MATHEMATICS OF COMPUTATION
VOLUME 50, NUMBER 182
APRIL 1988, PAGES 399-430

Testing a Class of Methods for Solving Minimization Problems with Simple Bounds on the Variables

By Andrew R. Conn,* Nicholas I. M. Gould, and Philippe L. Toint

Abstract. We describe the results of a series of tests for a class of new methods of trust region type for solving the simple bound constrained minimization problem. The results are encouraging and lead us to believe that the methods will prove useful in solving large-scale problems.

1. Introduction. We are concerned with solving the simple bound constrained minimization problem

$$\begin{aligned} &\underset{x\in\mathbf{R}^n}{\text{minimize}}\, f(x) \\ &\text{subject to the simple bounds } l \leq x \leq u \end{aligned} \tag{1.1}$$

by a method of the trust region type. In this problem, $f(x)$ is assumed to be sufficiently smooth, l and u are fixed vectors and the inequalities are taken componentwise.

Such problems arise quite naturally in a number of different circumstances. Indeed, some authors (see Gill, Murray and Wright [14]) maintain that the variables for the vast majority of optimization problems can only be considered meaningful within specific intervals and consequently should be solved with simple bound constraints.

Many algorithms for solving (1.1) have been proposed. Early methods were of the active set variety in which a sequence of problems are solved. In these methods a subset of the variables (the active set) are fixed at bounds and the objective function minimized with respect to the remaining variables. Such algorithms typically use line searches to force convergence, and both direct (Fletcher and Jackson [10], Gill and Murray [12]) and iterative (O'Leary [21]) methods have been suggested for solving the linear systems which arise during each iteration. A significant drawback of such methods for large-scale problems appears to be that the active sets can only change slowly, and many iterations are necessary to correct for a bad initial choice. More recently, methods which allow a rapid change of the active set to occur have been proposed (Bertsekas [1], Dembo and Tulowitzki [7]); these methods perform line searches along a search direction which is 'bent' to keep the iterates within the feasible region. Although a convergence theory for such methods can be given for convex problems, it is not obvious how to adapt the theory to more general problems. Alternatively, rapid changes in the active set are possible if sufficient effort

Received October 6, 1986; revised August 24, 1987.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 65K05; Secondary 90C30.
Key words and phrases. Trust regions, optimization with simple bounds, numerical results.
*The research of this author was supported in part by NSERC grant A8639.

©1988 American Mathematical Society
0025-5718/88 $1.00 + $.25 per page

is expended in computing the search direction at each iteration. For instance, if the search direction subproblem is one of minimizing a suitable quadratic model subject to appropriate bounds on the components of the direction, the combinatorial problem of deciding which variables lie on bounds may be relegated to the subproblem. The success of such methods is thus dependent on being able to solve the resulting subproblems efficiently, the subproblem of course being a special case of (1.1). Line-search methods based on this idea may be formulated as special cases of sequential quadratic programming methods; see, for example, Garcia-Palomares [11] or Gill et al. [13]. We know of no experience with this class of method.

In this paper we propose a new algorithm for which we have already produced a general convergence theory (Conn, Gould and Toint [3]) and which allows rapid changes in the active set. At each iteration of the algorithm, we define a (quadratic) approximation to the objective function and a region surrounding the current iterate in which we believe this approximation to be adequate. The algorithm then finds a feasible candidate for the next iterate in this region. This candidate is chosen so as to give a sufficient reduction in the value of the (quadratic) model of the objective function. If the function value calculated at this point matches its predicted value closely enough, the new point is accepted as the next iterate and the trust region is possibly enlarged; otherwise, the point is rejected and the trust region size decreased. We stress that *under no circumstances* is a line search performed to try to improve upon this new point.

A particularly attractive aspect of the algorithm proposed is that, by generalizing the standard notion of a Cauchy point to accommodate the bound constraints, in what seems to us a natural manner, one is able to extend the very general known convergence results for trust region methods applied to unconstrained problems (see Moré [18]). Moreover, the framework presented is well suited to large-dimensional problems and can be used in conjunction with partitioned secant updating techniques on the general class of partially separable problems (Griewank and Toint [15], [16]). However, the issues involved in solving large-scale problems are by no means trivial, and the design of sophisticated software for such problems is a major task that we postpone for a future paper. The purpose of the current work is to demonstrate the viability of the methods proposed (and whose convergence is studied) in our previous paper. Consequently, the numerical results presented below are for small dense problems—the largest containing 45 variables.

The paper is organized as follows. In Section 2 we give a general introduction to the framework of our algorithms and we discuss the issues involved in determining our generalization of the Cauchy point in Section 3. The specific algorithm used in our tests is then given in Section 4. In Section 5, details of our testing and the numerical results obtained are presented. They include a comparison of various possible choices in the context of the considered algorithmic class and also a comparison with existing quality software for the same problem. Both of these comparisons yield unexpected results that are commented on. Finally, a general discussion of some of the directions for future research follows in Section 6. A detailed list of the test problems that we have used is given in an appendix.

2. The Basic Algorithm. As we have already stated, our algorithm is essentially of the model-trust region variety. The description we give here is a special

case of the more general framework presented by Conn, Gould and Toint [3]. We need to define the following concepts. The set of all points x for which $l \leq x \leq u$ is the *feasible region* for the problem (1.1) and any point x in the feasible region is *feasible*. Let a and b be any two vectors for which $a \leq b$. The *active set with respect to the vectors a and b at the point x*, $I(x, a, b)$, is the index set $\{i$ for which $x_i \leq a_i$ or $x_i \geq b_i$ (or both)$\}$. In particular, we shall refer to $I(x, l, u)$ as *the active set* at x and denote it by $I(x)$. We now define the projection operator $P[\cdot]$ (componentwise) by

$$P[x, a, b]_i = \begin{cases} a_i & \text{if } x_i \leq a_i, \\ b_i & \text{if } x_i \geq b_i, \\ x_i & \text{otherwise.} \end{cases}$$

For brevity we shall write $P[x] = P[x, l, u]$ and will refer to $P[x]$ as the *projection of x onto the feasible region.*

At the kth stage of the algorithm, we suppose that we have a feasible point $x^{(k)}$, the gradient, $g^{(k)}$, of $f(x)$ at $x^{(k)}$ and a suitable symmetric approximation $B^{(k)}$ to the Hessian matrix of the objective function at this point. Furthermore, we require a scalar $\Delta^{(k)}$ which represents a bound upon displacements around $x^{(k)}$ within which we believe that a second-order approximation,

$$m^{(k)}(x^{(k)} + s) \equiv f(x^{(k)}) + g^{(k)T}s + \tfrac{1}{2}s^T B^{(k)}s, \tag{2.1}$$

to the objective function will effectively agree with the true function. A trial point $\overline{x}^{(k+1)} = x^{(k)} + s^{(k)}$ is constructed by finding an approximation to the solution of the trust region problem

$$\begin{aligned} &\underset{x \in \mathbf{R}^n}{\text{minimize}}\, m^{(k)}(x) \\ &\text{subject to the simple bounds } l \leq x \leq u \\ &\text{and the trust region constraint } \|x - x^{(k)}\| \leq \Delta^{(k)}, \end{aligned} \tag{2.2}$$

where $\|\cdot\|$ is a suitably chosen norm. The acceptance of the trial point as an improvement and the modification of $\Delta^{(k)}$ are treated in essentially the same way as in trust region methods for unconstrained problems (see Moré [18]). That is, we compute the ratio of the achieved to the predicted reduction of the objective function,

$$\rho^{(k)} = (f(x^{(k)}) - f(x^{(k)} + s^{(k)}))/(f(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)})), \tag{2.3}$$

and set

$$x^{(k+1)} = \begin{cases} x^{(k)} + s^{(k)} & \text{if } \rho^{(k)} > \mu, \\ x^{(k)} & \text{if } \rho^{(k)} \leq \mu, \end{cases}$$

and

$$\Delta^{(k+1)} = \begin{cases} \gamma_0 \Delta^{(k)} & \text{if } \rho^{(k)} \leq \mu, \\ \Delta^{(k)} & \text{if } \mu < \rho^{(k)} < \eta, \\ \gamma_2 \Delta^{(k)} & \text{if } \rho^{(k)} \geq \eta, \end{cases} \tag{2.4}$$

where $\gamma_0 < 1 < \gamma_2$, μ and η are appropriate numbers. It remains to describe our approximate solution of (2.2).

It is convenient to attempt to solve (2.2) in the case where we choose the infinity norm for the trust region constraint. For then the shape of the trust region is

aligned with the simple bounds (see, for example, Fletcher [9]). Thus, we replace the constraints in (2.2) by the "box" constraints

$$\max(l_i, x_i^{(k)} - \Delta^{(k)}) \equiv l_i^{(k)} \leq x_i \leq u_i^{(k)} \equiv \min(u_i, x_i^{(k)} + \Delta^{(k)}) \tag{2.5}$$

for $i = 1, \ldots, n$. In order to satisfy our global convergence theory, we need only find a feasible point within the trust region at which the value of the model function is no larger than its value at the Generalized Cauchy Point (GCP) (see Conn, Gould and Toint [3]). The GCP is defined as the first local minimizer of the univariate function

$$q^{(k)}(t) \equiv m^{(k)}(P[x^{(k)} - tg^{(k)}, l^{(k)}, u^{(k)}]). \tag{2.6}$$

That is, the first local minimizer of the model function, along the piecewise linear arc defined by projecting the steepest descent direction onto the region (2.5). We note that this definition is slightly different from that given by Conn, Gould and Toint [3], where the Generalized Cauchy Point is defined to be the first local minimizer of

$$m^{(k)}(P[x^{(k)} - tg^{(k)}, l, u])$$

within the trust region. The old definition forces the line minimization to cease at the first point at which the trust region boundary is encountered; in our new definition, further progress is possible along the boundary of the trust region provided that the quadratic model continues to decrease. Our existing theory covers the new definition, as the new GCP gives a lower model function value than the old GCP and the active set at the new point is no smaller than at the old point. The calculation of the GCP may be performed extremely efficiently as we shall shortly show. We note that the combinatorial part of our algorithm (the determination of which variables are to be fixed at one of their bounds during the current iteration) takes place in finding the GCP. This computation allows us to add and drop many bounds from the active set during a single iteration, if this is desirable. As we have already remarked, this is important for large-scale problems.

In order to induce a fast asymptotic rate of convergence of the method, we normally require a better approximation to the minimizer of (2.2) than the GCP. Consequently, we suppose that the GCP is $x^{(k)c}$ and that the active set with respect to $l^{(k)}$ and $u^{(k)}$ at $x^{(k)c}$ is $I(x^{(k)c}, l^{(k)}, u^{(k)})$. We now apply the conjugate gradient algorithm, starting from $x = x^{(k)c}$, to the problem

$$\underset{x \in \mathbf{R}^n}{\text{minimize}}\, m^{(k)}(x), \tag{2.7}$$

with the restriction that *the variables in the set* $I(x^{(k)c}, l^{(k)}, u^{(k)})$ *remain fixed throughout the process.*

Under a strict complementary slackness assumption, the strategy described above is sufficient to ensure that the set of constraints active at the solution (the *correct* active set) is identified after a finite number of iterations (see Conn, Gould and Toint [3]). The convergence of the algorithm can thereafter be analyzed as that of a purely unconstrained method. The conjugate gradient algorithm is terminated at the point $\overline{x}$ if

(i) the norm of the restricted gradient of the model function, that is the vector whose components are those elements of the gradient of $m^{(k)}(x)$ at $x = \overline{x}$

whose indices are not contained in $I(x^{(k)c}, l^{(k)}, u^{(k)})$, is less than $\eta^{(k)}$ for some $\eta^{(k)}$;

(ii) one or more of the unrestricted variables violates one of the bounds (2.5) ($\bar{x}$ is then the point at which the offending bound(s) is encountered); or

(iii) an excessive number of iterations has been taken.

In all cases, the final point reached is the new trial point, i.e., $\bar{x}^{(k+1)} = \bar{x}$. The rule (i) has already been considered by Toint [29] and Steihaug [27] for unconstrained minimization algorithms and is known to be useful for large-scale computations. Rule (ii) has three purposes. The first is to provide a natural stopping rule in the event that nonpositive curvature is encountered for the model. The second is to maintain feasibility of the iterates. The last is to avoid the expense of restarting the conjugate gradient iteration when a bound is encountered.

Once the correct active set has been determined, the asymptotic rate of convergence of the method will be controlled by the accuracy to which we attempt to solve (2.7). A superlinear rate of convergence can be assured provided that (a) the ratio of the norm of the restricted gradient at the final point to that at $x^{(k)}$ tends to zero as the iterates approach a Kuhn-Tucker point for the problem and (b) the matrices $B^{(k)}$ approach the true second derivative matrix at the solution in the direction that $x^{(k)}$ approaches the solution (see Dembo, Eisenstat and Steihaug [6], Dennis and Schnabel, [8]). A suitable choice for $\eta^{(k)}$ in order to satisfy (a) is given by

$$\eta^{(k)} = \min\left(0.1, \sqrt{||\bar{g}^{(k)}||}\right)||\bar{g}^{(k)}||, \tag{2.8}$$

where $\bar{g}^{(k)}$ is the projected gradient, $P[x^{(k)} - g^{(k)}] - x^{(k)}$, at $x^{(k)}$. The property (b) is certainly true if exact second derivatives are computed; the situation for secant updating formulae is closely related to that given by Steihaug [27]. In this case we must be careful that our updates obey the condition

$$\sum_{k=0}^{\infty} 1/\left(1 + \max_{0 \le i \le k} ||B^{(i)}||\right) = \infty \tag{2.9}$$

required in our global convergence theory. Such a condition is satisfied if, for instance,

$$||B^{(k)}|| \le c_1 + kc_2 \tag{2.10}$$

for some positive constants c_1 and c_2.

3. The Generalized Cauchy Point. In order to find the Generalized Cauchy Point, we need to be able to determine the first local minimizer of a quadratic function along a piecewise linear arc. In this section we describe an efficient algorithm for this calculation.

We shall consider a piecewise linear arc of the form

$$x(t) = x^0 + d(t), \quad \text{where } d(t)_i = \begin{cases} td_i & \text{if } t \le t_{j_i}, \\ t_{j_i} d_i & \text{otherwise,} \end{cases} \tag{3.1}$$

where the breakpoints $0 = t_0 < t_1 < \cdots < t_m$ and the indices $0 \le j_i \le m$ are well defined and may be calculated as required. We shall adopt the convention that

$j_i = 0$ if $d_i = 0$ and note that $d(t)$ is independent of t for all $t > t_m$. Notice that the arc

$$P[x^{(k)} - tg^{(k)}, l^{(k)}, u^{(k)}]$$

is exactly of this form, a breakpoint occurring whenever a variable reaches one of its bounds.

Our method for finding the GCP is simply to consider the intervals $[t_j, t_{j+1}]$ in order of increasing j until the one containing the GCP is located. We need only calculate t_{j+1} when the first j intervals have been examined and rejected. In order to calculate the GCP, we shall be concerned with the behavior of the quadratic function

$$m(x) = f + (x - x^0)^T g + \tfrac{1}{2}(x - x^0)^T B(x - x^0) \tag{3.2}$$

for points lying on (3.1). In particular, we shall be concerned with its behavior between adjacent pairs of breakpoints. If we define $x^j = x(t_j)$, we can express (3.1) in the interval $[t_j, t_{j+1}]$ as

$$x(t) = x^j + \Delta t\, d^j, \quad \text{where } d_i^j = \begin{cases} d_i & \text{if } j > j_i, \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

and $\Delta t = t - t_j$. Hence, defining

$$f_j = f + g^T z^j + \tfrac{1}{2} z^{jT} B z^j, \quad f_j' = g^T d^j + d^{jT} B z^j \quad \text{and} \quad f_j'' = d^{jT} B d^j,$$

where $z^j = x^j - x^0$, and combining (3.2) and (3.3), we obtain

$$m(x(t)) = f_j + \Delta t f_j' + \tfrac{1}{2}\Delta t^2 f_j''$$

for all $t = t_j + \Delta t$ in the interval $[t_j, t_{j+1}]$. It is then straightforward to deduce that the GCP lies at $t_j - f_j'/f_j''$ provided that this point lies in the interval (t_j, t_{j+1}) and that $f_j'' > 0$, lies at t_j if $f_j' \geq 0$ and lies at or beyond t_{j+1} in all other cases. We need now only be concerned with calculating f_j' and f_j''.

Let $J_j = \{i \text{ such that } j_i = j\}$. Then

$$d^j = d^{j-1} - \sum_{i \in J_j} d_i e^i,$$

where e^i is the ith column of the identity matrix. We may then obtain the identities

$$f_j' = f_{j-1}' + \Delta t_{j-1} f_{j-1}'' - b^{jT} x^j - \sum_{i \in J_j} d_i g_i^0$$

and

$$f_j'' = f_{j-1}'' + b^{jT}\left(\sum_{i \in J_j} d_i e^i - 2d^{j-1}\right), \tag{3.4}$$

where $g^0 = g - Bx^0$, $\Delta t_{j-1} = t_j - t_{j-1}$ and

$$b^j = B\left(\sum_{i \in J_j} d_i e^i\right) = \sum_{i \in J_j} d_i (Be^i). \tag{3.5}$$

Notice that the recurrences (3.4) require a single matrix-vector product (3.5) and two vector inner products. Moreover, the matrix-vector product only involves columns of B indexed by J_j; as it is usual for $|J_j|$ to be small (typically 1), the

product can normally be formed very efficiently. In the worst case, when we have to run through all of the breakpoints to find the GCP, each column of B will be accessed once. The recurrences are initialized with the values

$$f_0' = g^T d^0 \quad \text{and} \quad f_0'' = d^{0T} B d^0.$$

4. The Specific Algorithm. We are now in a position to specify our algorithm in almost its entirety. The only part that we do not give here is how the second derivative approximations $B^{(k)}$ are formed and updated. Details of these calculations are given in Section 5.

STEP 0 [Initialization]

The feasible starting point $x^{(0)}$, the function value $f(x^{(0)})$ and the gradient value $g^{(0)}$ are given, as well as an initial trust region radius, $\Delta^{(0)}$, and $B^{(0)}$, an initial symmetric approximation to the Hessian matrix of $f(x)$ at the starting point. The positive constants $\gamma_0 < 1 < \gamma_2, c_1, c_2, \mu, \eta$, and ε are specified. Set $k = 0$.

STEP 1 [Test for convergence]

Compute the projected gradient $\overline{g}^{(k)} = P[x^{(k)} - g^{(k)}] - x^{(k)}$. If $||\overline{g}^{(k)}|| < \varepsilon$, STOP.

STEP 2 [Find the GCP]

Calculate the bounds $l^{(k)}$ and $u^{(k)}$ from Eq. (2.5). Find the Generalized Cauchy Point, $x^{(k)c}$ (using the algorithm outlined in Section 3) as follows:

STEP 2.0 [Initialization]

Set

$$\begin{aligned} x &= x^{(k)}, \\ g &= g^{(k)} - B^{(k)} x^{(k)}, \\ d &= \lim_{t \to 0^+} P[x^{(k)} - t g^{(k)}, l, u] - x^{(k)}, \\ f' &= g^{(k)T} d \\ \text{and} \quad f'' &= d^T B^{(k)} d. \end{aligned}$$

If $f' \geq 0$, go to STEP 2.4.

STEP 2.1. [Find the next breakpoint]

Find the largest stepsize Δt for which $l^{(k)} \leq x + \Delta t\, d \leq u^{(k)}$ and the index set J of the indices of all the variables which first encounter their bounds at $x + \Delta t\, d$.

STEP 2.2 [Test whether the GCP has been found]

If $f'' > 0$ and $0 < -(f'/f'') < \Delta t$, reset $x := x - (f'/f'')d$ and go to STEP 2.4.

STEP 2.3. [Update line derivatives]

Compute $b = B^{(k)}(\sum_{i \in J} d_i e^i)$ and reset

$$\begin{aligned} x &:= x + \Delta t\; d \quad \text{and} \\ f' &:= f' + \Delta t f'' - b^T x - \sum_{i \in J} d_i g_i, \\ f'' &:= f'' + b^T \left(\sum_{i \in J} d_i e^i - 2d \right), \\ d_i &:= 0 \text{ for all } i \in J. \end{aligned}$$

If $f' \geq 0$, go to STEP 2.4. Otherwise, go to STEP 2.1.

STEP 2.4 [Termination with GCP]

Set $x^{(k)C} = x$.

STEP 3 [Find the new iterate]

Compute the active set $I(x^{(k)C}, l^{(k)}, u^{(k)})$.

Apply the following variant of the conjugate gradient algorithm to find an approximation $\overline{x}^{(k)}$ to the minimizer of the model function (2.1) over the feasible region (2.2) with the additional restriction that the variables with indices in $I(x^{(k)C}, l^{(k)}, u^{(k)})$ remain fixed at the corresponding values of $x^{(k)C}$:

STEP 3.0 [Initialization]

Set $x = x^{(k)C}$ and $r = -g^{(k)} - B^{(k)}(x - x^{(k)})$. Compute $\eta^{(k)}$ from Eq. (2.8). Set

$$\hat{I} = \{1, 2, \ldots, n\} \backslash I(x^{(k)C}, l^{(k)}, u^{(k)}).$$

Let $\hat{x}, \hat{r}, \hat{l}$ and $\hat{u}$ denote the vectors whose components are those of $x, r, l^{(k)}$ and $u^{(k)}$ indexed by $\hat{I}$. Furthermore, let $\hat{B}$ denote the matrix whose rows and columns are those of $B^{(k)}$ indexed by $\hat{I}$. Set $\hat{p} = 0, \rho_1 = 1$ and $\rho_2 = \hat{r}^T \hat{r}$.

STEP 3.1 [Test for the required accuracy in c.g. iteration]

If $\rho_2 < (\eta^{(k)})^2$, go to STEP 3.3.

STEP 3.2 [Conjugate gradient recurrences]

Set $\beta = \rho_2 / \rho_1$ and reset $\hat{p} := \hat{r} + \beta \hat{p}$. Compute $\hat{y} = \hat{B}\hat{p}$ and find α_1, the largest value of α for which $\hat{l} \leq \hat{x} + \alpha \hat{p} \leq \hat{u}$.

If $\hat{p}^T \hat{y} \leq 0$, reset $\hat{x} := \hat{x} + \alpha_1 \hat{p}$ and go to STEP 3.3.

Otherwise, calculate $\alpha_2 = \rho_2 / \hat{p}^T \hat{y}$.

If $\alpha_2 > \alpha_1$, reset $\hat{x} := \hat{x} + \alpha_1 \hat{p}$ and go to STEP 3.3.

Otherwise, reset

$$\begin{aligned} \hat{x} &:= \hat{x} + \alpha_2 \hat{p}, \\ \hat{r} &:= \hat{r} - \alpha_2 \hat{y}, \\ \rho_1 &:= \rho_2 \\ \text{and} \quad \rho_2 &:= \hat{r}^T \hat{r}. \end{aligned}$$

Return to STEP 3.1.

STEP 3.3 [Termination of conjugate gradient iteration]

Reset the components of x indexed by $\hat{I}$ to the values of the associated components of $\hat{x}$. Set $\overline{x}^{(k+1)} = x$.

STEP 4 [Compute the ratio of achieved to predicted reduction in the function]

Compute $f(\overline{x}^{(k+1)})$ and

$$\rho^{(k)} = (f(x^{(k)}) - f(\overline{x}^{(k+1)})) / (m^{(k)}(x^{(k)}) - f(\overline{x}^{(k+1)})).$$

STEP 5 [Updates]

Set

$$x^{(k+1)} = \begin{cases} \overline{x}^{(k+1)} & \text{if } \rho^{(k)} > \mu, \\ x^{(k)} & \text{if } \rho^{(k)} \leq \mu, \end{cases}$$

set $\Delta^{(k+1)}$ according to Eq. (2.4) and compute

$$g^{(k+1)} = \begin{cases} g(x^{(k+1)}) & \text{if } \rho^{(k)} > \mu, \\ g^{(k)} & \text{if } \rho^{(k)} \leq \mu. \end{cases}$$

Revise the approximation to the second derivative approximation $B^{(k+1)}$ while ensuring that Eq. (2.10) is maintained. Increment k by 1 and return to STEP 1.

5. Numerical Experiments. In order to investigate the behavior of the algorithm stated in Section 4, we have performed a substantial amount of numerical testing. We have attempted to solve forty six test problems using an assortment of methods, including existing library software for the problem and our algorithm. In testing our algorithm, we have allowed a variety of schemes for approximating second derivative information; we stress that this is the only way in which the alternative implementations of our algorithm differ from each other. The aim is naturally to indicate that our framework is an effective one for solving relatively small problems; our belief is that these tests give some indication as to how our methods could cope with larger problems.

We start with the various implementations of our algorithm. As has been stated, we have computed the second derivative matrices $B^{(k)}$ in a number of different ways. We refer the reader to Gill, Murray and Wright [14, Section 4.5.2.1] for a definition of the methods and terminology of this section.

First, we have tried using the exact second derivatives; this was intended to allow a comparison of other schemes with the ideal updating formula. Second, we have included the B.F.G.S.,

$$B^{(k+1)} = B^{(k)} + \frac{y^{(k)}y^{(k)T}}{y^{(k)T}s^{(k)}} - \frac{B^{(k)}s^{(k)}s^{(k)T}B^{(k)}}{s^{(k)T}B^{(k)}s^{(k)}},$$

and D.F.P.,

$$B^{(k+1)} = B^{(k)} + \frac{r^{(k)}y^{(k)T} + y^{(k)}r^{(k)T}}{y^{(k)T}s^{(k)}} - \frac{r^{(k)T}s^{(k)}y^{(k)}y^{(k)T}}{(y^{(k)T}s^{(k)})^2},$$

updating schemes. Here $s^{(k)}$ and $y^{(k)}$ are, respectively, the step $x^{(k+1)} - x^{(k)}$ and the change in gradient $g^{(k+1)} - g^{(k)}$, and $r^{(k)} \equiv y^{(k)} - B^{(k)}s^{(k)}$. In both cases, the update is only performed if the new approximation can be ensured to be positive definite. To be precise, the updates are only performed when the condition

$$y^{(k)T}s^{(k)}/y^{(k)T}y^{(k)} \geq 10^{-8} \tag{5.1}$$

is satisfied. We note that the B.F.G.S. (but not the D.F.P.) update has the property that the updates remain uniformly bounded (see Powell [23], Griewank and Toint [16]) for convex problems and hence automatically satisfy the growth requirement (2.10) in this case. Third, we have included the P.S.B. update,

$$B^{(k+1)} = B^{(k)} + \frac{r^{(k)}s^{(k)T} + s^{(k)}r^{(k)T}}{s^{(k)T}s^{(k)}} - \frac{r^{(k)T}s^{(k)}s^{(k)}s^{(k)T}}{(s^{(k)T}s^{(k)})^2},$$

as an example of a scheme that allows indefinite approximations to be generated and for which no effort needs be made to ensure that the growth requirement (2.10) is satisfied. Finally, we have tried the Symmetric Rank-one update,

$$B^{(k+1)} = B^{(k)} + \frac{r^{(k)}r^{(k)T}}{r^{(k)T}s^{(k)}},$$

as an example of an update which allows indefinite approximations but which must be controlled so that the growth requirement (2.10) is satisfied. We choose to skip

the update whenever the rank-one correction is too large. For simplicity, we have chosen to skip the update when the correction has norm larger than 10^8. In theory, this value, and the value 10^{-8} in the inequality (5.1), are arbitrary, but in practice it will almost certainly be better to chose the values to reflect problem scalings.

For each of these second-derivative schemes, we solved every test problem twice. For the first run, the problem is essentially unconstrained (U) but we have taken the precaution of including the simple bounds

$$-100.0 \leq x_i \leq 100.0, \qquad 1 \leq i \leq n,$$

to prevent an unbounded solution being found. For a few of the problems, typically those containing exponentials, we have provided tighter bounds in order to prevent numerical overflow when the problem functions are evaluated—notice that our framework is especially useful for imposing such safeguards. Details of these additional bounds are given in the Appendix. If x^* denotes the solution obtained for problem U, the second test of the problem (C) includes the additional bounds

$$x_i^* + 0.1 \leq x_i \leq x_i^* + 1.1 \quad \text{for all odd } i.$$

The starting point for the constrained problem is the projection onto the feasible region, $P[x^{(0)}]$, of the starting point for the unconstrained problem. Details of the test problems and their solutions are given in the appendix.

All computations were performed on the IBM 3084Q computer at Harwell; our code is written in Double Precision Fortran 77 (with modifications as required by WATFIV) and compiled using the WATFIV Fortran compiler. All timings reported are in seconds for time spent in the c.p.u. and appear to be correct to about one hundredth of a second. As the code is essentially a prototype for studying the effectiveness of the method, we did not feel it necessary to use a more sophisticated optimizing compiler; the timings quoted are supposed to allow comparisons of the relative merits of the schemes tested rather than trying to obtain the "best" timing for a particular scheme. The initial estimate $B^{(0)} = I$ was used for all of the updating schemes; the initial trust region radius was taken to be $\Delta^{(0)} = 0.1||\overline{g}^{(0)}||_2$. We also chose $\mu = 0.25$, $\eta = 0.75$, $\gamma_0 = 0.5$ and $\gamma_2 = 2$.

The results of the tests using our algorithm are given in Tables 5.1–5.5 and summarized in Tables 5.6–5.8. For each run we have given the number of variables (n), the number of iterations (it.) required to solve the problem (which is the same as the number of objective function evaluations), the total number of derivative evaluations (de.) (which is the same as the number of iterations for which the trial point is accepted), the total number of conjugate gradient iterations taken (c.g.), the norm of the projected gradient (gr. norm) at the final point, the last iteration on which the active set changes (cas) (which is an indication as to when the correct active set is obtained for successful runs) and the time taken to reach the final point (time). Additionally, for the B.F.G.S., D.F.P. and S.R.1 runs, we have recorded the number of times the second derivative update was skipped (usk). Every run was terminated if the norm of the projected gradient of the objective function was reduced below 10^{-6}. In addition, each unconstrained [constrained] run was terminated if more than $\max(20n, 600)$ [$\max(10n, 300)$] iterations were performed or if the trust region radius $\Delta^{(k)}$ was reduced below 10^{-16}, and these

failures are indicated by > in the iteration column. In Table 5.6, the performance of the updating schemes is compared with that obtained by the method which uses exact second derivative information. The relative number of iterations and the relative timings are given and indicate how much worse the respective updating schemes are than the use of exact second derivatives. In Tables 5.7 and 5.8, the methods are ranked according to the numbers of iterations and the c.p.u. times taken for each test problem. For each problem, the five methods are ranked from 1 (fewest iterations or shortest time) to 5 (most iterations or longest time), with failures being recorded separately. The table entries give the sums of these rankings over all the test problems. For instance, the method using the D.F.P. update ranks third for 19 of the test problems when the ranking is based upon the number of iterations taken.

As a comparison, we include results obtained on the same set of test problems when existing library software for solving (1.1) is used. The codes in question appear in the NAG subroutine library at mark 11 (1984) and are based upon earlier codes developed for the NPL optimization library (Gill and Murray [12]). The NAG algorithms are line-search and active-set based (see, for example, Gill, Murray and Wright [14]); we use routines E04LAF, a modified Newton method, and E04KAF, a quasi-Newton method based upon the B.F.G.S. update. Both routines perform derivative line searches and choose "tuning" parameters by default. The termination of the methods is controlled by specifying the accuracy required in the approximation to the solution x^*; we requested that x^* be computed correctly to five decimal places. The codes are written in Fortran 77, precompiled, and the tests were, once again, performed on the Harwell IBM 3084Q. The result of this testing is summarized in Table 5.9. For each problem and both algorithms we give the number of iterations required to solve the problem (it.), the number of function/gradient evaluations made (a function/gradient pair is required at every trial point by the line-search routine) (fde.) and the norm of the projected gradient (gr. norm) at the final point encountered. Quantitatively similar results for the unconstrained problems were also obtained from the Harwell subroutine VA13AD, a B.F.G.S. based line-search method, but are not reported here.

We now consider the conclusions that can be drawn from our tests. We first compare the various versions of our algorithm and then consider our algorithm in comparison with the NAG routines.

As one might expect, the use of exact second derivatives gives uniformly and significantly better results than the other approximating schemes—both in terms of the number of iterations and in terms of the overall timings. Of course, none of the problems solved has particularly complicated derivatives, and the calculation of exact second derivatives is often cheaper than performing a rank-one or rank-two update to an existing approximation. Nonetheless, we believe that the use of second derivatives in our framework is worthwhile and we recommend their use whenever they are available. The worst performance occurs on the problem DEGENSING for which the complementary slackness condition does not hold at the solution. A closer examination of the run shows that the degenerate bounds repeatedly enter and leave the active set as the solution is approached (as is to be expected). The insistence on terminating the conjugate gradient (c.g.) iteration when a previously inactive

TABLE 5.1
Results for tests using exact second derivatives

Problem	n	it.	de.	c.g.	gr.norm	cas	time
GENROSE U	8	42	31	184	2.2D-07	0	0.88
GENROSE C	8	15	15	54	2.8D-10	6	0.27
CHAINROSE U	25	20	17	98	1.3D-07	0	1.80
CHAINROSE C	25	18	13	26	3.9D-07	14	0.91
DEGENROSE U	25	95	94	34	6.3D-07	95	5.64
DEGENROSE C	25	17	14	13	1.2D-10	12	0.78
GENSING U	20	10	11	37	5.6D-07	0	0.58
GENSING C	20	4	5	3	1.6D-07	0	0.15
CHAINSING U	20	18	19	96	4.6D-07	0	1.21
CHAINSING C	20	3	4	8	2.5D-15	0	0.13
DEGENSING U	20	155	156	56	3.0D-07	148	6.45
DEGENSING C	20	3	4	8	2.5D-15	1	0.13
GENWOOD U	8	107	75	393	2.6D-10	0	2.03
GENWOOD C	8	5	6	2	1.0D-07	0	0.07
CHAINWOOD U	8	77	54	365	2.4D-09	0	1.64
CHAINWOOD C	8	5	6	7	7.8D-07	0	0.07
HOSC45 U	10	19	20	0	0.0D-01	19	0.26
HOSC45 C	10	12	13	0	0.0D-01	12	0.18
BROYDEN1A U	30	11	12	38	6.9D-07	0	1.13
BROYDEN1A C	30	8	9	10	3.5D-10	0	0.54
BROYDEN1B U	30	7	8	31	6.3D-08	0	0.79
BROYDEN1B C	30	6	7	5	3.3D-07	0	0.39
BROYDEN2A U	30	14	15	7	5.6D-07	0	1.48
BROYDEN2A C	30	10	11	11	1.8D-07	3	1.05
BROYDEN2B U	30	9	10	12	7.0D-07	0	1.04
BROYDEN2B C	30	9	10	9	6.5D-08	2	0.93
TOINTBROY U	30	8	9	52	3.3D-07	0	1.10
TOINTBROY C	30	8	9	8	2.2D-10	0	0.55
TRIG U	10	7	6	7	1.6D-07	0	0.28
TRIG C	10	8	8	6	8.8D-15	0	0.32
TOINTTRIG U	10	13	9	26	9.6D-08	0	0.28
TOINTTRIG C	10	10	9	18	2.8D-13	3	0.19
CRAGGLEVY U	8	24	25	106	4.4D-07	0	0.50
CRAGGLEVY C	8	20	20	65	1.7D-07	19	0.36
PENALTY U	15	27	25	58	2.0D-08	0	0.88
PENALTY C	15	80	81	31	1.2D-08	78	2.12
AUGMLAGN U	15	31	21	103	8.1D-07	8	1.15
AUGMLAGN C	15	47	37	186	5.2D-08	44	1.77
BROWN1 U	20	27	28	32	9.4D-10	1	1.22
BROWN1 C	20	27	28	0	3.6D-10	1	0.93
BROWN3 U	20	7	8	4	4.5D-08	0	0.31
BROWN3 C	20	6	7	6	3.9D-07	0	0.24
BVP(N=10) U	10	4	5	31	1.7D-09	0	0.14
BVP(N=10) C	10	4	5	18	1.4D-07	3	0.10
BVP(N=20) U	20	5	6	77	6.4D-10	0	0.63
BVP(N=20) C	20	9	10	62	6.1D-07	8	0.67
VAR(N=20) U	20	6	7	112	2.7D-09	0	0.87
VAR(N=20) C	20	6	7	61	1.7D-09	3	0.53
VAR(N=45) U	45	6	7	234	1.7D-07	0	5.35
VAR(N=45) C	45	12	13	90	1.8D-09	9	3.10

TABLE 5.2
Results for tests using the B.F.G.S. update

Problem	n	it.	de.	usk	c.g.	gr.norm	cas	time
GENROSE U	8	149	98	0	376	5.9D-07	0	2.82
GENROSE C	8	84	54	0	178	3.0D-08	49	1.33
CHAINROSE U	25	190	129	0	376	6.7D-07	0	15.50
CHAINROSE C	25	65	43	0	97	2.2D-07	16	4.10
DEGENROSE U	25	240	210	0	65	7.6D-07	240	17.39
DEGENROSE C	25	56	28	0	35	6.9D-07	22	2.75
GENSING U	20	125	77	0	503	3.8D-07	0	8.35
GENSING C	20	15	10	0	10	1.9D-07	12	0.58
CHAINSING U	20	165	109	0	512	6.3D-07	0	10.44
CHAINSING C	20	25	15	0	14	2.0D-08	0	0.93
DEGENSING U	20	572	545	0	203	4.4D-07	572	29.74
DEGENSING C	20	21	13	0	16	1.1D-07	7	0.77
GENWOOD U	8	201	136	0	720	6.5D-08	0	3.99
GENWOOD C	8	50	23	0	27	1.2D-07	21	0.59
CHAINWOOD U	8	221	148	0	894	4.9D-07	0	4.67
CHAINWOOD C	8	47	25	0	40	2.4D-08	22	0.61
HOSC45 U	10	>600	601	600	0	2.3D-04	0	7.57
HOSC45 C	10	86	87	86	0	0.0D-01	86	1.07
BROYDEN1A U	30	65	44	0	218	8.4D-07	0	7.68
BROYDEN1A C	30	54	23	0	17	4.1D-07	13	3.70
BROYDEN1B U	30	97	62	0	44	5.4D-07	0	8.78
BROYDEN1B C	30	65	31	0	30	3.3D-07	0	4.46
BROYDEN2A U	30	71	36	0	183	8.8D-07	0	7.66
BROYDEN2A C	30	65	29	0	18	1.9D-07	18	5.15
BROYDEN2B U	30	69	34	0	14	2.3D-07	0	6.29
BROYDEN2B C	30	67	29	0	25	4.2D-07	12	5.06
TOINTBROY U	30	122	80	0	76	7.5D-07	0	11.54
TOINTBROY C	30	68	34	0	29	3.8D-07	10	4.89
TRIG U	10	45	21	0	46	5.4D-07	0	1.09
TRIG C	10	17	11	0	13	3.3D-07	0	0.37
TOINTTRIG U	10	73	37	0	65	9.3D-07	0	1.50
TOINTTRIG C	10	57	25	0	51	1.3D-07	7	0.86
CRAGGLEVY U	8	99	83	0	574	6.3D-07	0	2.38
CRAGGLEVY C	8	63	56	0	151	7.6D-07	56	1.10
PENALTY U	15	139	104	2	537	7.0D-07	0	6.28
PENALTY C	15	74	59	0	156	1.7D-07	61	2.63
AUGMLAGN U	15	156	112	1	355	4.5D-07	41	5.83
AUGMLAGN C	15	139	98	0	306	9.7D-07	131	5.03
BROWN1 U	20	94	54	3	159	2.9D-07	8	4.68
BROWN1 C	20	33	29	3	0	9.0D-07	6	1.21
BROWN3 U	20	18	15	0	28	5.1D-07	0	0.94
BROWN3 C	20	9	9	0	13	7.3D-07	0	0.41
BVP(N=10) U	10	31	27	0	109	6.1D-07	0	0.80
BVP(N=10) C	10	33	23	0	82	2.4D-08	20	0.70
BVP(N=20) U	20	69	56	0	438	6.1D-07	0	5.67
BVP(N=20) C	20	61	47	0	277	1.8D-07	37	4.09
VAR(N=20) U	20	134	97	0	390	1.7D-07	0	8.35
VAR(N=20) C	20	128	86	0	267	1.9D-07	88	7.09
VAR(N=45) U	45	366	259	0	1252	5.9D-07	0	85.94
VAR(N=45) C	45	337	234	0	772	8.3D-07	279	69.54

TABLE 5.3
Results for tests using the D.F.P. update

Problem	n	it.	de.	usk	c.g.	gr.norm	cas	time
GENROSE U	8	>600	589	0	1154	4.7D 00	0	10.96
GENROSE C	8	39	31	0	99	5.4D-07	18	0.67
CHAINROSE U	25	95	70	0	410	7.1D-07	0	9.39
CHAINROSE C	25	45	30	0	47	4.0D-07	11	2.80
DEGENROSE U	25	183	164	0	202	9.4D-07	183	14.18
DEGENROSE C	25	47	20	0	43	2.6D-07	10	2.27
GENSING U	20	76	61	0	338	9.4D-07	0	5.30
GENSING C	20	14	10	0	8	3.0D-07	12	0.58
CHAINSING U	20	253	230	0	1635	4.4D-07	0	21.04
CHAINSING C	20	17	12	0	13	9.6D-07	0	0.69
DEGENSING U	20	>600	594	0	9	8.0D-03	601	32.08
DEGENSING C	20	19	12	0	13	1.7D-09	7	0.71
GENWOOD U	8	>600	587	92	1824	1.9D 00	0	11.46
GENWOOD C	8	28	16	0	10	1.7D-08	16	0.34
CHAINWOOD U	8	>600	590	0	1904	1.4D-01	0	12.10
CHAINWOOD C	8	30	18	0	24	1.7D-08	24	0.42
HOSC45 U	10	>600	601	600	0	2.3D-04	0	7.55
HOSC45 C	10	86	87	86	0	0.0D-01	86	1.07
BROYDEN1A U	30	96	69	0	819	9.5D-07	0	16.51
BROYDEN1A C	30	59	24	0	52	1.8D-07	13	4.11
BROYDEN1B U	30	65	36	0	62	6.0D-07	0	5.91
BROYDEN1B C	30	52	20	0	44	3.4D-07	0	3.49
BROYDEN2A U	30	83	51	0	432	9.3D-07	0	11.17
BROYDEN2A C	30	69	32	0	124	9.3D-08	14	5.69
BROYDEN2B U	30	66	26	0	69	4.9D-07	0	5.75
BROYDEN2B C	30	65	29	0	95	9.7D-07	12	5.07
TOINTBROY U	30	77	44	0	108	9.9D-07	0	7.41
TOINTBROY C	30	56	26	0	53	7.7D-07	10	4.03
TRIG U	10	28	16	0	20	6.1D-07	0	0.71
TRIG C	10	15	10	0	10	1.2D-07	3	0.34
TOINTTRIG U	10	41	24	0	71	1.7D-07	0	0.91
TOINTTRIG C	10	26	15	0	30	4.0D-07	7	0.44
CRAGGLEVY U	8	>600	592	0	2387	1.1D-04	0	12.70
CRAGGLEVY C	8	193	189	0	237	9.3D-07	170	3.06
PENALTY U	15	>600	572	1	2971	8.6D-01	0	30.19
PENALTY C	15	157	142	0	588	8.3D-07	63	5.78
AUGMLAGN U	15	>600	586	0	1346	3.6D-01	395	24.73
AUGMLAGN C	15	>300	291	0	998	2.5D-01	297	12.12
BROWN1 U	20	>600	565	3	2717	1.3D-04	8	41.86
BROWN1 C	20	33	29	3	0	9.0D-07	6	1.19
BROWN3 U	20	15	12	0	18	8.1D-07	0	0.75
BROWN3 C	20	9	9	0	11	7.0D-07	0	0.41
BVP(N=10) U	10	47	45	0	252	3.6D-08	0	1.41
BVP(N=10) C	10	25	22	0	83	1.4D-07	12	0.59
BVP(N=20) U	20	269	264	0	2645	8.5D-07	0	27.07
BVP(N=20) C	20	107	97	0	734	2.7D-07	30	8.46
VAR(N=20) U	20	56	43	0	226	1.6D-07	0	3.82
VAR(N=20) C	20	54	40	0	185	8.6D-08	21	3.40
VAR(N=45) U	45	130	99	0	695	1.5D-07	0	35.54
VAR(N=45) C	45	153	127	0	600	9.3D-07	136	36.88

TABLE 5.4
Results for tests using the P.S.B. update

Problem	n	it.	de.	c.g.	gr.norm	cas	time
GENROSE U	8	177	125	581	2.2D–07	0	3.62
GENROSE C	8	96	67	196	2.0D–07	42	1.52
CHAINROSE U	25	137	96	323	8.1D–07	0	12.21
CHAINROSE C	25	52	36	75	3.9D–07	12	3.17
DEGENROSE U	25	233	212	23	6.4D–07	233	17.08
DEGENROSE C	25	42	20	39	8.1D–08	10	2.01
GENSING U	20	171	132	387	8.4D–07	0	10.21
GENSING C	20	14	9	7	8.7D–07	9	0.51
CHAINSING U	20	217	173	1745	5.1D–07	0	19.91
CHAINSING C	20	24	14	21	5.2D–07	0	0.88
DEGENSING U	20	>600	559	547	5.4D–05	601	31.50
DEGENSING C	20	19	12	12	2.4D–09	7	0.68
GENWOOD U	8	296	230	1143	5.7D–07	0	6.19
GENWOOD C	8	32	15	19	2.8D–08	13	0.36
CHAINWOOD U	8	279	204	1099	9.2D–07	0	5.97
CHAINWOOD C	8	31	20	21	5.4D–09	23	0.44
HOSC45 U	10	29	30	0	0.0D–01	29	0.50
HOSC45 C	10	15	16	4	0.0D–01	15	0.28
BROYDEN1A U	30	172	143	1905	9.9D–07	0	34.55
BROYDEN1A C	30	60	28	58	5.7D–07	13	4.10
BROYDEN1B U	30	87	50	96	8.8D–07	0	8.23
BROYDEN1B C	30	57	23	42	1.1D–07	0	3.67
BROYDEN2A U	30	111	68	595	9.8D–07	0	15.22
BROYDEN2A C	30	69	37	60	7.4D–07	18	5.44
BROYDEN2B U	30	68	28	48	3.4D–07	0	5.88
BROYDEN2B C	30	62	26	69	7.1D–07	16	4.68
TOINTBROY U	30	99	59	112	5.9D–07	0	9.72
TOINTBROY C	30	54	23	28	9.7D–07	14	3.74
TRIG U	10	31	19	27	3.3D–07	0	0.79
TRIG C	10	16	10	11	9.6D–08	3	0.35
TOINTTRIG U	10	35	23	45	6.0D–08	0	0.78
TOINTTRIG C	10	22	14	21	3.3D–07	7	0.38
CRAGGLEVY U	8	454	422	2967	9.3D–07	0	11.82
CRAGGLEVY C	8	250	240	651	1.0D–06	186	4.14
PENALTY U	15	566	514	2886	9.6D–07	0	28.96
PENALTY C	15	>300	261	159	1.1D 00	109	8.96
AUGMLAGN U	15	549	493	4498	1.0D–06	24	33.26
AUGMLAGN C	15	127	103	315	8.1D–08	116	4.77
BROWN1 U	20	>600	539	7570	8.6D–04	528	69.40
BROWN1 C	20	41	37	0	5.3D–07	6	1.47
BROWN3 U	20	21	13	29	3.4D–08	0	0.99
BROWN3 C	20	9	9	13	3.8D–07	0	0.40
BVP(N=10) U	10	34	26	114	6.5D–07	0	0.86
BVP(N=10) C	10	38	27	92	4.9D–07	19	0.81
BVP(N=20) U	20	113	82	983	5.6D–07	0	10.62
BVP(N=20) C	20	131	88	867	1.6D–07	17	9.99
VAR(N=20) U	20	119	85	470	5.5D–07	0	8.15
VAR(N=20) C	20	102	72	353	9.3D–07	74	6.40
VAR(N=45) U	45	271	194	1390	7.7D–07	0	77.81
VAR(N=45) C	45	200	157	619	1.1D–07	168	47.60

TABLE 5.5

Results for tests using the symmetric rank-one update

Problem	n	it.	de.	usk	c.g.	gr.norm	cas	time
GENROSE U	8	195	115	0	467	4.8D-07	0	3.39
GENROSE C	8	70	37	0	118	8.8D-09	32	0.96
CHAINROSE U	25	140	77	0	220	1.6D-07	0	10.47
CHAINROSE C	25	39	23	0	55	4.0D-07	11	2.24
DEGENROSE U	25	152	120	0	58	9.2D-08	152	10.17
DEGENROSE C	25	34	16	0	22	1.0D-08	10	1.59
GENSING U	20	74	46	0	247	6.0D-07	0	4.49
GENSING C	20	12	8	0	7	2.3D-10	0	0.43
CHAINSING U	20	83	58	0	445	8.7D-07	0	6.09
CHAINSING C	20	14	9	0	9	7.1D-10	0	0.50
DEGENSING U	20	>600	570	0	539	9.4D-04	77	28.69
DEGENSING C	20	14	9	0	8	5.1D-09	7	0.48
GENWOOD U	8	486	307	0	814	6.8D-07	0	7.72
GENWOOD C	8	32	15	0	4	1.0D-07	28	0.37
CHAINWOOD U	8	411	261	0	1148	1.1D-10	0	7.57
CHAINWOOD C	8	23	15	0	15	1.4D-09	16	0.29
HOSC45 U	10	28	29	0	0	0.0D-01	28	0.45
HOSC45 C	10	14	15	0	2	0.0D-01	14	0.25
BROYDEN1A U	30	129	76	0	704	9.5D-07	0	17.46
BROYDEN1A C	30	54	25	0	46	4.2D-08	13	3.63
BROYDEN1B U	30	81	40	0	78	2.2D-07	0	7.33
BROYDEN1B C	30	45	19	0	45	3.0D-07	0	2.82
BROYDEN2A U	30	95	52	0	284	4.0D-07	0	10.47
BROYDEN2A C	30	63	30	0	44	5.3D-07	25	5.04
BROYDEN2B U	30	82	38	0	71	1.5D-07	0	7.68
BROYDEN2B C	30	57	28	0	45	9.3D-07	12	4.35
TOINTBROY U	30	62	36	0	119	4.2D-07	0	6.10
TOINTBROY C	30	44	22	0	35	6.2D-07	19	3.25
TRIG U	10	22	13	0	25	1.6D-08	0	0.55
TRIG C	10	13	9	0	9	3.3D-09	0	0.28
TOINTTRIG U	10	27	17	0	30	5.8D-07	0	0.57
TOINTTRIG C	10	20	12	0	19	1.9D-10	7	0.33
CRAGGLEVY U	8	142	92	0	398	2.6D-07	0	2.54
CRAGGLEVY C	8	56	48	0	182	3.2D-07	49	1.02
PENALTY U	15	163	105	0	742	5.2D-08	0	7.45
PENALTY C	15	91	71	0	126	4.9D-07	81	2.81
AUGMLAGN U	15	125	92	0	313	4.4D-07	37	4.47
AUGMLAGN C	15	97	72	0	240	2.1D-07	90	3.39
BROWN1 U	20	110	88	4	164	3.4D-07	8	5.41
BROWN1 C	20	33	29	3	0	9.0D-07	6	1.08
BROWN3 U	20	12	11	0	10	4.4D-09	0	0.54
BROWN3 C	20	9	9	0	12	1.8D-07	0	0.37
BVP(N=10) U	10	27	19	0	42	2.1D-08	0	0.54
BVP(N=10) C	10	18	13	0	45	4.9D-08	12	0.36
BVP(N=20) U	20	29	25	0	143	2.5D-07	0	2.01
BVP(N=20) C	20	35	26	0	172	2.6D-07	24	2.26
VAR(N=20) U	20	41	32	0	146	2.8D-08	0	2.52
VAR(N=20) C	20	35	25	0	92	9.4D-08	19	1.92
VAR(N=45) U	45	78	59	0	457	1.6D-08	0	22.21
VAR(N=45) C	45	85	67	0	245	9.2D-08	80	17.98

TABLE 5.6

A comparison of the performance of the methods relative to the method using exact second derivatives

	it.				time			
Problem	BFGS	DFP	PSB	SR1	BFGS	DFP	PSB	SR1
GENROSE U	3.55	>14.29	4.21	4.64	3.14	>12.12	4.01	3.75
GENROSE C	5.60	2.60	6.40	4.67	4.87	2.46	5.55	3.52
CHAINROSE U	9.50	4.75	6.85	7.00	8.49	5.15	6.69	5.74
CHAINROSE C	3.61	2.50	2.89	2.17	4.49	3.06	3.47	2.45
DEGENROSE U	2.53	1.93	2.45	1.60	3.08	2.51	3.02	1.80
DEGENROSE C	3.29	2.76	2.47	2.00	3.51	2.90	2.57	2.04
GENSING U	12.50	7.60	17.10	7.40	13.78	8.75	16.83	7.42
GENSING C	3.75	3.50	3.50	3.00	3.82	3.81	3.34	2.80
CHAINSING U	9.17	14.06	12.06	4.61	8.48	17.08	16.16	4.95
CHAINSING C	8.33	5.67	8.00	4.67	6.63	4.92	6.32	3.59
DEGENSING U	3.69	>3.87	>3.87	>3.87	4.59	>4.95	>4.86	>4.43
DEGENSING C	7.00	6.33	6.33	4.67	5.52	5.06	4.84	3.45
GENWOOD U	1.88	> 5.61	2.77	4.54	1.95	> 5.59	3.02	3.76
GENWOOD C	10.00	5.60	6.40	6.40	8.78	5.04	5.37	5.54
CHAINWOOD U	2.87	> 7.79	3.62	5.34	2.83	> 7.31	3.61	4.57
CHAINWOOD C	9.40	6.00	6.20	4.60	8.30	5.71	5.99	4.03
HOSC45 U	>31.58	>31.58	1.53	1.47	>26.58	>26.52	1.79	1.62
HOSC45 C	7.17	7.17	1.25	1.17	5.83	5.87	1.57	1.36
BROYDEN1A U	5.91	8.73	15.64	11.73	6.62	14.22	29.74	15.04
BROYDEN1A C	6.75	7.38	7.50	6.75	6.74	7.49	7.48	6.62
BROYDEN1B U	13.86	9.29	12.43	11.57	10.70	7.21	10.02	8.93
BROYDEN1B C	10.83	8.67	9.50	7.50	11.06	8.66	9.12	7.00
BROYDEN2A U	5.07	5.93	7.93	6.79	5.08	7.40	10.08	6.94
BROYDEN2A C	6.50	6.90	6.90	6.30	4.84	5.35	5.12	4.74
BROYDEN2B U	7.67	7.33	7.56	9.11	5.92	5.41	5.53	7.22
BROYDEN2B C	7.44	7.22	6.89	6.33	5.40	5.42	5.00	4.65
TOINTBROY U	15.25	9.63	12.38	7.75	10.27	6.59	8.65	5.44
TOINTBROY C	8.50	7.00	6.75	5.50	8.68	7.16	6.64	5.78
TRIG U	6.43	4.00	4.43	3.14	3.65	2.38	2.64	1.86
TRIG C	2.13	1.88	2.00	1.63	1.15	1.07	1.08	0.88
TOINTTRIG U	5.62	3.15	2.69	2.08	5.09	3.10	2.67	1.97
TOINTTRIG C	5.70	2.60	2.20	2.00	4.54	2.32	1.99	1.75
CRAGGLEVY U	4.13	>25.00	18.92	5.92	4.59	>24.36	22.68	4.89
CRAGGLEVY C	3.15	9.65	12.50	2.80	3.03	8.41	11.38	2.80
PENALTY U	5.15	>22.22	20.96	6.04	6.94	>33.34	31.98	8.24
PENALTY C	0.92	1.96	> 3.75	1.14	1.24	2.72	> 4.21	1.32
AUGMLAGN U	5.03	>19.35	17.71	4.03	4.97	>21.04	28.28	3.81
AUGMLAGN C	2.96	> 6.38	2.70	2.06	2.83	> 6.82	2.68	1.91
BROWN1 U	3.48	>22.22	>22.22	4.07	3.79	>33.74	>55.94	4.37
BROWN1 C	1.22	1.22	1.52	1.22	1.30	1.28	1.58	1.16
BROWN3 U	2.57	2.14	3.00	1.71	2.89	2.32	3.05	1.68
BROWN3 C	1.50	1.50	1.50	1.50	1.71	1.70	1.63	1.54
BVP(N=10) U	7.75	11.75	8.50	6.75	4.98	8.70	5.30	3.39
BVP(N=10) C	8.25	6.25	9.50	4.50	6.53	5.50	7.50	3.41
BVP(N=20) U	13.80	53.80	22.60	5.80	8.64	41.16	16.16	3.07
BVP(N=20) C	6.78	11.89	14.56	3.89	6.10	12.61	14.89	3.37
VAR(N=20) U	22.33	9.33	19.83	6.83	9.38	4.30	9.16	2.84
VAR(N=20) C	21.33	9.00	17.00	5.83	13.13	6.30	11.86	3.57
VAR(N=45) U	61.00	21.67	45.17	13.00	16.06	6.70	13.94	3.98
VAR(N=45) C	28.08	12.75	16.67	7.08	22.83	11.95	14.70	5.56

TABLE 5.7
A ranking of the various update methods according to the number of iterations taken

Ranking	Exact	B.F.G.S.	D.F.P.	P.S.B.	S.R.1
1st	49	1	0	0	0
2nd	1	12	7	1	35
3rd	0	7	19	16	7
4th	0	7	11	20	6
5th	0	22	3	10	1
Failure	0	1	10	3	1

TABLE 5.8
A ranking of the various update methods according to the time taken

Ranking	Exact	B.F.G.S.	D.F.P.	P.S.B.	S.R.1
1st	49	0	0	0	1
2nd	1	10	5	0	34
3rd	0	8	17	15	9
4th	0	9	11	23	4
5th	0	22	7	9	1
Failure	0	1	10	3	1

variable hits a bound slows down convergence in the presence of degenerate bounds, as it forces small steps to be taken. As stated in Section 3, the reasoning behind this termination criterion for the c.g. iteration is that for non dual degenerate problems the correct active set will eventually be found at the GCP and the c.g. iteration will thereafter be unhindered by the inactive bounds. As the c.g. scheme would have to be restarted every time a new bound is encountered (in order to restore conjugacy of the c.g. search directions) and as this could involve a lot of additional computation for large-scale problems, our termination criteria seemed reasonable. However, for this particular problem, the inactive degenerate bounds were all encountered during the *first* c.g. iteration, and there would be little overhead incurred in restarting the iteration under these conditions. The convergence theory of Conn, Gould and Toint [3] allows us to increase the active set in the c.g. iteration provided that it

always contains those variables active in the GCP. In Table 5.10, we indicate the improvements possible if we allow restarting within the conjugate gradient iteration for the DEGENSING (U) problem.

Somewhat more surprisingly, the simplest of the updating schemes, the symmetric rank-one method, appears to perform the best compared with the other methods in our tests. The use of a trust region removes the main disadvantage of such methods in allowing a meaningful step to be taken even when the approximation is indefinite. For large problems, it is desirable to allow indefinite approximations, as the combination of symmetric secant updating, positive definiteness and preservation of the sparsity structure can lead to severe numerical difficulties (see Sorensen [26]). Of course, such a scheme must be used with care, but the restriction (2.10) seems to provide a useful stabilizing effect—we note, moreover, that the update was almost never skipped. The use of S.R.1 for larger problems is thus quite appealing in view of its relatively good overall performance and its simplicity. We suspect that the success of the method is due in part to our observation that, in contrast to the other updating schemes, the second derivative approximations with respect to the inactive variables at the solution converged (or were very close) to their true values on almost every problem tested. Such a result has been theoretically established, under very mild conditions, for quadratic functions (see, e.g. Fletcher [9, p. 41]) and would appear to be true in general. It is noticeable that three of the poorest results, relative to the exact second derivative method, are on the problems BROYDEN1A (U), BROYDEN2A (U) and PENALTY (U) for which the second derivative approximations obtained were observed to be less accurate than normal. We believe that this is important for trust region methods, which are essentially based on being able to model the true function as accurately as possible in a larger subspace than just that given by the Newton direction; accuracy in the latter subspace is all that is really required for success in line-search algorithms. We have not tried to prove our conjecture here; we believe that such a result is likely to be rather difficult to establish and is a challenging and important open question. The fact that the minimization (2.7) is carried out inexactly may also be important in the appreciation of the performance of S.R.1.

We should mention here that Brayton and Cullum [2] have suggested the use of the rank-one formula in the context of a line-search algorithm for solving (1.1); they make use of a number of ingenious techniques for avoiding numerical difficulties in their approach [5] and provide some evidence that their method is effective. Exactly how their method compares with other line-search based techniques, such as the NAG algorithms considered here, is not reported but would be of some interest.

All of the updating methods failed on at least one problem. The method based upon the S.R.1 update failed on DEGENSING (U), but as we have already seen, this is caused by (dual) degeneracy of the solution and can be overcome (see Table 5.10). Of the three other methods, the B.F.G.S. update appears to be the most reliable—indeed the only failure for this problem, on HOSC45, is attributable to our insistence on maintaining a positive definite approximation to a matrix which is uniformly indefinite. As has been observed in the past, the B.F.G.S. method is more robust than the D.F.P. Although B.F.G.S. performs well on some problems, the results are surprisingly disappointing in comparison with S.R.1, especially in

TABLE 5.9
Results for the NAG routines E04LAF/KAF

	E04LAF			E04KAF		
Problem	it.	fde.	gr.norm	it.	fde.	gr.norm
GENROSE U	25	43	2.7D-08	68	155	1.3D-07
GENROSE C	13	64	2.9D-09	25	192	1.1D-06
CHAINROSE U	13	25	7.2D-12	61	176	1.6D-07
CHAINROSE C	14	56	2.7D-14	24	220	2.0D-07
DEGENROSE U	14	24	7.1D-11	128	350	2.4D-07
DEGENROSE C	12	47	2.9D-11	19	383	8.5D-08
GENSING U	18	25	6.3D-11	45	95	1.1D-11
GENSING C	3	35	4.7D-15	7	108	1.4D-07
CHAINSING U	24	31	1.2D-10	117	192	4.8D-11
CHAINSING C	3	41	2.3D-15	8	209	2.4D-08
DEGENSING U	24	33	1.4D-10	100	200	9.0D-08
DEGENSING C	3	43	3.7D-11	8	218	4.9D-09
GENWOOD U	42	70	3.6D-07	122	230	3.5D-08
GENWOOD C	1	78	1.4D-13	6	242	1.0D-09
CHAINWOOD U	37	53	6.0D-07	109	203	2.4D-07
CHAINWOOD C	1	61	1.8D-13	18	229	1.7D-07
HOSC45 U	8	16	0.0D+00	8	71	0.0D+00
HOSC45 C	9	32	0.0D+00	9	95	0.0D+00
BROYDEN1A U	13	20	1.3D-08	70	140	3.1D-07
BROYDEN1A C	7	34	7.5D-09	22	186	1.1D-08
BROYDEN1B U	6	13	3.6D-08	40	103	8.7D-08
BROYDEN1B C	6	26	2.0D-09	19	143	1.9D-06
BROYDEN2A U	15	22	4.6D-08	118	213	9.2D-07
BROYDEN2A C	10	39	6.7D-09	37	287	2.3D-07
BROYDEN2B U	8	15	2.3D-14	45	122	8.0D-07
BROYDEN2B C	8	30	2.2D-09	31	191	5.9D-07
TOINTBROY U	7	14	2.1D-12	80	176	9.0D-07
TOINTBROY C	6	27	1.7D-12	36	234	4.2D-07
TRIG U	4	11	3.1D-04	12	51	2.2D-08
TRIG C	4	23	5.0D-05	11	70	1.3D-06
TOINTTRIG U	7	17	3.6D-09	16	69	2.6D-06
TOINTTRIG C	6	30	1.3D-13	11	97	2.5D-07
CRAGGLEVY U	29	36	1.4D-10	109	206	6.7D-09
CRAGGLEVY C	25	69	1.4D-11	65	284	7.4D-09
PENALTY U	40	47	8.3D-09	248	357	5.6D-04
PENALTY C	18	72	2.1D-08	132	516	6.6D-08
AUGMLAGN U	15	24	3.7D-09	78	168	1.9D-07
AUGMLAGN C	27	66	3.2D-13	81	275	1.1D-07
BROWN1 U	49	56	4.5D-08	500	803	1.7D-09
BROWN1 C	28	91	3.4D-10	271	1109	5.6D-08
BROWN3 U	7	14	1.1D-17	16	49	4.0D-11
BROWN3 C	6	27	6.7D-10	9	63	1.4D-08
BVP(N=10) U	3	10	2.4D-11	17	56	9.8D-10
BVP(N=10) C	7	24	5.6D-16	24	92	4.3D-08
BVP(N=20) U	3	10	7.7D-14	41	87	2.5D-10
BVP(N=20) C	8	25	1.7D-12	50	160	3.7D-08
VAR(N=20) U	6	13	1.2D-08	29	78	3.8D-08
VAR(N=20) C	9	29	2.9D-12	34	135	4.3D-08
VAR(N=45) U	6	13	1.4D-11	55	134	1.1D-07
VAR(N=45) C	13	34	4.3D-12	87	277	1.3D-06

TABLE 5.10
Results for the DEGENSING (U, $n = 20$) *problem with restarts allowed in the conjugate gradient iteration*

Hessian	it.	de.	usk	c.g.	gr.norm	cas	time
Exact	20	21	-	142	9.3D-07	9	2.68
B.F.G.S.	104	78	0	531	5.2D-07	86	11.50
D.F.P	203	189	0	1203	7.5D-07	203	23.77
P.S.B.	160	127	-	1033	6.8D-07	160	17.69
S.R.1	85	64	0	459	4.2D-07	64	9.49

view of the high esteem in which the scheme is held when used in conjunction with line-search methods (see Dennis and Schnabel [8], or any other modern textbook on numerical optimization). Indeed, the number of problems for which this update performs worst is alarmingly high. We note that there have been previous instances reported where the B.F.G.S. (and D.F.P.) method(s) perform rather poorly but the rank-one formula would prove highly efficient (see Powell [24]). The P.S.B. update appears to give more reliable results than the D.F.P. However, it seems somewhat surprising, in view of the relative success of the S.R.1 formula and our belief that part of this success is attributable to the convergence of the updates to the true second derivatives, that the P.S.B. formula performs so poorly. Under the same weak conditions that are needed to prove the convergence of the updates in the S.R.1 case, the P.S.B. method should also generate convergent second-derivative approximations (see Powell [22]) but, in our experiments, this convergence often seems to be very slow—there are actually a couple of differences between the convergence results for the two methods, namely that, as mentioned above, the convergence result is only known for S.R.1 in the case of quadratic functions, whereas for P.S.B. the result is for a much larger class of functions but the convergence is actually finite (i.e., it occurs in theory after a finite number of updates) in the quadratic case for S.R.1 but not P.S.B. The D.F.P. update is the most unreliable. It is interesting to observe that it often fails or performs poorly on problems for which B.F.G.S. is efficient.

A number of different initial approximations $B^{(0)}$ have also been tried, including the use of the exact second derivatives (modified to be positive definite if necessary) and the suggestions of Shanno and Phua [25] and Dennis and Schnabel [8]. As was to be expected, such different initial approximations did not significantly alter the numerical results, since the problems cited are all reasonably well scaled.

We now turn to the comparison of the new algorithm with the existing library routines. We believe that the important figures in such a comparison are the number of function and gradient calls required to solve the problem. For the new algorithm, there is one function evaluation per iteration (it.) and the number of gradient evaluations is given in the column headed (de.) in Tables 5.1–5.5. For the NAG routines, function and gradient evaluations occur in pairs and the number of such evaluations occurs in the column maked (fde.) of Table 5.9.

The exact second-derivative method E04LAF is clearly superior to its quasi-Newton counterpart E04KAF. If we compare E04LAF with the exact second-derivative version of our new algorithm, we see that the two methods are roughly comparable, but with the edge slightly in favor of the new algorithm (the new algorithm requiring fewer function evaluations on 43 and fewer gradient evaluations on 44 of the 50 tests carried out, although the differences are not particularly large, as both methods are very efficient), especially on the constrained (C) problems. We attribute the particularly good performance on the constrained problems, in part, to being able to make significant changes in the active set during a single iteration in our framework. In view of our previous testing, we next compare E04KAF with the S.R.1 version of our algorithm. It is generally believed that the B.F.G.S. update is the best updating scheme for use with line-search based methods, and thus E04KAF should prove a good comparison for the updating versions of the new algorithm. As we have already stated, the S.R.1 version of our algorithm is our preferred choice when second derivatives are not available. We see that the S.R.1 version of the new algorithm performs very favorably against E04KAF, the new algorithm requiring fewer function evaluations on 46, and fewer gradient evaluations on 47, of the tests performed with the margin often being relatively high, particularly on the constrained (C) problems. Finally, when E04KAF is compared with the B.F.G.S. version of the new algorithm, the verdict is still in favor of our algorithm, but the difference is not so dramatic.

6. Discussion. We feel that the results indicate that our framework is a good one in which to consider solving problem (1.1), especially in view of the good theoretical properties of the methods given in our previous paper (Conn, Gould and Toint [3]). Moreover, they indicate that updating schemes which are held in good repute for line-search based methods are not necessarily the best within a trust-region context. This was rather surprising to us and, as yet, we cannot give a complete theoretical justification for the observations. We expect such a theory will not be easy to develop but feel that the experimental evidence that trust-region algorithms do not behave at all like line-search algorithms may well be an important insight. We recommend our algorithm with exact second derivatives, if these are available; otherwise, we recommend our algorithm with a (safeguarded) symmetric rank-one approximation to these derivatives.

The issues involved in solving larger problems are more complicated. We would not, for instance, imagine storing dense matrices. Our choice of a conjugate gradient "inner iteration" does not require that we access matrices, but merely that we are able to form matrix-vector products—the choice of algorithm has always had the large-scale case in mind. To this end, we envisage using the partial separability of the problem functions (see Griewank and Toint [15]) to allow efficient storage and updating of matrices in matrix-vector product form. This approach has the further advantage that accurate approximations to the second derivatives of the element functions, normally being of low rank, are easier to obtain than for assembled matrices. As we have suggested, this is important within our algorithmic framework for good performance. Of course, it is essential to use preconditioning when attempting to solve such large problems and we are currently experimenting

with a number of preconditioners. The theory given in our previous paper (Conn, Gould and Toint [3]) has this in mind.

The final aim of this research is actually to produce effective methods for solving general nonlinear programming problems. Our intention here is to solve problems of this form by combining the nonlinear constraints, in a suitable fashion, with the objective function (for instance, an augmented Lagrangian function) and solving the resulting (sequence of) bound constrained minimization problem(s) using the methods described in this paper. We anticipate an interesting tradeoff between the accuracy required in solving the sequence of bound constrained problems and the convergence of the overall method.

Acknowledgment. The authors would like to thank Bert Buckley, Iain Duff, Phil Gill, Jorge Moré, Mike Powell, John Reid and two anonymous referees for their helpful comments on a previous version of this paper.

Appendix. In our tests, we have attempted to solve the following problems. For each problem, we give (a) the function $f(x)$, (b) any bounds on the variables for the "unconstrained" problem, (c) the starting point $x^{(0)}$ and (d) and (e) the optimal solutions x_U^* and x_C^* obtained for the unconstrained and the constrained problems.

1. The Generalized Rosenbrock function (GENROSE) (Moré, Garbow and Hillstrom [19])

(a) $f(x) = 1 + \sum_{i=2}^{n} [100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2]$.

(c) $x^{(0)} = (-1.2, 1, -1.2, 1, 1, 1, \ldots, 1, 1)$.

(d) $x_U^* = (1, 1, 1, \ldots, 1)$.

(e) $x_C^* = (1.1, 1.0775, 1.1, 1.0972, 1.1528, 1.3075, 1.7026, 2.8987)$ $(n = 8)$.

2. The Chained Rosenbrock function (CHAINROSE) (Toint [28])

(a) $f(x) = 1 + \sum_{i=2}^{n} [4\alpha_i(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2]$,

where the constants α_i are as given by Toint [28].

(c) $x^{(0)} = (-1, -1, -1, \ldots, -1)$.

(d) $x_U^* = (1, 1, 1, \ldots, 1)$.

(e) $x_C^* = (1.1, 1.0659, 1.1, 1.0711, 1.1, 1.0645, 1.1, 1.0788, 1.1,$
$1.0691, 1.1, 1.0811, 1.1, 1.0759, 1.1, 1.0720, 1.1, 1.0714,$
$1.1, 1.0684, 1.1, 1.0652, 1.1, 1.1782, 1.3881)$ $(n = 25)$.

3. The Degenerate Chained Rosenbrock function (DEGENROSE)
The same as 2 except that

(b) $x_i \leq 1$ for all i such that $i \bmod 3 = 0$.

(e) $x_C^* = (1.1, 1.0659, 1.1, 1.0711, 1.1, 1, 1.1, 1.0788, 1.1, 1.0691,$
$1.1, 1, 1.1, 1.0759, 1.1, 1.0720, 1.1, 1, 1.1, 1.0684, 1.1,$
$1.0652, 1.1, 1, 1.3881)$ $(n = 25)$.

4. The Generalized Singular function (GENSING) (Moré, Garbow and Hillstrom [19])

(a) $f(x) = \sum_{i \in J}[(x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - 10x_{i+3})^4]$,

where n is a multiple of 4 and $J = \{1, 5, 9, \ldots, n-3\}$.

(c) $x^{(0)} = (3, -1, 0, 1, 3, -1, 0, 1, \ldots, 3, -1, 0, 1)$.

(d) $x_U^* = (0, 0, 0, \ldots, 0)$.

(e) $x_C^* = (0.1, \sigma, 0.1, 0.1, \ldots, 0.1, \sigma, 0.1, 0.1)$,
where $\sigma = -9.8153D - 3 \quad (n = 20)$.

5. The Chained Singular function (CHAINSING)

(a) $f(x) = \sum_{i \in J}[(x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - 10x_{i+3})^4]$,

where n is a multiple of 4 and $J = \{1, 3, 5, \ldots, n-3\}$.

(c) $x^{(0)} = (3, -1, 0, 1, 3, -1, 0, 1, \ldots, 3, -1, 0, 1)$.

(d) $x_U^* = (0, 0, 0, \ldots, 0)$.

(e) $x_C^* = (0.1, \sigma, 0.1, \omega, 0.1, \omega, 0.1, \omega, 0.1, \omega,$
$0.1, \omega, 0.1, \omega, 0.1, \omega, 0.1, \omega, 0.1, 0.1)$,
where $\sigma = -9.8153D - 3$ and $\omega = -4.3827D - 3$ $(n = 20)$.

6. The Degenerate Chained Singular function (DEGENSING)
The same as 5 except that

(b) $x_i \leq 0$ for all i such that $i \bmod 3 = 0$ and $i \bmod 4 = 2$
and $x_i \geq 0$ for all i such that $i \bmod 3 = 0$ and $i \bmod 4 \neq 2$.

(e) $x_C^* = (0.1, \sigma, 0.1, \omega, 0.1, \omega, 0.1, \omega, 0.1, \omega,$
$0.1, 0, 0.1, \omega, 0.1, \omega, 0.1, \omega, 0.1, 0.1)$,
where $\sigma = -9.8153D - 3$ and $\omega = -4.3827D - 3$ $(n = 20)$.

7. The Generalized Wood function (GENWOOD) (see, Moré, Garbow and Hillstrom [19] for the Wood function)

(a) $f(x) = 1 + \sum_{i \in J}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 + (1 - x_{i+2})^2 + 10(x_{i+1} + x_{i+3} - 2)^2 + 0.1(x_{i+1} - x_{i+3})^2]$,

where n is a multiple of 4 and $J = \{1, 5, 9, \ldots, n-3\}$.

(c) $x^{(0)} = (-3, -1, -3, -1, -2, 0, -2, 0, \ldots, -2, 0)$.

(d) $x_U^* = (1, 1, 1, \ldots, 1)$.

(e) $x_C^* = (1.1, 1.1753, 1.1, 1.1715, \ldots, 1.1, 1.1753, 1.1, 1.1715)$.

8. The Chained Wood function (CHAINWOOD)

(a) $f(x) = 1 + \sum_{i \in J} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 + (1 - x_{i+2})^2 + 10(x_{i+1} + x_{i+3} - 2)^2 + 0.1(x_{i+1} - x_{i+3})^2]$,

where n is a multiple of 4 and $J = \{1, 3, 5, \ldots, n-3\}$.

(c) $x^{(0)} = (-3, -1, -3, -1, -2, 0, -2, 0, \ldots, -2, 0)$.

(d) $x_U^* = (1, 1, 1, \ldots, 1)$.

(e) $x_C^* = (1.1, 1.1751, 1.1, 1.1734, 1.1, 1.1736, 1.1, 1.1716)$ $(n = 8)$.

9. A generalization of Hock and Schittkowski's 45th problem (HOSC45) (Hock and Schittkowski [17])

(a) $f(x) = 2 - \prod_{i=1}^{n} x_i/n!$.

(b) $0 \le x_i \le i$ for $1 \le i \le n$.

(c) $x^{(0)} = (2, 2, 2, \ldots, 2)$.

(d) $x_U^* = (1, 2, 3, \ldots, n)$.

(e) $x_C^* = (2.1, 2, 4.1, 4, 6.1, 6, \ldots)$.

10. A generalization of the Broyden Tridiagonal function (BROYDEN1A) (see Moré, Garbow and Hillstrom [19] for the Broyden Tridiagonal function)

(a) $f(x) = 1 + \sum_{i=1}^{n} |(3 - 2x_i)x_i - x_{i-1} - x_{i+1} + 1|^p$,

where $p = 7/3$ and $x_0 = x_{n+1} = 0$.

(c) $x^{(0)} = (-1, -1, -1, \ldots, -1)$.

(d) $x_U^* = (-0.5707, -0.6819, -0.7025, -0.7063, -0.7070, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7071, -0.7070, -0.7068, -0.7064, -0.7051, -0.7015, -0.6919, -0.6658, -0.5960, -0.4164)$ $(n = 30)$.

(e) $x_C^* = (-0.4707, -0.5909, -0.6025, -0.6196, -0.6070, -0.6200, -0.6071, -0.6200, -0.6071, -0.6200, -0.6071, -0.6200, -0.6017, -0.6200, -0.6071, -0.6200, -0.6071, -0.6200, -0.6017, -0.6200, -0.6071, -0.6200, -0.6068, -0.6193, -0.6050, -0.6146, -0.5919, -0.5758, -0.4960, -0.3570)$ $(n = 30)$.

11. Another generalization of the Broyden Tridiagonal function (BROYDEN1B)

The same as 10 except that $p = 2$ and

(e) $x_C^* = (-0.4707, -0.5952, -0.6025, -0.6233, -0.6070, -0.6239, -0.6071,$
$-0.6239, -0.6071, -0.6239, -0.6071, -0.6239, -0.6017, -0.6239,$
$-0.6071, -0.6239, -0.6071, -0.6239, -0.6017, -0.6239, -0.6071,$
$-0.6238, -0.6068, -0.6232, -0.6050, -0.6183, -0.5919, -0.5794,$
$-0.4960, -0.3625)$ $(n = 30)$.

12. A generalization of the Broyden Banded function (BROYDEN2A) (see Moré, Garbow and Hillstrom [19] for the Broyden Banded function)

(a) $$f(x) = 1 + \sum_{i=1}^{n} \left| (2 + 5x_i^2)x_i + 1 + \sum_{j \in J_i} x_j(1 + x_j) \right|^p,$$

where $p = 7/3$ and $J_i = \{j : \max(1, i-5) \leq j \leq \min(n, i+1)\}$.

(c) $x^{(0)} = (-1, -1, -1, \ldots, -1)$.

(d) $x_U^* = (-0.4774, -0.5204, -0.5584, -0.5921, -0.6223, -0.6505, -0.6481,$
$-0.6456, -0.6436, -0.6422, -0.6415, -0.6418, -0.6420, -0.6422,$
$-0.6422, -0.6422, -0.6422, -0.6422, -0.6422, -0.6422, -0.6422,$
$-0.6422, -0.6422, -0.6422, -0.6422, -0.6422, -0.6422, -0.6422,$
$-0.6430, -0.6140)$ $(n = 30)$.

(e) $x_C^* = (-0.3774, -0.5258, -0.4584, -0.6089, -0.5223, -0.6715, -0.5481,$
$-0.6702, -0.5436, -0.6682, -0.5415, -0.6681, -0.5420, -0.6682,$
$-0.5422, -0.6682, -0.5422, -0.6682, -0.5422, -0.6682, -0.5422,$
$-0.6684, -0.5422, -0.6687, -0.5422, -0.6649, -0.5422, -0.6610,$
$-0.5430, -0.6264)$ $(n = 30)$.

13. Another generalization of the Broyden Banded function (BROYDEN2B)

The same as 12 except that $p = 2$ and

(e) $x_C^* = (-0.3774, -0.5209, -0.4584, -0.6014, -0.5223, -0.6643, -0.5481,$
$-0.6630, -0.5436, -0.6611, -0.5415, -0.6611, -0.5420, -0.6612,$
$-0.5422, -0.6612, -0.5422, -0.6612, -0.5422, -0.6612, -0.5422,$
$-0.6613, -0.5422, -0.6616, -0.5422, -0.6585, -0.5422, -0.6557,$
$-0.5430, -0.6228)$ $(n = 30)$.

14. Toint's 7-diagonal generalization of the Broyden Tridiagonal function (TOINTBROY) (see Toint [28])

(a) $f(x) = 1 + \sum_{i=1}^{n} |(3 - 2x_i)x_i - x_{i-1} - x_{i+1} + 1|^p + \sum_{i=1}^{n/2} |x_i + x_{i+n/2}|^p,$

where n is even, $p = 7/3$ and $x_0 = x_{n+1} = 0$.

(c) $x^{(0)} = (-1, -1, -1, \ldots, -1)$.

(d) $x_U^* = (-0.4114, -0.4729, -0.4732, -0.4673, -0.4633, -0.4614, -0.4608,$
$-0.4614, -0.4630, -0.4657, -0.4700, -0.4761, -0.4838, -0.4914,$
$-0.4939, -0.4808, -0.4681, -0.4607, -0.4574, -0.4560, -0.4554,$
$-0.4546, -0.4532, -0.4506, -0.4459, -0.4374, -0.4221, -0.3938,$
$-0.3405, -0.2340)$ $(n = 30)$.

(e) $x_C^* = (-0.3114, -0.3802, -0.3732, -0.3780, -0.3632, -0.3712, -0.3608,$
$-0.3713, -0.3630, -0.3758, -0.3700, -0.3867, -0.3838, -0.4029,$
$-0.3939, -0.3919, -0.3681, -0.3706, -0.3574, -0.3658, -0.3554,$
$-0.3643, -0.3532, -0.3601, -0.3459, -0.3469, -0.3221, -0.2973,$
$-0.2405, -0.1811)$ $(n = 30)$.

15. A trigonometric function (TRIG) (Nazareth [20])

(a) $f(x) = \sum_{i=1}^{n} \left[n + i - \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j) \right]^2,$

where $a_{ij} = \delta_{ij}, b_{ij} = 1 + i\delta_{ij}$ and $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

(c) $x^{(0)} = (1/n, 1/n, 1/n, \ldots, 1/n)$.

(d) $x_U^* = (1.5708, 0.1, 0, 1.5708, 0.1, 0, 1.5708, 0.1, 0, 1.5708)$.

(e) $x_C^* = (1.6708, 0.1, 0.1, 1.5708, 0.2, 0, 1.6708, 0.1, 0.1, 1.5708)$.

16. Another trigonometric function (TOINTTRIG) (Toint [28])

(a) $f(x) = \sum_{(i,j) \in J} \alpha_{ij} \sin[\beta_i x_i + \beta_j x_j + \gamma_{ij}],$

where $\alpha_{ij} = 5[1 + \text{mod}(i, 5) + \text{ mod } (j, 5)], \beta_i = 1 + i/10$ and $\gamma_{ij} = (i + j)/10$. We selected $J = \{(i, j) : \text{ mod } (|i - j|, 4) = 0\}$.

(c) $x^{(0)} = (1, 1, 1, \ldots, 1)$.

(d) $x_U^* = (2.0511, 1.7968, 1.5817, 1.3973, 1.2375, 1.0976, 0.9742, 0.8645,$
$0.7664, 0.6781)$ $(n = 10)$.

(e) $x_C^* = (2.1511, 1.7968, 1.6817, 1.3973, 1.3375, 1.0976, 1.0742, 0.8645,$
$0.8664, 0.6781)$ $(n = 10)$.

17. A generalization of the Cragg and Levy function (CRAGGLEVY) (see Cragg and Levy [4] for the Cragg and Levy function)

(a) $f(x) = \sum_{i \in J}[(e^{x_i} - x_{i+1})^4 + 100(x_{i+1} - x_{i+2})^6 + \tan^4(x_{i+2} - x_{i+3}) + x_i^8 + (x_{i+3} - 1)^2]$,

where n is a multiple of 4 and $J = \{1, 5, 9, \ldots, n-3\}$.

(c) $x^{(0)} = (1, 2, 2, 2, 2, 2, 2, 2, \ldots, 2, 2, 2, 2)$.

(d) $x_U^* = (0, 1, 1, 1, \ldots, 0, 1, 1, 1)$.

(e) $x_C^* = (0.1, 1.1045, 1.1, 1.0019, \ldots, 0.1, 1.1045, 1.1, 1.0019)$.

18. A penalty function (PENALTY)

(a) $f(x) = 1 + \sum_{i=1}^{n} x_i + \mu\left(1 - \sum_{i=1}^{n} 1/x_i\right)^2 + \mu\left(1 - \sum_{i=1}^{n} i/x_i\right)^2$.

For our tests, we selected $\mu = 1000$.

(b) $-0.01 \leq x_i \leq 10000$ for $1 \leq i \leq n$.

(c) $x^{(0)} = (1, 1, 1, \ldots, 1)$.

(d) $x_U^* = 100(0.0371, 0.3346, 0.4718, 0.5772, 0.6662, 0.7446, 0.8155, 0.8807, 0.9414, 0.9984, 1.0524, 1.1037, 1.1527, 1.1997, 1.2450)$ $(n = 15)$.

(e) $x_C^* = 100(0.0381, 0.3302, 0.4728, 0.5732, 0.6672, 0.7404, 0.8165, 0.8762, 0.9424, 0.9936, 1.0534, 1.0986, 1.1537, 1.1944, 1.2460)$ $(n = 15)$.

19. An Augmented Lagrangian function for a generalization of Hock and Schittkowski's 80th problem (AUGMLAGN) (Hock and Schittkowski [17])

(a) $f(x) = 1 + \sum_{i \in J}\left[e^{x_i x_{i+1} x_{i+2} x_{i+3} x_{i+4}} + \frac{1}{2}\rho\left(\left(\sum_{j=0}^{4} x_{i+j}^2 - 10 - \lambda_1\right)^2 + (x_{i+1}x_{i+2} - 5x_{i+3}x_{i+4} - \lambda_2)^2 + (x_i^3 + x_{i+1}^3 + 1 - \lambda_3)^2\right)\right]$,

where n is a multiple of 5 and $J = \{1, 6, 11, \ldots, n-4\}$. For our tests, we selected $\rho = 20, \lambda_1 = -0.002008, \lambda_2 = -0.001900$ and $\lambda_3 = -0.000261$ (which makes $f(x)$ an exact penalty function for Hock and Schittkowski's problem).

(b) $-2.3 \leq x_i \leq 2.3$ for $1 \leq i \leq n$.

(c) $x^{(0)} = (-2, 2, 2, -1, -1, -1, -1, 2, -1, -1, \ldots, -1, -1, 2, -1, -1)$.

(d) $x_U^* = (-1.7171, 1.5957, -1.8273, -0.7636, -0.7636, \ldots, -1.7171, 1.5957, -1.8273, -0.7636, -0.7636)$.

(e) $x_C^* = (-1.6171, 1.4782, -1.9928, -0.8877, -0.6636, -1.8045, 1.6957,$
$-1.6488, -0.6636, -0.8425, -0.7290, -0.8553, 2.6161,$
$-1.3343, -0.3363)$ $(n = 15)$.

20. A generalization of a function due to A. Brown (BROWN1) (L. C. W. Dixon, private communication)

(a) $f(x) = \left[\sum_{i\in J}(x_i - 3)\right]^2 + \sum_{i\in J}[(0.0001(x_i - 3)^2 - (x_i - x_{i+1}) + e^{20(x_i - x_{i+1})}]$,
where n is a multiple of 2 and $J = \{1, 3, 5, \ldots, n-1\}$.

(b) $-1 \le x_i \le 4$ for $1 \le i \le n$.

(c) $x^{(0)} = (0, -1, 0, -1, \ldots, 0, -1)$.

(d) $x_U^* = (3, 3.1498, 3, 3.1498, \ldots, 3, 3.1498)$.

(e) $x_C^* = (3.1, 3.2498, 3.1, 3.2498, \ldots, 3.1, 3.2498)$.

21. A generalization of another function due to A. Brown (BROWN3) (L. C. W. Dixon, private communication)

(a) $f(x) = \sum_{i=1}^{n-1}[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}]$.

(c) $x^{(0)} = (-1, 1, -1, 1, \ldots, -1, 1)$.

(d) $x_U^* = (0, 0, 0, \ldots, 0)$.

(e) $x_C^* = (0.1, 0, 0.1, 0, \ldots, 0.1, 0)$.

22. A discrete boundary value problem (BVP) (Moré, Garbow and Hillstrom [19])

(a) $f(x) = \sum_{i=1}^{n}[2x_i - x_{i-1} - x_{i+1} + h^2(x_i + ih + 1)^3/2]$,
where $h = 1/(n+1)$ and $x_0 = x_{n+1} = 0$.

(b) $-0.2n \le x_i \le 0.2n$ for $1 \le i \le n$.

(c) $x_i^{(0)} = ih(ih - 1)$ for $1 \le i \le n$.

(d) $x_U^* = 0.1(-0.4317, -0.8158, -1.1449, -1.4097, -1.5991, -1.6988, -1.6909,$
$-1.5525, -1.2536, -0.7542)$ $(n = 10)$,
$x_U^* = 0.1(0.2321, -0.4520, -0.6588, -0.8514, -1.0288, -1.1895, -1.3322,$
$-1.4553, -1.5571, -1.6354, -1.6881, -1.7127, -1.7060, -1.6650,$
$-1.5856, -1.4636, -1.2938, -1.0702, -0.7858, -0.4323)$ $(n = 20)$.

(e) $x_C^* = 0.01(5.6835, 8.4100, 8.9057, 7.8272, 5.7611, 3.2315, 0.7129,$
$-1.3527, -2.5356, -2.3936)$ $(n = 10)$,
$x_C^* = 0.1(0.7679, 1.3625, 1.8041, 2.1121, 2.3047, 2.3990, 2.4107,$
$2.3542, 2.2425, 2.0875, 1.9000, 1.6895, 1.4648, 1.2337,$
$1.0034, 0.7805, 0.5710, 0.3050, 0.2142, 0.0773)$ $(n = 20)$.

23. The discretization of a variational problem (VAR) (Toint [28])

(a) $f(x) = 2\sum_{i=1}^{n}[x_i(x_i - x_{i+1})]/h + 2\lambda h \sum_{i=0}^{n}[(e^{x_{i+1}} - e^{x_i})/(x_{i+1} - x_i)]$,

where $h = 1/(n+1)$ and $x_0 = x_{n+1} = 0$. We selected $\lambda = -3.4$.

(b) $-0.2n \leq x_i \leq 0.2n$ for $1 \leq i \leq n$.

(c) $x_i^{(0)} = 0.1ih(1-i)$ for $1 \leq i \leq n$.

(d) $x_U^* = 0.1(1.4638, 2.8383, 4.1104, 5.2663, 6.2918, 7.1729, 7.8964,$
$8.4505, 8.8256, 9.0150, 9.0150, 8.8256, 8.4505, 7.8964,$
$7.1729, 6.2918, 5.2663, 4.1104, 2.8383, 1.4638)$ $(n = 20)$,

$x_U^* = (0.6812, 1.3452, 1.9909, 2.6169, 3.2220, 3.8050, 4.3645,$
$4.8991, 5.4075, 5.8883, 6.3401, 6.7617, 7.1517, 7.5089,$
$7.8320, 8.1200, 8.3718, 8.5865, 8.7633, 8.9016, 9.0007,$
$9.0604, 9.0803, 9.0604, 9.0007, 8.9016, 8.7633, 8.5865,$
$8.3718, 8.1200, 7.8320, 7.5089, 7.1517, 6.7617, 6.3401,$
$5.8883, 5.4075, 4.8991, 4.3645, 3.8050, 3.2220, 2.6169,$
$1.9909, 1.3452, 0.6812)$ $(n = 45)$.

(e) $x_C^* = 0.1(0.2464, 0.4253, 0.5925, 0.7456, 0.8826, 1.0010, 1.0984,$
$1.1728, 1.2224, 1.2459, 1.2427, 1.2129, 1.1573, 1.0773,$
$0.9747, 0.8517, 0.7107, 0.5540, 0.3838, 0.1966)$ $(n = 20)$,

$x_C^* = (0.1681, 0.3084, 0.4464, 0.5820, 0.7146, 0.8440, 0.9697,$
$1.0911, 1.2077, 1.3189, 1.4241, 1.5227, 1.6139, 1.6970,$
$1.7714, 1.8363, 1.8912, 1.9354, 1.9685, 1.9902, 2.0001,$
$2.0100, 2.0080, 2.0100, 2.0001, 1.9902, 1.9685, 1.9354,$
$1.8912, 1.8363, 1.7714, 1.6970, 1.6139, 1.5227, 1.4241,$
$1.3189, 1.2077, 1.0911, 0.9697, 0.8440, 0.7146, 0.5820,$
$0.4464, 0.3084, 0.1681)$ $(n = 45)$.

Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, Canada

Computer Science and Systems Division
A.E.R.E. Harwell
Oxford, England

Department of Mathematics
Facultés Universitaires ND de la Paix
B-5000 Namur, Belgium

1. D. P. BERTSEKAS, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, London and New York, 1982.

2. R. K. BRAYTON & J. CULLUM, "An algorithm for minimizing a differentiable function subject to box constraints and errors," *J. Optim. Theory Appl.*, v. 29, 1979, pp. 521–558.

3. A. R. CONN, N. I. M. GOULD & PH. L. TOINT, "Global convergence of a class of trust region algorithms for optimization with simple bounds," *SIAM J. Numer. Anal.*, v. 25, 1988, pp. 433–460.

4. E. E. CRAGG & A. V. LEVY, "Study on a supermemory gradient method for the minimization of functions," *J. Optim. Theory Appl.*, v. 4, 1969, pp. 191–205.

5. J. CULLUM & R. K. BRAYTON, "Some remarks on the symmetric rank-one update," *J. Optim. Theory Appl.*, v. 29, 1979, pp. 493–520.

6. R. S. DEMBO, S. C. EISENSTAT & T. STEIHAUG, "Inexact Newton methods," *SIAM J. Numer. Anal.*, v. 19, 1982, pp. 400–408.

7. R. S. DEMBO & U. TULOWITZKI, *On the Minimization of Quadratic Functions Subject to Box Constraints*, Working paper series B 71, School of Organization and Management, Yale University, 1983.

8. J. E. DENNIS & R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Ciffs, N.J., 1983.

9. R. FLETCHER, *Practical Methods of Optimization*, Vol. 1, Wiley, Chichester and New York, 1980.

10. R. FLETCHER & M. P. JACKSON, "Minimization of a quadratic function of many variables subject only to lower and upper bounds," *J. Inst. Math. Appl.*, v. 14, 1974, pp. 159–174.

11. U. M. GARCIA-PALOMARES, "Superlinearly convergent algorithms for linearly constrained optimization," in *Nonlinear Programming* 2 (O. L. Mangasarian, R. R. Meyer and S. M. Robinson, eds.), Academic Press, London and New York, 1975, pp. 101–119.

12. P. E. GILL & W. MURRAY, *Minimization Subject to Bounds on the Variables*, report NAC 71, National Physical Laboratory, England, 1976.

13. P. E. GILL, W. MURRAY, M. A. SAUNDERS & M. H. WRIGHT, *Some Theoretical Properties of an Augmented Lagrangian Function*, Technical Report SOL 86–6, Department of Operations Research, Stanford University, 1986.

14. P. E. GILL, W. MURRAY & M. H. WRIGHT, *Practical Optimization*, Academic Press, London and New York, 1981.

15. A. GRIEWANK & PH. L. TOINT, "On the unconstrained optimization of partially separable functions," in *Nonlinear Optimization*, 1981 (M. J. D. Powell, ed.), Academic Press, London and New York, 1982, pp. 301–312.

16. A. GRIEWANK & PH. L. TOINT, "Partitioned variable metric updates for large structured optimization problems," *Numer. Math.*, v. 39, 1982, pp. 119–137.

17. W. HOCK & K. SCHITTKOWSKI, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, Berlin, 1981.

18. J. J. MORÉ, "Recent developments in algorithms and software for trust region methods," *Mathematical Programming, The State of the Art, Bonn* 1982 (A. Bachem, M. Grötschel and B. Korte, eds.), Springer-Verlag, Berlin, 1983, pp. 258–287.

19. J. J. MORÉ, B. S. GARBOW & K. E. HILLSTROM, "Testing unconstrained optimization software," *ACM Trans. Math. Software*, v. 7, 1981, pp. 17–41.

20. J. L. NAZARETH, "Analogues of Dixon's and Powell's theorems for unconstrained minimization with inexact line searches," *SIAM J. Numer. Anal.*, v. 23, 1986, pp. 170–177.

21. D. P. O'LEARY, "A generalized conjugate gradient algorithm for solving a class of quadratic programming problems," *Linear Algebra Appl.*, v. 34, 1980, pp. 371–399.

22. 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 and New York, 1970, pp. 31–65.

23. M. J. D. POWELL, *Some Global Convergence Properties of a Variable Metric Algorithm for Minimization Without Exact Line Searches*, SIAM-AMS Proc., vol. 9, Amer. Math. Soc., Providence, R.I., 1976, pp. 53–72.

24. M. J. D. POWELL, "How bad are the BFGS and DFP methods when the objective function is quadratic?," *Math. Programming*, v. 34, 1986, pp. 34–47.

25. D. F. SHANNO & K. H. PHUA, "Matrix conditioning and nonlinear optimization," *Math. Programming*, v. 14, 1978, pp. 145–160.

26. D. C. SORENSEN, "An example concerning quasi-Newton estimation of a sparse Hessian," *SIGNUM Newsletter*, v. 16, 1981, pp. 8–10.

27. T. STEIHAUG, "The conjugate gradient method and trust regions in large scale optimization," *SIAM J. Numer. Anal.*, v. 20, 1983, pp. 626–637.

28. PH. L. TOINT, "Some numerical results using a sparse matrix updating formula in unconstrained optimization," *Math. Comp.*, v. 32, 1978, pp. 839–851.

29. PH. L. TOINT, "Towards an efficient sparsity exploiting Newton method for minimization," in *Sparse Matrices and Their Uses* (I. S. Duff, ed.), Academic Press, London and New York, 1981.