (LD and TD).

According to the results presented in our tables and the comments stated above, we can express several conclusions (which of course hold only for our collection of test problems):

(C1) When both function and gradient evaluations are inexpensive then methods based on sparse Hessian representation are most effective. It follows from the fact that often $\tilde{m} < \hat{n} < \hat{m}$ and, therefore, matrix operations connected with sparse Hessian representation are most economical.

(C2) Hybrid methods considerably outperform the Gauss-Newton method. They are sensitive to the condition for leaving the Gauss-Newton method, namely to constant $\hat{\eta}_1$ in (2.1). Cumulative update (CU) methods are usually better than SU methods. Simple update (SU) methods for sparse Jacobian representation could be easily generalized as limited memory CU methods.

(C3) We do not recommend PHH2 and SHH2 methods which have greater storage requirements and are not more efficient than PHH1, PHH3 and SHH1, SHH3 methods respectively. The PHH3 and SHH3 methods are very effective, especially in connection with trust region strategy, when gradient evaluations are not expensive.

(C4) Methods based on matrix decompositions are usually more advantageous for nonlinear equations then those based on truncated CG or CGLS subalgorithms in the sense that they find the global minima more frequently. Matrix direct methods are also more economical then unpreconditioned matrix iterative methods, measured by computational time, if fill-in is moderate.

Finally, let us recommend some areas for future research. First, since a condition for leaving Gauss-Newton method is a crucial point of hybrid methods and since we have used only a simple one, it could be useful to develop additional efficient possibilities. Furthermore, the sparse version of the factorized quasi-Newton update SJH1 could be studied and tested. Finally, the limited memory variants of both the SJH1 and SJH2 updates could be implemented and tested.

References

1. M.Al-Baali, R.Fletcher: Variational methods for nonlinear least squares. Journal of Optimization Theory and Applications 36 (1985) 405-421.

2. J.T.Betts: Solving the nonlinear least square problem: Application of a general method. Journal of Optimization Theory and Applications 18 (1976) 469-483.

3. M.C.Bartholomew-Biggs: The estimation of the Hessian matrix in nonlinear least squares problems with non-zero residuals. Mathematical Programming 12 (1977) 67-80.

4. A.Bjorck: Least squares methods. In: "Handbook of numerical analysis, Vol.1" (P.G.Ciarlet, J.L.Lions, eds.) Elsevier Science Publishers (North-Holland), Amsterdam 1990.

5. I.D.L.Bogle, J.D.Perkins: A new sparsity preserving quasi-Newton update for solving nonlinear equations. SIAM Journal on Scientific and Statistical Computations, Vol 11, pp. 621-630, 1990.

6. K.W.Brodlie, A.R.Gourlay, J.Greenstadt: Rank-one and rank-two corrections to positive definite matrices expressed in product form. J. Inst. Maths. Applics. 11 (1973), 73-82.

7. K.M.Brown, J.E.Dennis: A new algorithm for nonlinear least squares curve fitting. In: "Mathematical Software" (J.Rice ed.) Academic Press, London 1971.

8. A.R.Conn, N.I.M.Gould, P.L.Toint: Testing a class of methods for solving minimization problems with simple bounds on the variables. Mathematics of Computation, Vol. 50, pp.399-430, 1988.

9. R.S.Dembo, T.Steihaug: Truncated-Newton algorithms for large-scale unconstrained minimization. Mathematical Programming 26 (1983) 190-212.

10. J.E.Dennis: Some computational techniques for the nonlinear least squares problem. In: "Numerical solution of nonlinear algebraic equations" (G.D.Byrne, C.A.Hall, eds.) Academic Press, London 1974.

11. J.E.Dennis, D.M.Gay, R.E.Welsch: An adaptive nonlinear least-squares algorithm. ACM Transactions on Mathematical Software 7 (1981) 348-368.

12. J.E.Dennis, H.H.W.Mei: An unconstrained optimization algorithm which uses function and gradient values. Report No. TR-75-246. Dept. of Computer Science, Cornell University 1975.

13. J.E.Dennis, R.B.Schnabel: Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, New Jersey 1983.

14. J.E.Dennis, R.B.Schnabel: A view of unconstrained optimization. In: "Handbook in Operations Research and Mathematical Sciences, Vol.1" (G.L.Nemhauser, eds.) Elsevier Science Publishers (North-Holland), Amsterdam 1989.

15. J.E.Dennis, R.E.Welsch: Techniques for nonlinear least squares and robust regression. Communications in Statistics, B7 (1978) 345-359.

16. R.Fletcher: Practical Methods of Optimization. J.Wiley & Sons, Chichester, second edition, 1987.

17. R.Fletcher, C.Xu: Hybrid methods for nonlinear least squares. IMA Journal of Numerical Analysis 7 (1987) 371-389.

18. A.George, J.W.H.Liu: Computer solution of large sparse positive definite systems. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1981.

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

20. G.H.Golub, C.F.Van Loan: Matrix computations (second edition). Johns Hopkins University Press, Baltimore 1989.

21. M.A.Gomez-Ruggiero, J.M.Martinez, A.C.Moretti: Comparing algorithms for solving sparse nonlinear systems of equations. SIAM Journal on Scientific and Statistical Computations, Vol. 13, pp. 459-483, 1992.

22. A.Griewank, P.L.Toint: Partitioned variable metric updates for large scale structured optimization problems. Numerische Mathematik 39 (1982) 119-137.

23. G.Li: Successive column correction algorithms for solving sparse nonlinear systems of equations. Mathematical Programming, Vol. 43, pp. 187-207, 1989.

24. L.Lukšan: Inexact trust region method for large sparse nonlinear least squares. Kybernetika 29 (1993) 305-324.

25. L.Lukšan: Computational experience with known variable metric updates. Journal of Optimization Theory and Applications 83 (1994) XXX-XXX.

26. L.Lukšan: Inexact trust region method for large sparse systems of nonlinear equations. Journal of Optimization Theory and Applications 81 (1994) 569-590..

27. L.Lukšan: Combined trust region methods for nonlinear least squares. Report No. 555, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague 1993.

28. L.Lukšan, M.Šiška, M.Tůma, N.Ramešová: Interactive system for universal functional optimization (UFO). Report No. 545, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague 1992.

29. E.S.Marwill: Exploiting sparsity in Newton-like methods. Ph.D. Thesis, Cornell University, Ithaca 1978.

30. J.J.Moré, B.S.Garbow, K.E.Hillström: Testing unconstrained optimization software. ACM Transactions on Mathematical Software, Vol. 7, pp. 17-41, 1981.

31. J.J.Moré, D.C.Sorensen: Computing a trust region step. SIAM Journal on Scientific and Statistical Computations 4 (1983) 553-572.

32. S.S.Oren, D.G.Luenberger: Self scaling variable metric (SSVM) algorithms. Part 1 - criteria and sufficient condition for scaling a class of algorithms. Management Sci. 20 (1974) 845-862.

33. C.C.Paige and M.A.Saunders: LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Transactions on Mathematical Software 8 (1982) 43-71.

34. M.J.D.Powell: A new algorithm for unconstrained optimization. In: "Nonlinear Programming" (J.B.Rosen O.L.Mangasarian, K.Ritter, eds.) Academic Press, London 1970.

35. M.J.D.Powell: On the Global Convergence of Trust Region Algorithms for Unconstrained Minimization. Mathematical Programming 29 (1984) 297-303.

36. H.Ramsin, P.A.Wedin: A comparison of some algorithms for the nonlinear least squares problem. BIT 17 (1977) 72-90.

37. G.A.Shultz, R.B.Schnabel, R.H.Byrd: A Family of Trust-Region-Based Algorithms for Unconstrained Minimization with Strong Global Convergence Properties. SIAM Journal on Numerical Analysis 22 (1985) 47-67.

38. T.Steihaug: The conjugate gradient method and trust regions in large-scale optimization. SIAM Journal on Numerical Analysis 20 (1983) 626-637.

39. P.L.Toint: Numerical solution of large sets of algebraic equations. Mathematics of Computation, Vol. 46, pp. 175-189, 1986.

40. P.L.Toint: On large scale nonlinear least squares calculations. SIAM J. on Scientific and Statistical Computations 8 (1987) 416-435.

41. M.Tůma: Sparse fractioned variable metric updates. Report No. 497, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague 1991.

42. M.Tůma: Intermediate fill-in in sparse QR decomposition. In: "Linear Algebra for Large Scale and Real-Time Applications", (B.de Moor, G.H.Golub, M.Moonen, eds.), Kluwer Academic Publishers, London 1993, pp. 475-476.

43. H.Yabe, T.Takahashi: Factorized quasi-Newton methods for nonlinear least squares problems. Programming 51 (1991) 75-100.

44. S.J.Wright, J.N.Holt: An inexact Levenberg-Marquardt method for large sparse nonlinear least squares. J. Australian Mathematical Society, Ser. B 26 (1985) 387-403.