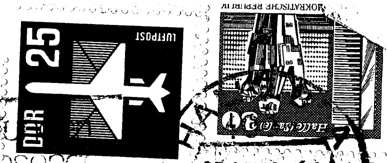
Sender: Dr. O. Knoth

Martin-Luther-Universität
Halle-Wittenberg
Sektion Mathematik
Universitätsplatz 6
DDR - Halle
4 0 2 0
Deutsche Demokratische Republik

MIT LUFTPOST
PAR AVION

IV/10/23 P 792/86 4000 7300

~~Postkarte~~



27.11.88-24

Dr. A. R. Conn
University of Waterloo
Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1
Kanada

Halle, 25.11.88

Dear Dr. Conn

I should greatly appreciate receiving a reprint of your paper entitled:

Performance of a multifrontal scheme for partially separable optimization

which appeared in CS-88-04

(and also of any other related papers).

With many thanks for your kindness,

Yours sincerely,

Donald Knuth

sent with complements Dec. 12/88

Printing Requisition / Graphic Services

10349

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services, Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION

Performance of a Multifrontal Scheme for Partially Seperable Optimization CS-88-04

DATE REQUISITIONED	DATE REQUIRED	ACCOUNT NO.
Feb. 16/88	ASAP	1216 6011741

REQUISITIONER - PRINT	PHONE	SIGNING AUTHORITY
A.R. Conn	x3688	

MAILING INFO - NAME	DEPT.	BLDG. & ROOM NO.	DELIVER / PICK-UP
Sue DeAngelis	Computer Science	MC 6081E	xx DELIVER ☐ PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 20 NUMBER OF COPIES: 100

TYPE OF PAPER STOCK: [X] BOND [ ] NCR ___ PT. [X] COVER [ ] BRISTOL [X] SUPPLIED [ ] ___

PAPER SIZE: [X] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17 [ ] ___

PAPER COLOUR: [X] WHITE [ ] ___ INK: [X] BLACK [ ] ___

PRINTING: [ ] 1 SIDE ___ PGS. [X] 2 SIDES ___ PGS. NUMBERING: FROM ___ TO ___

BINDING/FINISHING: 3 down left side
[X] COLLATING [X] STAPLING [ ] ___ HOLE PUNCHED [ ] PLASTIC RING

FOLDING/PADDING: CUTTING SIZE:

Special Instructions
Math fronts and backs enclosed.

COPY CENTRE — OPER. NO. 50 | BLDG. KWB | MACH. NO. 905

DESIGN & PASTE-UP

OPER. NO.	TIME	LABOUR CODE
		D01
		D01
		D01

TYPESETTING

	QUANTITY			
PAP000000				T01
PAP000000				T01
PAP000000				T01

PROOF

PRF				
PRF				

NEGATIVES

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
FLM				C01
FLM				C01
FLM				C01
FLM				C01
FLM				C01

PMT

PMT				C01
PMT				C01
PMT				C01

PLATES

PLT				P01
PLT				P01
PLT				P01

STOCK

				001
				001
				001
				001

BINDERY

RNG		26	20	B01
RNG				B01
RNG				B01
MIS000000				B01

OUTSIDE SERVICES

$ ___ COST

PERFORMANCE OF A MULTIFRONTAL SCHEME FOR PARTIALLY SEPERABLE OPTIMIZATION

A.R. Conn† , N.I.M. Gould‡ , M. Lescrenier¶ and
Ph. L. Toint§

Research Report CS-88-04

February, 1988

† Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada.
‡ Harwell Laboratory, Harwell, England.
¶ Belgian National Fund for Scientific Research, Facultés Universitaires ND de la Paix, B-5000 Namur, Belgium.
§ Department of Mathematics, Facultés Universitaires ND de la Paix, B-5000 Namur, Belgium.

PERFORMANCE OF A MULTIFRONTAL SCHEME FOR PARTIALLY SEPERABLE OPTIMIZATION

by A.R. Conn†, N.I.M. Gould‡, M. Lescrenier¶ and Ph.L. Toint§

Abstract. We consider the solution of partially seperable minimization problems subject to simple bounds. At each iteration, a quadratic model is used to approximate the objective function within a trust region. To minimize this model, the iterative method of conjugate gradients has usually been used. The aim of this paper is to compare the performance of a direct method, a multifrontal scheme, with this iterative method (with and without preconditioning). To assess our conclusions, a set of numerical experiments, including large dimensional problems, is presented.

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

‡Harwell Laboratory,
Harwell, England

¶Belgian National Fund for Scientific Research
Facultés Universitaires ND de la Paix
B-5000 Namur, Belgium §Department of Mathematics
Facultés Universitaires ND de la Paix
B-5000 Namur, Belgium

This report is being issued simultaneously by the Department of Combinatorics and Optimization , University of Waterloo, Waterloo, Ontario, Canada, the Computer Science and Systems Division, Harwell Laboratory, Oxfordshire, U.K. and the Department of Mathematics, Facultés Notre Dame de la Paix, University of Namur, Namur, Belgium

1. Introduction.

This paper is concerned with the solution of partially separable optimization problems (defined in Section 2). Such problems appear in a large majority of nonlinear minimization applications, for example finite elements, network problems and others. The formalism was first introduced by Griewank and Toint (1982) and methods using this particular structure have proved to be extremely successful for large dimensional problems (see Griewank and Toint (1984) for instance).

To solve these problems, a trust region type algorithm may be applied which requires the partial solution of a quadratic minimization problem at each step of an iterative scheme. Up to now, only iterative methods, specifically (preconditioned and truncated) conjugate gradient schemes, have been used in practice to solve the quadratic minimization problem. The aim of this paper is to test the use of direct methods, particularly multifrontal schemes, for this purpose.

The authors are aware that this type of direct method can only be used when the solution of the quadratic problem can be found by solving a linear system, that is, when the quadratic has a finite solution. Such a situation normally arises when the Hessian matrix of the quadratic is positive definite. In the indefinite case, the authors use a simple strategy which consists of computing directions of negative curvature. However, they realize that a more sophisticated strategy like the Levenberg-Marquardt algorithm (see Moré 1977) or an attempt to solve the trust region problem by another method may be more appropriate.

The paper is organized as follows. Section 2 defines the concept of partial separability. Section 3 describes the trust region algorithm used to solve partially separable problems. Indirect and direct methods to solve the quadratic minimization problem are proposed in Sections 4 and 5 respectively. In Section 6 we discuss the numerical experiments and conclusions are drawn in Section 7. We end with an appendix which describes some additional test functions not available in the literature.

2. Partial separability.

We consider the simple bound constrained minimization problem

$$\begin{aligned} &\text{Min } f(x) \\ &a_i \le x_i \le b_i \end{aligned} \tag{2.1}$$

where x is a vector of R^n and f is a so called *partially separable function*, that is a function of the form

$$f(x) = \sum_{i=1}^{m} f_i(x) \tag{2.2}$$

where the *element functions* $f_i(x)$ have Hessian matrices of low rank compared with n, the dimension of the problem.

A typical case is when each element function only depends on a small subset of the variables called the *elemental variables*. Frequently, it is possible to further reduce the number of elemental variables applying a linear transformation of the type

$$y_i = U_i x \tag{2.3}$$

where the variables of y_i are called the *internal variables* for the i-th element function.

The adaptability of partially separable methods to problems of large sizes comes mainly from a compact storage scheme for the Hessian approximation and the corresponding updating technique. The change of variables (2.3) allows us to consider new element functions $\hat{f}_i$ such that

$$f_i(x)=\hat{f}_i(y_i). \tag{2.4}$$

The gradients of f_i and $\hat{f}_i$ satisfy

$$g_i(x)=U_i^{\mathrm{T}}\,\hat{g}_i(y_i), \tag{2.5}$$

while the Hessians satisfy

$$H_i(x)=U_i^{\mathrm{T}}\,\hat{H}_i(y_i)U_i. \tag{2.6}$$

The so called *partitioned updating technique* consists in storing and accessing only the gradients and Hessians in internal variables, that is, the $\hat{g}_i(y_i)$ and $\hat{H}_i(y_i)$ from (2.5) and (2.6) respectively. The advantage clearly comes from the fact that the number of internal variables is much smaller than the total dimension of the problem and that the Hessian of f is never explicitly assembled.

Finally, since in many practical problems, the matrices U_i only differ in the elemental variables that they select, they are not stored explicitly, but the operations involving them,

$$u=U_i v \quad \text{and} \quad u=U_i^{\mathrm{T}} v, \tag{2.7}$$

are performed in a user provided subroutine.

3. A trust region algorithm for partially separable problems.

The algorithm we propose for solving problem (2.1) is of trust region type and belongs to the class of methods described by Conn, Gould and Toint (1988a).

At each iteration, we suppose that we have a feasible point $x^{(k)}$, the gradient $g^{(k)}$ of the objective function (2.2) at this point and a suitable symmetric approximation $H^{(k)}$ to the Hessian of f at $x^{(k)}$. In the remainder of this paper, by 'Hessian' we will always mean an approximation of the true Hessian. This approximation may be computed by a secant updating formula such as the Broyden-Fletcher-Goldfarb-Shanno or the symmetric rank-one update for instance (see Dennis and Schnabel (1983)). The gradient and Hessian of the partially separable function f are stored as described in Section 2. We approximate the objective function by a quadratic model

$$m^{(k)}(x^{(k)}+s)=f(x^{(k)})+s^{\mathrm{T}}g^{(k)}+\tfrac{1}{2}s^{\mathrm{T}}H^{(k)}s, \tag{3.1}$$

in a region surrounding the current iterate defined by its radius $\Delta^{(k)}$. This region, called the trust region, is of the form

$$\|x-x^{(k)}\|\le\Delta^{(k)}. \tag{3.2}$$

It is convenient to choose the infinity norm for then the shape of the trust region is aligned with the simple bounds of the problem (2.1). If we define

$$\begin{aligned} l_i^{(k)}&=\max(a_i,x_i^{(k)}-\Delta^{(k)}),\\ u_i^{(k)}&=\min(b_i,x_i^{(k)}+\Delta^{(k)}), \end{aligned} \tag{3.3}$$

a trial point $x^{(k)}+s^{(k)}$ is constructed by finding an approximation to the solution of the trust region problem, which is

$$\mathrm{Min}\; m^{(k)}(x) \tag{3.4.a}$$

subject to

$$l_i^{(k)}\le x_i\le u_i^{(k)}. \tag{3.4.b}$$

If the function value calculated at this new point matches its predicted value (the one given by the model), the 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. More precisely, we compute the ratio of the achieved to the predicted reduction of the objective function,

$$\rho^{(k)} = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{f(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)})} \tag{3.5}$$

and set

$$x^{(k+1)} = \begin{cases} x^{(k)} + s^{(k)} & \text{if } \rho^{(k)} > \mu, \\ x^{(k)} & \text{if } \rho^{(k)} \leq \mu, \end{cases} \tag{3.6}$$

and

$$\Delta^{(k+1)} = \begin{cases} \gamma_0 \Delta^{(k)} & \text{if } \rho^{(k)} \leq \mu, \\ \Delta^{(k)} & \text{if } \mu < \rho^{(k)} < \eta, \\ \gamma_2 \Delta^{(k)} & \text{if } \rho^{(k)} \geq \eta, \end{cases} \tag{3.7}$$

where $\gamma_0 < 1 < \gamma_2$, μ and η are appropriate numbers.

It remains now to describe our approximate solution of (3.4). We first compute the Generalized Cauchy Point $x_C^{(k)}$ which is defined as the first local minimizer of the univariate function

$$m^{(k)}(P[x^{(k)} - t g^{(k)}]) \tag{3.8}$$

where $P[\cdot]$ is the projection operator componentwise computed as

$$(P[x])_i = \begin{cases} l_i^{(k)} & \text{if } x_i \leq l_i^{(k)}, \\ u_i^{(k)} & \text{if } x_i \geq u_i^{(k)}, \\ x_i & \text{otherwise.} \end{cases} \tag{3.9}$$

The Generalized Cauchy Point (GCP) is then the first local minimizer of the model, along the piecewise linear arc defined by projecting the steepest descent direction into the feasible domain of problem (3.4). We define $I(x_C^{(k)}, l^{(k)}, u^{(k)})$ as the active set of $x_C^{(k)}$ with respect to the bounds $l^{(k)}$ and $u^{(k)}$, that is the set of indices of the variables at the GCP violating or lying on a constraint of (3.4), and solve the sub-problem (3.4) with the restriction that the variables in the active set remain fixed. The solution of this restricted sub-problem gives us the trial point $x^{(k)} + s^{(k)}$ and can be computed by using an iterative or a direct method. This is the subject of the next two Sections.

4. Minimization of the quadratic model by iterative methods.

If the Hessian $H^{(k)}$ is positive definite and the bounds (3.4.b) which are inactive at the Cauchy point remain inactive, the solution of the sub-problem (3.4) may be obtained as the solution of a system of linear equations. However, this may be prohibitively expensive in the context of large scale optimization unless care is taken to solve the resulting linear system as efficiently as possible. To date, the use of iterative schemes, in particular truncated, preconditioned conjugate gradients methods, seem to have been the most popular approach to solving the sub-problem as it is reflected in the works of Toint (1981), Steihaug (1983) or Stoer (1983).

Two main reasons explain this interest. The first one is that a truncated strategy which asymptotically takes the exact quasi-Newton step can save a significant amount of computation during the early iterations, when we are still far from the optimum. The second reason, and probably the main one, is that these types of methods do not involve operations on the matrix entries but only require matrix-vector products. This calculation, which represents the major cost of the algorithm, can be efficiently performed

when the sparsity pattern of the matrix is taken into account. In the context of partially separable optimization, the matrix is the Hessian of the objective function, and the matrix-vector product does not require the assembly of the Hessian since it can be computed as

$$(\sum_{i=1}^{m} H_i)\, d = \sum_{i=1}^{m} (U_i^T \hat{H}_i U_i)\, d. \tag{4.1}$$

The conjugate gradient algorithm is applied, starting from $x = x_C^{(k)}$, to the sub-problem (3.4) with the restriction that the variables in the set $I(x_C^{(k)}, l^{(k)}, u^{(k)})$ are kept fixed throughout the process. The algorithm we choose uses the diagonal of the Hessian as preconditioner and is terminated at the point $\bar{x}$ if

(i) the norm of the restricted gradient of the model, that is the vector whose components are those of the gradient of $m^{(k)}(x)$ at $x = \bar{x}$ not indexed by $I(x_C^{(k)}, l^{(k)}, u^{(k)})$, is less than $\eta^{(k)}$, for some $\eta^{(k)}$; or

(ii) one or more of the unrestricted variables violate one of the bounds of (3.4). $\bar{x}$ is then the point at which the offending bound(s) is (are) encountered.

A superlinear rate of convergence can be assured provided that 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 the matrices $H^{(k)}$ restricted to the set of variables active at the solution satisfy the Dennis-Moré condition (see Dennis and Schnabel (1983)). A suitable choice for $\eta^{(k)}$ in order to satisfy the first condition is given by

$$\eta^{(k)} = \min(0.1, \sqrt{\|\bar{g}^{(k)}\|})\, \|\bar{g}^{(k)}\|, \tag{4.2}$$

where $\bar{g}^{(k)}$ is the projected gradient, $P[x^{(k)} - g^{(k)}] - x^{(k)}$, at $x^{(k)}$.

5. Minimization of the quadratic model by a direct method.

Let us now consider the solution of the sub-problem (3.4) (where some of the variables are fixed) by a direct method.

We already remarked that the advantage of using the conjugate gradient method to solve (3.4) is that there is no need to assemble explicitly the Hessian. Surprisingly, this advantage can be maintained for a class of direct methods called the frontal methods, introduced by Irons (1970).

These methods solve symmetric positive-definite systems of linear equations

$$Ax = b \tag{5.1}$$

by Gaussian elimination where the matrix A has a finite-element representation

$$A = \sum_r B^{(r)}, \tag{5.2}$$

and where each $B^{(r)}$ is zero except in a small number of rows and columns.

In frontal methods, advantage is taken of the fact that elimination steps

$$a_{ij} := a_{ij} - a_{il}\, a_{ll}^{-1}\, a_{lj} \tag{5.3}$$

do not have to wait for all the assembly steps

$$a_{ij} := a_{ij} + b_{ij}^{(r)} \tag{5.4}$$

from (5.2) to be complete. The operation (5.3) can be performed as soon as the pivot row (and column) is fully summed, that is, as soon as all the operations (5.4) have been completed for them. The fully summed rows and columns (not yet eliminated) are stored in a so-called frontal matrix whose size can be maintained sufficiently small.

This scheme is perfectly adaptable to our framework except that we have no assumption of positive definiteness of the matrix A. Duff and Reid (1983) overcame this difficulty by following Bunch and Parlett (1971), using a mixture of 1x1 and 2x2 pivots chosen during the numerical factorization of A. This allows stable Gaussian elimination for indefinite systems.

This method has been implemented by Duff and Reid (1982) as a set of Fortran subroutines, currently available through the HARWELL library under the name MA27 and is the one we used for our implementation.

During the assembly steps of the frontal method, one needs to access the entries of each element Hessian $H_i^{(k)}$. Since the element Hessians are stored in internal variables, each time we need one of them we must restore it in elemental variables. The user is then asked to write a code to perform operation (2.6), in addition to the operations (2.7). Numerical experience shows that this can often be done efficiently since (2.6) requires few floating point operations. Furthermore, we maintain the advantages of a reduction of storage for the Hessian $H^{(k)}$ and of an efficient updating technique.

One must point out however, that the current version of MA27 does not assume a finite-element representation for the matrix A but requires instead its storage in compact form. Consequently, for our experimental code, one had to assemble the Hessian, even if this type of method does not require it. One must point out, however, that this does not inhibit the proof of the viability of the frontal approach and does not affect our conclusions. Moreover there are plans to introduce a new version of MA27 which allows an elemental representation of the coefficient matrix (Iain Duff, private communication).

Given the factorization of the Hessian (restricted to only the free variables)

$$H^{(k)} = L^{(k)} D^{(k)} L^{(k)\,\mathrm{T}}, \tag{5.5}$$

we check the 1x1 and 2x2 pivots stored in $D^{(k)}$ to decide whether it is positive definite, indefinite, or singular. Different strategies will be applied in these different cases.

If the restricted Hessian is positive definite, we can solve the Newton equations

$$H^{(k)} s = -g^{(k)} \tag{5.6}$$

for the direction s along which a step will be taken. The actual step is then computed as

$$s^{(k)} = \min(1, \alpha^{(k)})\, s \tag{5.7}$$

where $\alpha^{(k)}\, s$ is the largest admissible step for which (3.4.b) is satisfied. MA27 provides the code for the forward and backward substitutions that compute this direction s.

If $H^{(k)}$ is indefinite, the solution of (5.6) is no longer that of the sub-problem (3.4). Fortunately, the decomposition (5.5) allows us to compute a direction of negative curvature for (3.4), along which we will take the largest admissible step.

To compute a direction of negative curvature, s say, we first chose λ, the most negative eigenvalue of $D^{(k)}$ and computed the corresponding eigenvector v. The direction s would then be given at the cost of a backward substitution

$$s = L^{(k)\,-\mathrm{T}}\, v \tag{5.8}$$

and the corresponding curvature would be

$$s^{\mathrm{T}} H^{(k)} s = \lambda \|v\|^2. \tag{5.9}$$

Numerical experiments indicate a drawback of this method. When the restricted Hessian remains indefinite in successive iterations, the direction s often lies in the same subspace and the number of iterations required to reach optimality is abnormally high. To avoid this defect, when successive

directions of negative curvature are encountered, instead of choosing the most negative eigenvalue of $D^{(k)}$, we cycle on its negative eigenvalues. By cycling we mean that we choose the negative eigenvalue ordered next to the one used at the previous iteration until we reach the last, in which case we repeat the cycle starting with the most negative again.

The last case to consider is when the restricted Hessian is singular and semi-positive definite, although we noticed that it occurs very rarely in practice (and indeed, it never occured in our experiments of the next Section). The strategy used is the one described by Conn and Gould (1984) which consists in solving the linear system if it is consistent (trying to take a Newton step along this direction in the feasible domain), or finding a descent direction s otherwise, satisfying

$$H^{(k)}s=0 \text{ and } s^{\mathrm{T}}g^{(k)} \; 0, \tag{5.10}$$

along which the maximal admissible step will be taken.

6. Numerical experiments.

The test problems we used for our experiments come mainly from the set of functions used by Toint for testing partially separable codes. They are fully described in Toint (1983) and the numbering we use here refers to that paper. We considered the problems 8 (Tridiagonal quadratic), 10 (Rosenbrock function), 11 (Linear minimum surface), 16 (Boundary value problem), 17 (Broyden tridiagonal), 19 (Extended Powell singular function), 20 (Wrong extended Wood's function), 22 (Diagonal quadratic), 27 (Extended Wood's bounded problem), 29 (Extended McCormick's bounded problem), 31 (Extended ENGVL1), 33 (Extended Freudenstein), 36 (Cube problem) and 47 (PSPDOC example, with n–2 elements). We also considered five additional problems, described in the appendix to this paper, so as to allow for other sparsity structures in our test set.

For starting points, we decided to use the ones provided in Toint (1983). However, to test the sensitivity of our code to changes in the initial points, we tried a second point for some of the test problems (see Table 8). For these problems, the two different starting points are given in the appendix.

All the experiments were performed on the CRAY 2 supercomputer of Harwell Laboratory. Our code, called hereafter SBMIN, is written in Fortran 77 and compiled using the CFT77 Fortran compiler (without optimization). All timings reported are in seconds for time spent in the C.P.U.

The initial trust region radius $\Delta^{(0)}=0.1\,\|\tilde{g}^{(0)}\|_2$ and we chose $\mu=0.25$, $\eta=0.75$, $\gamma_0=1/\sqrt{10}$ and $\gamma_2=\sqrt{10}$. The stopping criteria we used was based on the order of magnitude of the gradient (projected on the feasible domain); we required its norm to be of less than 10^{-6}.

In the tables, the symbol * indicates that the trust region radius has become too small and that the routine has consequently decided to stop. One must point out however, that in those cases, the projected gradient norm was of the order of 10^{-4} or even 10^{-5} and the failure should be attributed to numerical rounding errors preventing the accurate calculation of the ratio ρ (equation 3.5). The symbol # means that the maximum number of iterations has been reached. In the cases where this occured, we again observed that the norm of the projected gradient was small.

For each test problem, we consider three different methods, conjugate gradients (cg), preconditioned conjugate gradients (pcg) and multifrontal (multif), to obtain an approximate solution to the subproblem (3.4). We also consider three ways of computing the element Hessians; exact derivatives (exact), the Broyden-Fletcher-Goldfarb-Shanno update (BFGS) and the symmetric rank-one update (rk1). Specifically, in the latter two cases the i–th element Hessian, $\hat{H}_i^{(k)}$, stored in terms of its internal variables, is updated from the BFGS formula

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

or from the rank-one formula

$$\hat{H}_i^{(k+1)} = \hat{H}_i^{(k)} + \frac{\hat{r}_i^{(k)}\,\hat{r}_i^{(k)T}}{\hat{r}_i^{(k)T}\,\hat{s}_i^{(k)}}.$$

Here $\hat{s}_i^{(k)}$ and $\hat{y}_i^{(k)}$ are, respectively, the change in the internal variables $U_i(x^{(k+1)} - x^{(k)})$, and the change in the elemental gradient $\hat{g}_i^{(k+1)} - \hat{g}_i^{(k)}$ and $\hat{r}_i^{(k)} \equiv \hat{y}_i^{(k)} - \hat{H}_i^{(k)}\,\hat{s}_i^{(k)}$. The BFGS update is only performed for a given element if the new approximation can be ensured to be positive definite, and this is implemented by only allowing an update if the condition

$$\hat{y}_i^{(k)T}\,\hat{s}_i^{(k)} / \hat{y}_i^{(k)T}\,\hat{y}_i^{(k)} \geq 10^{-8}$$

is satisfied. The rank-one update is only made when the correction has norm smaller than 10^8. The initial estimate of each element Hessian $H_i^{(0)}$ is set to the identity matrix when updating schemes are used. This choice is considered satisfactory as the test problems are reasonably well scaled.

The figures we report in each Table are the number of function evaluations (fct), the number of gradient evaluations (grad), the time spent in the main optimization routine (sbmin cpu) and the time spent in the main optimization routine added to the time for function and gradient evaluations (total cpu). Under the label ' syst. iter. ' we give the number of conjugate gradient iterations in the case of the iterative method or information of the type *pd nc* (*ratio*) for the direct methods where *pd* is the number of positive definite linear systems solved, *nc* is the number of directions of negative curvature taken and *ratio* gives an idea of the fill-in during the Gaussian elimination and is equal to the storage space needed to store the factor of the Gaussian decomposition divided by the space needed for the original matrix. In our tests, no singular linear systems were encountered despite our having code specially to deal with such eventualities.

Table 1 presents the performance of the different methods on a set of 19 problems ran with 100 variables.

pb	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
8	exact	cg	7	8	160	0.57	0.57
		pcg	6	7	32	0.19	0.19
		multif	1	2	1 0 (1.00)	0.03	0.04
	BFGS	cg	10	7	120	0.61	0.61
		pcg	10	7	30	0.36	0.36
		multif	6	3	1 0 (1.00)	0.20	0.20
	rk1	cg	10	7	120	0.61	0.61
		pcg	10	7	30	0.35	0.36
		multif	6	3	1 0 (1.00)	0.20	0.20
10	exact	cg	441	289	2064	5.68	5.88
		pcg	492	259	2133	6.19	6.38
		multif	527	272	512 11 (1.00)	12.42	12.63
	BFGS	cg	495	334	3133	8.55	8.76
		pcg	639	393	2088	7.91	8.17
		multif	363	319	356 0 (1.00)	9.21	9.38
	rk1	cg	1072	624	2098	11.70	12.13
		pcg	1093	641	2007	10.78	11.22
		multif	1619	935	586 993 (1.03)	39.87	40.51
11	exact	cg	35	29	80	0.62	0.63
		pcg	38	31	71	0.85	0.87
		multif	15	14	15 0 (1.89)	0.45	0.46
	BFGS	cg	17	18	154	0.64	0.65

		pcg	21	19	135	0.79	0.81
		multif	17	18	17 0 (1.89)	0.58	0.59
	rk1	cg	63	36	105	1.17	1.20
		pcg	49	35	74	1.22	1.24
		multif	109	67	16 93 (1.91)	3.40	3.45
16	exact	cg	3	4	254	0.93	0.93
		pcg	4	5	377	1.46	1.47
		multif	3	4	3 0 (1.00)	0.11	0.11
	BFGS	cg	17	17	2016	7.36	7.37
		pcg	18	18	2603	9.78	9.79
		multif	21	21	20 0 (1.00)	0.79	0.80
	rk1	cg	12	9	508	1.96	1.96
		pcg	10	9	911	3.52	3.53
		multif	6	6	4 1 (1.00)	0.21	0.21
17	exact	cg	7	8	29	0.17	0.17
		pcg	6	7	28	0.21	0.22
		multif	5	6	5 0 (1.00)	0.18	0.18
	BFGS	cg	11	9	30	0.25	0.25
		pcg	11	9	32	0.33	0.33
		multif	11	9	9 0 (1.00)	0.38	0.38
	rk1	cg	15	9	40	0.33	0.33
		pcg	11	9	35	0.33	0.34
		multif	10	8	8 0 (1.00)	0.33	0.34
19	exact	cg	15	16	95	0.19	0.20
		pcg	15	16	83	0.18	0.19
		multif	15	16	15 0 (1.00)	0.31	0.32
	BFGS	cg	51	45	2726	4.30	4.31
		pcg	51	44	1717	2.94	2.95
		multif	42	36	35 0 (1.00)	0.89	0.90
	rk1	cg	53	33	632	1.20	1.21
		pcg	70	38	746	1.52	1.53
		multif	72	48	27 40 (1.00)	1.47	1.48
20	exact	cg	11	12	35	0.12	0.12
		pcg	9	10	20	0.09	0.09
		multif	9	10	8 0 (1.00)	0.21	0.21
	BFGS	cg	53	37	122	0.64	0.65
		pcg	66	38	166	0.82	0.84
		multif	43	30	38 0 (1.01)	1.07	1.09
	rk1	cg	43	26	45	0.42	0.43
		pcg	55	32	69	0.54	0.56
		multif	72	39	13 53 (1.02)	1.73	1.75
22	exact	cg	6	7	8	0.06	0.06
		pcg	2	3	1	0.02	0.02
		multif	3	4	1 0 (1.00)	0.05	0.05
	BFGS	cg	22	15	76	0.38	0.39
		pcg	33	18	77	0.50	0.52
		multif	25	16	19 0 (1.00)	0.72	0.73
	rk1	cg	10	8	10	0.12	0.12
		pcg	10	8	4	0.11	0.11
		multif	6	4	1 0 (1.00)	0.08	0.09
27	exact	cg	9	10	9	0.07	0.07
		pcg	6	7	5	0.05	0.05
		multif	4	5	3 1 (1.18)	0.09	0.09
	BFGS	cg	21	16	37	0.23	0.24
		pcg	26	21	34	0.26	0.27
		multif	26	19	18 0 (1.00)	0.45	0.45
	rk1	cg	25	17	26	0.25	0.26
		pcg	28	16	19	0.24	0.25
		multif	34	23	9 18 (1.06)	0.57	0.58
29	exact	cg	9	8	12	0.06	0.08
		pcg	9	8	10	0.07	0.08
		multif	8	7	4 0 (1.00)	0.10	0.11
	BFGS	cg	12	11	15	0.13	0.14

		pcg	11	10	12	0.12	0.13
		multif	11	10	9 0 (1.00)	0.28	0.29
	rk1	cg	8	9	11	0.08	0.09
		pcg	8	9	9	0.09	0.09
		multif	17	8	4 10 (1.00)	0.39	0.40
31	exact	cg	8	9	21	0.07	0.08
		pcg	8	9	13	0.07	0.07
		multif	7	8	5 0 (1.00)	0.14	0.14
	BFGS	cg	13	11	31	0.16	0.16
		pcg	13	11	25	0.16	0.16
		multif	11	9	7 0 (1.00)	0.24	0.25
	rk1	cg	13	10	27	0.15	0.15
		pcg	25	10	25	0.22	0.23
		multif	15	10	6 5 (1.00)	0.33	0.34
33	exact	cg	11	12	25	0.09	0.10
		pcg	7	8	18	0.07	0.08
		multif	10	11	4 0 (1.00)	0.13	0.14
	BFGS	cg	19	15	26	0.19	0.21
		pcg	17	13	23	0.18	0.19
		multif	17	13	6 0 (1.00)	0.25	0.26
	rk1	cg	17	13	28	0.17	0.18
		pcg	18	13	27	0.18	0.19
		multif	37	14	7 20 (1.00)	0.75	0.77
36	exact	cg	1008	612	7615	17.18	17.61
		pcg	1028	596	4031	11.46	11.89
		multif	994	562	932 7 (1.00)	21.84	22.24
	BFGS	cg	1566	971	9092	25.41	26.04
		pcg	1401	936	4443	16.72	17.30
		multif	1237	853	1224 7 (1.00)	31.06	31.58
	rk1	cg	3154	1833	8788	34.16	35.40
		pcg	3017	1750	4732	27.07	28.26
		multif	4043	2293	1928 1943 (1.03)	97.54	99.12
47	exact	cg	8	9	34	0.21	0.21
		pcg	10	9	20	0.26	0.27
		multif	8	8	8 0 (1.00)	0.28	0.29
	BFGS	cg	8	8	25	0.21	0.21
		pcg	9	9	32	0.32	0.33
		multif	9	9	9 0 (1.00)	0.37	0.37
	rk1	cg	8	8	27	0.21	0.22
		pcg	8	8	26	0.27	0.28
		multif	9	9	7 0 (1.00)	0.30	0.30
55	exact	cg	5	6	4	0.03	0.04
		pcg	5	6	4	0.04	0.04
		multif	5	6	2 0 (1.00)	0.08	0.08
	BFGS	cg	12	9	2	0.10	0.10
		pcg	13	10	9	0.13	0.13
		multif	13	10	3 0 (1.00)	0.18	0.19
	rk1	cg	22	10	14	0.16	0.16
		pcg	15	10	11	0.13	0.13
		multif	22	10	12 0 (1.00)	0.21	0.21
56	exact	cg	12	13	14	0.16	0.17
		pcg	12	13	0	0.21	0.23
		multif	12	13	11 0 (1.00)	0.41	0.42
	BFGS	cg	19	20	430	1.86	1.88
		pcg	19	20	452	2.09	2.11
		multif	19	20	13 0 (1.00)	0.61	0.63
	rk1	cg	39	25	17	0.68	0.71
		pcg	32	22	26	0.70	0.73
		multif	33	22	1 22 (1.01)	0.96	1.01
57	exact	cg	107	68	1380	6.09	6.13
		pcg	133	80	971	5.62	5.67
		multif	15	16	11 0 (1.00)	0.50	0.51
	BFGS	cg	93	64	1140	5.32	5.36

		pcg	110	88	397	3.73	3.77
		multif	24	22	16 0 (1.00)	0.82	0.83
	rk1	cg	107	76	1378	6.22	6.26
		pcg	176	130	523	5.22	5.28
		multif	24	22	16 0 (1.00)	0.79	0.80
59	exact	cg	8	7	16	0.11	0.12
		pcg	10	9	18	0.14	0.16
		multif	12	9	4 5 (2.69)	0.45	0.47
	BFGS	cg	13	12	23	0.26	0.29
		pcg	13	12	13	0.25	0.27
		multif	11	10	8 0 (2.75)	0.47	0.49
	rk1	cg	10	9	8	0.17	0.18
		pcg	15	12	10	0.24	0.27
		multif	19	9	3 12 (2.48)	0.73	0.76
61	exact	cg	13	14	56	0.35	0.36
		pcg	11	12	25	0.22	0.23
		multif	11	12	10 0 (1.00)	0.55	0.56
	BFGS	cg	34	26	150	1.12	1.14
		pcg	26	20	66	0.70	0.71
		multif	26	20	18 0 (1.00)	1.20	1.21
	rk1	cg	45	22	131	1.19	1.21
		pcg	56	25	45	1.08	1.10
		multif	117	72	19 90 (1.00)	6.21	6.27

Table 1. Performance of the methods on a set of test problems with 100 variables.

This table indicates the viability of the multifrontal method in the context of partially separable optimization and more than that, we already see that the method seems to be really competitive with the iterative schemes in some cases. It is not rare that the number of function and gradient evaluations is significantly reduced and this can lead to significant improvements in terms of computation time. In order to investigate more carefully the relative performance of the methods, we increased the dimension n up to a maximum of 5000. The results of those experiments are given in Tables 2 to 7 for a subset of our test problems. The problems were chosen as being fairly representative of the larger set.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
484	exact	cg	169	125	357	14.62	15.05
		pcg	153	103	202	16.63	17.01
		multif	24	19	24 0 (3.45)	5.32	5.38
	BFGS	cg	32	26	540	10.56	10.64
		pcg	78	58	294	12.52	12.71
		multif	26	21	26 0 (3.43)	6.18	6.25
	rk1	cg	264	163	442	25.97	26.56
		pcg	214	136	198	23.05	23.53
		multif	498	321	27 471 (3.58)	113.65	114.77
961	exact	cg	517	470	712	165.29	171.78
		pcg	184	128	298	81.78	83.79
		multif	26	20	26 0 (4.17)	16.79	17.10
	BFGS	cg	51	38	538	48.16	48.66
		pcg	100	75	568	76.95	77.99
		multif	26	21	26 0 (4.17)	18.24	18.51
	rk1	cg	562	358	803	311.78	316.91
		pcg	440	286	371	179.01	183.01
		multif	2274	1414	58 2215 (4.32)	1589.50	1609.06
4900	exact	cg	2039	1755	3077	3521.02	3651.55
		pcg	580	441	769	1324.17	1358.05
		multif	36	29	36 0 (6.43)	136.29	138.49
	BFGS	cg	127	96	1601	741.55	748.15
		pcg	376	269	1448	1247.58	1266.58
		multif	32	25	32 0 (6.43)	131.93	133.63

	rk1	cg	–	–	–	–	> 20000
		pcg	2035	1327	1178	5957.36	6052.71
		multif	–	–	–	–	> 20000

Table 2. Performance of the methods on the test problem 11 for increasing dimensions.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
500	exact	cg	3	4	1634	29.06	29.07
		pcg	3	4	1839	33.34	33.35
		multif	5	6	5 0 (1.00)	0.87	0.88
	BFGS	cg	15	15	10280	181.61	181.65
		pcg	16	16	13131	240.93	240.97
		multif	23	23	22 0 (1.00)	4.40	4.46
	rk1	cg	17	11	495	9.91	9.95
		pcg	5	5	937	17.47	17.48
		multif	9	9	4 4 (1.00)	1.60	1.63
1000	exact	cg	2	3	256	18.19	18.24
		pcg	2	3	612	44.33	44.39
		multif	5	6	5 0 (1.00)	2.54	2.64
	BFGS	cg	15	15	14435	945.39	945.63
		pcg	19	19	26667	1772.99	1773.29
		multif	44	32	30 13 (1.00)	22.02	22.61
	rk1	cg	12	8	822	56.62	56.78
		pcg	7	6	3088	207.01	207.12
		multif	8	8	4 3 (1.01)	3.94	4.08
5000	exact	cg	2	3	1151	395.78	396.05
		pcg	2	3	1830	643.32	643.59
		multif	7	8	7 0 (1.00)	17.15	17.86
	BFGS	cg	19	11	4510	1581.51	1582.73
		pcg	19	14	14069	4743.19	4744.55
		multif	197	115	10 186 (1.01)	1365.62	1377.51
	rk1	cg	17	10	282	114.01	115.09
		pcg	23	13	158	88.42	89.84
		multif	125	71	4 121 (1.01)	657.81	665.37

Table 3. Performance of the methods on the test problem 16 for increasing dimensions.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
500	exact	cg	13	14	0	0.57	0.65
		pcg	13	14	0	1.15	1.22
		multif	13	14	0 0 (–)	0.58	0.65
	BFGS	cg	20	21	1015	19.92	20.02
		pcg	20	21	1431	28.79	28.89
		multif	20	21	12 0 (1.00)	2.96	3.06
	rk1	cg	30	22	8	2.31	2.44
		pcg	35	25	34	3.93	4.08
		multif	60	37	4 45 (1.00)	18.45	18.69
1000	exact	cg	13	14	14	2.36	2.64
		pcg	13	14	0	4.15	4.43
		multif	13	14	0 0 (–)	2.29	2.56
	BFGS	cg	21	22	1454	107.38	107.72
		pcg	21	22	2628	194.66	195.00
		multif	21	22	12 0 (1.00)	8.84	9.18
	rk1	cg	36	24	17	15.05	15.51
		pcg	35	26	29	13.86	14.33
		multif	42	27	0 30 (1.00)	48.71	49.24
5000	exact	cg	14	15	0	11.98	13.41
		pcg	14	15	0	21.11	22.54
		multif	14	15	0 0 (–)	11.94	13.37

	BFGS	cg	22	23	2170	747.86	749.58
		pcg	22	23	5998	2079.67	2081.38
		multif	22	23	11 0 (1.00)	42.53	44.25
	rk1	cg	36	25	13	92.25	94.55
		pcg	40	29	42	79.64	82.23
		multif	43	29	0 30 (1.00)	1588.61	1591.31

Table 4. Performance of the methods on the test problem 56 for increasing dimensions.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
500	exact	cg	115	75	1715	36.46	36.66
		pcg	180	120	852	29.80	30.13
		multif	16	17	10 0 (1.00)	3.45	3.49
	BFGS	cg	172	124	1751	42.91	43.22
		pcg	653	454	851	78.06	79.22
		multif	18	16	14 0 (1.00)	4.98	5.02
	rk1	cg	164	116	1572	38.16	38.45
		pcg	548	414	775	66.27	67.27
		multif	18	16	14 0 (1.00)	4.93	4.96
1000	exact	cg	141	88	1946	151.90	153.39
		pcg	224	140	1073	128.60	130.97
		multif	17	18	10 0 (1.00)	25.58	25.82
	BFGS	cg	276	164	2512	214.40	216.67
		pcg	568	405	852	231.57	237.16
		multif	19	17	15 0 (1.00)	38.34	38.56
	rk1	cg	213	150	2209	180.86	182.87
		pcg	554	430	825	218.12	223.56
		multif	19	17	15 0 (1.00)	41.18	41.39
5000	exact	cg	164	105	2296	879.14	887.70
		pcg	199	125	838	529.40	539.69
		multif	18	19	10 0 (1.00)	507.83	509.08
	BFGS	cg	260	181	2835	1192.99	1205.22
		pcg	1133	787	1418	2167.72	2221.16
		multif	20	18	16 0 (1.00)	844.19	845.28
	rk1	cg	252	178	2446	1069.75	1081.73
		pcg	993	695	1318	1900.92	1947.81
		multif	20	18	16 0 (1.00)	840.97	842.05

Table 5. Performance of the methods on the test problem 57 for increasing dimensions.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
500	exact	cg	10	7	18	0.66	0.74
		pcg	8	7	13	0.52	0.59
		multif	19	13	3 14 (7.35)	7.30	7.46
	BFGS	cg	14	13	39	1.63	1.76
		pcg	12	11	12	1.15	1.26
		multif	14	13	13 0 (7.23)	5.85	5.98
	rk1	cg	19	11	6	1.29	1.43
		pcg	19	11	14	1.44	1.58
		multif	33	15	3 23 (7.08)	11.24	11.46
1000	exact	cg	10	7	23	3.21	3.54
		pcg	14	11	31	4.67	5.16
		multif	40	24	6 31 (1.23)	62.80	63.99
	BFGS	cg	14	13	28	6.06	6.53
		pcg	14	13	15	5.63	6.10
		multif	13	12	12 0 (1.20)	19.15	19.58
	rk1	cg	24	12	9	7.42	8.01
		pcg	21	10	5	6.40	6.91
		multif	23	15	2 14 (1.20)	26.69	27.33

5000	exact	cg	20	14	40	29.15	32.33
		pcg	10	8	16	13.42	15.16
		multif	26	15	4 20 (5.38)	844.83	848.66
	BFGS	cg	15	14	58	48.11	50.56
		pcg	13	12	15	34.35	36.47
		multif	15	14	14 0 (5.23)	500.06	502.57
	rk1	cg	41	18	55	72.48	77.10
		pcg	28	17	7	56.45	60.08
		multif	39	19	2 27 (5.31)	1086.27	1090.98

Table 6. Performance of the methods on the test problem 59 for increasing dimensions.

n	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
500	exact	cg	13	14	55	1.77	1.83
		pcg	11	12	24	1.10	1.15
		multif	11	12	10 0 (1.00)	4.01	4.06
	BFGS	cg	37	26	139	5.79	5.89
		pcg	27	21	67	3.73	3.80
		multif	30	23	21 0 (1.00)	9.57	9.66
	rk1	cg	49	23	115	5.96	6.07
		pcg	54	26	68	5.60	5.72
		multif	397	234	91 296 (1.20)	156.62	157.56
1000	exact	cg	14	15	64	9.39	10.00
		pcg	11	12	24	5.47	5.96
		multif	12	13	10 0 (1.00)	41.39	41.92
	BFGS	cg	68	31	135	35.74	37.18
		pcg	74	35	61	39.84	* 41.42
		multif	68	32	49 0 (1.00)	129.13	130.59
	rk1	cg	49	25	138	31.92	33.00
		pcg	75	25	76	37.02	* 38.46
		multif	428	250	85 328 (1.00)	1784.35	* 1793.84
5000	exact	cg	44	21	79	104.15	* 110.02
		pcg	11	22	23	42.68	45.08
		multif	12	13	10 0 (1.00)	894.85	897.45
	BFGS	cg	75	35	137	234.24	* 239.94
		pcg	30	21	67	169.22	172.86
		multif	30	22	20 0 (1.00)	1915.80	1919.52
	rk1	cg	41	25	103	197.91	202.61
		pcg	54	25	68	205.04	210.64
		multif	–	–	–	–	> 20000

Table 7. Performance of the methods on the test problem 61 for increasing dimensions.

When exact derivatives are available, the multifrontal method seems competitive with respect to the conjugate gradients type algorithms in many cases. It appears to be also the case when the Broyden-Fletcher-Goldfarb-Shanno update is applied. If the symmetric rank-one update is used however, conjugate gradients methods are preferable.

We observe the excellent performances of the multifrontal method when the quadratic model is convex (see Table 5 for instance). However, if the fill-in of the factorization is too high, the number of operations for the Gaussian elimination is dominant and the multifrontal scheme is no longer competitive. A good example of this is shown in Table 6 where the sparsity pattern of the Hessian is randomly generated.

We observe particularly poor performances of the direct method when directions of negative curvature are taken. This happens, obviously, more often with the symmetric rank-one update of the Hessian. Illustration of this behaviour can be found is Tables 2, 3, 6 and 7. The proposed strategy seems to be clearly inefficient and other type of strategies must definitely be used to handle such cases, if direct methods are to be used.

We note that the negative curvature directions generated by the rank-one update are always taken into account when using our multifrontal approach, in contrast with the (preconditioned) conjugate gradient technique. This last calculation only considers the first Krylov subspaces spanned by the gradient and its successive images by the Hessian approximation, especially in the early iterations when this approximation may be quite poor and the trust region radius small. The comparison of the performances of the rank-one update with the (preconditioned) conjugate gradient and the multifrontal scheme, in cases where negative curvature is encountered, tends to show that this possible ignorance of the negative curvature in the early stages of the computation might be advantageous.

We were also interested in the behaviour of our methods with respect to the starting point, since we aimed for globally convergent algorithms. Table 8 shows the results of these experiments. For each method, the two lines refer respectively to the first and second starting points given in the appendix. It is important to note that the choice of the starting point does not affect the conclusions that we have drawn in this Section.

Finally, in Conn, Gould and Toint (1988b), the simplest of the updating schemes, the symmetric rank-one method, appeared to perform better than the BFGS method for many of the problems tested. It seems that, in the context of partial separability, this conclusion does not appear to apply. One could attempt to explain this phenomenon by the fact that it is not uncommon for the projection of successive search directions into the range space of certain elements to be close to linear dependence despite the independence of the overall search directions. Linear dependence of the search directions can be highly undesirable for the rank-one update.

pb	Hessian	method	fct.	grad.	syst. iter.	sbmin cpu	total cpu
11	exact	cg	169	125	357	14.62	15.05
			145	103	365	27.15	27.92
		pcg	153	103	202	16.63	17.01
			135	93	227	29.63	30.34
		multif	24	19	24 0 (3.45)	5.32	5.38
			19	16	19 0 (3.45)	5.56	5.67
	BFGS	cg	32	26	540	10.56	10.64
			24	23	477	17.93	18.06
		pcg	78	58	294	12.52	12.71
			80	55	413	26.84	27.21
		multif	26	21	26 0 (3.43)	6.18	6.25
			26	21	26 0 (3.43)	8.20	8.33
	rk1	cg	264	163	442	25.97	26.56
			299	184	438	71.67	72.99
		pcg	214	136	198	23.05	23.53
			229	151	219	45.76	46.79
		multif	498	321	27 471 (3.58)	113.65	114.77
			811	532	30 780 (3.59)	260.33	263.96
16	exact	cg	3	4	1634	29.06	29.07
			4	5	3088	105.86	105.90
		pcg	3	4	1839	33.34	33.35
			3	4	1666	58.42	58.46
		multif	5	6	5 0 (1.00)	0.87	0.88
			5	6	5 0 (1.00)	1.27	1.32
	BFGS	cg	15	15	10280	181.61	181.65
			17	17	14008	482.41	482.56
		pcg	16	16	13131	240.93	240.97
			18	18	15615	542.59	542.74
		multif	23	23	22 0 (1.00)	4.40	4.46
			33	33	32 0 (1.00)	8.72	8.99
	rk1	cg	17	11	495	9.91	9.95
			15	10	1850	65.70	65.81
		pcg	5	5	937	17.47	17.48
			8	7	3084	110.00	110.07

		multif	9	9	4 4 (1.00)	1.60	1.63
			13	11	4 8 (1.00)	3.35	3.45
56	exact	cg	13	14	0	0.57	0.65
			2	3	0	0.16	0.18
		pcg	13	14	0	1.15	1.22
			2	3	0	0.34	0.36
		multif	13	14	0 0 (–)	0.58	0.65
			2	3	0 0 (–)	1.16	1.18
	BFGS	cg	20	21	1015	19.92	20.02
			6	7	2	0.50	0.54
		pcg	20	21	1431	28.79	28.89
			22	23	38	2.28	2.35
		multif	20	21	12 0 (1.00)	2.96	3.06
			6	7	2 0 (1.00)	0.51	0.55
	rk1	cg	30	22	8	2.31	2.44
			10	8	2	0.45	0.48
		pcg	35	25	34	3.93	4.08
			6	7	13	0.83	0.86
		multif	60	37	4 45 (1.00)	18.45	18.69
			10	8	2 0 (1.00)	0.45	0.48
57	exact	cg	115	75	1715	36.46	36.66
			1379	842	28824	1041.14	1048.35
		pcg	180	120	852	29.80	30.13
			4839	2924	17447	1209.96	1235.47
		multif	16	17	10 0 (1.00)	3.45	3.49
			22	23	11 0 (1.00)	8.96	9.12
	BFGS	cg	172	124	1751	42.91	43.22
			1709	1225	27124	1063.56	1072.01
		pcg	653	454	851	78.06	79.22
			10000	6610	12206	1873.86	# 1921.59
		multif	18	16	14 0 (1.00)	4.98	5.02
			26	24	16 0 (1.00)	13.57	13.71
	rk1	cg	164	116	1572	38.16	38.45
			1614	1157	24370	966.64	974.67
		pcg	548	414	775	66.27	67.27
			10000	6485	12458	1904.63	# 1952.06
		multif	18	16	14 0 (1.00)	4.93	4.96
			26	24	16 0 (1.00)	13.65	13.79
59	exact	cg	10	7	18	0.66	0.74
			3	4	1	0.29	0.37
		pcg	8	7	13	0.52	0.59
			3	4	1	0.32	0.40
		multif	19	13	3 14 (7.35)	7.30	7.46
			3	4	1 0 (6.96)	0.68	0.76
	BFGS	cg	14	13	39	1.63	1.76
			10	11	19	1.90	2.09
		pcg	12	11	12	1.15	1.26
			10	11	11	1.76	1.95
		multif	14	13	13 0 (7.23)	5.85	5.98
			10	11	10 0 (7.23)	5.51	5.70
	rk1	cg	19	11	6	1.29	1.43
			19	9	8	2.69	2.92
		pcg	19	11	14	1.44	1.58
			16	10	2	2.04	2.26
		multif	33	15	3 23 (7.08)	11.24	11.46
			27	12	1 19 (7.30)	10.97	11.29
61	exact	cg	13	14	55	1.77	1.83
			19	20	77	5.54	5.95
		pcg	11	12	24	1.10	1.15
			16	17	32	3.40	3.74
		multif	11	12	10 0 (1.00)	4.01	4.06
			16	17	15 0 (1.00)	18.24	18.59
	BFGS	cg	37	26	139	5.79	5.89
			65	28	146	16.78	* 17.45
		pcg	27	21	67	3.73	3.80
			41	25	164	16.15	16.63

	multif	30	23	21 0 (1.00)	9.57	9.66
		29	21	24 0 (1.00)	31.88	32.24
rk1	cg	49	23	115	5.96	6.07
		53	28	147	16.48	17.06
	pcg	54	26	68	5.60	5.72
		40	19	69	11.41	11.84
	multif	397	234	91 296 (1.20)	156.62	157.56
		350	228	37 308 (1.24)	428.64	432.79

Table 8. Changing the starting point (n=500, except for problem 11 where n=484).

7. Conclusions.

The numerical experiments show that the use of a direct method instead of an iterative one can lead to very significant improvements in terms of computation time. They also clearly demonstrated that these improvements can only be achieved in cases where the quadratic model of the function at the current iterate is convex (or at least not too often non convex) and the structure of the Hessian is sufficiently regular to avoid high fill-in during the factorization.

When the approximation of the function Hessian is indefinite, the use of directions of negative curvature inhibits fast convergence of direct methods and can even lead to dramatic increase of computation time. This conclusion convinced the authors that in this particular case, other types of strategies must definitely be used.

The main conclusion is that an efficient code for partially separable optimization must provide the user a choice of methods more adapted to the specific problem being solved. The authors intend to provide such a code through the HARWELL library in the future.

Acknowledgements.

The project has been financially supported by the Belgian National Fund for Scientific Research, Harwell Laboratory (U.K.A.E.A.) and by grant A8639 from the Natural Sciences and Engineering Research Council of Canada. This support is gratefully acknowledged.

Appendix.

The five problems that we added to introduce other type of sparsity structures are now given. For each of them we mention (a) the element functions, (b) any bounds on the variables and (c) the starting point.

1. Test problem 55.

(a) $f_i(x)=(x_i^2+x_n^2)^2-4x_i+3$, for i=1,...,n–1.

(c) $x=(1,1,...,1)$.

2. Test problem 56.

(a) $f_i(x)=(x_i+x_{i+1})\,e^{-x_{i+2}(x_i+x_{i+1})}$, for i=1,...,n–2.

(b) $x_i\ 0, i$=1,...n.

(c) $x=(1,1,...,1)$.

3. Test problem 57.

(a) $f_i(x)=(x_i+x_{i+1}+x_n)^4$, for i=1,...,n–2,

$f_{n-1}(x)=(x_1-x_2)^2$,

$f_n(x)=(x_{n-1}-x_n)^2$.

(c) $x=(1,-1,1,-1,...)$.

4. Test problem 59.

(a) $f_i(x)=x_i^2 e^{-x_{j_i}}$, for $i=1,...,n$,

$f_i(x)=x_{i-n}^2 e^{-x_{j_i}}$, for $i=n+1,...,2n$,

where the indices j_i are randomly generated between 1 and n in order by subroutine FA04BS of the Harwell Subroutine library, starting with the default seed, but with the provision that any j_i equal to i is rejected and the next random number in the sequence taken.

(c) $x=(1,-1,1,-1,...)$.

5. Test problem 61.

(a) $f_i(x)=(x_i^2+2x_{i+1}^2+3x_{i+2}^2+4x_{i+3}^2+5x_n^2)^2-4x_i+3$, for $i=1,...,n-4$.

(c) $x=(1,1,1,...)$.

Below, we give the initial points used for the experiments reported in Table 8. For each problem, we have two different starting points.

1. Test problem 11.

The variables on the boundaries of the unit square are kept fixed at the values :

$x_i=1+4t$,

$x_{(i-1)p+1}=1+8t$,

$x_{i+p(p-1)}=9+4t$,

$x_{ip}=5+8t$,

for $i=1,...,p$. Here $t=(i-1)/(p-1)$, while p is given by the relation $n=p^2$. The two different starting points are determined by the initialization of the free variables, namely

(1) free variables initialized to 0, and

(2) free variables initialized to 1.

2. Test problem 16.

(1) $x_i=ih(ih-1)$ for $i=2,...n-1$,

$x_1=0$,

$x_n=0$.

(2) $x_i=2ih(ih-1)$ for $i=2,...n-1$,

$x_1 = 0,$

$x_n = 0.$

3. Test problem 56.

(1) $x=(1,1,...1).$

(2) $x=(1,0,1,0,...).$

4. Test problem 57.

(1) $x=(1,-1,1,-1,...).$

(2) $x=(10,-10,10,-10,...).$

5. Test problem 59.

(1) $x=(1,-1,1,-1,...).$

(2) $x=(0.1,-0.1,0.1,-0.1,...).$

6. Test problem 61.

(1) $x=(1,1,...,1).$

(2) $x=(10,10,...,10).$

References.

Bunch, J.R. and Parlett, B.N. (1971). Direct methods for solving symmetric indefinite systems of linear equations. SIAM J. Numer. Anal. 8, 639-655.

Conn, A.R. and Gould, N.I.M. (1984). On the location of directions of infinite descent for nonlinear programming algorithms. SIAM J. Numer. Anal., Vol. 21, No. 6, 302-325.

Conn, A.R., Gould, N.I.M. and Toint, Ph.L. (1988a). Global convergence of a class of trust region algorithms for optimization with simple bounds. SIAM J. Numer. Anal., Vol. 25, No. 2, 433-460.

Conn, A.R., Gould, N.I.M. and Toint, Ph.L. (1988b). Testing a class of methods for solving minimization problems with simple bounds on the variables. To appear in Mathematics of Computation.

Dennis, J.E. and Schnabel, R.B. (1983). Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, New Jersey.

Duff, I.S. and Reid, J.K. (1982). MA27 – A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Report HL82/2225 (C13),

Duff, I.S. and Reid, J.K. (1983). The multifrontal solution of indefinite sparse symmetric linear equations. ACM Transactions on Mathematical Software, Vol. 9, No. 3, 302-325.

Griewank, A. and Toint, Ph.L. (1982). Partitioned variable metric updates for large structured optimization problems, Numerische Mathematik, vol. 39, 119-137.

Griewank, A. and Toint, Ph.L. (1984). Numerical experiments with partially separable optimization problems, in Numerical Analysis: Proceedings Dundee 1983 (D.F. Griffiths,ed.), Lecture Notes in Mathematics 1066, Springer Verlag, Berlin, 203-220.

Irons, B.M. (1970). A frontal solution program for finite element analysis. Int. J. Numer. Meth. Eng. 2, 5-32.

Moré, J.J. (1977). The Levenberg-Marquardt algorithm : implementation and theory. Numerical Analysis, G.A. Watson, ed., Lecture Notes in Math. 630, Springer-Verlag, Berlin, 105-116.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. SIAM Journal on Numerical Analysis, Vol.20, No. 3, 626-637.

Stoer, J. (1983). Solution of large linear systems of equations by conjugate gradient type methods, in Mathematical Programming : The State of the Art (Bonn 1982), A. Bachem, M. Grötschel and B. Korte (eds.), Springer Verlag, Berlin, 540-565.

Toint, Ph.L. (1981). Towards an efficient sparsity exploiting Newton method for minimization, in Sparse Matrices and their Uses (I. S. Duff, ed.), Academic Press, London.

Toint, Ph.L. (1983). Test problems for partially separable optimization and results for the routine PSPMIN. Report 83/4 of the FUNDP, Namur.