MATHEMATICS OF COMPUTATION, VOLUME 32, NUMBER 143
JULY 1978, PAGES 839–851

Some Numerical Results Using a Sparse Matrix Updating Formula in Unconstrained Optimization

By Ph. L. Toint*

Abstract. This paper presents a numerical comparison between algorithms for unconstrained optimization that take account of sparsity in the second derivative matrix of the objective function. Some of the methods included in the comparison use difference approximation schemes to evaluate the second derivative matrix and others use an approximation to it which is updated regularly using the changes in the gradient. These results show what method to use in what circumstances and also suggest interesting future developments.

1. Introduction. Numerical procedures for finding the minimum of a general function of several variables have been studied extensively in the last few years; and many powerful algorithms have been proposed, which require the calculation of the vector of first derivatives (see Davidon [2], Fletcher and Powell [4], Huang [6], Powell [8], for example). One common feature of these procedures is the fact that they use a matrix which can be regarded as an approximation to the second derivative matrix of the objective function. This matrix is updated regularly, so that the updated matrix satisfies a linear equation which is known usually as the "quasi-Newton equation" or "DFP condition". Because all the classical updating formulae revise all the elements of the matrix, the size of the problem that can be treated is often limited by the amount of computer storage that is available.

However, a way to take any present sparsity into account has been suggested by Powell [11]. It does not change the elements of the approximating matrix corresponding to known zero entries in the hessian of the objective function. Toint [14] shows that the calculation can be arranged so that its main part is only the solution of a sparse, symmetric and positive definite system of linear equations.

The main purpose of this paper is to present numerical results that compare this procedure with some other methods that also use first derivative information, in order to show the actual and computational value of the proposed method.

The other methods included in the comparison are Powell's method [8], since the new formula may be viewed as an extension of it to the sparse case, and methods using difference approximations to the hessian. When sparsity conditions

Received August 15, 1977.
AMS (MOS) subject classifications (1970). Primary 65K05; Secondary 90C30.
Key words and phrases. Numerical results, sparsity, quasi-Newton methods, unconstrained optimization.
*This work was done during a visit of the author at the Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England.

are known, the use of differences usually requires much fewer gradient calculations than in the full case. The actual criterion in the comparison is the number of gradient evaluations, since most of the computing time is spent on them when the function is somewhat complex.

It is shown that the sparse updating formula is a valuable contribution to unconstrained optimization; and also that a more sophisticated procedure is still desirable in order to exploit its full power.

Section 2 of this paper is concerned with the general problem formulation and presents the different minimization procedures that are tested. Section 3 considers the test functions that are used. Finally, numerical results are presented and discussed in Section 4.

2. Problem Formulation and Presentation of the Algorithms. We consider a function $f(x)$ from R^n into R which is twice differentiable. We assume also that some sparsity pattern is known for $\nabla^2 f(x) = H(x)$, the matrix of second derivatives, i.e. that some entries of $H(x)$ are known to be zero for all $x \in R^n$. Given any starting point x_0 in R^n, we are trying to find a point x^* such that

$$f(x^*) = \min_{x \in V(x^*)} f(x), \tag{1}$$

where $V(x^*)$ is a neighborhood of x^*. That is to say that x^* is a local minimum of $f(x)$.

All the algorithms we consider for finding x^* are iterative and use, at the iteration k, an approximation B_k to $H(x_k)$ where x_k is the current approximation to x^*. The gradient vector $f(x)$ will be denoted by $g(x)$ and the symbol $\langle \cdot, \cdot \rangle$ will stand for the inner product in R^n.

We compare different formulae for obtaining a new hessian approximation, i.e. different ways of deriving B_{k+1}. The formulae we consider are the following ones.

General Powell's Algorithm (GPA). In [8], Powell proposed an updating formula for obtaining B_{k+1} from B_k. It uses the difference in points $x_{k+1} - x_k$ and the difference in gradients $g(x_{k+1}) - g(x_k)$. His formula, in fact, solves the problem

$$\min \| B_k - B_{k+1} \|_F \tag{2}$$

subject to the conditions

$$B_{k+1} = B_{k+1}^T, \tag{3}$$

$$B_{k+1}(x_{k+1} - x_k) = g(x_{k+1}) - g(x_k), \tag{4}$$

where the subscript F denotes the Frobenius matrix norm and where (4) is the "quasi-Newton equation". For a proof of this statement, see Dennis and Moré [3]. The formula does not take any sparsity into account.

Sparse Updating Algorithm (*SUA*). This formula is obtained by solving the problem (2)–(4) when sparsity conserving constraints are added. These constraints are of the form

$$[B_{k+1}]_{ij} = 0, \quad (i, j) \in I, \tag{5}$$

where I is a set of pairs of indices denoting the positions of the known zero entries in $H(x)$. A derivation of a good method of calculation may be found in [14], with a discussion and proof of some of its properties.

Algorithms Using Explicit Second Derivative Information. Since sparsity is present in $H(x)$, estimating its nonzero entries by using differences in first derivatives may be relatively cheap. Therefore, algorithms that use this approach may be competitive with the first two methods. In the tests presented in this paper, exact second derivatives are used in order to avoid the difficulties that arise from the choice of steplength in derivative estimation. An interpretation of the amount of work required by difference approximations is given with the numerical results. The following three approaches are studied, which depend on the frequency of the second derivative calculations.

The first one is to evaluate the second derivative matrix at each iteration. This is done by Newton's method, so we refer to the resulting procedure as the Newton algorithm (NA).

The second possibility is to evaluate second derivatives every l*p*th iteration ($l = 1, 2, \ldots$) and to let $B_{k+1} = B_k$, otherwise. In this comparison, all tests were run with $p = 6$. The resulting procedure will be referred to as the constant Newton algorithm (CNA).

The third possibility is to evaluate the second derivative matrix every l*p*th iteration ($l = 1, 2, \ldots$) as in CNA, but to update it on all other iterations using the sparse updating formula, as in SUA, to include the information given by the successive gradient evaluations. Here also $p = 6$ was chosen for all the tests. The resulting procedure will be called the updated Newton algorithm (UNA).

It is important to observe that none of the procedures given above can guarantee the positive definiteness of the approximation B_k, and thus the usual variable metric implementation of the updating formulae cannot be used. However, since our aim is to compare these formulae, a common implementation is needed for them. Therefore, we use an implementation that is similar to the one proposed in [8] for the GPA update. It may be described as follows.

Step 1. Let $x_0 \in R^n$ be any starting point, Δ_0 an initial steplength, B_0 an initial approximation to the hessian of $f(x)$ and ϵ a small constant representing the required gradient accuracy. Compute $f(x_0)$ and $g(x_0)$ and set $k = 0$.

Step 2. If $\|g(x_k)\| < \epsilon$, stop. Otherwise, continue.

Step 3. Compute a vector δ_k such that $x_k + \delta_k$ is the value of x that minimizes

$$q(x, k) = \tfrac{1}{2}\langle x - x_k, B_k(x - x_k)\rangle + \langle g(x_k), x - x_k\rangle + f(x_k), \tag{6}$$

where $\|\delta_k\|_2$ is bounded by Δ_k.

Step 4. Compute $f(x_k + \delta_k)$ and $g(x_k + \delta_k)$.

Step 5. *If the values of* $f(x_k + \delta_k)$ and $g(x_k + \delta_k)$ are found to be close to the predicted values $q(x_k + \delta_k, k)$ and $\nabla q(x_k + \delta_k, k)$, then set $\Delta_{k+1} = \|\delta_k\|$ or $\Delta_{k+1} = 2\|\delta_k\|$. Otherwise, set $\Delta_{k+1} = \frac{1}{2}\|\delta_k\|$.

Step 6. Compute the new approximation to the hessian, namely B_{k+1} by using one of the methods that have been described.

Step 7. If $f(x_k + \delta_k) < f(x_k)$, set $x_{k+1} = x_k + \delta_k$. Otherwise, set $x_{k+1} = x_k$. Set $k = k + 1$ and go to *Step* 2.

In *Step* 1, B_0 is chosen as $H(x_0)$ when explicit second derivative information is available (in NA, CNA and UNA). In the other cases (GPA and SUA), B_0 is chosen as 0.01 $\|g(x_0)\|/\Delta_0$ times the unit matrix. *Step* 3 is implemented using a Levenberg-Marquardt procedure, very similar to the one described in Hebden [5]. It is interesting to observe that because in this procedure only linear equation solutions are needed, advantage may be taken of sparsity by using a method such as [12]. *Step* 5 is implemented as proposed by Powell in [9]. The five different formulae are used in *Step* 6 and give the five algorithms that are compared. We are now ready to examine the test functions that are used in the comparison.

i	α_i	i	α_i	i	α_i	i	α_i	i	α_i
1	1.25	11	1.50	21	1.80	31	1.25	41	2.20
2	1.40	12	1.60	22	0.75	32	1.25	42	1.40
3	2.40	13	1.25	23	1.25	33	1.25	43	1.50
4	1.40	14	1.25	24	1.40	34	3.00	44	1.25
5	1.75	15	1.20	25	1.60	35	1.50	45	2.00
6	1.20	16	1.20	26	2.00	36	2.00	46	1.50
7	2.25	17	1.40	27	1.00	37	1.25	47	1.25
8	1.20	18	0.50	28	1.60	38	1.40	48	1.40
9	1.00	19	0.50	29	1.25	39	1.80	49	0.60
10	1.10	20	1.25	30	2.75	40	1.50	50	1.50

TABLE 1. *Parameters* α_i

3. Test Functions.

3.1. *Two Functions Arising from Operations Research.* The two functions below arise from a very simple operations research model of a market of interconnected producers and consumers. The first one is derived from a simple model and is the quadratic function

$$\text{(QOR)} \qquad f(x) = \sum_{i=1}^{50} \alpha_i x_i^2 + \sum_{i=1}^{33} \beta_i \left[d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j \right]^2,$$

where the constants α_i, β_i and d_i are given in Table 1 and Table 2, together with the $A(i)$ and $B(i)$ which are subsets of the indices $\{1, \ldots, 50\}$. The sparsity pattern of the hessian is determined by these sets and is shown in Figure 1. The starting point is the zero vector.

i	β_i	d_i	A(i)	B(i)	i	β_i	d_i	A(i)	B(i)
1	1.0	5	31	1	18	0.1	2	32	33, 34
2	1.5	5	1	2, 3	19	3.0	9	3, 33	35
3	1.0	5	2	4, 5	20	0.1	2	35	21, 36
4	0.1	2.5	4	6, 7	21	1.2	5	36	37, 38
5	1.5	6	6	8, 9	22	1.0	5	30, 37	39
6	2.0	6	8	10, 11	23	0.1	2.5	38, 39	40
7	1.0	5	10	12, 13	24	2.0	5	40	41, 42
8	1.5	6	12	14, 15	25	1.2	6	41	43, 44, 50
9	3.0	10	11, 13, 14	16, 17	26	3.0	10	44	45, 46, 47
10	2.0	6	16	18, 19	27	1.5	7	46	48
11	1.0	5	9, 18	20	28	3.0	10	42, 45, 48, 50	49
12	3.0	9	5, 20, 21	---	29	2.0	6	26, 34, 43	---
13	0.1	2	19	22, 23, 24	30	1.0	5	15, 17, 24, 47	---
14	1.5	7	23	25, 26	31	1.2	4	49	---
15	0.15	2.5	7, 25	27, 28	32	2.0	4	22	---
16	2.0	6	28	29, 30	33	1.0	4	27	---
17	1.0	5	29	31, 32					

TABLE 2. *Parameters β_i, d_i, $A(i)$ and $B(i)$*

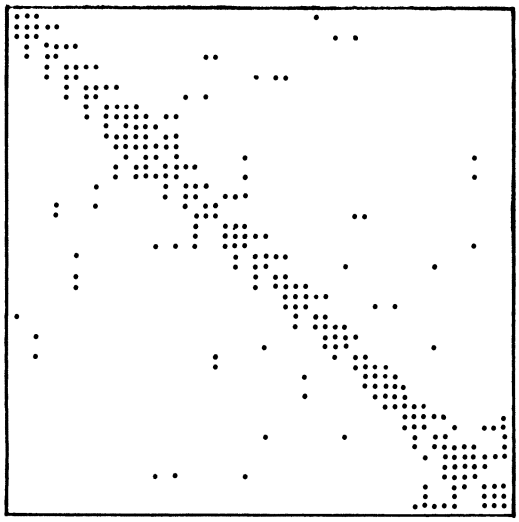

FIGURE 1

A more elaborate model gives the objective function

(GOR) $$f(x) = \sum_{i=1}^{50} c_i(x_i) + \sum_{i=1}^{33} b_i(y_i),$$

where

$$c_i(x_i) = \begin{cases} \alpha_i x_i \ln(1 + x_i), & x_i \geqslant 0, \\ -\alpha_i x_i \ln(1 - x_i), & x_i < 0, \end{cases}$$

$$y_i = d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j \tag{7}$$

and

$$b_i(y_i) = \begin{cases} \beta_i y_i^2 \ln(1 + y_i), & y_i \geqslant 0, \\ \beta_i y_i^2, & y_i < 0. \end{cases}$$

The constants α_i, β_i, d_i and the sets $A(i)$ and $B(i)$ are the same as in QOR and so is the sparsity pattern of its second derivative matrix (see Figure 1). As in QOR, the starting point is the zero vector.

Both these functions (QOR and GOR) are convex. There are discontinuities in the second derivative of GOR.

3.2. *A Pseudo Penalty Function.* Another function was constructed with the same sparsity pattern as QOR, but with a very nonquadratic behavior. It looks very much like a penalty function and is given by the relation

$$f(x) = \sum_{i=1}^{50} \alpha_i (x_i - 5)^2 + \sum_{i=1}^{33} \beta_i h_i(y_i), \tag{PSP}$$

where the quantities y_i are defined by (7) and where

$$h_i(y_i) = \begin{cases} 1/y_i, & y_i \geqslant 0.1, \\ 100(0.1 - y_i) + 10, & y_i < 0.1. \end{cases}$$

The constants α_i, β_i and the sets $A(i)$ and $B(i)$ occurring in (7) are the same as in QOR. The starting point is again the zero vector.

3.3. *A Chained Rosenbrock Function.* In order to provide a nonconvex function, an extension of the well-known Rosenbrock function of two variables was designed. It is the function

$$f(x) = \sum_{i=2}^{25} [4\alpha_i (x_{i-1} - x_i^2)^2 + (1 - x_i)^2], \tag{CR}$$

where the constants α_i are still chosen from Table 1 for $i = 2, \ldots, 25$. The hessian of this function is tridiagonal. The starting point is $x_0 = (-1, -1, \ldots, -1)$.

3.4. *A 7-Diagonal Function.* This function was built from a nonlinear system of equations suggested originally by Broyden (for this system, see [13], for example). It is defined as

$$f(x) = \sum_{i=1}^{60} |y_i|^{7/3} + \sum_{i=1}^{30} |x_i + x_{i+30}|^{7/3}, \tag{G7D}$$

where

$$y_1 = -\left(3 - \frac{x_1}{2}\right) x_1 + 2x_2 - 1,$$

$$y_i = x_{i-1} - \left(3 - \frac{x_i}{2}\right) x_i + 2x_{i+1} - 1, \qquad i = 2, \ldots, 59,$$

$$y_{60} = x_{59} - \left(3 - \frac{x_{60}}{2}\right) x_{60} - 1.$$

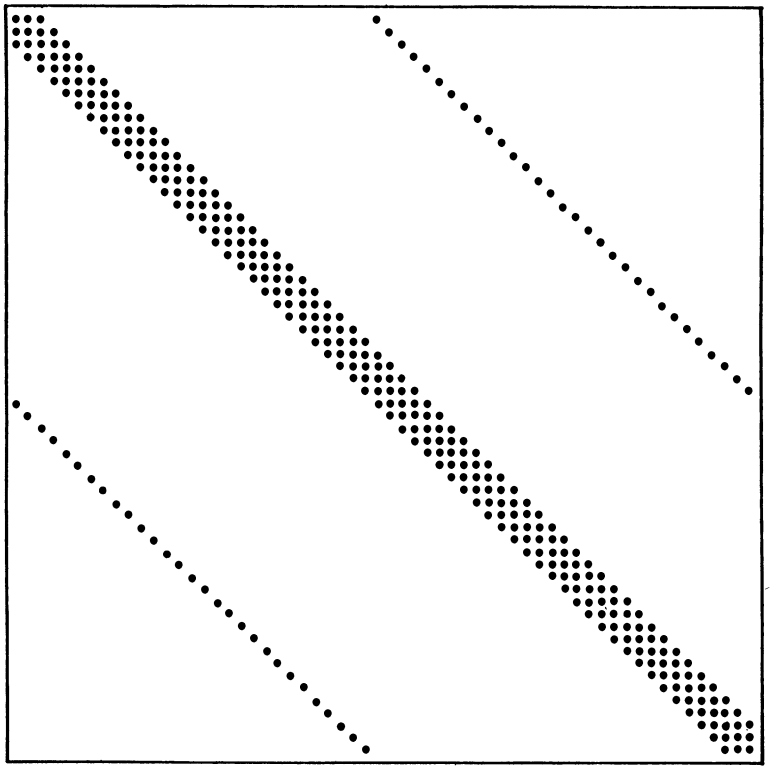

FIGURE 2

The sparsity pattern of the hessian of this function is shown in Figure 2. We note that some nonzero elements are far from the diagonal. The starting point is $x_0 = (-1, -1, \ldots, -1)$.

3.5. *A Variational Problem.* The next function arises from the Rayleigh-Ritz discretization of the following variational problem:

$$\min_{y \in C^1[0,1]} \int_0^1 \left\{ \left[\frac{dy(t)}{dt} \right]^2 + 2\lambda e^{y(t)} \right\} dt$$

subject to the constraints (boundary conditions)

$$y(0) = y(1) = 0,$$

where $\lambda \in [-3.5, +\infty]$. This problem is considered in Osborne [7]. For the discretization, we use the "hat functions" as a basis of the solution. They are defined by the equations

$$\phi_i(t) = \begin{cases} (t - t_{i-1})/h, & t \in [t_{i-1}, t_i], \\ (t_{i+1} - t)/h, & t \in [t_i, t_{i+1}], \\ 0, & \text{elsewhere}, \end{cases} \qquad i = 1, \ldots, n,$$

where $\{t_i\}_{i=0}^{n+1}$ are the discretization points with $t_i = ih$ $(i = 0, 1, \ldots, n+1)$ and $t_{n+1} = 1$. Expressing the solution in the form $y(t) = \Sigma_{i=1}^{n} \alpha_i \phi_i(t)$, which incorporates

the boundary conditions, we obtain the following objective function to minimize

(VAR(λ))
$$f(\alpha) = \frac{2}{h}\left[\sum_{i=1}^{n} \alpha_i^2 - \sum_{i=1}^{n-1} \alpha_i\alpha_{i+1}\right] + 2\lambda h\left[\frac{e^{\alpha_1} - 1}{\alpha_1} + \sum_{i=1}^{n-1} \frac{e^{\alpha_{i+1}} - e^{\alpha_i}}{\alpha_{i+1} - \alpha_i} + \frac{e^{\alpha_n} - 1}{\alpha_n}\right].$$

This function has a tridiagonal hessian matrix. It is convex for all the considered values of λ and its quadratic when $\lambda = 0$. A truncated Taylor series is used when the rounding errors due to cancellation become large. Tests were run with the following choices of parameters:

$n = 75$ $\lambda = 20$,
$\lambda = 6$,
$\lambda = -0.3$,
$\lambda = -3$,
$\lambda = -3.4$,
$n = 45$ $\lambda = -3.4$,
$n = 20$ $\lambda = -3.4$.

The starting point is

$$[\alpha_0]_i = \frac{0.1i}{n+1}(1 - i) \quad \text{for } i = 1, \ldots, n.$$

3.6. *A Trigonometric Function.* The last function is a trigonometric function with a second derivative matrix of general sparsity pattern. A set of pairs of indices is chosen: $J = \{(i, j) \mid 1 \leqslant i \leqslant n \text{ and } 1 \leqslant j < i\}$. The function is then defined as

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

where the x_{ij}, β_i and c_{ij} are defined by

$$\alpha_{ij} = 5[1 + \text{mod}(i, 5) + \text{mod}(j, 5)],$$
$$\beta_i = 1 + i/10,$$
$$c_{ij} = (i + j)/10.$$

The sparsity pattern of the second derivative matrix depends on J. Figure 3 shows the pattern that was used for $n = 100$. For smaller values of n ($n = 25, 50$) the sparsity pattern is the leading $n \times n$ submatrix of the one given in Figure 3. The function TRIG is nonconvex because it is bounded above and below. The starting point is $x = (1, 1, \ldots, 1)$.

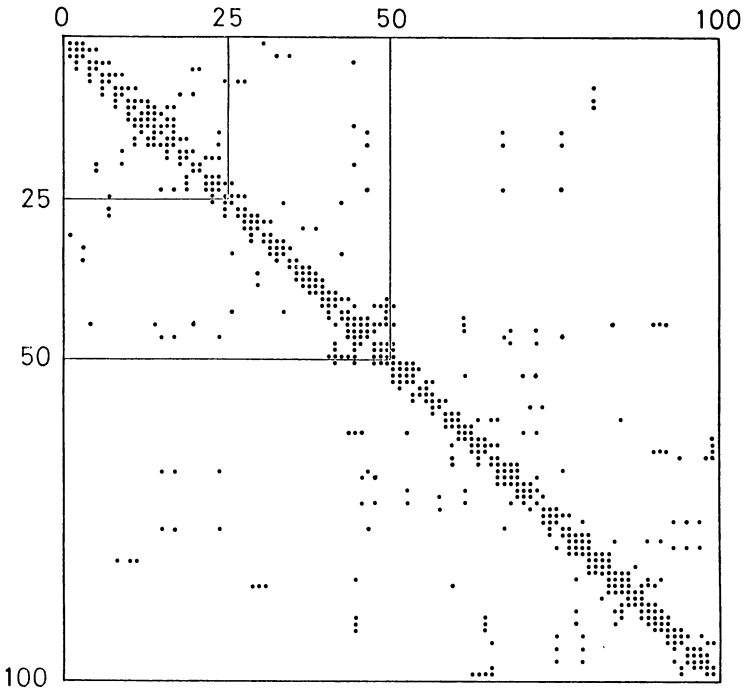

FIGURE 3

4. Numerical Results and Discussion. As stated in the introduction, the comparison between the algorithms is based on the number of gradient evaluations. This criterion is justified by the fact that, for complex functions, the work required to obtain one gradient is usually of an order greater than n^2 while the internal work required by the algorithms is usually of this order, except for GPA. In that case indeed the work needed to solve the linear systems in the Levenberg-Marquardt procedure of *Step* 3 is of the order of n^3 while for the other algorithms it is usually of the order of n^2, depending on the sparsity that can be exploited in the solution of these linear systems.

However, if the evaluation of the function requires an amount of work comparable to the evaluation of the gradient and may be done separately (i.e. one does not need to compute the function each time the gradient is evaluated), the number of function evaluations, which is the number of iterations of the algorithm plus one, becomes more and more important. The number of iterations is also important when the work devoted to the internal matrix calculations becomes comparable to the work of evaluating the function and the gradient. Therefore, this number is also considered in the comparison.

All tests were run with the same accuracy requirement ($\epsilon = 10^{-5}$). For all the functions, the global minimum was found, except for TRIG and G7D where different local minima were found by the different algorithms. All tests were run on the Cambridge University IBM 370/165 in double length arithmetic.

In Tables 3 and 4, n is the number of variables while s is the number of nonzero entries of the hessian that require explicit computation divided by n. It gives a

measure of the sparsity present in the problem: high sparsity corresponds to low s. Δ_0 is the initial steplength used on the different functions. A sign ">" in the results means that the algorithm was stopped at the corresponding number of gradient evaluations or iterations without having converged to any local minimum.

Table 3 compares GPA and SUA.

Function	n	s	Δ_o	number of gradient evaluations	
				GPA	SUA
QOR	50	3.30	1	50	23
GOR	50	3.30	1	102	49
PSP	50	3.30	1	113	132
CR	25	1.96	1	>93	40
VAR(-3.4)	20	1.95	2	69	31
VAR(-3.4)	45	1.98	2	>150	46

TABLE 3. *Comparison of GPA and SUA*

The number of iterations is not given in the table because it is always one less than the number of gradient evaluations.

One can see a clear advantage in taking account of sparsity in the evaluation of the approximation to the hessian. This is to be expected because SUA is given more information about the true second derivative matrix.

However, GPA is superior to SUA on the function PSP, which is most unlike a quadratic function. The explanation may be that approximating more closely the local quadratic behavior is, in this case, of little use to the global convergence.

Before comparing SUA with NA, CNA and UNA, it is necessary to make a few comments on the way in which the number of gradient evaluations is calculated. Since our aim is to compare methods using first derivatives, we assume that the second derivative information in NA, CNA and UNA can be obtained by using a forward difference approximation scheme in the gradients. For this interpretation, we consider two different approximation schemes.

The first scheme consists of evaluating each one of the entries of the hessian separately by calculating differences in the relevant first derivative component. The number of these entries is generally a small multiple of n since sparsity is present. Moreover, only the nonzero entries below and on the diagonal need to be computed because of symmetry. It is easy to observe that this method is optimal in terms of gradient component evaluations. (Thus an interpretation of the parameter s is that it is a lower bound on the number of full gradient evaluations required to obtain one approximation to the hessian.) However, this method supposes that it is possible to calculate each component of the gradient separately. The parameter k_1 in Table 4 is relevant to this scheme. It is the sum of the number of gradient evaluations occurring in *Step* 4 plus $1/n$ times the number of gradient components that are

evaluated to form the difference approximations to the hessian. It may be regarded as the overall number of gradient evaluations needed by the algorithms NA, CNA and UNA to achieve convergence.

Another suitable scheme for estimating the hessian matrix by difference, is proposed by Curtis, Powell and Reid [1]. It uses full gradient evaluations in a way that takes advantage of sparsity, but it does not take account of symmetry. No procedure of this type that takes symmetry into account is known to the author. The parameter k_2 in Table 4 is the number of gradient evaluations when the Curtis-Powell-Reid strategy is used to evaluate the hessian. The parameter m is the number of gradient evaluations needed to obtain one hessian using this strategy.

In Table 4, the parameter "*it*" is the number of iterations required by each method to find a local minimum. The number of function evaluations has not been reported since it is always equal to $it + 1$. The parameter k_3 for SUA is the number of gradient evaluations required to reach the local minimum. Some of its values appear already in Table 3.

Examining Table 4, we find that NA is less efficient than UNA and, except perhaps for TRIG with 100 variables, less efficient than CNA. Therefore, it is not advantageous to evaluate the hessian matrix by finite differences on every iteration. The effort required to calculate one approximation should be used on more than one iteration.

Secondly, we note that in all these calculations, UNA is superior to CNA. Therefore, it is valuable to incorporate in the second derivative matrix the changes in gradient that occur from the intermediate gradient evaluations.

For the comparison between SUA and UNA, two points are worthy to be made: 1) on all convex problems, UNA performed better than SUA. It may mean that in this case it is worthwhile to calculate more accurate second derivative estimates. On the other hand, SUA performed better on the problems CR and TRIG which are not convex. Therefore, far from the optimum and the zone surrounding it where the function is convex, it is not worthwhile to put much effort in accurate second derivative calculations. In this case, first derivatives make a greater contribution to the steps of the algorithms. 2) On the convex problem VAR(− 3.4), the amount of work required by SUA increases about linearly with the dimension of the problem, while the work required by UNA remains more or less constant. This may indicate another superiority of UNA in the convex case.

When the number of iterations of function evaluations becomes important, NA is clearly superior. However, UNA seems to remain a reasonable choice because it combines a small number of gradient evaluations with a relatively small number of iterations.

It is also to be observed that the performance of UNA depends on the frequency at which the hessian matrix is evaluated. The number $p = 6$ chosen here is unlikely to be optimal for all the examples.

An interesting idea is to develop a procedure that chooses this frequency automatically, depending on the convexity and uniformity of the function. Perhaps

Function	n	s	m	Δ_o	SUA		NA			CNA			UNA		
					it	k_3	it	k_1	k_2	it	k_1	k_2	it	k_1	k_2
QOR	50	3.30	10	20	22*	23	1	5.30	12	1	5.30	12	1	5.30	12
GOR	50	3.30	10	20	48*	49	6	26.80	77	13	23.90	44	12	22.90	43
CR	25	1.96	3	1	39	40	18	54.28	76	45	61.68	70	32	44.76	51
G7D	60	3.45	10	5	86	87	14	63.30	155	31	52.70	92	18	32.80	59
VAR(20)	75	1.99	3	2	30	31	4	12.96	17	8	12.90	15	6	10.90	13
VAR(6)	75	1.99	3	2	57	58	3	9.97	16	6	10.98	13	4	6.99	8
VAR(-0.3)	75	1.99	3	2	63	64	1	3.99	5	1	3.99	5	1	3.99	5
VAR(-3)	75	1.99	3	2	>90	>90	4	12.96	17	6	10.98	13	5	7.99	9
VAR(-3.4)	20	1.95	3	2	30	31	4	12.80	17	8	12.90	15	6	10.90	13
VAR(-3.4)	45	1.98	3	2	45	46	5	15.90	21	9	13.96	16	6	10.96	13
VAR(-3.4)	75	1.99	3	2	60	61	5	15.96	21	9	13.98	16	6	10.98	13
TRIG	25	3.04	9	2	24	25	15	61.60	151	35	54.24	78	18	31.16	46
TRIG	50	3.36	10	2	44	45	24	105.64	265	42	69.88	123	30	51.16	91
TRIG	100	3.69	20	2	70	71	56	263.64	1177	>87	>142.35	>837	53	87.21	234

*A steplength $\Delta_0 = 1$ was used in this case.

TABLE 4. *Comparison between SUA, NA, CNA and UNA*

such a procedure could blend the advantages of UNA and SUA.

Finally, we note that under conditions on $f(x)$ similar to those assumed in [10], the sparse updating algorithm (SUA) can be proved to converge superlinearly to a local minimum of $f(x)$. Since the proof is not relevant to a numerical comparison and is similar to [10], it has been omitted here.

5. Conclusion. This paper shows that the use of the sparse updating formula SUA for unconstrained optimization problems may be very useful, not only because the amount of storage needed is low when applied on large sparse problems, but also to reduce the number of function and gradient evaluations. This is especially true when the updating formula is combined with some difference approximation scheme for the second derivative matrix.

Nevertheless, a more sophisticated procedure using this sparse update seems still to be desirable, in order to use all the available information in an optimal way.

6. Acknowledgements. The author wishes to express his gratitude to Professor M. J. D. Powell for considerable encouragements and many useful comments on the early drafts of this paper. The author also thanks the Royal Society and the European Science Exchange Program for their financial support.

Department of Mathematics
FNDP
Belgium

1. A. R. CURTIS, M. J. D. POWELL & J. K. REID, "On the estimation of sparse Jacobian matrices," *J. Inst. Math. Appl.*, v. 13, 1974, pp. 117–119.

2. W. C. DAVIDON, *Variable Metric Method for Minimization,* A. N. L. Research and Development Report, ANL-5990 (Rev.), 1959.

3. J. E. DENNIS & J. MORÉ, "Quasi Newton methods, motivation and theory," *SIAM Rev.*, v. 19, 1977, pp. 46–89.

4. R. FLETCHER & M. J. D. POWELL, "A rapidly convergent descent method for minimization," *Comput. J.*, v. 6, 1963, pp. 163–168.

5. M. D. HEBDEN, *An Algorithm for Minimization Using Exact Second Derivatives,* Report T. P. 515, AERE, Harwell, 1973.

6. H. Y. HUANG, "Unified quadratically convergent algorithms for function minimization," *J. Optimization Theory Appl.*, v. 5, 1970, No. 6.

7. M. R. OSBORNE, "Variational methods and extrapolation procedures," in *Methods in Computational Physics* (R. S. Anderssen & R. O. Watts, Editors), Univ. of Queensland Press, Brisbane, 1975.

8. M. J. D. POWELL, "A new algorithm for unconstrained optimization" in *Nonlinear Programming* (J. B. Rosen, O. L. Mangasarian & K. Ritter, Editors), Academic Press, New York, 1970.

9. M. J. D. POWELL, *A Fortran Subroutine for Unconstrained Minimization, Requiring First Derivatives of the Objective Function,* Report R-6469, AERE, Harwell, 1970.

10. M. J. D. POWELL, "Convergence properties of a class of minimization algorithms" in *Nonlinear Programming*, 2 (O. L. Mangasarian, R. R. Meyer & S. M. Robinson, Editors), Academic Press, New York, 1975.

11. M. J. D. POWELL, "A view of unconstrained optimization" in *Optimization in Action* (L.C.W. Dixon, Editor), Academic Press, New York, 1976.

12. J. K. REID, *Two Fortran Subroutines for Direct Solution of Linear Equations, whose Matrix is Sparse, Symmetric and Positive Definite,* Report R-7119, AERE, Harwell, 1972.

13. L. K. SCHUBERT, "Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian," *Math. Comp.*, v. 24, 1970, pp. 27-30.

14. Ph. L. TOINT, "On sparse and symmetric matrix updating subject to a linear equation," *Math. Comp.*, v. 31, 1977, pp. 954-961.